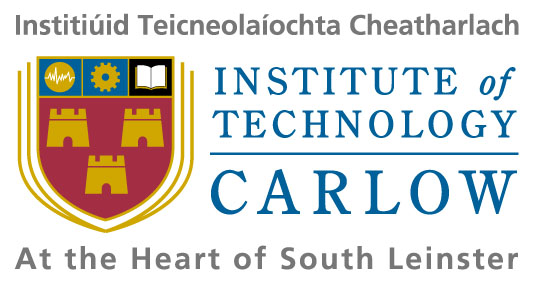
2020





Technical Manual

Holly Gunning

C014010

Supervisor: DR. Hisain Elshaafi

# Abstract

The purpose of this project is to create a progressive, single page web application to aid small to medium size clinics and their patients with a more online based experience, so as to reduce the need for as many physical consultations, enabling a more direct approach to prognosis and thereby limiting the spread of contagions amongst patients.

The application allows patients to book appointments online, keep track of their personal medical record, request prescriptions digitally rather than in person as well as record any medication they are taking. Doctors can easily view a database of all registered patients and make amendments to their medical records as well as review prescription requests and either grant or deny them.

Table of Contents

[Abstract 1](#_Toc50078428)

[Set Up Instructions 5](#_Toc50078429)

[Accessing Web Application 5](#_Toc50078430)

[Instructions to Download 5](#_Toc50078431)

[Login 6](#_Toc50078432)

[Login.vue 6](#_Toc50078433)

[TsAndCs.vue 29](#_Toc50078434)

[Firebase 34](#_Toc50078435)

[Firebase.js 34](#_Toc50078436)

[Firestore.rules 36](#_Toc50078437)

[Firestore.indexes.json 38](#_Toc50078438)

[Storage.rules 41](#_Toc50078439)

[Firebase Functions - Index.js 42](#_Toc50078440)

[Store 44](#_Toc50078441)

[Index.js 44](#_Toc50078442)

[Navbars 48](#_Toc50078443)

[Navbar.vue 48](#_Toc50078444)

[DoctorNavbar.vue 51](#_Toc50078445)

[AdminNavbar.vue 53](#_Toc50078446)

[Central App 54](#_Toc50078447)

[App.vue 54](#_Toc50078448)

[Main.js 55](#_Toc50078449)

[Admin 57](#_Toc50078450)

[Admin.vue 57](#_Toc50078451)

[Navigation Router 62](#_Toc50078452)

[Router index.js 62](#_Toc50078453)

[Profile 68](#_Toc50078454)

[Profile.vue 68](#_Toc50078455)

[PatientAppCalendar.vue 82](#_Toc50078456)

[Dashboard 89](#_Toc50078457)

[Dashboard.vue 89](#_Toc50078458)

[Medical Record 96](#_Toc50078459)

[MedicalRecord.vue 96](#_Toc50078460)

[Doctor.vue 117](#_Toc50078461)

[Allergy Component – Allergies.vue 172](#_Toc50078462)

[Condition Component – Conditions.vue 180](#_Toc50078463)

[Immunisation Component – Immunisation.vue 191](#_Toc50078464)

[Track Medication 202](#_Toc50078465)

[Medication.vue 202](#_Toc50078466)

[Calendar Component – MedicationCalendar.vue 203](#_Toc50078467)

[Appointments 221](#_Toc50078468)

[Appointments.vue 221](#_Toc50078469)

[View Component - ViewAppointments.vue 239](#_Toc50078470)

[DoctorAppointment.vue 250](#_Toc50078471)

[Appointment Calendar Component – Calendar.vue 251](#_Toc50078472)

[Prescriptions 258](#_Toc50078473)

[Prescriptions.vue 258](#_Toc50078474)

[ViewPrescriptions.vue 300](#_Toc50078475)

[DoctorPrescription.vue 310](#_Toc50078476)

[Chat 331](#_Toc50078477)

[Chat.vue 331](#_Toc50078478)

[DoctorRooms.vue 343](#_Toc50078479)

[Plagiarism Declaration 356](#_Toc50078480)

Introduction

The purpose of this document is to provide any necessary instructions to the reader as well as to provide them with all code relevant to the creation of the application. Due to the nature of the application being web based, there are no necessary installations required before usage. The user must simply go to the web address listed in the document and register as a user. They will also be supplied with the admin credentials for this application so that they can see the differences between the patient and doctor screens. Any code generated by Vue or firebase has been omitted from this document. If the user wishes to see the full code in a more development friendly environment, they need only visit the repository on GitHub at <https://github.com/HollyGunning/E-Dochtuir>.

# Set Up Instructions

## Accessing Web Application

The web application can be accessed by going to <https://e-dochtuir.web.app/login>, which will direct the user to the login screen of the application. Any user wishing to view the application can simply register and immediately see the patient view.

To view the admin page, an admin account has been created with the credentials:

* Email – holly.jade.gunning@gmail.com
* Password – SoftEng2020

## Instructions to Download

The application can be accessed through the web browser on any device. On a desktop, in the search bar a plus button on the far right allows the user to download it to their laptop or tablet. On a mobile when the user access the application on their browser, a banner appears with the message “Add to home” on the bottom of the screen, and allows the user to install the application to their mobile so that they may use it as if it were a native mobile application.

# Login

## Login.vue

<template>

<v-container>

<div id="login">

<div :class="{ 'signup-form': !showLoginForm }" class="col2">

<v-card flat class="mt-0">

<!-- Login Form -->

<v-form class="px-2" v-if="showLoginForm" @submit.prevent="login">

<v-card-title class="text-uppercase">Login</v-card-title>

<v-divider class="mx4"></v-divider>

<v-card-text>

<v-row>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

type="email"

name="email"

v-model.trim="loginForm.email"

:error-messages="loginEmailErrors"

label="E-mail Address"

required

outlined

@input="$v.loginForm.email.$touch()"

@blur="$v.loginForm.email.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

name="password"

:append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"

v-model.trim="loginForm.password"

:error-messages="loginPasswordErrors"

label="Password"

:type="showPassword ? 'text' : 'password'"

required

outlined

@click:append="showPassword = !showPassword"

@input="$v.loginForm.password.$touch()"

@blur="$v.loginForm.password.$touch()"

>

</v-text-field>

</v-col>

<v-col></v-col>

<v-col class="mt-n8 text-right" cols="12" md="6">

<v-dialog v-model="dialog" max-width="500px">

<template v-slot:activator="{ on, attrs }">

<v-btn

right

color="primary"

dark

text

v-bind="attrs"

v-on="on">

Forgot Password?

</v-btn>

</template>

<!-- Dialog box for forgot password -->

<v-card v-if="!showSuccess">

<v-form class="px-2" @submit.prevent="resetPassword">

<v-card-title class="text-uppercase mt-2">Forgot Password</v-card-title>

<v-divider class="mx4"></v-divider>

<v-card-text>

<h2 class="text-center">Enter your email to reset your password</h2>

<v-row class="mt-4">

<v-col class="mt-n2" cols="12" md="12">

<v-text-field

type="email"

name="email"

v-model.trim="forgotForm.email"

:error-messages="emailErrors"

label="E-Mail Address"

required

outlined

@input="$v.forgotForm.email.$touch()"

@blur="$v.forgotForm.email.$touch()"

>

</v-text-field>

</v-col>

</v-row>

<v-card-actions>

<v-row class="mt-n2">

<v-btn

type="submit"

block

class="primary white--text"

@click="resetPassword()">

<span>Reset Password</span>

</v-btn>

</v-row>

</v-card-actions>

<v-flex v-if="errorMsg !== ''">

<v-alert type="error" dismissible v-model="alert3">

{{ errorMsg }}

</v-alert>

</v-flex>

</v-card-text>

</v-form>

</v-card>

<!-- Show success message on password reset email -->

<v-card v-else>

<v-card-title class="text-uppercase">Success!</v-card-title>

<v-divider class="mx4"></v-divider>

<v-card-text>

<v-row>

<v-col class="mt-n2" cols="12" md="12">

<h2 class="pa-2 d-flex justify-center mt-2">Check your email for a reset link.</h2>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-dialog>

</v-col>

</v-row>

<v-card-actions>

<v-row>

<v-btn

type="submit"

:disabled="loading"

block class="primary white--text"

@click="login()">

<span>Login</span>

</v-btn>

</v-row>

</v-card-actions>

<v-flex>

<v-alert type="error" dismissible v-model="alert2">

{{ loginError }}

</v-alert>

</v-flex>

</v-card-text>

<v-divider class="mt-2"></v-divider>

<v-card-actions>

<v-btn

color="primary"

text

@click="toggleForm()">

Create Account

</v-btn>

</v-card-actions>

</v-form>

<!-- SIGNUP FORM -->

<v-form class="px-2" v-else @submit.prevent="signup">

<v-card-title class="text-uppercase">Register</v-card-title>

<v-divider class="mx4"></v-divider>

<v-card-text>

<v-row>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

type="text"

name="firstname"

v-model.trim="signupForm.firstname"

:error-messages="signupFirstNameErrors"

label="First Name\*"

:counter="15"

required

outlined

@input="$v.signupForm.firstname.$touch()"

@blur="$v.signupForm.firstname.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

type="text"

name="surname"

v-model.trim="signupForm.surname"

:error-messages="signupSurnameErrors"

label="Surname\*"

:counter="15"

required

outlined

@input="$v.signupForm.surname.$touch()"

@blur="$v.signupForm.surname.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-menu

v-model="menu"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

:value="formattedDate"

clearable

:error-messages="dateErrors"

label="Date of Birth\*"

hint="Date of Birth must be 16 or older"

persistent-hint

readonly

v-bind="attrs"

v-on="on"

required

outlined

@input="$v.date.$touch()"

@blur="$v.date.$touch()"

@click:clear="date = null"

>

</v-text-field>

</template>

<v-date-picker

v-model.trim="date"

:max="getLowestPossible()"

:min="getEarliestPossible()"

@change="menu = false"

>

</v-date-picker>

</v-menu>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

type="text"

name="ppsn"

hint="7 numerical characters, followed by either 1 or 2 letters E.g. 1234567RW"

persistent-hint

v-model.trim="signupForm.ppsn"

:error-messages="signupPPSNErrors"

label="PPSN\*"

required

outlined

@input="$v.signupForm.ppsn.$touch()"

@blur="$v.signupForm.ppsn.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

type="email"

name="email"

v-model.trim="signupForm.email"

:error-messages="signupEmailErrors"

label="E-Mail Address\*"

required

outlined

@input="$v.signupForm.email.$touch()"

@blur="$v.signupForm.email.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

name="mobile"

v-model.trim="signupForm.mobile"

:error-messages="signupMobileErrors"

label="Mobile Number\*"

required

outlined

@input="$v.signupForm.mobile.$touch()"

@blur="$v.signupForm.mobile.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

name="password"

:append-icon="showPassword2 ? 'mdi-eye' : 'mdi-eye-off'"

:type="showPassword2 ? 'text' : 'password'"

hint="At least 8 characters"

v-model.trim="signupForm.password"

:error-messages="signupPasswordErrors"

label="Password\*"

:counter="16"

required

outlined

@click:append="showPassword2 = !showPassword2"

@input="$v.signupForm.password.$touch()"

@blur="$v.signupForm.password.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n2" cols="12" md="6">

<v-text-field

name="confirmPassword"

:append-icon="showPassword2 ? 'mdi-eye' : 'mdi-eye-off'"

:type="showPassword2 ? 'text' : 'password'"

v-model.trim="signupForm.confirmPassword"

:error-messages="signupConfirmErrors"

label="Confirm Password\*"

required

outlined

@click:append="showPassword2 = !showPassword2"

@input="$v.signupForm.confirmPassword.$touch()"

@blur="$v.signupForm.confirmPassword.$touch()"

>

</v-text-field>

</v-col>

<v-col class="mt-n8" cols="12" md="6">

<v-checkbox

type="checkbox"

name="checkbox"

v-model.trim="signupForm.checkbox"

:error-messages="checkboxErrors"

required

@input="$v.signupForm.checkbox.$touch()"

@blur="$v.signupForm.checkbox.$touch()"

>

<template v-slot:label>

<span>I have read and agree to the

<a @click.stop.prevent="dialog2 = true">Terms of Service\*</a>

</span>

</template>

</v-checkbox>

</v-col>

<!-- Ts & Cs Field -->

<v-dialog

v-model="dialog2"

absolute

max-width="400"

persistent

>

<v-card>

<v-card-title class="text-uppercase">Terms & Conditions</v-card-title>

<v-divider></v-divider>

<TsAndCs />

<v-card-actions>

<v-btn

text

@click="signupForm.checkbox = false, dialog2 = false"

>

No

</v-btn>

<v-spacer></v-spacer>

<v-btn

class="white--text"

color="primary"

@click="signupForm.checkbox = true, dialog2 = false"

>

Yes

</v-btn>

</v-card-actions>

</v-card>

</v-dialog>

<v-spacer></v-spacer>

<caption class="mr-4">\*indicates required field</caption>

</v-row>

<v-card-actions>

<v-row>

<v-btn

type="submit"

:disabled="loading"

block class="primary white--text"

@click.prevent="signup()">

<span>Register</span>

</v-btn>

</v-row>

</v-card-actions>

<v-flex>

<v-alert type="error" dismissible v-model="alert1">

{{ registerError }}

</v-alert>

</v-flex>

</v-card-text>

<v-divider class="mt-2"></v-divider>

<v-card-actions>

<v-btn

color="primary"

text

@click="toggleForm()">

Already Registered?

</v-btn>

</v-card-actions>

</v-form>

</v-card>

</div> <!-- switch from login to sign up -->

</div> <!-- id=login close -->

</v-container>

</template>

<script>

import { auth } from '@/firebase'

import { required, email, maxLength, minLength, alpha, numeric, sameAs } from "vuelidate/lib/validators"

import format from 'date-fns/format'

import parseISO from 'date-fns/parseISO'

import TsAndCs from '../components/TsAndCs'

export default {

components: {

TsAndCs,

},

computed: {

// Returning the state of the errors

registerError () {

return this.$store.state.registerError

},

loginError () {

return this.$store.state.loginError

},

loading () {

return this.$store.state.loading

},

formattedDate () {

return this.date ? format(parseISO(this.date), 'do MMM yyyy') : ''

},

// VALIDATION ERROR MESSAGES

emailErrors () {

const errors = []

if (!this.$v.forgotForm.email.$dirty) return errors

!this.$v.forgotForm.email.email && errors.push('Invalid Email Address')

!this.$v.forgotForm.email.required && errors.push('E-mail is required')

return errors

},

loginEmailErrors () {

const errors = []

if (!this.$v.loginForm.email.$dirty) return errors

!this.$v.loginForm.email.email && errors.push('Invalid Email Address')

!this.$v.loginForm.email.required && errors.push('E-mail is required')

return errors

},

loginPasswordErrors () {

const errors = []

if (!this.$v.loginForm.password.$dirty) return errors

!this.$v.loginForm.password.maxLength && errors.push('Password exceeds maximum length of 16 characters')

!this.$v.loginForm.password.required && errors.push('Password is required')

return errors

},

signupFirstNameErrors () {

const errors = []

if (!this.$v.signupForm.firstname.$dirty) return errors

!this.$v.signupForm.firstname.alpha && errors.push('Alphabetical characters only')

!this.$v.signupForm.firstname.minLength && errors.push('First Name must be at least 3 characters long')

!this.$v.signupForm.firstname.maxLength && errors.push('First Name should not exceed 15 characters')

!this.$v.signupForm.firstname.required && errors.push('First Name is required')

return errors

},

signupSurnameErrors () {

const errors = []

if (!this.$v.signupForm.surname.$dirty) return errors

!this.$v.signupForm.surname.alpha && errors.push('Alphabetical characters only')

!this.$v.signupForm.surname.minLength && errors.push('Surname must be at least 3 characters long')

!this.$v.signupForm.surname.maxLength && errors.push('Surname should not exceed 15 characters')

!this.$v.signupForm.surname.required && errors.push('Surname is required')

return errors

},

signupEmailErrors () {

const errors = []

if (!this.$v.signupForm.email.$dirty) return errors

!this.$v.signupForm.email.email && errors.push('Invalid Email Address')

!this.$v.signupForm.email.required && errors.push('E-mail is required')

return errors

},

signupMobileErrors () {

const errors = []

if (!this.$v.signupForm.mobile.$dirty) return errors

!this.$v.signupForm.mobile.numeric && errors.push('Numerical values only')

!this.$v.signupForm.mobile.minLength && errors.push('Mobile number appears too short')

!this.$v.signupForm.mobile.maxLength && errors.push('Mobile number appears too long')

!this.$v.signupForm.mobile.required && errors.push('Mobile number is required')

return errors

},

signupPasswordErrors () {

const errors = []

if (!this.$v.signupForm.password.$dirty) return errors

!this.$v.signupForm.password.minLength && errors.push('Password must be at least 8 characters long')

!this.$v.signupForm.password.maxLength && errors.push('Password should not exceed 16 characters')

!this.$v.signupForm.password.required && errors.push('Password is required')

return errors

},

signupConfirmErrors () {

const errors = []

if (!this.$v.signupForm.confirmPassword.$dirty) return errors

!this.$v.signupForm.confirmPassword.sameAs && errors.push('Passwords do not match')

!this.$v.signupForm.confirmPassword.required && errors.push('Password is required')

return errors

},

dateErrors () {

const errors = []

if (!this.$v.date.$dirty) return errors

!this.$v.date.required && errors.push('Date of Birth is required')

return errors

},

signupPPSNErrors () {

const errors = []

if (!this.$v.signupForm.ppsn.$dirty) return errors

!this.$v.signupForm.ppsn.required && errors.push('PPSN is required')

!this.$v.signupForm.ppsn.ppsnValidate && errors.push('PPSN is invalid')

return errors

},

checkboxErrors () {

const errors = []

if (!this.$v.signupForm.checkbox.$dirty) return errors

!this.$v.signupForm.checkbox.checked && errors.push('You must agree to continue!')

return errors

},

},

data() {

return {

//icons to show password

showPassword: false,

showPassword2: false,

// dialog and menu for forgot password

dialog: false,

dialog2: false,

menu: false,

// date for dob

date: '',

showSuccess: false,

errorMsg: {},

alert1: false,

alert2: false,

alert3: false,

forgotForm: {

email: '',

},

loginForm: {

email: '',

password: ''

},

signupForm: {

firstname: '',

surname: '',

ppsn: '',

email: '',

mobile: '',

password: '',

confirmPassword: '',

checkbox: false,

},

showLoginForm: true,

showPasswordReset: false

}

},

validations: {

loginForm: {

email: { required, email },

password: { required, maxLength: maxLength(16) },

},

signupForm: {

firstname: { required, minLength: minLength(3), maxLength: maxLength (15), alpha},

surname: { required, minLength: minLength(3), maxLength: maxLength (15), alpha},

ppsn: {

required,

// Validation to check for exactly 7 alpha characters followed by 1/2 num

ppsnValidate(ppsn){

return (

/^[0-9]{7}[a-zA-Z]{1,2}$/.test(ppsn)

);

},

},

email: { required, email },

mobile: { required, numeric, minLength: minLength(9), maxLength: maxLength(14) },

password: { required, minLength: minLength(8), maxLength: maxLength(16)},

confirmPassword: { required, sameAs: sameAs(function () { return this.signupForm.password})},

checkbox: {checked (val) {return val}},

},

date: { required },

forgotForm: {

email: { required, email },

},

},

methods: {

// Method to calculate 16 years back from todays date

// in order to limit calendar selection to 16s and over

getLowestPossible () {

let latest = new Date ()

latest.setFullYear(latest.getFullYear() - 16)

//Filter

var d = new Date(latest),

month = '' + (d.getMonth() + 1),

day = '' + d.getDate(),

year = d.getFullYear();

if (month.length < 2) month = '0' + month;

if (day.length < 2) day = '0' + day;

return [year, month, day].join('-');

},

// Get the lateset possible age for the app, currently set to 90

getEarliestPossible () {

let latest = new Date ()

latest.setFullYear(latest.getFullYear() - 90)

//Filter

var d = new Date(latest),

month = '' + (d.getMonth() + 1),

day = '' + d.getDate(),

year = d.getFullYear();

if (month.length < 2) month = '0' + month;

if (day.length < 2) day = '0' + day;

return [year, month, day].join('-');

},

async resetPassword() {

this.errorMsg = ''

this.$v.$touch()

try {

await auth.sendPasswordResetEmail(this.forgotForm.email)

this.showSuccess = true

} catch (error) {

this.errorMsg = error.message

}

},

// Reset values when user toggles between register and login

toggleForm() {

this.showLoginForm = !this.showLoginForm

this.$v.$reset()

this.loginForm.email = ''

this.loginForm.password = ''

this.signupForm.firstname = ''

this.signupForm.surname = ''

this.signupForm.ppsn = ''

this.signupForm.email = ''

this.signupForm.mobile = ''

this.signupForm.password = ''

this.signupForm.confirmPassword = ''

this.signupForm.checkbox = false

this.forgotForm.email = ''

this.alert1 = false

this.alert2 = false

},

login() {

this.$v.$touch()

this.$store.dispatch('login', {

email: this.loginForm.email,

password: this.loginForm.password

})

},

signup() {

this.$v.$touch()

this.formTouched = !this.$v.signupForm.$anyDirty

this.errors = this.$v.signupForm.$anyError

if (this.errors === false && this.formTouched === false){

// send users registered info into a collection

this.$store.dispatch('signup', {

firstname: this.signupForm.firstname,

surname: this.signupForm.surname,

date: this.date,

ppsn: this.signupForm.ppsn,

email: this.signupForm.email,

mobile: this.signupForm.mobile,

password: this.signupForm.password,

})

}

},

},

watch: {

// Register firebase error alert

registerError (value) {

if (value) {

this.alert1 = true

}

},

alert (value) {

if (!value) {

this.$store.commit('setregisterError', null)

}

},

// Login firebase error alert

loginError (value){

if (value){

this.alert2 = true

}

},

alert2 (value){

if (!value){

this.$store.commit('setLoginError', null)

}

},

// Forgot Password firebase error alert

errorMsg (value) {

if (!value) {

this.alert3 = true

}

},

},

}

</script>

<style scoped>

.textCss input {

text-transform: capitalize;

}

</style>

## TsAndCs.vue

<template>

<v-card-text class="mt-3">

<p>Welcome to E-DOCHTÚIR!</p>

<p>These terms and conditions outline the rules and regulations for the use of E-DOCHTÚIR's Website, located at https://e-dochtuir.web.app/.</p>

<p>By accessing this website we assume you accept these terms and conditions. Do not continue to use E-DOCHTÚIR if you do not agree to take all of the terms and conditions stated on this page. Our Terms and Conditions were created with the help of the <a href="https://www.termsandconditionsgenerator.com">Terms And Conditions Generator</a> and the <a href="https://www.privacypolicyonline.com/terms-conditions-generator/">Free Terms & Conditions Generator</a>.</p>

<p>The following terminology applies to these Terms and Conditions, Privacy Statement and Disclaimer Notice and all Agreements: "Client", "You" and "Your" refers to you, the person log on this website and compliant to the Company’s terms and conditions. "The Company", "Ourselves", "We", "Our" and "Us", refers to our Company. "Party", "Parties", or "Us", refers to both the Client and ourselves. All terms refer to the offer, acceptance and consideration of payment necessary to undertake the process of our assistance to the Client in the most appropriate manner for the express purpose of meeting the Client’s needs in respect of provision of the Company’s stated services, in accordance with and subject to, prevailing law of Netherlands. Any use of the above terminology or other words in the singular, plural, capitalization and/or he/she or they, are taken as interchangeable and therefore as referring to same.</p>

<h3><strong>Cookies</strong></h3>

<p>We employ the use of cookies. By accessing E-DOCHTÚIR, you agreed to use cookies in agreement with the E-DOCHTÚIR's Privacy Policy.</p>

<p>Most interactive websites use cookies to let us retrieve the user’s details for each visit. Cookies are used by our website to enable the functionality of certain areas to make it easier for people visiting our website. Some of our affiliate/advertising partners may also use cookies.</p>

<h3><strong>License</strong></h3>

<p>Unless otherwise stated, E-DOCHTÚIR and/or its licensors own the intellectual property rights for all material on E-DOCHTÚIR. All intellectual property rights are reserved. You may access this from E-DOCHTÚIR for your own personal use subjected to restrictions set in these terms and conditions.</p>

<p>You must not:</p>

<ul>

<li>Republish material from E-DOCHTÚIR</li>

<li>Sell, rent or sub-license material from E-DOCHTÚIR</li>

<li>Reproduce, duplicate or copy material from E-DOCHTÚIR</li>

<li>Redistribute content from E-DOCHTÚIR</li>

</ul>

<p>This Agreement shall begin on the date hereof.</p>

<p>Parts of this website offer an opportunity for users to post and exchange opinions and information in certain areas of the website. E-DOCHTÚIR does not filter, edit, publish or review Comments prior to their presence on the website. Comments do not reflect the views and opinions of E-DOCHTÚIR,its agents and/or affiliates. Comments reflect the views and opinions of the person who post their views and opinions. To the extent permitted by applicable laws, E-DOCHTÚIR shall not be liable for the Comments or for any liability, damages or expenses caused and/or suffered as a result of any use of and/or posting of and/or appearance of the Comments on this website.</p>

<p>E-DOCHTÚIR reserves the right to monitor all Comments and to remove any Comments which can be considered inappropriate, offensive or causes breach of these Terms and Conditions.</p>

<p>You warrant and represent that:</p>

<ul>

<li>You are entitled to post the Comments on our website and have all necessary licenses and consents to do so;</li>

<li>The Comments do not invade any intellectual property right, including without limitation copyright, patent or trademark of any third party;</li>

<li>The Comments do not contain any defamatory, libelous, offensive, indecent or otherwise unlawful material which is an invasion of privacy</li>

<li>The Comments will not be used to solicit or promote business or custom or present commercial activities or unlawful activity.</li>

</ul>

<p>You hereby grant E-DOCHTÚIR a non-exclusive license to use, reproduce, edit and authorize others to use, reproduce and edit any of your Comments in any and all forms, formats or media.</p>

<h3><strong>Hyperlinking to our Content</strong></h3>

<p>The following organizations may link to our Website without prior written approval:</p>

<ul>

<li>Government agencies;</li>

<li>Search engines;</li>

<li>News organizations;</li>

<li>Online directory distributors may link to our Website in the same manner as they hyperlink to the Websites of other listed businesses; and</li>

<li>System wide Accredited Businesses except soliciting non-profit organizations, charity shopping malls, and charity fundraising groups which may not hyperlink to our Web site.</li>

</ul>

<p>These organizations may link to our home page, to publications or to other Website information so long as the link: (a) is not in any way deceptive; (b) does not falsely imply sponsorship, endorsement or approval of the linking party and its products and/or services; and (c) fits within the context of the linking party’s site.</p>

<p>We may consider and approve other link requests from the following types of organizations:</p>

<ul>

<li>commonly-known consumer and/or business information sources;</li>

<li>dot.com community sites;</li>

<li>associations or other groups representing charities;</li>

<li>online directory distributors;</li>

<li>internet portals;</li>

<li>accounting, law and consulting firms; and</li>

<li>educational institutions and trade associations.</li>

</ul>

<p>We will approve link requests from these organizations if we decide that: (a) the link would not make us look unfavorably to ourselves or to our accredited businesses; (b) the organization does not have any negative records with us; (c) the benefit to us from the visibility of the hyperlink compensates the absence of E-DOCHTÚIR; and (d) the link is in the context of general resource information.</p>

<p>These organizations may link to our home page so long as the link: (a) is not in any way deceptive; (b) does not falsely imply sponsorship, endorsement or approval of the linking party and its products or services; and (c) fits within the context of the linking party’s site.</p>

<p>If you are one of the organizations listed in paragraph 2 above and are interested in linking to our website, you must inform us by sending an e-mail to E-DOCHTÚIR. Please include your name, your organization name, contact information as well as the URL of your site, a list of any URLs from which you intend to link to our Website, and a list of the URLs on our site to which you would like to link. Wait 2-3 weeks for a response.</p>

<p>Approved organizations may hyperlink to our Website as follows:</p>

<ul>

<li>By use of our corporate name; or</li>

<li>By use of the uniform resource locator being linked to; or</li>

<li>By use of any other description of our Website being linked to that makes sense within the context and format of content on the linking party’s site.</li>

</ul>

<p>No use of E-DOCHTÚIR's logo or other artwork will be allowed for linking absent a trademark license agreement.</p>

<h3><strong>iFrames</strong></h3>

<p>Without prior approval and written permission, you may not create frames around our Webpages that alter in any way the visual presentation or appearance of our Website.</p>

<h3><strong>Content Liability</strong></h3>

<p>We shall not be hold responsible for any content that appears on your Website. You agree to protect and defend us against all claims that is rising on your Website. No link(s) should appear on any Website that may be interpreted as libelous, obscene or criminal, or which infringes, otherwise violates, or advocates the infringement or other violation of, any third party rights.</p>

<h3><strong>Your Privacy</strong></h3>

<p>Please read Privacy Policy</p>

<h3><strong>Reservation of Rights</strong></h3>

<p>We reserve the right to request that you remove all links or any particular link to our Website. You approve to immediately remove all links to our Website upon request. We also reserve the right to amen these terms and conditions and it’s linking policy at any time. By continuously linking to our Website, you agree to be bound to and follow these linking terms and conditions.</p>

<h3><strong>Removal of links from our website</strong></h3>

<p>If you find any link on our Website that is offensive for any reason, you are free to contact and inform us any moment. We will consider requests to remove links but we are not obligated to or so or to respond to you directly.</p>

<p>We do not ensure that the information on this website is correct, we do not warrant its completeness or accuracy; nor do we promise to ensure that the website remains available or that the material on the website is kept up to date.</p>

<h3><strong>Disclaimer</strong></h3>

<p>To the maximum extent permitted by applicable law, we exclude all representations, warranties and conditions relating to our website and the use of this website. Nothing in this disclaimer will:</p>

<ul>

<li>limit or exclude our or your liability for death or personal injury;</li>

<li>limit or exclude our or your liability for fraud or fraudulent misrepresentation;</li>

<li>limit any of our or your liabilities in any way that is not permitted under applicable law; or</li>

<li>exclude any of our or your liabilities that may not be excluded under applicable law.</li>

</ul>

<p>The limitations and prohibitions of liability set in this Section and elsewhere in this disclaimer: (a) are subject to the preceding paragraph; and (b) govern all liabilities arising under the disclaimer, including liabilities arising in contract, in tort and for breach of statutory duty.</p>

<p>As long as the website and the information and services on the website are provided free of charge, we will not be liable for any loss or damage of any nature.</p>

</v-card-text>

</template>

# Firebase

## Firebase.js

import firebase from 'firebase/app'

import 'firebase/auth'

import 'firebase/firestore'

import 'firebase/functions'

import 'firebase/storage'

import 'firebase/messaging'

import 'firebase/performance'

const firebaseConfig = {

apiKey: "AIzaSyAuaVEBK2WfHRjQj60JaDIuaMlDut7XRbk",

authDomain: "e-dochtuir.firebaseapp.com",

databaseURL: "https://e-dochtuir.firebaseio.com",

projectId: "e-dochtuir",

storageBucket: "e-dochtuir.appspot.com",

messagingSenderId: "73425071156",

appId: "1:73425071156:web:55d7625dd6599fad7e6e61",

measurementId: "G-NPJX6TG19X"

}

firebase.initializeApp(firebaseConfig)

// utils

const db = firebase.firestore()

const auth = firebase.auth()

const functions = firebase.functions()

const fieldValue = firebase.firestore.FieldValue

const storage = firebase.storage()

const messaging = firebase.messaging()

const performance = firebase.performance()

// collections

const usersCollection = db.collection('users')

const rolesCollection = db.collection('roles')

export{

db,

auth,

functions,

fieldValue,

storage,

messaging,

performance,

usersCollection,

rolesCollection,

}

## Firestore.rules

service cloud.firestore {

match /databases/{database}/documents {

// match logged in users doc with users collection

match /users/{userId}{

allow create, read, write: if request.auth.uid !=null;

}

match /roles/{userId}{

allow create: if request.auth.uid !=null;

allow read: if request.auth.uid !=null;

allow write: if request.auth.token.admin == true;

}

match /appointments/{appointmentsId}{

allow read, write: if request.auth.uid !=null;

}

match /prescriptions/{prescriptionsId}{

allow read, write: if request.auth.uid !=null;

}

match /tickets/{ticketsId}{

allow write: if request.auth.uid !=null;

}

match /rooms/{roomsId}{

allow create: if user.auth.token.doctor == true;

allow read, write: if request.auth.uid !=null;

}

}

}

## Firestore.indexes.json

{

"indexes": [

{

"collectionGroup": "prescriptions",

"queryScope": "COLLECTION",

"fields": [

{

"fieldPath": "patientID",

"order": "ASCENDING"

},

{

"fieldPath": "dateRequested",

"order": "ASCENDING"

}

]

},

{

"collectionGroup": "prescriptions",

"queryScope": "COLLECTION",

"fields": [

{

"fieldPath": "doctor",

"order": "ASCENDING"

},

{

"fieldPath": "dateRequested",

"order": "ASCENDING"

}

]

},

{

"collectionGroup": "prescriptions",

"queryScope": "COLLECTION",

"fields": [

{

"fieldPath": "doctor",

"order": "ASCENDING"

},

{

"fieldPath": "dateRequested",

"order": "ASCENDING"

},

{

"fieldPath": "patientName",

"order": "ASCENDING"

}

]

},

{

"collectionGroup": "appointments",

"queryScope": "COLLECTION",

"fields": [

{

"fieldPath": "patientID",

"order": "ASCENDING"

},

{

"fieldPath": "appointmentDate",

"order": "ASCENDING"

}

]

},

{

"collectionGroup": "appointments",

"queryScope": "COLLECTION",

"fields": [

{

"fieldPath": "patientID",

"order": "ASCENDING"

},

{

"fieldPath": "appointmentDate",

"order": "ASCENDING"

},

{

"fieldPath": "appointmentTime",

"order": "ASCENDING"

}

]

}

],

"fieldOverrides": []

}

## Storage.rules

rules\_version = '2';

service firebase.storage {

match /b/{bucket}/o {

match /{allPaths=\*\*} {

allow read, write: if request.auth!=null;

}

}

}

## Firebase Functions - Index.js

const functions = require('firebase-functions');

const admin = require('firebase-admin') // Require the admin package from firebase

admin.initializeApp() //Initialise admin server side

const db = admin.firestore()

exports.AddUserRole = functions.auth.user().onCreate(async (user) => {

if (user.email) {

const customClaims = {

patient: true,

};

// const customClaims = {

// role: 'patient',

// };

try {

var \_ = await admin.auth().setCustomUserClaims(user.uid, customClaims)

return db.collection("roles").doc(user.uid).set({

email: user.email,

role: customClaims

})

} catch (error) {

console.log(error)

}

}

})

exports.setUserRole = functions.https.onCall(async (data, context) => {

if (!context.auth.token.admin) return

try {

var \_ = await admin.auth().setCustomUserClaims(data.uid, data.role)

return db.collection("roles").doc(data.uid).update({

role: data.role

})

} catch (error) {

console.log(error)

}

});

# Store

## Index.js

import Vue from 'vue'

import Vuex from 'vuex'

import \* as fb from '../firebase'

import router from '../router/index'

Vue.use(Vuex)

export default new Vuex.Store({

state: {

userProfile: {},

registerError: {},

loginError: {},

loading: false,

},

mutations: {

setUserProfile(state, payload) {

state.userProfile = payload

},

setRegisterError(state, payload){

state.registerError = payload

},

setLoginError(state,payload){

state.loginError = payload

},

setLoading(state, payload){

state.loading = payload

},

},

actions: {

// SIGNUP

async signup({ commit }, payload) {

commit('setLoading', true)

await fb.auth.createUserWithEmailAndPassword(payload.email, payload.password).then(user => {

commit('setLoading', false)

// fetch user profile and set in state

commit('fetchUserProfile', user)

// Store form data in a patients collection

return fb.usersCollection.doc(user.user.uid).set({

firstname: payload.firstname,

surname: payload.surname,

date: payload.date,

ppsn: payload.ppsn,

email: payload.email,

mobile: payload.mobile,

});

}).catch(registerError => {

console.log(registerError.message)

commit('setRegisterError', registerError.message)

commit('setLoading', false)

})

},

// LOGIN

async login({ commit }, payload) {

commit('setLoading', true)

// sign user in

await fb.auth.signInWithEmailAndPassword(payload.email, payload.password).then( user => {

commit('setLoading', false)

console.log(user)

}).catch(loginError => {

console.log(loginError.message)

commit('setLoginError', loginError.message)

commit('setLoading', false)

})

},

// LOGOUT

async logout({ commit }){

await fb.auth.signOut()

// clear userProfile and redirect to /login

commit('setUserProfile', {})

router.push('/login')

},

// FETCH USER PROFILE

async fetchUserProfile({ commit }, user){

// fetch user profile

const userProfile = await fb.usersCollection.doc(user.uid).get()

// set user profile in state

commit('setUserProfile', userProfile.data())

// change route to dashboard

if (router.currentRoute.path === '/login') {

router.push('/')

}

},

async updateProfile({ commit, dispatch }, user) {

const userId = fb.auth.currentUser.uid

commit('setLoading', true)

// update user object

await fb.usersCollection.doc(userId).update({

firstname: user.firstname,

surname: user.surname,

date: user.date,

ppsn: user.ppsn,

mobile: user.mobile,

}).catch(error => {

console.log(error)

commit('setLoading', false)

})

commit('setLoading', false)

dispatch('fetchUserProfile', { uid: userId })

}

}, // actions end

getters: {

userProfile (state) {

return state.userProfile

},

},

modules: {

}

})

# Navbars

## Navbar.vue

<template>

<nav>

<!-- Toolbar -->

<v-app-bar app class="primary">

<v-app-bar-nav-icon large class="grey--text" @click="drawer = !drawer"></v-app-bar-nav-icon>

<v-toolbar-title class="grey--text text-uppercase">E-Dochtúir</v-toolbar-title>

<v-spacer></v-spacer>

<v-btn depressed class="primary grey--text" @click="logout()">

<span>Log Out</span>

<v-icon right> exit\_to\_app </v-icon>

</v-btn>

</v-app-bar>

<!-- Navigation Drawer -->

<v-navigation-drawer app v-model="drawer" temporary class="teal darken-3">

<v-list>

<v-list-item route-link :to="{ name: 'profile'}">

<v-list-item-icon>

<v-icon left class="white--text">fa-user-circle</v-icon>

</v-list-item-icon>

<v-list-item-content>

<v-list-item-title class="white--text">My Profile</v-list-item-title>

</v-list-item-content>

</v-list-item>

<v-divider></v-divider>

<v-list-item v-for="link in links" :key="link.text" router :to="link.route">

<v-list-item-icon>

<v-icon left class="white--text">{{ link.icon }}</v-icon>

</v-list-item-icon>

<v-list-item-content>

<v-list-item-title class="white--text">{{ link.text}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-list>

</v-navigation-drawer>

</nav>

</template>

<script>

export default {

data(){

return{

drawer: false,

links: [

{ icon: 'dashboard', text: 'Dashboard', route: '/' },

{ icon: 'fa-book-medical', text: 'Medical Record', route: '/medicalRecord'},

{ icon: 'fa-pills', text: 'Track Medication', route: '/medication' },

{ icon: 'fa-stethoscope', text: 'Appointments', route: '/appointments' },

{ icon: 'fa-prescription', text: 'Prescriptions', route: '/prescriptions' },

{ icon: 'fa-user-md', text: 'Chat', route: '/chat' },

]

}

},

methods: {

logout() {

this.$store.dispatch('logout')

}

}

}

</script>

## DoctorNavbar.vue

<template>

<nav>

<!-- Toolbar -->

<v-app-bar app class="primary">

<v-app-bar-nav-icon large class="grey--text" @click="drawer = !drawer"></v-app-bar-nav-icon>

<v-toolbar-title class="grey--text text-uppercase">E-Dochtúir - Doctor Portal</v-toolbar-title>

<v-spacer></v-spacer>

<v-btn depressed class="primary grey--text" @click="logout()">

<span>Log Out</span>

<v-icon right> exit\_to\_app </v-icon>

</v-btn>

</v-app-bar>

<!-- Navigation Drawer -->

<v-navigation-drawer app v-model="drawer" temporary class="teal darken-3">

<v-list>

<v-list-item v-for="link in links" :key="link.text" router :to="link.route">

<v-list-item-icon>

<v-icon left class="white--text">{{ link.icon }}</v-icon>

</v-list-item-icon>

<v-list-item-content>

<v-list-item-title class="white--text">{{ link.text}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-list>

</v-navigation-drawer>

</nav>

</template>

<script>

export default {

data(){

return{

drawer: false,

links: [

{ icon: 'fa-user-injured', text: 'Patient Database', route: '/doctor'},

{ icon: 'fa-calendar', text: 'Appointments', route: '/doctorAppointment' },

{ icon: 'fa-file-prescription', text: 'Prescription Requests', route: '/doctorPrescription' },

{ icon: 'fa-comment-medical', text: 'Patient Chat', route: '/doctorRooms' },

]

}

},

methods: {

logout() {

this.$store.dispatch('logout')

}

}

}

</script>

## AdminNavbar.vue

<template>

<nav>

<!-- Toolbar -->

<v-app-bar app class="primary">

<v-toolbar-title class="grey--text text-uppercase">E-Dochtúir</v-toolbar-title>

<v-spacer></v-spacer> <!-- Anything after the spacer will align to the right of the page -->

<v-btn depressed class="primary grey--text" @click="logout()">

<span>Log Out</span>

<v-icon right> exit\_to\_app </v-icon>

</v-btn>

</v-app-bar>

</nav>

</template>

<script>

export default {

data(){

return{

}

},

methods: {

logout() {

this.$store.dispatch('logout')

}

}

}

</script>

# Central App

## App.vue

<template>

<v-app> <!-- Route Vuetify Component use only once to wrap everything else -->

<v-main class="mx-2 mb-4">

<router-view></router-view> <!-- This will load the specific route the user is on e.g. Home, About etc. -->

</v-main>

</v-app>

</template>

<script>

export default {

name: 'App',

};

</script>

## Main.js

import Vue from 'vue'

import App from './App.vue'

import Vuelidate from 'vuelidate'

import './registerServiceWorker'

import router from './router'

import store from './store'

import { auth } from './firebase'

import vuetify from './plugins/vuetify';

import VueTextareaAutosize from 'vue-textarea-autosize'

Vue.config.productionTip = false

Vue.use(Vuelidate, VueTextareaAutosize)

let app

// wait for auth to init before creating the app

auth.onAuthStateChanged(user => {

// init app if not already created

if(!app){

app = new Vue({

router,

store,

vuetify,

render: h => h(App)

}).$mount('#app')

}

if (user) {

// Get the token of the user to see if they are admin or not

auth.currentUser.getIdTokenResult().then( tokenResult=> {

console.log(tokenResult.claims)

})

// User is signed in

store.dispatch('fetchUserProfile', user)

}

})

# Admin

## Admin.vue

<template>

<v-container >

<AdminNavbar />

<!-- Display the admin users email -->

<v-row>

<v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-col class="mt-n4">

<v-card outlined>

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">Admin</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="8">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1" v-if="user">{{user.email}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

</v-card>

</v-col>

<!-- Card for User Records Search Bar -->

<v-card outlined>

<v-card-title class="text-uppercase grey--text text--darken-1">

User Management

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="fa-search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

</v-card>

<!-- Card for the Data Table -->

<v-card outlined>

<!-- Table -->

<v-data-table

:headers="headers"

:items="users"

:search="search"

:items-per-page="10"

class="elevation-1"

>

<!-- selector for the user role -->

<template v-slot:[`item.role`]="{item}">

<v-select

:key="item.id"

v-model="item.currentRole"

:items="items"

item-text="roleName"

item-value="value"

@change="changeRole(item.id, $event)"

></v-select>

</template>

</v-data-table>

</v-card>

</v-col>

</v-row>

</v-container>

</template>

<script> // import the admin navbar as well as auth, db and functions from firebase

import AdminNavbar from '../components/Navbars/AdminNavbar'

import {auth, db, functions} from '../firebase'

export default {

components: {

AdminNavbar,

},

data() {

return {

users: [],

user: null,

search: '',

headers: [

{ text: 'User Email', align: 'start', value: 'email' },

{ text: 'User Role ', value: 'role', width: '300' },

],

select: {roleName: 'Patient'},

items : [

{ roleName: 'Patient', role:'patient', value: 'patient'},

{ roleName: 'Doctor', role: 'doctor', value: 'doctor'}

],

};

},

created() {

var self = this;

auth.onAuthStateChanged(function(user) {

self.user = user;

});

this.users = [];

db.collection("roles").get().then(snap => {

snap.forEach(doc => {

var user = doc.data()

user.id = doc.id

user.currentRole = user.role.patient ? "patient" : "doctor"

if (!user.role.admin) this.users.push(user)

});

console.log(this.users)

});

},

methods: {

// Update the role of the current user, used to update them from a patient to doctor

changeRole(uid, event) {

// Fires of a call to the cloud functions to setUserRole

var addMessage = functions.httpsCallable("setUserRole");

var data = { uid: uid, role: { [event]: true } };

addMessage(data)

.then(function(result) {

console.log(result);

})

.catch(function(error) {

console.log(error);

});

},

}

};

</script>

# Navigation Router

## Router index.js

import Vue from 'vue'

import VueRouter from 'vue-router'

import 'material-design-icons-iconfont/dist/material-design-icons.css'

import {auth} from '../firebase'

import Admin from '@/views/Admin.vue'

import Doctor from '@/views/Doctor.vue'

import DoctorAppointment from '@/views/DoctorAppointment.vue'

import DoctorPrescription from '@/views/DoctorPrescription.vue'

import Dashboard from '@/views/Dashboard.vue'

import Profile from '@/views/Profile.vue'

import MedicalRecord from '@/views/MedicalRecord.vue'

import Medication from '@/views/Medication.vue'

import Appointments from '@/views/Appointments.vue'

import Prescriptions from '@/views/Prescriptions.vue'

import Chat from '@/views/Chat.vue'

import DoctorRooms from '@/views/DoctorRooms.vue'

import Login from '@/views/Login.vue'

import NotFound from '../components/404.vue'

Vue.use(VueRouter)

// Each of the individual page routes

const routes = [

{

path: '/login',

name: 'Login',

component: Login,

},

{

path: '/admin',

name: 'admin',

component: Admin,

meta: {

requiresAuth: true,

}

},

{

path: '/doctor',

name: 'doctor',

component: Doctor,

meta: {

requiresAuth: true

}

},

{

path: '/doctorAppointment',

name: 'doctorAppointment',

component: DoctorAppointment,

meta: {

requiresAuth: true

}

},

{

path: '/doctorPrescription',

name: 'doctorPrescription',

component: DoctorPrescription,

meta: {

requiresAuth: true

}

},

{

path: '/',

name: 'dashboard',

component: Dashboard,

meta: {

requiresAuth: true

}

},

{

path: '/profile',

name: 'profile',

component: Profile,

meta: {

requiresAuth: true

}

},

{

path: '/medicalRecord',

name: 'medicalRecord',

component: MedicalRecord,

meta: {

requiresAuth: true

}

},

{

path: '/medication',

name: 'Medication',

component: Medication,

meta: {

requiresAuth: true

}

},

{

path: '/appointments',

name: 'Appointments',

component: Appointments,

meta: {

requiresAuth: true

}

},

{

path: '/prescriptions',

name: 'Prescriptions',

component: Prescriptions,

meta: {

requiresAuth: true

}

},

{

path: '/chat',

name: 'Chat',

component: Chat,

meta: {

requiresAuth: true

}

},

{

path: '/doctorRooms',

name: 'DoctorRooms',

component: DoctorRooms,

meta: {

requiresAuth: true

}

},

{

path: '\*',

name: 'NotFound',

component: NotFound

}

]

const router = new VueRouter({

mode: 'history',

base: process.env.BASE\_URL,

routes

})

router.beforeEach((to, from, next) => {

// Check for requiredAuth and current user

const requiresAuth = to.matched.some(x => x.meta.requiresAuth)

if (requiresAuth && !auth.currentUser) {

next('/login')

} else {

next()

}

// if auth state is authenticated user then do checks for different roles otherwise redirect to login

auth.onAuthStateChanged(user => {

if (user != null && user != undefined ) {

auth.currentUser.getIdTokenResult()

.then(function ({claims}) {

if (claims.patient) {

if (to.path != '/profile' && to.path != '/appointments' && to.path != '/prescriptions' && to.path != '/chat' && to.path != '/medication' && to.path != '/medicalRecord')

return next ({

path: './'

})

} else if (claims.admin) {

if (to.path != '/admin')

return next ({

path: './admin'

})

} else if (claims.doctor) {

if (to.path != '/doctor' && to.path != '/doctorAppointment' && to.path != '/doctorPrescription' && to.path != '/doctorRooms')

return next ({

path: './doctor'

})

}

})

}

else {

if (to.matched.some(record => record.meta.auth)) {

next({

path: '/login',

query: {

redirect: to.fullPath

}

})

}

}

})

})

export default router

# Profile

## Profile.vue

<template>

<v-container >

<Navbar />

<v-row>

<!-- Left Card - Profile -->

<v-col cols="12" s="12" sm="6" md="6" lg="6">

<v-card width="800" >

<div v-if="showEditForm">

<v-card flat> <!-- Full Name -->

<v-card-title class="text-uppercase primary lighten-1 white--text">

Profile

<v-spacer></v-spacer>

<v-icon right class="white--text" @click="toggleEditProfile()">fa-edit</v-icon>

</v-card-title>

<v-divider></v-divider>

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">Full Name</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1">{{userProfile.firstname}} {{userProfile.surname}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

<v-divider></v-divider>

</v-card>

<v-card flat> <!-- Date of Birth -->

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">Date of Birth</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1">{{formattedDate}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

<v-divider></v-divider>

</v-card>

<v-card flat> <!-- PPSN -->

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">PPSN</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1">{{userProfile.ppsn}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

<v-divider></v-divider>

</v-card>

<v-card flat> <!-- EMAIL -->

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">Email</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="6">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1">{{userProfile.email}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

<v-divider></v-divider>

</v-card>

<v-card flat> <!-- MOBILE -->

<v-row>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text">Mobile</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

<v-col cols="12" md="4">

<v-list-item>

<v-list-item-content>

<v-list-item-title class="subtitle-1">{{userProfile.mobile}}</v-list-item-title>

</v-list-item-content>

</v-list-item>

</v-col>

</v-row>

</v-card>

</div>

<!-- EDIT FORM -->

<v-form v-else @submit.prevent="updateProfile">

<v-card flat>

<v-card-title class="text-uppercase primary lighten-1 white--text">Edit Profile

<v-spacer></v-spacer>

<v-btn depressed color="primary lighten-1" @click="toggleEditProfile()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-divider></v-divider>

<v-card-text>

<v-row>

<v-col class="mt-n2" cols="12" md="6" lg="6">

<!-- Firstname Edit -->

<v-text-field

type="text"

name="firstname"

label="First Name"

:counter="15"

v-model.trim="firstname"

:error-messages="firstNameErrors"

:placeholder="userProfile.firstname"

outlined

@input="$v.firstname.$touch()"

@blur="$v.firstname.$touch()"

></v-text-field>

</v-col>

<!-- Surname Edit -->

<v-col class="mt-n2" cols="12" md="6" lg="6">

<v-text-field

type="text"

name="surname"

label="Surname"

:counter="15"

v-model.trim="surname"

:error-messages="surnameErrors"

:placeholder="userProfile.surname"

outlined

@input="$v.surname.$touch()"

@blur="$v.surname.$touch()"

></v-text-field>

</v-col>

<!-- DOB Edit -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-menu

v-model="menu"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Date of Birth"

:value="formattedDate"

:placeholder="userProfile.date"

clearable

readonly

v-bind="attrs"

v-on="on"

@click:clear="date = null"

outlined

>

</v-text-field>

</template>

<v-date-picker

v-model.trim="date"

:min="getLowestPossible()"

:max="getEarliestPossible()"

@change="menu = false"

></v-date-picker>

</v-menu>

</v-col>

<!-- PPSN Edit -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="ppsn"

label="PPSN"

hint="7 numerical characters, followed by either 1 or 2 letters E.g. 1234567RW"

v-model.trim="ppsn"

:error-messages="ppsnErrors"

:placeholder="userProfile.ppsn"

outlined

@input="$v.ppsn.$touch()"

@blur="$v.ppsn.$touch()"

></v-text-field>

</v-col>

<!-- Mobile Edit -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="mobile"

label="Mobile"

v-model.trim="mobile"

:error-messages="mobileErrors"

:placeholder="userProfile.mobile"

outlined

@input="$v.mobile.$touch()"

@blur="$v.mobile.$touch()"

></v-text-field>

</v-col>

</v-row>

<v-card-actions>

<v-row>

<v-btn

type="button"

:disabled="loading"

block class="primary white--text"

@click.prevent="updateProfile()">

<span>Update</span>

</v-btn>

</v-row>

</v-card-actions>

</v-card-text>

</v-card>

</v-form>

</v-card>

<!-- Dark Mode Card -->

<v-card width="585" height="140" class="mt-5">

<v-card-title class="primary lighten-1 white--text">Application Theme</v-card-title>

<v-card-text class="ml-15">

<v-row>

<v-col class="mt-n4">

<v-switch

v-model="$vuetify.theme.dark"

hide-details

inset

label="Dark Mode"

></v-switch>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" s="12" sm="6" md="6" lg="6">

<v-card width="600">

<v-card-text>

<CalendarAppointments /> <!-- Import Calendar -->

</v-card-text>

</v-card>

</v-col>

</v-row>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import CalendarAppointments from '../components/PatientAppCalendar'

import { mapState } from 'vuex'

import { maxLength, minLength, alpha, numeric } from "vuelidate/lib/validators"

import { auth, db } from '../firebase'

import format from 'date-fns/format'

import parseISO from 'date-fns/parseISO'

export default {

components: {

Navbar,

CalendarAppointments

},

computed: {

...mapState(['userProfile']),

user () {

return this.$store.getters.user

},

loading () {

return this.$store.state.loading

},

formattedDate () {

return this.date ? format(parseISO(this.date), 'do MMM yyyy') : ''

},

// VALIDATION ERROR MESSAGES

firstNameErrors () {

const errors = []

if(!this.$v.firstname.$dirty) return errors

!this.$v.firstname.alpha && errors.push('Alphabetical characters only')

!this.$v.firstname.minLength && errors.push('First Name must be at least 3 characters long')

!this.$v.firstname.maxLength && errors.push('First Name should not exceed 15 characters')

return errors

},

surnameErrors() {

const errors = []

if(!this.$v.surname.$dirty) return errors

!this.$v.surname.alpha && errors.push('Alphabetical characters only')

!this.$v.surname.minLength && errors.push('Surname must be at least 3 characters long')

!this.$v.surname.maxLength && errors.push('Surname should not exceed 15 characters')

return errors

},

ppsnErrors () {

const errors = []

if (!this.$v.ppsn.$dirty) return errors

!this.$v.ppsn.ppsnValidate && errors.push('PPSN is invalid')

return errors

},

mobileErrors () {

const errors = []

if (!this.$v.mobile.$dirty) return errors

!this.$v.mobile.numeric && errors.push('Numerical values only')

!this.$v.mobile.minLength && errors.push('Mobile number appears too short')

!this.$v.mobile.maxLength && errors.push('Mobile number appears too long')

return errors

},

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

},

data() {

//

return{

firstname: this.$store.state.userProfile.firstname,

surname: this.$store.state.userProfile.surname,

date: this.$store.state.userProfile.date,

ppsn: this.$store.state.userProfile.ppsn,

mobile: this.$store.state.userProfile.mobile,

menu: false,

showEditForm: true,

dob: null,

}

},

validations: {

firstname: {minLength: minLength(3), maxLength: maxLength(15), alpha},

surname: {minLength: minLength(3), maxLength: maxLength(15), alpha},

ppsn: {

ppsnValidate(ppsn){

return (

/^[0-9]{7}[a-zA-Z]{1,2}$/.test(ppsn)

);

},

},

mobile: {numeric, minLength: minLength(9), maxLength: maxLength(14)},

},

methods: {

getEarliestPossible () {

db.collection("users").doc(this.currentUser).onSnapshot(doc => {

let patientRecord = doc.data()

patientRecord.id = doc.id

// Get DoB

this.dob = patientRecord.date

})

let latest = new Date ()

latest.setFullYear(latest.getFullYear() - 16)

//Filter

var d = new Date(latest),

month = '' + (d.getMonth() + 1),

day = '' + d.getDate(),

year = d.getFullYear();

if (month.length < 2) month = '0' + month;

if (day.length < 2) day = '0' + day;

return [year, month, day].join('-');

},

getLowestPossible () {

let latest = new Date ()

latest.setFullYear(latest.getFullYear() - 90)

//Filter

var d = new Date(latest),

month = '' + (d.getMonth() + 1),

day = '' + d.getDate(),

year = d.getFullYear();

if (month.length < 2) month = '0' + month;

if (day.length < 2) day = '0' + day;

return [year, month, day].join('-');

},

toggleEditProfile(){

this.showEditForm = !this.showEditForm

this.$v.$reset()

this.firstname = this.$store.state.userProfile.firstname

this.surname = this.$store.state.userProfile.surname

this.date = this.$store.state.userProfile.date

this.ppsn = this.$store.state.userProfile.ppsn

this.mobile = this.$store.state.userProfile.mobile

},

updateProfile(){

this.$v.$touch()

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

if (this.errors === false && this.formTouched === false){

// Update the users info in their users collection

this.$store.dispatch('updateProfile', {

firstname: this.firstname !== '' ? this.firstname : this.userProfile.firstname,

surname: this.surname !== '' ? this.surname : this.userProfile.surname,

date: this.date !== '' ? this.date : this.userProfile.date,

ppsn: this.ppsn !== '' ? this.ppsn : this.userProfile.ppsn,

mobile: this.mobile !== '' ? this.mobile : this.userProfile.mobile,

}).then ( () =>{

this.showEditForm = !this.showEditForm

})

}

},

}

};

</script>

## PatientAppCalendar.vue

<template>

<v-row class="fill-height">

<v-col>

<!-- Contains the top toolbar with the today, prev, next btn options and a menu that switches the calendar view-->

<v-sheet height="64">

<v-toolbar flat>

<v-btn outlined color="primary" @click="setToday">Today</v-btn>

<v-btn fab text small color="primary" @click="prev">

<v-icon small >mdi-chevron-left</v-icon>

</v-btn>

<v-btn fab text small color="primary" @click="next">

<v-icon small>mdi-chevron-right</v-icon>

</v-btn>

<v-toolbar-title v-if="$refs.calendar">{{ $refs.calendar.title }}</v-toolbar-title>

<v-spacer></v-spacer>

<v-menu bottom right>

<template v-slot:activator="{ on, attrs }">

<v-btn

outlined

color="primary"

v-bind="attrs"

v-on="on"

>

<span>{{ typeToLabel[type] }}</span>

<v-icon right>mdi-menu-down</v-icon>

</v-btn>

</template>

<v-list>

<v-list-item @click="type = 'day'">

<v-list-item-title>Day</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'week'">

<v-list-item-title>Week</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'month'">

<v-list-item-title>Month</v-list-item-title>

</v-list-item>

</v-list>

</v-menu>

</v-toolbar>

</v-sheet>

<!-- Calendar sheet -->

<v-sheet height="500">

<v-calendar

ref="calendar"

color="primary"

:type="type"

v-model="focus"

:events="events"

@click:date="viewDay"

@click:event="showEvent"

>

</v-calendar>

<v-menu

v-model="selectedOpen"

:close-on-content-click="false"

:activator="selectedElement"

offset-x

>

<v-card

color="grey lighten-4"

min-width="350px"

flat

>

<v-toolbar class="primary"

:color="selectedEvent.color"

dark

>

<v-toolbar-title class="primary white--text" v-html="selectedEvent.name"></v-toolbar-title>

<v-spacer></v-spacer>

<v-btn

text

color="white"

@click="selectedOpen = false"

>

Close

</v-btn>

</v-toolbar>

<v-card-text>

<v-row class="mt-n8">

<v-col cols="12" md="12" class="mt-4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clinic-medical</v-icon>

Appointment Type</v-list-item-title>

<h3 v-html="selectedEvent.type"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" md="12">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Details</v-list-item-title>

<h3 v-html="selectedEvent.details"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-menu>

</v-sheet>

</v-col>

</v-row>

</template>

<script>

import { auth, db } from '../firebase'

export default {

data() {

return {

focus: '',

type: 'month',

typeToLabel: {

month: 'Month',

week: 'Week',

day: 'Day',

},

selectedEvent: {},

selectedElement: null,

selectedOpen: false,

events: [],

doctorName: null,

}

},

mounted() {

this.getAppointments()

},

methods: {

getAppointments () {

// Set current user to the currently logged in user

var currentUser = auth.currentUser.uid

db.collection("appointments").where("patientID", "==", currentUser).onSnapshot( snap => {

let appointment = snap.docChanges()

appointment.forEach(appointment => {

let record = appointment.doc.data()

let time = record.appointmentTime.replace(".", ":")

let endTime = time.substr(0, time.length -2)

let endTimeMinutes = time.substr(time.length -2, 2)

endTime = endTime + (parseInt(endTimeMinutes) + 25)

if(this.doctorName == null){

db.collection("users").doc(record.doctorID).get().then(doc => {

this.doctorName = 'Dr. ' + doc.data().firstname + ' ' + doc.data().surname

}).then(() => {

let event = {

name: this.doctorName,

details : record.appointmentDetails,

type: record.appointmentType + ' Appointment',

start: record.appointmentDate + " " + time,

end: record.appointmentDate + " " + endTime

}

this.events.push(event)

})

}

})

})

},

viewDay ({ date }) {

this.focus = date

this.type = 'day'

},

setToday () {

this.focus = ''

},

prev () {

this.$refs.calendar.prev()

},

next () {

this.$refs.calendar.next()

},

showEvent ({ nativeEvent, event }) {

const open = () => {

this.selectedEvent = event

this.selectedElement = nativeEvent.target

setTimeout(() => this.selectedOpen = true, 10)

}

if (this.selectedOpen) {

this.selectedOpen = false

setTimeout(open, 10)

} else {

open()

}

nativeEvent.stopPropagation()

},

},

}

</script>

# Dashboard

## Dashboard.vue

<template>

<v-container>

<Navbar />

<v-container class="text-center">

<v-card flat>

<v-card-title class="text-uppercase primary lighten-1 white--text"><h1>Features</h1></v-card-title>

</v-card>

<v-row>

<v-col v-for="({ icon, title, text }, i) in features" :key="i" cols="12" md="6">

<v-card

class="py-12 px-4"

flat

>

<v-theme-provider dark>

<div>

<v-avatar

color="primary"

size="88"

>

<v-icon

large

v-text="icon"

></v-icon>

</v-avatar>

</div>

</v-theme-provider>

<v-card-title

class="justify-center text-center text-uppercase"

v-text="title"

></v-card-title>

<v-card-text

class="subtitle-1 text-justify text-center"

v-text="text"

>

</v-card-text>

</v-card>

</v-col>

</v-row>

</v-container>

<v-row><v-col cols="12">

<v-card flat>

<v-card-title class="text-uppercase primary lighten-1 white--text">Contact Us

<v-spacer></v-spacer>

</v-card-title>

<v-form @submit.prevent="submitQuery">

<v-row class="mt-4">

<v-col cols="12" s="12" sm="6" md="6" lg="6">

<v-text-field

label="Name"

v-model="this.name"

outlined

readonly

></v-text-field>

</v-col>

<v-col cols="12" s="12" sm="6" md="6" lg="6">

<v-text-field

label="Email"

v-model="this.email"

outlined

readonly

></v-text-field>

</v-col>

<v-col cols="12">

<v-textarea

label="Message"

v-model="message"

outlined

auto-grow

:counter="150"

:minLength="50"

:maxlength="150"

:error-messages="messageErrors"

@input="$v.message.$touch()"

@blur="$v.message.$touch()"

></v-textarea>

</v-col>

</v-row>

</v-form>

<v-card-actions>

<v-row align="center" justify="center">

<v-col cols="6">

<v-btn

type="submit"

block

class="primary white--text"

@click.prevent="submitQuery()"

>

<span>Submit</span>

</v-btn>

</v-col>

</v-row>

</v-card-actions>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-card>

</v-col></v-row>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import { auth, db } from '../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

components: {

Navbar,

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

db.collection("users").doc(this.currentUser).onSnapshot(doc => {

let userRecord = doc.data()

userRecord.id = doc.id

this.name = userRecord.firstname + ' ' + userRecord.surname

this.email = userRecord.email

})

},

computed: {

messageErrors () {

const errors = []

if (!this.$v.message.$dirty) return errors

!this.$v.message.required && errors.push('A Message Is Required')

return errors

},

},

data() {

return {

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

currentUser: null,

email: null,

name: null,

message: null,

features: [

{

icon: 'fa-book-medical',

title: 'Medical Record',

text: 'Store your personal medical records digitally and receive medical updates in real-time.'

},

{

icon: 'fa-pills',

title: 'Track Medication',

text: 'Enter any medication taken and visually track your medication timetable through the app.'

},

{

icon: 'fa-stethoscope',

title: 'Book Appointments',

text: 'Book appointments with your doctors online with E-Dochtúirs intuitive booking system. Select online consultations to chat online.'

},

{

icon: 'fa-prescription-bottle-alt',

title: 'Request Prescriptions',

text: 'Request prescriptions online rather than having to make a trip to the clinic to pick up a physical copy of your prescriptions.'

},

],

}

},

validations: {

message: { required }

},

methods: {

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

submitQuery () {

this.$v.$touch()

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

if (this.errors === false && this.formTouched === false){

var document = {

userID: this.currentUser,

userName: this.name,

email: this.email,

message: this.message,

}

db.collection("tickets").doc().set(document).then(() => {

this.message = null

this.triggerSnackbar("Your Query Has Been Successfully Logged!", "success")

}).catch(error => {

console.log(error)

})

}

else{

this.triggerSnackbar("There Were Errors With Your Form!", "error")

}

},

},

};

</script>

# Medical Record

## MedicalRecord.vue

<template>

<v-container>

<Navbar /> <!-- Import Patient Nav toolbar & drawer -->

<v-row><v-col cols="12" md="12" lg="12">

<v-card flat>

<v-card-title class="primary lighten-1 white--text">Medical Record

<v-spacer></v-spacer>

</v-card-title>

<v-row>

<!-- Gender Field -->

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title class="overline">

<v-subheader class="overline ml-n5">Gender

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-btn-toggle

v-model="gender"

color="primary"

group

mandatory

@change="onGenderChange(gender)"

> <!-- onGenderChange method method carries over selected value -->

<v-btn depressed x-large color="primary--text darken-1" value="Female">

<v-icon class="mr-2">fa-venus</v-icon>Female

</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="Male">

<v-icon class="mr-2">fa-mars</v-icon>Male

</v-btn>

</v-btn-toggle>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Type Field -->

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title class="overline">

<v-subheader class="overline ml-n5">Blood Type

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-select

v-model="selectedBlood"

:items="bloods"

label="Blood Type"

prepend-icon="fa-syringe"

outlined

@change="onBloodsChange(selectedBlood)"

></v-select> <!-- onBloodsChange method carries over selected value -->

</v-card-text>

</v-card>

</v-col>

<!-- Weight & Height Field -->

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Weight & Height

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Weight"

v-model="weight"

outlined

prepend-icon="fa-weight"

:maxlength="3"

suffix="lbs"

@change="onWeightChange(weight)"

:error-messages="weightError"

@input="$v.weight.$touch(weight)"

@blur="$v.weight.$touch(weight)"

></v-text-field> <!-- onWeightChange method carries over inputted value -->

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Height"

v-model="height"

outlined

prepend-icon="fa-ruler-vertical"

:maxlength="3"

suffix="inches"

@change="onHeightChange(height)"

:error-messages="heightError"

@input="$v.height.$touch(height)"

@blur="$v.height.$touch(height)"

></v-text-field> <!-- onHeightChange method carries over inputted value -->

</v-card-text>

</v-card>

</v-col>

<!-- Pulse Field (readonly) -->

<v-col cols="12" sm="4" md="4" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Pulse

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Pulse"

v-model="pulse"

outlined

prepend-icon="fa-heartbeat"

suffix="bpm"

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Pressure & Pulse Fields (readonly) -->

<v-col cols="12" sm="4" md="4" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Blood Pressure

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Systolic"

v-model="systolic"

outlined

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" sm="4" md="4" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Diastolic"

v-model="diastolic"

outlined

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Glucose Level Field (readonly) -->

<v-col cols="12" sm="4" md="4" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Blood Glucose Level

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Level"

v-model="bloodGlucoseLevel"

outlined

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Cholesterol Fields (readonly) -->

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Cholesterol

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Overall Cholesterol"

v-model="cholesterol"

outlined

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="LDL"

v-model="cholesterolLDL"

outlined

suffix="mmol/l"

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="HDL"

v-model="cholesterolHDL"

outlined

suffix="mmol/l"

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Triglycerides"

v-model="cholesterolTriglycerides"

outlined

suffix="mmol/l"

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Tabs for Allergies, Conditions & Immunisations -->

<v-col cols="12">

<v-tabs v-model="tab">

<v-tab v-for="tab in tabs" :key="tab.tabName">{{ tab.tabName }}</v-tab>

</v-tabs>

<v-tabs-items v-model="tab">

<v-tab-item> <!-- ALLERGIES -->

<v-card >

<Allergies /> <!-- Import Allergies Component -->

<v-col cols="12" md="12">

<!-- For every allergy in the allergies array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(allergy, index) in allergies" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-allergies</v-icon>

{{ allergy.allergyName }}

<v-subheader class="overline primary lighten-1 white--text">

{{ allergy.allergyType }}

</v-subheader>

<v-spacer></v-spacer> <!-- deleteAllergy method called on click of icon -->

<v-icon right @click="deleteAllergy(allergy)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Severity

</v-list-item-title>

<h3>{{ allergy.allergySeverity }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Descirption

</v-list-item-title>

<h3>{{ allergy.allergyDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col></v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

<v-tab-item> <!-- CONDITIONS -->

<v-card>

<Conditions /> <!-- Import Conditions Component -->

<v-col cols="12" md="12">

<!-- For every condition in the conditions array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(condition, index) in conditions" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-file-medical-alt</v-icon>

{{ condition.conditionName }}

<v-subheader class="overline primary lighten-1 white--text"></v-subheader>

<v-spacer></v-spacer> <!-- deleteCondition method called on click of icon -->

<v-icon right @click="deleteCondition(condition)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Date of Diagnosis

</v-list-item-title>

<h3>{{ condition.conditionDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Description

</v-list-item-title>

<h3>{{ condition.conditionDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

<v-tab-item><!-- immunisations -->

<v-card>

<immunisations /> <!-- Import Immunisations Component -->

<v-col cols="12" md="12">

<!-- For every immunisation in the immunisations array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(immunisation, index) in immunisations" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-syringe</v-icon>

{{ immunisation.immunisationName }}

<v-subheader class="overline primary lighten-1 white--text"></v-subheader>

<v-spacer></v-spacer> <!-- deleteImmunisation method called on click of icon -->

<v-icon right @click="deleteImmunisation(immunisation)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Date Received

</v-list-item-title>

<h3>{{ immunisation.immunisationDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Reaction

</v-list-item-title>

<h3>{{ immunisation.immunisationsReaction }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Additional Details

</v-list-item-title>

<h3>{{ immunisation.immunisationDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

</v-tabs-items>

</v-col>

<!-- Snackbar to display any success or error messages -->

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-row>

</v-card>

</v-col></v-row> <!-- End of display -->

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import Allergies from '../components/MedicalRecord/Allergies'

import Conditions from '../components/MedicalRecord/Conditions'

import Immunisations from '../components/MedicalRecord/Immunisations'

import { auth, db, fieldValue } from '../firebase'

import { numeric } from 'vuelidate/lib/validators'

export default {

components: {

Navbar,

Allergies,

Conditions,

Immunisations,

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

// Realtime listen to the patients medical record fields in the users collection

db.collection("users").doc(this.currentUser).onSnapshot(doc => {

let storedRecord = doc.data() // store the doc.data() into a value to get each field

storedRecord.id = doc.id

// Populate these values to display to the user what fields are currently holding info

this.gender = storedRecord.gender

this.selectedBlood = storedRecord.bloodType

this.weight = storedRecord.weight

this.height = storedRecord.height

this.pulse = storedRecord.pulse

this.systolic = storedRecord.systolic

this.diastolic = storedRecord.diastolic

this.bloodGlucoseLevel = storedRecord.bloodGlucoseLevel

this.cholesterol = storedRecord.cholesterol

this.cholesterolLDL = storedRecord.cholesterolLDL

this.cholesterolHDL = storedRecord.cholesterolHDL

this.cholesterolTriglycerides = storedRecord.cholesterolTriglycerides

// Populate arrays for allergies, conditions & immunisations

this.allergies = storedRecord.allergy

this.conditions = storedRecord.condition

this.immunisations = storedRecord.immunisation

})

},

computed: {

// Validation Errors

weightError () {

const errors = []

if(!this.$v.weight.$dirty) return errors

!this.$v.weight.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

heightError () {

const errors = []

if(!this.$v.height.$dirty) return errors

!this.$v.height.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

},

data() {

return {

currentUser: null,

// Snackbar fields

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

// Tabs to switch between allergies, conditions & immunisations

tab: null,

tabs: [

{ tabName: 'Allergies' },

{ tabName: 'Conditions' },

{ tabName: 'Immunisations' },

],

gender: null, // Gender value passes from db to buttons and highlights active one

weight: null, // Used to store weight

height: null, // Used to store height

selectedBlood: null, // Used to store the selected blood type

// Blood choices

bloods: [

{text: 'None'},

{text: 'A Positive'},

{text: 'A Negative'},

{text: 'B Positive'},

{text: 'B Negative'},

{text: 'O Positive'},

{text: 'O Negative'},

{text: 'AB Positive'},

{text: 'AB Negative'},

],

pulse: null, // Used to store pulse

systolic: null, // Used to store systolic

diastolic: null, // Used to store diastolic

bloodGlucoseLevel: null, // Used to store bloodGlucoseLevel

cholesterol: null, // Used to store cholesterol

cholesterolLDL: null, // Used to store cholesterolLDL

cholesterolHDL: null, // Used to store cholesterolHDL

cholesterolTriglycerides: null, // Used to store cholesterolTriglycerides

// Arrays

allergies: [], // Store allergies here to display to user

conditions: [], // Store conditions here to display to user

immunisations: [], // Store immunisations here to display to user

}

},

validations: {

// numeric values only

weight: { numeric },

height: { numeric }

},

methods: {

// Error and success messages

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// Gender button changed

onGenderChange (gender) {

// Access the users collection then update gender value

db.collection("users").doc(this.currentUser).get().then(() =>{

// so long as gender is not null

if(gender != null){

db.collection("users").doc(this.currentUser).update({

gender: gender

})

}

})

},

// Blood selection changed

onBloodsChange(selectedBlood) {

// Access the users collection then update blood type

db.collection("users").doc(this.currentUser).get().then(() =>{

// so long as blood type is not null

if(selectedBlood != null){

db.collection("users").doc(this.currentUser).update({

bloodType: selectedBlood

})

}

})

},

// Weight value changed

onWeightChange (weight) {

// Check to see if entered data is dirty before updating the db

this.$v.$touch()

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

// if no errors and no dirty values

if(this.errors === false && this.formTouched === false){

db.collection("users").doc(this.currentUser).get().then(() =>{

// Access the users collection then update weight if not null

if(weight != null){

db.collection("users").doc(this.currentUser).update({

weight: weight

})

}

})

}

},

// Height value changed

onHeightChange (height) {

// Check to see if entered data is dirty before updating the db

this.$v.$touch()

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update height if not null

db.collection("users").doc(this.currentUser).get().then(() =>{

if(height != null){

db.collection("users").doc(this.currentUser).update({

height: height

})

}

})

}

},

// NB. fieldValue - a sentinel value that can be used to tell server to remove

// given elements from any array value that exists on the server

// NB. arrayRemove - removes the exact item specified from the array

// Delete an allergy

deleteAllergy(allergy){

db.collection("users").doc(this.currentUser).update({

// remove the allergy from the object allergies from array

allergy: fieldValue.arrayRemove(allergy)

})

},

// Delete a condition

deleteCondition (condition) {

db.collection("users").doc(this.currentUser).update({

// remove the condition from the object conditions from array

condition: fieldValue.arrayRemove(condition)

})

},

// Delete an immunisation

deleteImmunisation (immunisation) {

db.collection("users").doc(this.currentUser).update({

// remove the immunisation from the object immunisations from array

immunisation: fieldValue.arrayRemove(immunisation)

})

},

},

};

</script>

## Doctor.vue

<template>

<v-container>

<DoctorNavbar />

<v-col cols="12" s="12" sm="12" md="12" lg="12">

<!-- Content is wrapped with a dialog model which when opened covers page -->

<v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">

<template v-slot:activator="{ on, attrs }">

<!-- Card for Patient Records Search Bar -->

<v-card outlined>

<v-card-title class="text-uppercase grey--text text--darken-1">Patients

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="fa-search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

</v-card>

<!-- Card for the Data Table -->

<v-card outlined>

<v-data-table

:headers="headers"

:items="patients"

:search="search"

:items-per-page="10"

class="elevation-1"

>

<!-- Action field, each record corresponds to a seperate ID -->

<template v-slot:[`item.actions`]="{ item }">

<v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-icon

class="mr-2"

:key="item.id"

:items="patients"

v-bind="attrs"

v-on="on"

@click="storeID(item.id)"

>fa-notes-medical

</v-icon>

</v-col>

</template>

</v-data-table>

</v-card>

</template>

<!-- Dialog opens this card where the doctor can update the patients medical record -->

<v-card>

<v-card-title class="primary white--text">

<v-btn class="ml-0" icon dark @click="close()">

<v-icon class="ml-0" fab dark color="white--text darken-1 ">fa-long-arrow-alt-left</v-icon>

</v-btn>

<span class="ml-10">Update Medical Record</span>

</v-card-title>

<v-row>

<v-col cols="12" md="12" lg="12">

<!-- ALL MEDICAL RECORD FIELDS HERE -->

<v-card flat class="mt-n5">

<v-form @submit.prevent="updateMedicalRecord">

<v-card-text>

<v-row>

<!-- Gender Field -->

<v-col class="mt-n3" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title class="overline">

<v-subheader class="overline ml-n5">Gender

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Gender"

v-model="gender"

outlined

readonly

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Type Field -->

<v-col class="mt-n3" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title class="overline">

<v-subheader class="overline ml-n5">Blood Type

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-select

v-model="selectedBlood"

:items="bloods"

label="Blood Type"

prepend-icon="fa-syringe"

outlined

@change="onBloodsChange(selectedBlood)"

></v-select>

</v-card-text>

</v-card>

</v-col>

<!-- Weight & Height Fields -->

<v-col class="mt-n3" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Weight & Height

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Weight"

v-model="medicalForm.weight"

outlined

prepend-icon="fa-weight"

:maxlength="3"

suffix="lbs"

@change="onWeightChange(medicalForm.weight)"

:error-messages="weightError"

@input="$v.medicalForm.weight.$touch()"

@blur="$v.medicalForm.weight.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Height"

v-model="medicalForm.height"

outlined

prepend-icon="fa-ruler-vertical"

:maxlength="3"

suffix="inches"

@change="onHeightChange(medicalForm.height)"

:error-messages="heightError"

@input="$v.medicalForm.height.$touch()"

@blur="$v.medicalForm.height.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Pulse Field -->

<v-col class="mt-n6" cols="12" sm="3" md="3" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Pulse

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Pulse"

v-model="medicalForm.pulse"

outlined

prepend-icon="fa-heartbeat"

:maxlength="3"

suffix="bpm"

@change="onPulseChange(medicalForm.pulse)"

:error-messages="pulseError"

@input="$v.medicalForm.pulse.$touch()"

@blur="$v.medicalForm.pulse.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Pressure & Pulse Field -->

<v-col class="mt-n6" cols="12" sm="3" md="3" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Blood Pressure

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Systolic"

v-model="medicalForm.systolic"

outlined

:maxlength="3"

@change="onSystolicChange(medicalForm.systolic)"

:error-messages="systolicError"

@input="$v.medicalForm.systolic.$touch()"

@blur="$v.medicalForm.systolic.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col class="mt-n6" cols="12" sm="3" md="3" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Diastolic"

v-model="medicalForm.diastolic"

outlined

:maxLength="3"

@change="onDiastolicChange(medicalForm.diastolic)"

:error-messages="diastolicError"

@input="$v.medicalForm.diastolic.$touch()"

@blur="$v.medicalForm.diastolic.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Blood Glucose Level Field -->

<v-col class="mt-n6" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Blood Glucose Level

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-select

label="Levels"

v-model="medicalForm.bloodGlucoseLevel"

:items="bloodGlucoseLevelList"

outlined

suffix="mmol/l"

@change="onLevelChange(medicalForm.bloodGlucoseLevel)"

:error-messages="bloodGlucoseLevelError"

@input="$v.medicalForm.bloodGlucoseLevel.$touch()"

@blur="$v.medicalForm.bloodGlucoseLevel.$touch()"

></v-select>

</v-card-text>

</v-card>

</v-col>

<!-- Cholesterol Fields -->

<v-col class="mt-n6" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">Cholesterol

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-select

label="Total Cholesterol"

v-model="medicalForm.cholesterol"

:items="totalCholList"

outlined

suffix="mmol/l"

prepend-icon="fa-heart-broken"

@change="onTotalCholChange(medicalForm.cholesterol)"

:error-messages="cholesterolError"

@input="$v.medicalForm.cholesterol.$touch()"

@blur="$v.medicalForm.cholesterol.$touch()"

></v-select>

</v-card-text>

</v-card>

</v-col>

<v-col class="mt-n6" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="LDL"

v-model="medicalForm.cholesterolLDL"

outlined

:maxLength="3"

suffix="mmol/l"

@change="onLDLChange(medicalForm.cholesterolLDL)"

:error-messages="cholesterolLDLError"

@input="$v.medicalForm.cholesterolLDL.$touch()"

@blur="$v.medicalForm.cholesterolLDL.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col class="mt-n6" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="HDL"

v-model="medicalForm.cholesterolHDL"

outlined

:maxLength="3"

suffix="mmol/l"

@change="onHDLChange(medicalForm.cholesterolHDL)"

:error-messages="cholesterolHDLError"

@input="$v.medicalForm.cholesterolHDL.$touch()"

@blur="$v.medicalForm.cholesterolHDL.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<v-col class="mt-n6" cols="12" sm="6" md="6" lg="3">

<v-card flat>

<v-card-title>

<v-subheader class="overline ml-n5">

<v-divider></v-divider>

</v-subheader>

</v-card-title>

<v-card-text>

<v-text-field

label="Triglycerides"

v-model="medicalForm.cholesterolTriglycerides"

outlined

:maxLength="3"

suffix="mmol/l"

@change="onTriglyceridesChange(medicalForm.cholesterolTriglycerides)"

:error-messages="cholesterolTriglyceridesError"

@input="$v.medicalForm.cholesterolTriglycerides.$touch()"

@blur="$v.medicalForm.cholesterolTriglycerides.$touch()"

></v-text-field>

</v-card-text>

</v-card>

</v-col>

<!-- Tabs for Allergies, Conditions & Immunisations -->

<v-col class="mt-n4" cols="12">

<v-tabs v-model="tab">

<v-tab v-for="tab in tabs" :key="tab.tabName">{{ tab.tabName }}</v-tab>

</v-tabs>

<v-tabs-items v-model="tab">

<v-tab-item> <!-- ALLERGIES -->

<v-card >

<!-- Any Allergies -->

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogAllergies" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add Allergy

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveAllergy()">

<v-card-title class="primary lighten-1 white--text">Allergy

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-text-field

label="Allergen Name\*"

v-model.trim="allergyForm.allergyName"

:counter="25"

outlined

required

:error-messages="allergyNameErrors"

@input="$v.allergyForm.allergyName.$touch()"

@blur="$v.allergyForm.allergyName.$touch()"

></v-text-field>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-select

v-model="allergyForm.selectedAllergyType"

:items="allergyList"

label="Allergy Type\*"

outlined

required

:error-messages="allergyTypeErrors"

@input="$v.allergyForm.selectedAllergyType.$touch()"

@blur="$v.allergyForm.selectedAllergyType.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="1"

label="Details\*"

v-model="allergyForm.allergyDetails"

:counter="150"

:maxlength="150"

outlined

required

:error-messages="allergyDetailsErrors"

@input="$v.allergyForm.allergyDetails.$touch()"

@blur="$v.allergyForm.allergyDetails.$touch()"

></v-textarea>

</v-col>

<v-col cols="12" sm="6">

<v-select

v-model="allergyForm.selectedSeverity"

:items="severityList"

label="Severity\*"

outlined

required

:error-messages="allergySeverityErrors"

@input="$v.allergyForm.selectedSeverity.$touch()"

@blur="$v.allergyForm.selectedSeverity.$touch()"

></v-select>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

type="submit"

block class="primary white--text"

@click.prevent="saveAllergy()">

<span>Add Allergy</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

</v-card>

</v-dialog>

</v-card>

</v-col> <!-- allergies end -->

<v-col cols="12" md="12">

<!-- For every allergy in the allergies array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(allergy, index) in allergies" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-allergies</v-icon>

{{ allergy.allergyName }}

<v-subheader class="overline primary lighten-1 white--text">

{{ allergy.allergyType }}

</v-subheader>

<v-spacer></v-spacer> <!-- deleteAllergy method called on click of icon -->

<v-icon right @click="deleteAllergy(allergy)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Severity

</v-list-item-title>

<h3>{{ allergy.allergySeverity }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Descirption

</v-list-item-title>

<h3>{{ allergy.allergyDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col></v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

<v-tab-item> <!-- CONDITIONS -->

<v-card>

<!-- Any Conditions -->

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogConditions" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add Condition

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveCondition()">

<v-card-title class="primary lighten-1 white--text">Condition

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-autocomplete

label="Search Condition Name"

v-model="conditionForm.conditionName"

:items="conditionsList"

item-text="name"

item-value="value"

outlined

:error-messages="conditionNameErrors"

@input="$v.conditionForm.conditionName.$touch()"

@blur="$v.conditionForm.conditionName.$touch()"

></v-autocomplete>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="1"

label="Details\*"

v-model.trim="conditionForm.conditionDetails"

:counter="150"

:maxlength="150"

outlined

required

:error-messages="conditionDetailsErrors"

@input="$v.conditionForm.conditionDetails.$touch()"

@blur="$v.conditionForm.conditionDetails.$touch()"

></v-textarea>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-menu

v-model="menu"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Date\*"

readonly

:value="conditionForm.conditionDate"

v-bind="attrs"

v-on="on"

required

outlined

:error-messages="conditionDateErrors"

@input="$v.conditionForm.conditionDate.$touch()"

@blur="$v.conditionForm.conditionDate.$touch()"

@click:clear="conditionForm.conditionDate = null"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

:min="getUsersDob()"

:max="getTodaysDate()"

v-model="conditionForm.conditionDate"

@input="menu = false"

>

</v-date-picker>

</v-menu>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

block class="primary white--text"

@click.prevent="saveCondition()">

<span>Add Condition</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

<v-snackbar

:color= "color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

> {{ snackbarText }}

</v-snackbar>

</v-card>

</v-dialog>

</v-card>

</v-col> <!-- conditions end -->

<v-col cols="12" md="12">

<!-- For every condition in the conditions array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(condition, index) in conditions" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-file-medical-alt</v-icon>

{{ condition.conditionName }}

<v-subheader class="overline primary lighten-1 white--text"></v-subheader>

<v-spacer></v-spacer> <!-- deleteCondition method called on click of icon -->

<v-icon right @click="deleteCondition(condition)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Date of Diagnosis

</v-list-item-title>

<h3>{{ condition.conditionDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Description

</v-list-item-title>

<h3>{{ condition.conditionDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

<v-tab-item><!-- immunisations -->

<v-card>

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogImmunisation" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add immunisation

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveImmunisation()">

<v-card-title class="primary lighten-1 white--text">Immunisation

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-autocomplete

label="Search Immunisations\*"

v-model="immunisationForm.selectedImmunisation"

:items="immunisationsList"

:filter="customFilter"

item-text="name"

item-value="value"

persistent-hint

hint="Immunisation Name Or Abbreviation"

outlined

:error-messages="selectedImmunisationErrors"

@input="$v.immunisationForm.selectedImmunisation.$touch()"

@blur="$v.immunisationForm.selectedImmunisation.$touch()"

></v-autocomplete>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-menu

v-model="menu2"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Date\*"

readonly

:value="immunisationForm.immunisationDate"

v-bind="attrs"

v-on="on"

required

outlined

:error-messages="immunisationDateErrors"

@input="$v.immunisationForm.immunisationDate.$touch()"

@blur="$v.immunisationForm.immunisationDate.$touch()"

@click:clear="immunisationForm.immunisationDate = null"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

:min="getUsersDob()"

:max="getTodaysDate()"

v-model="immunisationForm.immunisationDate"

@input="menu2 = false"

>

</v-date-picker>

</v-menu>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-select

label="Reaction"

v-model="immunisationForm.immunisationReaction"

:items="reactionList"

outlined

required

:error-messages="immunisationReactionErrors"

@input="$v.immunisationForm.immunisationReaction.$touch()"

@blur="$v.immunisationForm.immunisationReaction.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="2"

name="immunisationForm.immunisationDetails"

label="Additional Details"

v-model="immunisationForm.immunisationDetails"

:counter="150"

:maxlength="150"

required

outlined

:error-messages="additionalDetailsError"

@input="$v.immunisationForm.immunisationDetails.$touch()"

@blur="$v.immunisationForm.immunisationDetails.$touch()"

></v-textarea>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

type="submit"

block class="primary white--text"

@click.prevent="saveImmunisation()">

<span>Add Immunisation</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

<v-snackbar

:color= "color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

> {{ snackbarText }}

</v-snackbar>

</v-card>

</v-dialog>

</v-card>

</v-col>

<v-col cols="12" md="12">

<!-- For every immunisation in the immunisations array, recognised seperately by index position -->

<v-card outlined class="mt-2" v-for="(immunisation, index) in immunisations" :key="index">

<v-card-title class="primary lighten-1 white--text">

<v-icon class="mr-4 white--text">fa-syringe</v-icon>

{{ immunisation.immunisationName }}

<v-subheader class="overline primary lighten-1 white--text"></v-subheader>

<v-spacer></v-spacer> <!-- deleteImmunisation method called on click of icon -->

<v-icon right @click="deleteImmunisation(immunisation)">fa-trash</v-icon>

</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Date Received

</v-list-item-title>

<h3>{{ immunisation.immunisationDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Reaction

</v-list-item-title>

<h3>{{ immunisation.immunisationsReaction }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title

class="overline grey--text mb-4">Additional Details

</v-list-item-title>

<h3>{{ immunisation.immunisationDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

</v-card>

</v-tab-item>

</v-tabs-items>

</v-col>

<!-- Snackbar to display any success or error messages -->

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-row>

</v-card-text>

</v-form>

</v-card>

</v-col></v-row>

</v-card>

</v-dialog>

</v-col>

</v-container>

</template>

<script>

import DoctorNavbar from '../components/Navbars/DoctorNavbar'

import Allergies from '../components/MedicalRecord/Allergies'

import Conditions from '../components/MedicalRecord/Conditions'

import Immunisations from '../components/MedicalRecord/Immunisations'

import { auth, db, fieldValue } from '../firebase'

import { required, numeric, minLength, between } from 'vuelidate/lib/validators'

export default {

components: {

DoctorNavbar,

Allergies,

Conditions,

Immunisations,

},

created () {

this.currentUser = auth.currentUser.uid // Get current users ID

// Gets the patients through roles by filtering only patient roles IDs

db.collection("roles").get().then(snap => {

snap.forEach(doc => {

let user = doc.data()

// if user is not doctor or admin retrieve the record

if (!user.role.admin && !user.role.doctor) {

// Then set user.id to the doc.id to refer back to their record in users

user.id = doc.id

db.collection("users").doc(user.id).get().then(doc => {

let patientRecord = doc.data()

patientRecord.id = doc.id

// Set the patient info for the data table

let patient = {

id: patientRecord.id,

name: patientRecord.firstname + ' ' + patientRecord.surname,

dob: patientRecord.date,

gender: patientRecord.gender,

ppsn: patientRecord.ppsn

}

this.patients.push(patient) // Store the patient info to the patients array

})

}

})

})

},

computed: {

// Validation Errors

weightError () {

const errors = []

if(!this.$v.medicalForm.weight.$dirty) return errors

!this.$v.medicalForm.weight.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.weight.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

heightError () {

const errors = []

if(!this.$v.medicalForm.height.$dirty) return errors

!this.$v.medicalForm.height.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.height.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

pulseError () {

const errors = []

if(!this.$v.medicalForm.pulse.$dirty) return errors

!this.$v.medicalForm.pulse.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.pulse.numeric && errors.push('Only Numeric Values Will Save')

!this.$v.medicalForm.pulse.minLength && errors.push('Minimum Pulse Should Be In Double Digits')

!this.$v.medicalForm.pulse.between && errors.push('Pulse Should Be Between Values 60 To 120')

return errors

},

systolicError () {

const errors = []

if(!this.$v.medicalForm.systolic.$dirty) return errors

!this.$v.medicalForm.systolic.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.systolic.numeric && errors.push('Only Numeric Values Will Save')

!this.$v.medicalForm.systolic.minLength && errors.push('Systolic Should Be In Double Digits')

!this.$v.medicalForm.systolic.between && errors.push('Systolic Should Be Between Values 70 To 190')

return errors

},

diastolicError () {

const errors = []

if(!this.$v.medicalForm.diastolic.$dirty) return errors

!this.$v.medicalForm.diastolic.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.diastolic.numeric && errors.push('Only Numeric Values Will Save')

!this.$v.medicalForm.diastolic.minLength && errors.push('Diastolic Should Be In Double Digits')

!this.$v.medicalForm.diastolic.between && errors.push('Diastolic Should Be Between Values 40 To 100')

return errors

},

bloodGlucoseLevelError () {

const errors = []

if(!this.$v.medicalForm.bloodGlucoseLevel.$dirty) return errors

!this.$v.medicalForm.bloodGlucoseLevel.required && errors.push('Empty Fields Will Not Save')

return errors

},

cholesterolError () {

const errors = []

if(!this.$v.medicalForm.cholesterol.$dirty) return errors

!this.$v.medicalForm.cholesterol.required && errors.push('Empty Fields Will Not Save')

return errors

},

cholesterolLDLError () {

const errors = []

if(!this.$v.medicalForm.cholesterolLDL.$dirty) return errors

!this.$v.medicalForm.cholesterolLDL.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.cholesterolLDL.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

cholesterolHDLError () {

const errors = []

if(!this.$v.medicalForm.cholesterolHDL.$dirty) return errors

!this.$v.medicalForm.cholesterolHDL.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.cholesterolHDL.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

cholesterolTriglyceridesError () {

const errors = []

if(!this.$v.medicalForm.cholesterolTriglycerides.$dirty) return errors

!this.$v.medicalForm.cholesterolTriglycerides.required && errors.push('Empty Fields Will Not Save')

!this.$v.medicalForm.cholesterolTriglycerides.numeric && errors.push('Only Numeric Values Will Save')

return errors

},

allergyNameErrors () {

const errors = []

if(!this.$v.allergyForm.allergyName.$dirty) return errors

!this.$v.allergyForm.allergyName.required && errors.push('An Allergy Name Is Required')

return errors

},

allergyTypeErrors () {

const errors = []

if(!this.$v.allergyForm.selectedAllergyType.$dirty) return errors

!this.$v.allergyForm.selectedAllergyType.required && errors.push('Select The Type Of Allergy')

return errors

},

allergyDetailsErrors () {

const errors = []

if(!this.$v.allergyForm.allergyDetails.$dirty) return errors

!this.$v.allergyForm.allergyDetails.required && errors.push('Include Some Details About Your Allergy')

return errors

},

allergySeverityErrors () {

const errors = []

if(!this.$v.allergyForm.selectedSeverity.$dirty) return errors

!this.$v.allergyForm.selectedSeverity.required && errors.push('Select The Severity Of Your Allergy')

return errors

},

conditionNameErrors () {

const errors = []

if(!this.$v.conditionForm.conditionName.$dirty) return errors

!this.$v.conditionForm.conditionName.required && errors.push('A Condition Name Is Required')

return errors

},

conditionDetailsErrors () {

const errors = []

if(!this.$v.conditionForm.conditionDetails.$dirty) return errors

!this.$v.conditionForm.conditionDetails.required && errors.push('Include Some Details About Your Condition')

return errors

},

conditionDateErrors () {

const errors = []

if(!this.$v.conditionForm.conditionDate.$dirty) return errors

!this.$v.conditionForm.conditionDate.required && errors.push('Select Date Of Condition Diagnosis')

return errors

},

selectedImmunisationErrors () {

const errors = []

if(!this.$v.immunisationForm.selectedImmunisation.$dirty) return errors

!this.$v.immunisationForm.selectedImmunisation.required && errors.push('Select An Immunisation')

return errors

},

immunisationDateErrors () {

const errors = []

if(!this.$v.immunisationForm.immunisationDate.$dirty) return errors

!this.$v.immunisationForm.immunisationDate.required && errors.push('Select Date Of Immunisation')

return errors

},

immunisationReactionErrors () {

const errors = []

if(!this.$v.immunisationForm.immunisationReaction.$dirty) return errors

!this.$v.immunisationForm.immunisationReaction.required && errors.push('Select Your Reaction To the Immunisation')

return errors

},

additionalDetailsError () {

const errors = []

if(!this.$v.immunisationForm.immunisationDetails.$dirty) return errors

!this.$v.immunisationForm.immunisationDetails.required && errors.push('Include Additional Details Of Immunisation')

return errors

},

},

data() {

return {

currentUser: null, // Used to store the current logged in user

patientID: null, // Store the patient ID when doctor clicks on actions

dialog: false,

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

patients: [], // Patients array for data table

patient: null, // Individual patient info

search: '', // Search bar

/// Headers define the labels for the data table and the values take the values from patient

headers: [

{ text: 'Patient Name', value: 'name' },

{ text: 'DoB', value: 'dob'},

{ text: 'Gender', value: 'gender' },

{ text: 'PPSN', value: 'ppsn' },

{ text: 'Actions', value: 'actions', sortable: false },

],

gender: null, // Gender value passes from db to buttons and highlights active one

selectedBlood: null, // Used to store the selected blood type

// for blood type select

bloods: [

{text: 'None'},

{text: 'A Positive'},

{text: 'A Negative'},

{text: 'B Positive'},

{text: 'B Negative'},

{text: 'O Positive'},

{text: 'O Negative'},

{text: 'AB Positive'},

{text: 'AB Negative'},

],

// for blood glucose level select

bloodGlucoseLevelList: [

{ text: 'Less Than 3', value: 'Severe Hypoglycemia'},

{ text: 'Less Than 3.9', value: 'Hypoglycemia'},

{ text: 'Less Than 7', value: 'Normal'},

{ text: 'Less Than 10', value: 'High (Take Action)'},

{ text: 'Above 10 - 27.7 +', value: 'Metabolic Consequences (Take Action)'}

],

// for total cholesterol select

totalCholList: [

{ text: 'Less Than 200', value: 'Desirable' },

{ text: 'Between 200 & 239', value: 'Borderline High' },

{ text: '240 & Above', value: 'High' },

],

medicalForm: {

weight: null, // Used to store weight

height: null, // Used to store height

pulse: null, // Used to store pulse

systolic: null, // Used to store systolic

diastolic: null, // Used to store diastolic

bloodGlucoseLevel: null, // Used to store bloodGlucoseLevel

cholesterol: null, // Used to store cholesterol

cholesterolLDL: null, // Used to store cholesterolLDL

cholesterolHDL: null, // Used to store cholesterolHDL

cholesterolTriglycerides: null, // Used to store cholesterolTriglycerides

},

allergies: [], // Store allergies here to display to user

conditions: [], // Store conditions here to display to user

immunisations: [], // Store immunisations here to display to user

// Tabs for upcoming and past

tab: null,

tabs: [

{ tabName: 'Allergies' },

{ tabName: 'Conditions' },

{ tabName: 'Immunisations' },

],

// Everything for Allergies

dialogAllergies: false,

allergyList: [

{ text: 'Drug Allergy', value: 'Drug Allergy'},

{ text: 'Food Allergy', value: 'Food Allergy'},

{ text: 'Pollen Allergy', value: 'Pollen Allergy'},

{ text: 'Insect Allergy', value: 'Insect Allergy'},

{ text: 'Latex Allergy', value: 'Latex Allergy'},

{ text: 'Other', value: 'Other'},

],

severityList: [

{ text: 'Mild', value: 'Mild'},

{ text: 'Moderate', value: 'Moderate'},

{ text: 'Severe', value: 'Severe'},

{ text: 'Life Threatening', value: 'Life Threatening'},

],

allergyForm: {

allergyName: null,

selectedAllergyType: null, // Value holds the allergy selected by the user

allergyDetails: null, // Value holds user details

selectedSeverity: null, // Value holds the severity selected by the user

},

// Everything for Conditions

dialogConditions: false,

menu: false,

// Condition Name Autocomplete Selection Field Name & Values

conditionsList: [

{ name: 'Alzheimer`s Disease', value: 'Alzheimer`s Disease'},

{ name: 'Anemia', value: 'Anemia'},

{ name: 'Anxiety Disorder', value: 'Anxiety Disorder'},

{ name: 'Asthma', value: 'Asthma'},

{ name: 'Atopic Dermatitis', value: 'Atopic Dermatitis'},

{ name: 'Atrial Fibrillation', value: 'Atrial Fibrillation'},

{ name: 'Attention Deficit Hyperactivity Disorder', value: 'Attention Deficit Hyperactivity Disorder'},

{ name: 'Autism or Asperger`s Syndrome', value: 'Autism or Asperger`s Syndrome'},

{ name: 'Bipolar Mood Disorder', value: 'Bipolar Mood Disorder'},

{ name: 'Bronchiectasis', value: 'Bronchiectasis'},

{ name: 'Cancer', value: 'Cancer'},

{ name: 'Cardiac Failure', value: 'Cardiac Failure'},

{ name: 'Cardiomyopathy', value: 'Cardiomyopathy'},

{ name: 'Cerebral Palsy', value: 'Cerebral Palsy'},

{ name: 'Cerebrovascular Disease', value: 'Cerebrovascular Disease'},

{ name: 'Chronic Fatigue Syndrome', value: 'Chronic Fatigue Syndrome'},

{ name: 'Chronic Hepatitis', value: 'Chronic Hepatitis'},

{ name: 'Chronic Kidney Disease', value: 'Chronic Kidney Disease'},

{ name: 'Chronic Obstructive Pulmonary Disorder', value: 'Chronic Obstructive Pulmonary Disorder'},

{ name: 'Chronic Renal Disease', value: 'Chronic Renal Disease'},

{ name: 'Coeliac Disease', value: 'Coeliac Disease'},

{ name: 'Complex Regional Pain Syndrome', value: 'Complex Regional Pain Syndrome'},

{ name: 'Coronary Artery Disease', value: 'Coronary Artery Disease'},

{ name: 'Crohn`s Disease', value: 'Crohn`s Disease'},

{ name: 'Deafness & Hearing Impairment', value: 'Deafness & Hearing Impairment'},

{ name: 'Dermatological Conditions', value: 'Dermatological Conditions'},

{ name: 'Diabetes Insipidus', value: 'Diabetes Insipidus'},

{ name: 'Diabetes Mellitus Types 1 & 2', value: 'Diabetes Mellitus Types 1 & 2'},

{ name: 'Dysrhythmias', value: 'Dysrhythmias'},

{ name: 'Ehlers-Danlos Syndrome', value: 'Ehlers-Danlos Syndrome'},

{ name: 'Endometriosis', value: 'Endometriosis'},

{ name: 'Epilepsy', value: 'Epilepsy'},

{ name: 'Fetal Alcohol Spectrum Disorder', value: 'Fetal Alcohol Spectrum Disorder'},

{ name: 'Fibromyalgia', value: 'Fibromyalgia'},

{ name: 'Gestational Diabetes', value: 'Gestational Diabetes'},

{ name: 'Glaucoma ', value: 'Glaucoma '},

{ name: 'Haemophilia', value: 'Haemophilia'},

{ name: 'Hashimoto`s Thyroiditis / Hashimoto`s Disease', value: 'Hashimoto`s Thyroiditis / Hashimoto`s Disease'},

{ name: 'HIV/AIDS', value: 'HIV/AIDS'},

{ name: 'Huntington`s Disease', value: 'Huntington`s Disease'},

{ name: 'Hyperlipidaemia', value: 'Hyperlipidaemia'},

{ name: 'Hypertension', value: 'Hypertension'},

{ name: 'Hypothyroidism', value: 'Hypothyroidism'},

{ name: 'Ischemic Cardiomyopathy', value: 'Ischemic Cardiomyopathy'},

{ name: 'Mental Illness', value: 'Mental Illness'},

{ name: 'Migraines', value: 'Migraines'},

{ name: 'Multiple Sclerosis', value: 'Multiple Sclerosis'},

{ name: 'Myasthenia Gravis', value: 'Myasthenia Gravis'},

{ name: 'Narcolepsy', value: 'Narcolepsy'},

{ name: 'Obesity', value: 'Obesity'},

{ name: 'Osteoarthritis', value: 'Osteoarthritis'},

{ name: 'Osteoporosis', value: 'Osteoporosis'},

{ name: 'Parkinson`s Disease', value: 'Parkinson`s Disease'},

{ name: 'Periodontal Disease', value: 'Periodontal Disease'},

{ name: 'Polycystic Ovary Syndrome', value: 'Polycystic Ovary Syndrome'},

{ name: 'Postural Orthostatic Tachycardia Syndrome', value: 'Postural Orthostatic Tachycardia Syndrome'},

{ name: 'Post-Vasectomy Pain Syndrome', value: 'Post-Vasectomy Pain Syndrome'},

{ name: 'Pre-Diabetes', value: 'Pre-Diabetes'},

{ name: 'Psoriasis', value: 'Psoriasis'},

{ name: 'Relapsing Polychondritis', value: 'Relapsing Polychondritis'},

{ name: 'Rheumatoid Arthritis', value: 'Rheumatoid Arthritis'},

{ name: 'Schizophrenia', value: 'Schizophrenia'},

{ name: 'Sleep Apnea', value: 'Sleep Apnea'},

{ name: 'Substance Abuse Disorders', value: 'Substance Abuse Disorders'},

{ name: 'Systemic Lupus Erythematosus', value: 'Systemic Lupus Erythematosus'},

{ name: 'Thyroid Disease', value: 'Thyroid Disease'},

{ name: 'Ulcerative Colitis', value: 'Ulcerative Colitis'},

],

conditionForm: {

condtionName: null,

conditionDetails: null,

conditionDate: null,

},

// Everything for Immunisations

dialogImmunisation: false,

menu2: false,

selectedImmunisation: null,

immunisationsList: [

{ name: 'Anthrax Vaccine Adsorbed', abbr: 'AVA', value: 'Anthrax Vaccine Adsorbed', id: 1 },

{ name: 'Bacille Calmette-Guérin (Tuberculosis) Vaccine', abbr: 'BCG', value: 'Bacille Calmette-Guérin (Tuberculosis) Vaccine', id: 2},

{ name: 'Diphtheria, Tetanus & Pertussis Vaccine', abbr: 'DTaP', value: 'Diphtheria, Tetanus & Pertussis / Whopping Cough Vaccine', id: 3},

{ name: 'Hepatitis A Vaccine', abbr: 'HepA', value: 'Hepatitis A Vaccine', id: 4 },

{ name: 'Hepatitis B Vaccine', abbr: 'HepB', value: 'Hepatitis B Vaccine', id: 5 },

{ name: 'Haemophilus Influenzae Type B', abbr: 'Hib', value: 'Haemophilus Influenzae Type B', id: 6 },

{ name: 'Human Papillomavirus', abbr: 'HPV', value: 'Human Papillomavirus', id: 7 },

{ name: 'Inactivated Poliovirus Vaccine', abbr: 'IPV', value: 'Inactivated Poliovirus Vaccine', id: 8 },

{ name: 'Inactivated Influenza Vaccine', abbr: 'IIV', value: 'Inactivated Influenza Vaccine', id: 9 },

{ name: 'Japanese Encephalitis', abbr: 'JE', value: 'Japanese Encephalitis', id: 10 },

{ name: 'Live, Attenuated Influenza Vaccine (Quadrivalent)', abbr: 'LAIV4 ', value: 'Live, Attenuated Influenza Vaccine (Quadrivalent)', id: 11 },

{ name: 'Meningococcal A, C, W, Y', abbr: 'MenACWY', value: 'Meningococcal A, C, W, Y', id: 12 },

{ name: 'Meningococcal B', abbr: 'MenB', value: 'Meningococcal B', id: 13 },

{ name: 'Measles, Mumps, & Rubella', abbr: 'MMR', value: 'Measles, Mumps, & Rubella', id: 14 },

{ name: 'Measles-Rubella Vaccine', abbr: 'MR', value: 'Measles-Rubella Vaccine', id: 15 },

{ name: 'Pneumococcal Conjugate Vaccine', abbr: 'PCV13', value: 'Pneumococcal Conjugate Vaccine', id: 16 },

{ name: 'Pneumococcal Polysaccharide Vaccine', abbr: 'PPSV23', value: 'Pneumococcal Polysaccharide Vaccine', id: 17 },

{ name: 'Recombinant Influenza Vaccine', abbr: 'RIV3', value: 'Recombinant Influenza Vaccine', id: 18 },

{ name: 'Rotavirus Vaccine', abbr: 'ROTA', value: 'Rotavirus Vaccine', id: 19 },

{ name: 'Recombinant Zoster Vaccine', abbr: 'RZV', value: 'Recombinant Zoster Vaccine', id: 20 },

{ name: 'Tetanus & Diphtheria Vaccine', abbr: 'Td', value: 'Tetanus & Diphtheria Vaccine', id: 21 },

{ name: 'Tetanus, Diphtheria & Acellular Pertussis Vaccine', abbr: 'Tdap', value: 'Tetanus, Diphtheria & Acellular Pertussis Vaccine', id: 22 },

{ name: 'Varicella Vaccine', abbr: 'VAR', value: 'Varicella Vaccine', id: 23 },

{ name: 'Varicella Zoster Virus', abbr: 'VZV', value: 'Varicella Zoster Virus', id: 24 },

{ name: 'Yellow Fever', abbr: 'YF', value: 'Yellow Fever', id: 25 },

{ name: 'Zoster Vaccine Live', abbr: 'ZVL', value: 'Zoster Vaccine Live', id: 26 },

],

// reaction to immunisation

reactionList: [

{ text: 'None', value: 'None'},

{ text: 'Pain', value: 'Pain'},

{ text: 'Swelling', value: 'Swelling'},

{ text: 'Redness', value: 'Redness'},

{ text: 'Fever', value: 'Fever'},

{ text: 'Irritability', value: 'Irritability'},

{ text: 'Malaise', value: 'Malaise'},

{ text: 'Systemic Symptoms', value: 'Systemic Symptoms'},

],

immunisationForm: {

immunisationSelected: null,

immunisationDate: null,

immunisationReaction: null,

immunisationDetails: null

},

}

},

validations: {

medicalForm: {

weight: { required, numeric },

height: { required, numeric },

pulse: { required, numeric, minLength: minLength(2), between: between(60, 120) },

systolic: { required, numeric, minLength: minLength(2), between: between(70, 190) },

diastolic: { required, numeric, minLength: minLength(2), between: between(40, 100) },

bloodGlucoseLevel: { required },

cholesterol: { required },

cholesterolLDL: { required, numeric },

cholesterolHDL: { required, numeric },

cholesterolTriglycerides: { required, numeric },

},

allergyForm: {

allergyName: { required },

selectedAllergyType: { required },

allergyDetails: { required },

selectedSeverity: { required },

},

conditionForm: {

conditionName: { required },

conditionDetails: { required },

conditionDate: { required }

},

immunisationForm: {

selectedImmunisation: { required },

immunisationDate: { required },

immunisationReaction: { required },

immunisationDetails: { required },

}

},

methods: {

storeID (id) {

// Store the ID of the record of which the action icon is selected

this.patientID = id

db.collection("users").doc(this.patientID).onSnapshot(doc => {

let patientInfo = doc.data()

patientInfo.id = doc.id

// Populate these values to display patient medical record fields that are currently holding info

this.gender = patientInfo.gender

this.selectedBlood = patientInfo.bloodType

this.medicalForm.weight = patientInfo.weight

this.medicalForm.height = patientInfo.height

this.medicalForm.pulse = patientInfo.pulse

this.medicalForm.systolic = patientInfo.systolic

this.medicalForm.diastolic = patientInfo.diastolic

this.medicalForm.bloodGlucoseLevel = patientInfo.bloodGlucoseLevel

this.medicalForm.cholesterol = patientInfo.cholesterol

this.medicalForm.cholesterolLDL = patientInfo.cholesterolLDL

this.medicalForm.cholesterolHDL = patientInfo.cholesterolHDL

this.medicalForm.cholesterolTriglycerides = patientInfo.cholesterolTriglycerides

// Populate the arrays with corresponding data from users record

this.allergies = patientInfo.allergy

this.conditions = patientInfo.condition

this.immunisations = patientInfo.immunisation

})

},

onBloodsChange(selectedBlood) {

// Access the users collection then update blood type of patient

db.collection("users").doc(this.patientID).get().then(() =>{

if(selectedBlood != null){

db.collection("users").doc(this.patientID).update({

bloodType: selectedBlood

})

}

})

},

onWeightChange (weight) {

// Check to see if entered data is dirty before updating the db

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.weight.$anyDirty

this.errors = this.$v.medicalForm.weight.$anyError

if(this.errors === false && this.formTouched === false){

db.collection("users").doc(this.patientID).get().then(() =>{

// Access the users collection then update weight of patient if not null

if(weight != null){

db.collection("users").doc(this.patientID).update({

weight: weight

})

}

})

}

},

onHeightChange (height) {

// Check to see if entered data is dirty before updating the db

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.height.$anyDirty

this.errors = this.$v.medicalForm.height.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update height of patient if not null

db.collection("users").doc(this.patientID).get().then(() =>{

if(height != null){

db.collection("users").doc(this.patientID).update({

height: height

})

}

})

}

},

onPulseChange (pulse) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.pulse.$anyDirty

this.errors = this.$v.medicalForm.pulse.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update pulse of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(pulse != null){

db.collection("users").doc(this.patientID).update({

pulse: pulse

})

}

})

}

},

onSystolicChange (systolic) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.systolic.$anyDirty

this.errors = this.$v.medicalForm.systolic.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update systolic of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(systolic != null){

db.collection("users").doc(this.patientID).update({

systolic: systolic

})

}

})

}

},

onDiastolicChange (diastolic) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.diastolic.$anyDirty

this.errors = this.$v.medicalForm.diastolic.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update diastolic of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(diastolic != null){

db.collection("users").doc(this.patientID).update({

diastolic: diastolic

})

}

})

}

},

onLevelChange (bloodGlucoseLevel) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.bloodGlucoseLevel.$anyDirty

this.errors = this.$v.medicalForm.bloodGlucoseLevel.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update bloodGlucoseLevel of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(bloodGlucoseLevel != null){

db.collection("users").doc(this.patientID).update({

bloodGlucoseLevel: bloodGlucoseLevel

})

}

})

}

},

onTotalCholChange (cholesterol) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.cholesterol.$anyDirty

this.errors = this.$v.medicalForm.cholesterol.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update cholesterol of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(cholesterol != null){

db.collection("users").doc(this.patientID).update({

cholesterol: cholesterol

})

}

})

}

},

onLDLChange (cholesterolLDL) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.cholesterolLDL.$anyDirty

this.errors = this.$v.medicalForm.cholesterolLDL.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update cholesterolLDL of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(cholesterolLDL != null){

db.collection("users").doc(this.patientID).update({

cholesterolLDL: cholesterolLDL

})

}

})

}

},

onHDLChange (cholesterolHDL) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.cholesterolHDL.$anyDirty

this.errors = this.$v.medicalForm.cholesterolHDL.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update cholesterolHDL of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(cholesterolHDL != null){

db.collection("users").doc(this.patientID).update({

cholesterolHDL: cholesterolHDL

})

}

})

}

},

onTriglyceridesChange (cholesterolTriglycerides) {

this.$v.$touch()

this.formTouched = !this.$v.medicalForm.cholesterolTriglycerides.$anyDirty

this.errors = this.$v.medicalForm.cholesterolTriglycerides.$anyError

if(this.errors === false && this.formTouched === false){

// Access the users collection then update cholesterolTriglycerides of patient

db.collection("users").doc(this.patientID).get().then(()=> {

if(cholesterolTriglycerides != null){

db.collection("users").doc(this.patientID).update({

cholesterolTriglycerides: cholesterolTriglycerides

})

}

})

}

},

close() {

this.$v.$reset()

this.dialog = false

},

cancel() {

this.snackbar = false

this.dialogAllergies = false

this.dialogConditions = false

this.dialogImmunisation = false

this.$v.$reset()

this.allergyForm.allergyName = null

this.allergyForm.selectedAllergyType = null

this.allergyForm.allergyDetails = null

this.allergyForm.selectedSeverity = null

this.conditionForm.conditionName = null

this.conditionForm.conditionDetails = null

this.conditionForm.conditionDate = null

this.immunisationForm.selectedImmunisation = null

this.immunisationForm.immunisationDate = null

this.immunisationForm.immunisationReaction = null

this.immunisationForm.immunisationDetails = null

},

saveAllergy () {

this.$v.$touch()

this.formTouched = !this.$v.allergyForm.$anyDirty

this.errors = this.$v.allergyForm.$anyError

if(this.errors === false && this.formTouched === false){

var addAllergy = {

allergyName: this.allergyForm.allergyName,

allergyType: this.allergyForm.selectedAllergyType,

allergyDetails: this.allergyForm.allergyDetails,

allergySeverity: this.allergyForm.selectedSeverity

}

var allergyRecord = {

allergy: fieldValue.arrayUnion(addAllergy)

}

this.triggerSnackbar("Allergy Was Successfully Added!", "success")

db.collection("users").doc(this.patientID).update(allergyRecord).then(() => {

// Clear the form values

this.$v.$reset()

this.allergyForm.allergyName = null

this.allergyForm.selectedAllergyType = null

this.allergyForm.allergyDetails = null

this.allergyForm.selectedSeverity = null

}).catch(error => {

console.log(error)

}).then(() => {

this.dialogAllergies = false

this.snackbar = false

})

}else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

},

deleteAllergy(allergy){

// Use arrayRemove to remove all instances of the record

db.collection("users").doc(this.patientID).update({

allergy: fieldValue.arrayRemove(allergy)

})

},

saveCondition () {

this.$v.$touch()

this.formTouched = !this.$v.conditionForm.$anyDirty

this.errors = this.$v.conditionForm.$anyError

if(this.errors === false && this.formTouched === false){

var addCondition = {

conditionName: this.conditionForm.conditionName,

conditionDetails: this.conditionForm.conditionDetails,

conditionDate: this.conditionForm.conditionDate,

}

var conditionRecord = {

condition: fieldValue.arrayUnion(addCondition)

}

this.triggerSnackbar("Condition Was Successfully Added!", "success")

db.collection("users").doc(this.patientID).update(conditionRecord).then(() => {

// Clear the form values

this.$v.$reset()

this.conditionForm.conditionName = null

this.conditionForm.conditionDetails = null

this.conditionForm.conditionDate = null

}).catch(error => {

console.log(error)

}).then(() =>{

this.dialogConditions = false

this.snackbar = false

})

}

else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

},

deleteCondition (condition) {

// Use arrayRemove to remove all instances of the record

db.collection("users").doc(this.patientID).update({

condition: fieldValue.arrayRemove(condition)

})

},

saveImmunisation () {

this.$v.$touch()

this.formTouched = !this.$v.immunisationForm.$anyDirty

this.errors = this.$v.immunisationForm.$anyError

if(this.errors === false && this.formTouched === false){

var addImmunisation = {

immunisationName: this.immunisationForm.selectedImmunisation,

immunisationDate: this.immunisationForm.immunisationDate,

immunisationsReaction: this.immunisationForm.immunisationReaction,

immunisationDetails: this.immunisationForm.immunisationDetails,

}

var immunisationRecord = {

immunisation: fieldValue.arrayUnion(addImmunisation)

}

this.triggerSnackbar("Immunisation Was Successfully Added!", "success")

db.collection("users").doc(this.patientID).update(immunisationRecord).then(() => {

this.$v.$reset()

this.immunisationForm.selectedImmunisation = null

this.immunisationForm.immunisationDate = null

this.immunisationForm.immunisationReaction = null

this.immunisationForm.immunisationDetails = null

}).catch(error => {

console.log(error)

}).then(() => {

this.dialogImmunisation = false

this.snackbar = false

})

}else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

},

deleteImmunisation (immunisation) {

// Use arrayRemove to remove all instances of the record

db.collection("users").doc(this.patientID).update({

immunisation: fieldValue.arrayRemove(immunisation)

})

},

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

getUsersDob () {

if(this.patientID != null){

db.collection("users").doc(this.patientID).get().then(doc => {

let patientRecord = doc.data()

patientRecord.id = doc.id

// Get DoB

this.dob = patientRecord.date

})

return this.dob

}

},

getTodaysDate () {

let today = new Date ()

today.setDate(today.getDate())

return today.toISOString()

},

customFilter (item, queryText) {

// convert any text or abreviation input

// to lower case in order to query it

const textOne = item.name.toLowerCase()

const textTwo = item.abbr.toLowerCase()

const searchText = queryText.toLowerCase()

// return if any letters match an item

return textOne.indexOf(searchText) > -1 ||

textTwo.indexOf(searchText) > -1

},

},

};

</script>

## Allergy Component – Allergies.vue

<template>

<!-- ANY ALLERGIES -->

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogAllergies" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add Allergy

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveAllergy()">

<v-card-title class="primary lighten-1 white--text">Allergy

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancelAllergy()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-text-field

label="Allergen Name\*"

v-model.trim="allergyForm.allergyName"

:counter="25"

outlined

required

:error-messages="allergyNameErrors"

@input="$v.allergyForm.allergyName.$touch()"

@blur="$v.allergyForm.allergyName.$touch()"

></v-text-field>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-select

v-model="allergyForm.selectedAllergyType"

:items="allergyList"

label="Allergy Type\*"

outlined

required

:error-messages="allergyTypeErrors"

@input="$v.allergyForm.selectedAllergyType.$touch()"

@blur="$v.allergyForm.selectedAllergyType.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="1"

label="Details\*"

v-model="allergyForm.allergyDetails"

:counter="150"

:maxlength="150"

outlined

required

:error-messages="allergyDetailsErrors"

@input="$v.allergyForm.allergyDetails.$touch()"

@blur="$v.allergyForm.allergyDetails.$touch()"

></v-textarea>

</v-col>

<v-col cols="12" sm="6">

<v-select

v-model="allergyForm.selectedSeverity"

:items="severityList"

label="Severity\*"

outlined

required

:error-messages="allergySeverityErrors"

@input="$v.allergyForm.selectedSeverity.$touch()"

@blur="$v.allergyForm.selectedSeverity.$touch()"

></v-select>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

type="submit"

block class="primary white--text"

@click.prevent="saveAllergy()">

<span>Add Allergy</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

<v-snackbar

:color= "color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

> {{ snackbarText }}

</v-snackbar>

</v-card>

</v-dialog>

</v-card>

</v-col> <!-- allergies end -->

</template>

<script>

import {auth, db, fieldValue} from '../../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

},

computed: {

allergyNameErrors () {

const errors = []

if(!this.$v.allergyForm.allergyName.$dirty) return errors

!this.$v.allergyForm.allergyName.required && errors.push('An Allergy Name Is Required')

return errors

},

allergyTypeErrors () {

const errors = []

if(!this.$v.allergyForm.selectedAllergyType.$dirty) return errors

!this.$v.allergyForm.selectedAllergyType.required && errors.push('Select The Type Of Allergy')

return errors

},

allergyDetailsErrors () {

const errors = []

if(!this.$v.allergyForm.allergyDetails.$dirty) return errors

!this.$v.allergyForm.allergyDetails.required && errors.push('Include Some Details About Your Allergy')

return errors

},

allergySeverityErrors () {

const errors = []

if(!this.$v.allergyForm.selectedSeverity.$dirty) return errors

!this.$v.allergyForm.selectedSeverity.required && errors.push('Select The Severity Of Your Allergy')

return errors

},

},

data() {

return {

currentUser: null,

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

dialogAllergies: false,

allergyList: [

{ text: 'Drug Allergy', value: 'Drug Allergy'},

{ text: 'Food Allergy', value: 'Food Allergy'},

{ text: 'Pollen Allergy', value: 'Pollen Allergy'},

{ text: 'Insect Allergy', value: 'Insect Allergy'},

{ text: 'Latex Allergy', value: 'Latex Allergy'},

{ text: 'Other', value: 'Other'},

],

severityList: [

{ text: 'Mild', value: 'Mild'},

{ text: 'Moderate', value: 'Moderate'},

{ text: 'Severe', value: 'Severe'},

{ text: 'Life Threatening', value: 'Life Threatening'},

],

allergyForm: {

allergyName: null,

selectedAllergyType: null, // Value holds the allergy selected by the user

allergyDetails: null, // Value holds user details

selectedSeverity: null, // Value holds the severity selected by the user

},

}

},

validations: {

allergyForm: {

allergyName: { required },

selectedAllergyType: { required },

allergyDetails: { required },

selectedSeverity: { required },

},

},

methods: {

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

cancelAllergy() {

this.snackbar = false

this.dialogAllergies = false

this.$v.$reset()

this.allergyForm.allergyName = null

this.allergyForm.selectedAllergyType = null

this.allergyForm.allergyDetails = null

this.allergyForm.selectedSeverity = null

},

saveAllergy () {

this.$v.$touch()

this.formTouched = !this.$v.allergyForm.$anyDirty

this.errors = this.$v.allergyForm.$anyError

if(this.errors === false && this.formTouched === false){

var addAllergy = {

allergyName: this.allergyForm.allergyName,

allergyType: this.allergyForm.selectedAllergyType,

allergyDetails: this.allergyForm.allergyDetails,

allergySeverity: this.allergyForm.selectedSeverity

}

var allergyRecord = {

allergy: fieldValue.arrayUnion(addAllergy)

}

this.triggerSnackbar("Allergy Was Successfully Added!", "success")

db.collection("users").doc(this.currentUser).update(allergyRecord).then(() => {

// Clear the form values

this.$v.$reset()

this.allergyForm.allergyName = null

this.allergyForm.selectedAllergyType = null

this.allergyForm.allergyDetails = null

this.allergyForm.selectedSeverity = null

}).catch(error => {

console.log(error)

}).then(() => {

this.dialogAllergies = false

this.snackbar = false

})

}else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

},

},

}

</script>

## Condition Component – Conditions.vue

<template>

<!-- Any Conditions -->

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogCondition" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add Condition

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveCondition()">

<v-card-title class="primary lighten-1 white--text">Condition

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancelCondition()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-autocomplete

label="Search Condition Name"

v-model="conditionForm.conditionName"

:items="conditions"

item-text="name"

item-value="value"

outlined

:error-messages="conditionNameErrors"

@input="$v.conditionForm.conditionName.$touch()"

@blur="$v.conditionForm.conditionName.$touch()"

></v-autocomplete>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="1"

label="Details\*"

v-model.trim="conditionForm.conditionDetails"

:counter="150"

:maxlength="150"

outlined

required

:error-messages="conditionDetailsErrors"

@input="$v.conditionForm.conditionDetails.$touch()"

@blur="$v.conditionForm.conditionDetails.$touch()"

></v-textarea>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-menu

v-model="menu"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Date\*"

readonly

:value="conditionForm.conditionDate"

v-bind="attrs"

v-on="on"

required

outlined

:error-messages="conditionDateErrors"

@input="$v.conditionForm.conditionDate.$touch()"

@blur="$v.conditionForm.conditionDate.$touch()"

@click:clear="conditionForm.conditionDate = null"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

:min="getUsersDob()"

:max="getTodaysDate()"

v-model="conditionForm.conditionDate"

@input="menu = false"

>

</v-date-picker>

</v-menu>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

block class="primary white--text"

@click.prevent="saveCondition()">

<span>Add Condition</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

<v-snackbar

:color= "color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

> {{ snackbarText }}

</v-snackbar>

</v-card>

</v-dialog>

</v-card>

</v-col> <!-- conditions end -->

</template>

<script>

import {auth, db, fieldValue} from '../../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

},

computed: {

conditionNameErrors () {

const errors = []

if(!this.$v.conditionForm.conditionName.$dirty) return errors

!this.$v.conditionForm.conditionName.required && errors.push('A Condition Name Is Required')

return errors

},

conditionDetailsErrors () {

const errors = []

if(!this.$v.conditionForm.conditionDetails.$dirty) return errors

!this.$v.conditionForm.conditionDetails.required && errors.push('Include Some Details About Your Condition')

return errors

},

conditionDateErrors () {

const errors = []

if(!this.$v.conditionForm.conditionDate.$dirty) return errors

!this.$v.conditionForm.conditionDate.required && errors.push('Select Date Of Condition Diagnosis')

return errors

},

},

data() {

return {

currentUser: null,

dob: null,

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

menu: false,

dialogCondition: false,

// Condition Name Autocomplete Selection Field Name & Values

conditions: [

{ name: 'Alzheimer`s Disease', value: 'Alzheimer`s Disease'},

{ name: 'Anemia', value: 'Anemia'},

{ name: 'Anxiety Disorder', value: 'Anxiety Disorder'},

{ name: 'Asthma', value: 'Asthma'},

{ name: 'Atopic Dermatitis', value: 'Atopic Dermatitis'},

{ name: 'Atrial Fibrillation', value: 'Atrial Fibrillation'},

{ name: 'Attention Deficit Hyperactivity Disorder', value: 'Attention Deficit Hyperactivity Disorder'},

{ name: 'Autism or Asperger`s Syndrome', value: 'Autism or Asperger`s Syndrome'},

{ name: 'Bipolar Mood Disorder', value: 'Bipolar Mood Disorder'},

{ name: 'Bronchiectasis', value: 'Bronchiectasis'},

{ name: 'Cancer', value: 'Cancer'},

{ name: 'Cardiac Failure', value: 'Cardiac Failure'},

{ name: 'Cardiomyopathy', value: 'Cardiomyopathy'},

{ name: 'Cerebral Palsy', value: 'Cerebral Palsy'},

{ name: 'Cerebrovascular Disease', value: 'Cerebrovascular Disease'},

{ name: 'Chronic Fatigue Syndrome', value: 'Chronic Fatigue Syndrome'},

{ name: 'Chronic Hepatitis', value: 'Chronic Hepatitis'},

{ name: 'Chronic Kidney Disease', value: 'Chronic Kidney Disease'},

{ name: 'Chronic Obstructive Pulmonary Disorder', value: 'Chronic Obstructive Pulmonary Disorder'},

{ name: 'Chronic Renal Disease', value: 'Chronic Renal Disease'},

{ name: 'Coeliac Disease', value: 'Coeliac Disease'},

{ name: 'Complex Regional Pain Syndrome', value: 'Complex Regional Pain Syndrome'},

{ name: 'Coronary Artery Disease', value: 'Coronary Artery Disease'},

{ name: 'Crohn`s Disease', value: 'Crohn`s Disease'},

{ name: 'Deafness & Hearing Impairment', value: 'Deafness & Hearing Impairment'},

{ name: 'Dermatological Conditions', value: 'Dermatological Conditions'},

{ name: 'Diabetes Insipidus', value: 'Diabetes Insipidus'},

{ name: 'Diabetes Mellitus Types 1 & 2', value: 'Diabetes Mellitus Types 1 & 2'},

{ name: 'Dysrhythmias', value: 'Dysrhythmias'},

{ name: 'Ehlers-Danlos Syndrome', value: 'Ehlers-Danlos Syndrome'},

{ name: 'Endometriosis', value: 'Endometriosis'},

{ name: 'Epilepsy', value: 'Epilepsy'},

{ name: 'Fetal Alcohol Spectrum Disorder', value: 'Fetal Alcohol Spectrum Disorder'},

{ name: 'Fibromyalgia', value: 'Fibromyalgia'},

{ name: 'Gestational Diabetes', value: 'Gestational Diabetes'},

{ name: 'Glaucoma ', value: 'Glaucoma '},

{ name: 'Haemophilia', value: 'Haemophilia'},

{ name: 'Hashimoto`s Thyroiditis / Hashimoto`s Disease', value: 'Hashimoto`s Thyroiditis / Hashimoto`s Disease'},

{ name: 'HIV/AIDS', value: 'HIV/AIDS'},

{ name: 'Huntington`s Disease', value: 'Huntington`s Disease'},

{ name: 'Hyperlipidaemia', value: 'Hyperlipidaemia'},

{ name: 'Hypertension', value: 'Hypertension'},

{ name: 'Hypothyroidism', value: 'Hypothyroidism'},

{ name: 'Ischemic Cardiomyopathy', value: 'Ischemic Cardiomyopathy'},

{ name: 'Mental Illness', value: 'Mental Illness'},

{ name: 'Migraines', value: 'Migraines'},

{ name: 'Multiple Sclerosis', value: 'Multiple Sclerosis'},

{ name: 'Myasthenia Gravis', value: 'Myasthenia Gravis'},

{ name: 'Narcolepsy', value: 'Narcolepsy'},

{ name: 'Obesity', value: 'Obesity'},

{ name: 'Osteoarthritis', value: 'Osteoarthritis'},

{ name: 'Osteoporosis', value: 'Osteoporosis'},

{ name: 'Parkinson`s Disease', value: 'Parkinson`s Disease'},

{ name: 'Periodontal Disease', value: 'Periodontal Disease'},

{ name: 'Polycystic Ovary Syndrome', value: 'Polycystic Ovary Syndrome'},

{ name: 'Postural Orthostatic Tachycardia Syndrome', value: 'Postural Orthostatic Tachycardia Syndrome'},

{ name: 'Post-Vasectomy Pain Syndrome', value: 'Post-Vasectomy Pain Syndrome'},

{ name: 'Pre-Diabetes', value: 'Pre-Diabetes'},

{ name: 'Psoriasis', value: 'Psoriasis'},

{ name: 'Relapsing Polychondritis', value: 'Relapsing Polychondritis'},

{ name: 'Rheumatoid Arthritis', value: 'Rheumatoid Arthritis'},

{ name: 'Schizophrenia', value: 'Schizophrenia'},

{ name: 'Sleep Apnea', value: 'Sleep Apnea'},

{ name: 'Substance Abuse Disorders', value: 'Substance Abuse Disorders'},

{ name: 'Systemic Lupus Erythematosus', value: 'Systemic Lupus Erythematosus'},

{ name: 'Thyroid Disease', value: 'Thyroid Disease'},

{ name: 'Ulcerative Colitis', value: 'Ulcerative Colitis'},

],

conditionForm: {

condtionName: null,

conditionDetails: null,

conditionDate: null,

},

}

},

validations: {

conditionForm: {

conditionName: { required },

conditionDetails: { required },

conditionDate: { required }

},

},

methods: {

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

cancelCondition() {

this.snackbar = false

this.dialogCondition = false

this.$v.$reset()

this.conditionForm.conditionName = null

this.conditionForm.conditionDetails = null

this.conditionForm.conditionDate = null

},

getUsersDob () {

db.collection("users").doc(this.currentUser).onSnapshot(doc => {

let patientRecord = doc.data()

patientRecord.id = doc.id

// Get DoB

this.dob = patientRecord.date

})

return this.dob

},

getTodaysDate () {

let today = new Date ()

today.setDate(today.getDate())

return today.toISOString()

},

saveCondition () {

this.$v.$touch()

this.formTouched = !this.$v.conditionForm.$anyDirty

this.errors = this.$v.conditionForm.$anyError

if(this.errors === false && this.formTouched === false){

var addCondition = {

conditionName: this.conditionForm.conditionName,

conditionDetails: this.conditionForm.conditionDetails,

conditionDate: this.conditionForm.conditionDate,

}

var conditionRecord = {

condition: fieldValue.arrayUnion(addCondition)

}

this.triggerSnackbar("Condition Was Successfully Added!", "success")

db.collection("users").doc(this.currentUser).update(conditionRecord).then(() => {

// Clear the form values

this.$v.$reset()

this.conditionForm.conditionName = null

this.conditionForm.conditionDetails = null

this.conditionForm.conditionDate = null

}).catch(error => {

console.log(error)

}).then(() =>{

this.dialogCondition = false

this.snackbar = false

})

}

else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

},

}

}

</script>

## Immunisation Component – Immunisation.vue

<template>

<v-col cols="12" md="12" lg="12">

<v-card flat>

<v-dialog v-model="dialogImmunisation" persistent max-width="600px">

<template v-slot:activator="{ on, attrs }">

<v-card flat>

<v-card-title class="overline">

<v-spacer></v-spacer>

<v-btn

small

v-bind="attrs"

v-on="on"

><v-icon>mdi-plus</v-icon>

Add immunisation

</v-btn>

</v-card-title>

</v-card>

</template>

<v-card>

<v-form @submit.prevent="saveImmunisation()">

<v-card-title class="primary lighten-1 white--text">Immunisation

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancelImmunisation()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<v-col cols="12" sm="6" md="6">

<v-autocomplete

label="Search Immunisations\*"

v-model="immunisationForm.selectedImmunisation"

:items="immunisations"

:filter="customFilter"

item-text="name"

item-value="value"

persistent-hint

hint="Immunisation Name Or Abbreviation"

outlined

:error-messages="selectedImmunisationErrors"

@input="$v.immunisationForm.selectedImmunisation.$touch()"

@blur="$v.immunisationForm.selectedImmunisation.$touch()"

></v-autocomplete>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-menu

v-model="menu"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Date\*"

readonly

:value="immunisationForm.immunisationDate"

v-bind="attrs"

v-on="on"

required

outlined

:error-messages="immunisationDateErrors"

@input="$v.immunisationForm.immunisationDate.$touch()"

@blur="$v.immunisationForm.immunisationDate.$touch()"

@click:clear="immunisationForm.immunisationDate = null"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

:min="getUsersDob()"

:max="getTodaysDate()"

v-model="immunisationForm.immunisationDate"

@input="menu = false"

>

</v-date-picker>

</v-menu>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-select

label="Reaction"

v-model="immunisationForm.immunisationReaction"

:items="reactionList"

outlined

required

:error-messages="immunisationReactionErrors"

@input="$v.immunisationForm.immunisationReaction.$touch()"

@blur="$v.immunisationForm.immunisationReaction.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6">

<v-textarea

auto-grow

rows="2"

name="immunisationForm.immunisationDetails"

label="Additional Details"

v-model="immunisationForm.immunisationDetails"

:counter="150"

:maxlength="150"

required

outlined

:error-messages="additionalDetailsError"

@input="$v.immunisationForm.immunisationDetails.$touch()"

@blur="$v.immunisationForm.immunisationDetails.$touch()"

></v-textarea>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

type="submit"

block class="primary white--text"

@click.prevent="saveImmunisation()">

<span>Add Immunisation</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

<v-snackbar

:color= "color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

> {{ snackbarText }}

</v-snackbar>

</v-card>

</v-dialog>

</v-card>

</v-col>

</template>

<script>

import {auth, db, fieldValue} from '../../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

},

computed: {

selectedImmunisationErrors () {

const errors = []

if(!this.$v.immunisationForm.selectedImmunisation.$dirty) return errors

!this.$v.immunisationForm.selectedImmunisation.required && errors.push('Select An Immunisation')

return errors

},

immunisationDateErrors () {

const errors = []

if(!this.$v.immunisationForm.immunisationDate.$dirty) return errors

!this.$v.immunisationForm.immunisationDate.required && errors.push('Select Date Of Immunisation')

return errors

},

immunisationReactionErrors () {

const errors = []

if(!this.$v.immunisationForm.immunisationReaction.$dirty) return errors

!this.$v.immunisationForm.immunisationReaction.required && errors.push('Select Your Reaction To the Immunisation')

return errors

},

additionalDetailsError () {

const errors = []

if(!this.$v.immunisationForm.immunisationDetails.$dirty) return errors

!this.$v.immunisationForm.immunisationDetails.required && errors.push('Include Additional Details Of Immunisation')

return errors

},

},

data() {

return {

dialogImmunisation: false,

menu: false,

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

loading: false,

items: [],

search: null,

selectedImmunisation: null,

immunisations: [

{ name: 'Anthrax Vaccine Adsorbed', abbr: 'AVA', value: 'Anthrax Vaccine Adsorbed', id: 1 },

{ name: 'Bacille Calmette-Guérin (Tuberculosis) Vaccine', abbr: 'BCG', value: 'Bacille Calmette-Guérin (Tuberculosis) Vaccine', id: 2},

{ name: 'Diphtheria, Tetanus & Pertussis Vaccine', abbr: 'DTaP', value: 'Diphtheria, Tetanus & Pertussis / Whopping Cough Vaccine', id: 3},

{ name: 'Hepatitis A Vaccine', abbr: 'HepA', value: 'Hepatitis A Vaccine', id: 4 },

{ name: 'Hepatitis B Vaccine', abbr: 'HepB', value: 'Hepatitis B Vaccine', id: 5 },

{ name: 'Haemophilus Influenzae Type B', abbr: 'Hib', value: 'Haemophilus Influenzae Type B', id: 6 },

{ name: 'Human Papillomavirus', abbr: 'HPV', value: 'Human Papillomavirus', id: 7 },

{ name: 'Inactivated Poliovirus Vaccine', abbr: 'IPV', value: 'Inactivated Poliovirus Vaccine', id: 8 },

{ name: 'Inactivated Influenza Vaccine', abbr: 'IIV', value: 'Inactivated Influenza Vaccine', id: 9 },

{ name: 'Japanese Encephalitis', abbr: 'JE', value: 'Japanese Encephalitis', id: 10 },

{ name: 'Live, Attenuated Influenza Vaccine (Quadrivalent)', abbr: 'LAIV4 ', value: 'Live, Attenuated Influenza Vaccine (Quadrivalent)', id: 11 },

{ name: 'Meningococcal A, C, W, Y', abbr: 'MenACWY', value: 'Meningococcal A, C, W, Y', id: 12 },

{ name: 'Meningococcal B', abbr: 'MenB', value: 'Meningococcal B', id: 13 },

{ name: 'Measles, Mumps, & Rubella', abbr: 'MMR', value: 'Measles, Mumps, & Rubella', id: 14 },

{ name: 'Measles-Rubella Vaccine', abbr: 'MR', value: 'Measles-Rubella Vaccine', id: 15 },

{ name: 'Pneumococcal Conjugate Vaccine', abbr: 'PCV13', value: 'Pneumococcal Conjugate Vaccine', id: 16 },

{ name: 'Pneumococcal Polysaccharide Vaccine', abbr: 'PPSV23', value: 'Pneumococcal Polysaccharide Vaccine', id: 17 },

{ name: 'Recombinant Influenza Vaccine', abbr: 'RIV3', value: 'Recombinant Influenza Vaccine', id: 18 },

{ name: 'Rotavirus Vaccine', abbr: 'ROTA', value: 'Rotavirus Vaccine', id: 19 },

{ name: 'Recombinant Zoster Vaccine', abbr: 'RZV', value: 'Recombinant Zoster Vaccine', id: 20 },

{ name: 'Tetanus & Diphtheria Vaccine', abbr: 'Td', value: 'Tetanus & Diphtheria Vaccine', id: 21 },

{ name: 'Tetanus, Diphtheria & Acellular Pertussis Vaccine', abbr: 'Tdap', value: 'Tetanus, Diphtheria & Acellular Pertussis Vaccine', id: 22 },

{ name: 'Varicella Vaccine', abbr: 'VAR', value: 'Varicella Vaccine', id: 23 },

{ name: 'Varicella Zoster Virus', abbr: 'VZV', value: 'Varicella Zoster Virus', id: 24 },

{ name: 'Yellow Fever', abbr: 'YF', value: 'Yellow Fever', id: 25 },

{ name: 'Zoster Vaccine Live', abbr: 'ZVL', value: 'Zoster Vaccine Live', id: 26 },

],

reactionList: [

{ text: 'None', value: 'None'},

{ text: 'Pain', value: 'Pain'},

{ text: 'Swelling', value: 'Swelling'},

{ text: 'Redness', value: 'Redness'},

{ text: 'Fever', value: 'Fever'},

{ text: 'Irritability', value: 'Irritability'},

{ text: 'Malaise', value: 'Malaise'},

{ text: 'Systemic Symptoms', value: 'Systemic Symptoms'},

],

immunisationForm: {

immunisationSelected: null,

immunisationDate: null,

immunisationReaction: null,

immunisationDetails: null

},

}

},

validations: {

immunisationForm: {

selectedImmunisation: { required },

immunisationDate: { required },

immunisationReaction: { required },

immunisationDetails: { required },

}

},

methods: {

customFilter (item, queryText) {

const textOne = item.name.toLowerCase()

const textTwo = item.abbr.toLowerCase()

const searchText = queryText.toLowerCase()

return textOne.indexOf(searchText) > -1 ||

textTwo.indexOf(searchText) > -1

},

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

cancelImmunisation () {

this.snackbar = false

this.dialogImmunisation = false

this.$v.$reset()

this.immunisationForm.selectedImmunisation = null

this.immunisationForm.immunisationDate = null

this.immunisationForm.immunisationReaction = null

this.immunisationForm.immunisationDetails = null

},

getUsersDob () {

db.collection("users").doc(this.currentUser).onSnapshot(doc => {

let patientRecord = doc.data()

patientRecord.id = doc.id

// Get DoB

this.dob = patientRecord.date

})

return this.dob

},

getTodaysDate () {

let today = new Date ()

today.setDate(today.getDate())

return today.toISOString()

},

saveImmunisation () {

this.$v.$touch()

this.formTouched = !this.$v.immunisationForm.$anyDirty

this.errors = this.$v.immunisationForm.$anyError

if(this.errors === false && this.formTouched === false){

var addImmunisation = {

immunisationName: this.immunisationForm.selectedImmunisation,

immunisationDate: this.immunisationForm.immunisationDate,

immunisationsReaction: this.immunisationForm.immunisationReaction,

immunisationDetails: this.immunisationForm.immunisationDetails,

}

var immunisationRecord = {

immunisation: fieldValue.arrayUnion(addImmunisation)

}

this.triggerSnackbar("Immunisation Was Successfully Added!", "success")

db.collection("users").doc(this.currentUser).update(immunisationRecord).then(() => {

this.$v.$reset()

this.immunisationForm.selectedImmunisation = null

this.immunisationForm.immunisationDate = null

this.immunisationForm.immunisationReaction = null

this.immunisationForm.immunisationDetails = null

}).catch(error => {

console.log(error)

}).then(() => {

this.dialogImmunisation = false

this.snackbar = false

})

}else{

this.triggerSnackbar("There Are Errors Preventing You From Submitting This Form", "error")

}

}

},

}

</script>

# Track Medication

## Medication.vue

<template>

<v-container>

<Navbar />

<!-- Import the medication calendar component -->

<v-card id="Medication Calendar" flat>

<MedCalendar />

</v-card>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import MedCalendar from '../components/MedicationCalendar'

export default {

components: {

Navbar,

MedCalendar,

},

}

</script>

## Calendar Component – MedicationCalendar.vue

<template>

<v-row class="fill-height">

<v-col>

<!-- Contains the top toolbar with the today, prev, next btn options and a menu that switches the calendar view-->

<v-sheet height="64">

<v-toolbar flat>

<v-btn outlined color="primary" class="mr-2" @click="dialog = true">

Add Medication

</v-btn>

<v-btn outlined color="primary" @click="setToday">Today</v-btn>

<v-btn fab text small color="primary" @click="prev">

<v-icon small >mdi-chevron-left</v-icon>

</v-btn>

<v-btn fab text small color="primary" @click="next">

<v-icon small>mdi-chevron-right</v-icon>

</v-btn>

<v-toolbar-title v-if="$refs.calendar">{{ $refs.calendar.title }}</v-toolbar-title>

<v-spacer></v-spacer>

<v-menu bottom right>

<template v-slot:activator="{ on, attrs }">

<v-btn

outlined

color="primary"

v-bind="attrs"

v-on="on"

>

<span>{{ typeToLabel[type] }}</span>

<v-icon right>mdi-menu-down</v-icon>

</v-btn>

</template>

<v-list>

<v-list-item @click="type = 'day'">

<v-list-item-title>Day</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'week'">

<v-list-item-title>Week</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'month'">

<v-list-item-title>Month</v-list-item-title>

</v-list-item>

</v-list>

</v-menu>

</v-toolbar>

</v-sheet>

<!-- Dialog for adding medication -->

<v-dialog v-model="dialog" persistent max-width="600px">

<v-card>

<v-form @submit.prevent="saveMedication()">

<!-- <v-card-title class="primary lighten-1 white--text">Add Medication -->

<v-card-title class="primary white--text">Add Medication

<v-spacer></v-spacer>

<v-btn class="mr-6" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-card-text>

<v-container>

<v-row>

<!-- Name -->

<v-col cols="12" sm="5" md="5">

<v-text-field

label="Medication Name\*"

v-model="medicationName"

type="text"

outlined

:error-messages="medicationNameError"

@input="$v.medicationName.$touch()"

@blur="$v.medicationName.$touch()"

></v-text-field>

</v-col>

<!-- Dose Amount -->

<v-col cols="12" sm="3" md="3">

<v-text-field

label="Dosage\*"

type="number"

v-model="dose"

outlined

:error-messages="doseError"

@input="$v.dose.$touch()"

@blur="$v.dose.$touch()"

></v-text-field>

</v-col>

<!-- Select Type -->

<v-col cols="12" sm="4" md="4">

<v-select

label="Unit\*"

v-model="dosageUnit"

:items="units"

outlined

:error-messages="dosageUnitError"

@input="$v.dosageUnit.$touch()"

@blur="$v.dosageUnit.$touch()"

></v-select>

</v-col>

<!-- Select Time -->

<v-col cols="12" sm="6" md="6">

<v-menu

ref="menu"

v-model="menu"

:close-on-content-click="false"

:nudge-right="40"

:return-value.sync="time"

transition="scale-transition"

offset-y

max-width="290px"

min-width="290px"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

v-model="time"

label="Dose Time\*"

prepend-icon="access\_time"

readonly

v-bind="attrs"

v-on="on"

outlined

:error-messages="timeError"

@input="$v.time.$touch()"

@blur="$v.time.$touch()"

></v-text-field>

</template>

<v-time-picker

v-if="menu"

v-model="time"

full-width

@click:minute="$refs.menu.save(time)"

></v-time-picker>

</v-menu>

</v-col>

<!-- Date Picker -->

<v-col cols="12" sm="6" md="6">

<v-menu

v-model="menu2"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Day\*"

readonly

:value="medDates"

v-bind="attrs"

v-on="on"

required

outlined

:error-messages="medDatesError"

@input="$v.medDates.$touch()"

@blur="$v.medDates.$touch()"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

:min="getTodaysDate()"

:max="getFutureDate()"

v-model="medDates"

@input="menu2 = false"

>

</v-date-picker>

</v-menu>

</v-col>

<!-- Additional Details -->

<v-col cols="12" sm="6" md="6">

<v-textarea

label="Additional Detail"

v-model="medicationDetails"

auto-grow

rows="1"

outlined

:error-messages="medicationDetailsError"

@input="$v.medicationDetails.$touch()"

@blur="$v.medicationDetails.$touch()"

></v-textarea>

</v-col>

<!-- Select a Colour -->

<v-col cols="12" sm="6" md="6">

<v-select

label="Choose a Colour\*"

v-model="picker"

:items="colourList"

outlined

:error-messages="colourError"

@input="$v.picker.$touch()"

@blur="$v.picker.$touch()"

></v-select>

</v-col>

</v-row>

</v-container>

</v-card-text>

<v-card-actions>

<v-col cols="12" class="mt-n6">

<v-btn

block class="primary white--text"

@click.prevent="saveMedication()">

<span>Add Medication</span>

</v-btn>

</v-col>

</v-card-actions>

</v-form>

</v-card>

</v-dialog>

<v-sheet height="500">

<!-- Calendar is inserted here -->

<v-calendar

ref="calendar"

color="primary"

:type="type"

v-model="focus"

:events="events"

@click:date="viewDay"

@click:event="showEvent"

>

</v-calendar>

<v-menu

v-model="selectedOpen"

:close-on-content-click="false"

:activator="selectedElement"

offset-x

>

<v-card color="grey lighten-4" min-width="350px" flat>

<v-toolbar :color="selectedEvent.color" dark>

<v-toolbar-title v-html="selectedEvent.name"></v-toolbar-title>

<v-spacer></v-spacer>

<v-btn icon @click="deleteEvent(selectedEvent)">

<v-icon>fa-trash</v-icon>

</v-btn>

<v-btn

text

color="white"

@click="selectedOpen = false"

>

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-toolbar>

<v-card-text>

<v-row class="mt-n8">

<v-col cols="12" md="12" class="mt-4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Medication Details</v-list-item-title>

<h3 v-html="selectedEvent.details"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" md="12" class="mt-4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-syringe</v-icon>

Dose</v-list-item-title>

<h3 v-html="selectedEvent.dose"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" md="12" class="mt-4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-pills</v-icon>

Dosage Unit</v-list-item-title>

<h3 v-html="selectedEvent.dosageUnit"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-menu>

</v-sheet>

</v-col>

</v-row>

</template>

<script>

import { auth, db, fieldValue } from '../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

this.getMedication() // Calls the getMedication method to populate cal

},

computed: {

medicationNameError () {

const errors = []

if(!this.$v.medicationName.$dirty) return errors

!this.$v.medicationName.required && errors.push('Please Select A Medication Name')

return errors

},

doseError () {

const errors = []

if(!this.$v.dose.$dirty) return errors

!this.$v.dose.required && errors.push('Dose Required')

return errors

},

dosageUnitError () {

const errors = []

if(!this.$v.dosageUnit.$dirty) return errors

!this.$v.dosageUnit.required && errors.push('Dosage Unit Required')

return errors

},

timeError () {

const errors = []

if(!this.$v.time.$dirty) return errors

!this.$v.time.required && errors.push('Dose Time Required')

return errors

},

medDatesError () {

const errors = []

if(!this.$v.medDates.$dirty) return errors

!this.$v.medDates.required && errors.push('Dose Date Required')

return errors

},

medicationDetailsError () {

const errors = []

if(!this.$v.medicationDetails.$dirty) return errors

!this.$v.medicationDetails.required && errors.push('Dose Details Required')

return errors

},

colourError () {

const errors = []

if(!this.$v.picker.$dirty) return errors

!this.$v.picker.required && errors.push('Colour is Required')

return errors

},

},

data() {

return {

currentUser: null,

focus: '',

type: 'month',

typeToLabel: {

month: 'Month',

week: 'Week',

day: 'Day',

},

// selected event on the calendar

selectedEvent: {},

selectedElement: null,

selectedOpen: false,

events: [],

dialog: false, // add medication

menu: false, // time picker

medicationName: null,

dose: null,

dosageUnit: null,

units: [

{ text: 'Tablet(s)', value: 'tablet' },

{ text: 'Capsule(s)', value: 'capsule' },

{ text: 'Injection(s)', value: 'injection' },

{ text: 'Drop(s)', value: 'drop' },

{ text: 'Spray(s)', value: 'spray' },

{ text: 'Suppository(ies)', value: 'suppository' },

{ text: 'Tablespoon(s)', value: 'tbs' },

{ text: 'Teaspoon(s)', value: 'tsp' },

{ text: 'Milligram(s)', value: 'milligram' },

{ text: 'Milliliter(s)', value: 'milliliter' },

],

time: null,

menu2: false, // date-picker

medDates: null, // Stored date from date=picker

medicationDetails: null,

picker: null, // Stored colour

colourList: [

{ text: "Red", value: "red"},

{ text: "Yelllow", value: "yellow"},

{ text: "Green", value: "green"},

{ text: "Blue", value: "blue"},

{ text: "Purple", value: "purple"},

{ text: "Pink", value: "pink"},

{ text: "Orange", value: "orange"},

{ text: "Cyan", value: "cyan"},

{ text: "Indigo", value: "indigo"},

],

}

},

validations: {

medicationName: { required },

dose: { required },

dosageUnit: { required },

time: { required },

medDates: { required },

medicationDetails: { required },

picker: { required },

},

methods: {

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

getTodaysDate () {

let today = new Date()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

return formattedDate

},

getFutureDate () {

let latest = new Date ()

latest.setMonth(latest.getMonth() + 1)

return latest.toISOString()

},

// Load in medication entries

getMedication () {

db.collection("users").doc(this.currentUser).onSnapshot(snap => {

let medication = snap.data().medication

// so long as there are medication entried in the array in users collection

if(medication != null){

this.events = [] // clear on each check

medication.forEach(medication => {

// Set the values to pass to the events

let time = medication.startTime.replace(".", ":")

let event = {

name: medication.medication,

details: medication.details,

dose: medication.dose,

dosageUnit: medication.doseUnit,

start: medication.dateTaken + " " + time,

end: medication.dateTaken + " " + time,

color: medication.color,

}

this.events.push(event)

})

}

})

},

cancel () {

this.dialog = false

this.clearForms()

},

clearForms() {

this.$v.$reset()

this.medicationName = null

this.dose = null

this.dosageUnit = null

this.time = null

this.medDates= null

this.medicationDetails = null

this.picker = null

},

// Delete a selected event by removing it from array

deleteEvent (event) {

db.collection("users").doc(this.currentUser).update({

medication: fieldValue.arrayRemove(this.eventToRecord(event))

})

this.selectedOpen = false

},

// Every value must be passed back into its original form

// and will only be deleted if values are identical

eventToRecord (calendarEvent) {

let datetime = calendarEvent.start.split(" ")

let date = datetime[0] // Split datatime

let time = datetime[1] // to two individual values

// must match perfectly to object

let record = {

medication: calendarEvent.name,

details: calendarEvent.details,

dose: calendarEvent.dose,

doseUnit: calendarEvent.dosageUnit,

dateTaken: date,

startTime: time,

color: calendarEvent.color

}

return record

},

saveMedication () {

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

var addMedication = {

medication: this.medicationName,

dose: this.dose,

doseUnit: this.dosageUnit,

startTime: this.time,

dateTaken: this.medDates,

details: this.medicationDetails,

color: this.picker,

}

var medicationRecord = {

medication: fieldValue.arrayUnion(addMedication)

}

// Update the array with the new medication entry

db.collection("users").doc(this.currentUser).update(medicationRecord).then(() => {

// Clear the form values

this.clearForms()

}).then(() => {

console.log("Submitted!")

this.dialog = false

}).catch(error => {

console.log("Med Add ", error)

})

}

else{

console.log("NoG")

}

},

viewDay ({ date }) {

this.focus = date

this.type = 'day'

},

setToday () {

this.focus = ''

},

prev () {

this.$refs.calendar.prev()

},

next () {

this.$refs.calendar.next()

},

showEvent ({ nativeEvent, event }) {

const open = () => {

this.selectedEvent = event

this.selectedElement = nativeEvent.target

setTimeout(() => this.selectedOpen = true, 10)

}

if (this.selectedOpen) {

this.selectedOpen = false

setTimeout(open, 10)

} else {

open()

}

nativeEvent.stopPropagation()

},

},

}

</script>

# Appointments

## Appointments.vue

<template>

<v-container >

<Navbar />

<v-row><v-col cols="12" s="12" sm="12" md="12" lg="12">

<!-- V-Dialog is used to encompass the view page so that when the appointment add button is

pressed a full dialog with the form opens -->

<v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">

<template v-slot:activator="{ on, attrs }">

<!-- Viewing of appointments occurs here -->

<v-card flat>

<v-card-title class="primary lighten-1 white--text">Appointments

<v-spacer></v-spacer>

<v-btn

class="mx-2"

fab small

dark color="primary darken-1"

v-bind="attrs"

v-on="on"

>

<v-icon>fa-plus</v-icon>

</v-btn>

</v-card-title>

<!-- View Appointments Component is imported here, makes it easier to keep the code tidier -->

<ViewAppointments />

</v-card>

</template>

<!-- Book appointment starts here -->

<v-card>

<v-card-title class="primary white--text">

Book Appointment

<v-spacer></v-spacer>

<v-btn class="mr-4" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-divider class="mx4"></v-divider>

<v-form @submit.prevent="bookAppointment">

<v-card-text>

<v-row>

<v-expansion-panels flat accordion v-model="panel">

<!-- Personal Details Field -->

<v-expansion-panel>

<v-expansion-panel-header>

<v-subheader class="overline ml-n5">Personal Details

<v-divider></v-divider>

</v-subheader>

</v-expansion-panel-header>

<v-expansion-panel-content>

<v-row>

<!-- Firstname Field -->

<v-col class="mt-n0" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="firstname"

label="First Name"

v-model.trim="firstname"

readonly

outlined

></v-text-field>

</v-col>

<!-- Surname Field -->

<v-col class="mt-n0" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="surname"

label="Surname"

v-model.trim="surname"

readonly

outlined

></v-text-field>

</v-col>

<!-- Date of Birth Field -->

<v-col class="mt-n0" cols="12" md="4" lg="4">

<v-text-field

label="Date of Birth"

:value="formattedDate"

readonly

outlined

></v-text-field>

</v-col>

<!-- Email Field -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-text-field

type="email"

name="email"

label="Email"

v-model.trim="email"

readonly

outlined

></v-text-field>

</v-col>

<!-- PPSN Field -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="ppsn"

label="PPSN"

v-model.trim="ppsn"

readonly

outlined

></v-text-field>

</v-col>

<!-- Mobile Field -->

<v-col class="mt-n2" cols="12" md="4" lg="4">

<v-text-field

type="text"

name="mobile"

label="Mobile"

v-model.trim="mobile"

readonly

outlined

></v-text-field>

</v-col>

</v-row>

</v-expansion-panel-content>

</v-expansion-panel>

<!-- Account Details -->

<v-expansion-panel>

<v-expansion-panel-header>

<v-subheader class="overline ml-n5">Appointment Details

<v-divider></v-divider>

</v-subheader>

</v-expansion-panel-header>

<v-expansion-panel-content>

<v-row>

<!-- Appointment Type Field -->

<v-col>

<v-select

label="Select Appointment Type"

v-model="appointmentSelected"

:items="appointmentTypes"

outlined

:error-messages="selectTypeError"

@input="$v.appointmentSelected.$touch()"

@blur="$v.appointmentSelected.$touch()"

></v-select>

</v-col>

<!-- Appointment Doctor Field -->

<v-col class="mt-n0" cols="12" md="6">

<v-select

name="selectDoctor"

v-model="chosenDoc"

:items="doctors"

item-text="name"

item-value="value"

label="Select a Doctor"

outlined

:error-messages="selectDocError"

@input="$v.chosenDoc.$touch()"

@blur="$v.chosenDoc.$touch()"

@change="onDropdownChanged($event)"

></v-select>

</v-col>

<!-- Appointment Date Field -->

<v-col class="mt-n0" cols="12" md="6">

<v-menu

v-model="menu2"

:close-on-content-click="false"

max-width="290"

>

<template v-slot:activator="{ on, attrs }">

<v-text-field

label="Select Date"

:error-messages="selectDateError"

readonly

:value="formattedAppointmentDate"

v-bind="attrs"

v-on="on"

required

outlined

@input="$v.appointmentDate.$touch()"

@blur="$v.appointmentDate.$touch()"

@click:clear="appointmentDate = null"

></v-text-field>

</template>

<v-date-picker

full-width

scrollable

show-current

v-model="appointmentDate"

@input="menu2 = false"

@change="onDateChanged()"

:min="getTomorrowsDate()"

:max="getLatestDate()"

>

</v-date-picker>

</v-menu>

</v-col>

<!-- Appointment Details Field -->

<v-col cols="12" md="6">

<v-textarea

auto-grow

rows="1"

name="additionalDetails"

label="Additional Details"

:error-messages="additionalDetailsError"

v-model="additionalDetails"

:counter="150"

:maxlength="150"

required

outlined

@input="$v.additionalDetails.$touch()"

@blur="$v.additionalDetails.$touch()"

></v-textarea>

</v-col>

<!-- Appointment Time Field -->

<v-col class="mt-0" cols="12" md="12">

<v-row>

<v-card flat v-if="showSelectTime">

<v-card-text class="mt-n4">

<span class="subheading">Select Appointment Time</span>

<v-chip-group

column

v-model="selectedTime"

mandatory

active-class="primary--text"

>

<!-- Displays times available -->

<v-chip class="mr-5" outlined default v-for="time in displayedTimeSlots" :key="time" :value="time">

{{ time }}

</v-chip>

</v-chip-group>

</v-card-text>

</v-card>

</v-row>

</v-col>

</v-row>

</v-expansion-panel-content>

</v-expansion-panel>

</v-expansion-panels>

</v-row>

<v-card-actions>

<v-row>

<v-btn

type="submit"

:disabled="loading"

block class="primary white--text"

@click.prevent="bookAppointment()">

<span>Book Appointment</span>

</v-btn>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-row>

</v-card-actions>

</v-card-text>

</v-form>

</v-card> <!-- End of book appointment form within dialog -->

</v-dialog>

</v-col></v-row>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import ViewAppointments from "../components/Appointments/ViewAppointments"

import { mapState } from 'vuex'

import { auth, db } from '../firebase'

import { format, parseISO } from 'date-fns'

import { required, maxLength } from "vuelidate/lib/validators"

export default {

components: {

Navbar,

ViewAppointments,

},

computed: {

...mapState(['userProfile']),

loading () {

return this.$store.state.loading

},

// For b-day format

formattedDate () {

return this.date ? format(parseISO(this.date), 'do MMM yyyy') : ''

},

// More friendly date format display

formattedAppointmentDate () {

return this.appointmentDate ? format(parseISO(this.appointmentDate), 'do MMM yyyy') : ''

},

// VALIDATION ERROR MESSAGES

selectTypeError () {

const errors = []

if(!this.$v.appointmentSelected.$dirty) return errors

!this.$v.appointmentSelected.required && errors.push('Please Select An Appointment Type')

return errors

},

selectDocError () {

const errors = []

if(!this.$v.chosenDoc.$dirty) return errors

!this.$v.chosenDoc.required && errors.push('Please Select A Doctor')

return errors

},

selectDateError () {

const errors = []

if(!this.$v.appointmentDate.$dirty) return errors

!this.$v.appointmentDate.required && errors.push('Please Select A Date For Your Appointment')

return errors

},

additionalDetailsError () {

const errors = []

if(!this.$v.additionalDetails.$dirty) return errors

!this.$v.additionalDetails.maxLength && errors.push('Cannot exceed 150 characters')

!this.$v.additionalDetails.required && errors.push('Please provide us with the details of your appointment')

return errors

},

},

data () {

return {

// Snackbar properties

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

dialog: false, // Dialog for overall dialog panel

panel: 1, // Appointment panel is open by default

menu2: false, // Menu for picking the appointment date

// Personal Details, pre-populated

firstname: this.$store.state.userProfile.firstname,

surname: this.$store.state.userProfile.surname,

date: this.$store.state.userProfile.date,

email: this.$store.state.userProfile.email,

ppsn: this.$store.state.userProfile.ppsn,

mobile: this.$store.state.userProfile.mobile,

appointmentSelected: null, // Stores the value selected for the Appointment Type

// All appointment type options

appointmentTypes: [

{ text: 'Clinic Consultation', value: 'Clinic'},

{ text: 'Online Consultation', value: 'Online'},

],

currentUser: null, // currentUser is used for the patientID

doctors: [], // Doctors array contains a list of doctors

chosenDoc: null, // Chosen Doc stores the value of selected doctor, derived of onDropdownChanged()

appointmentDate: null,

additionalDetails: '',

showSelectTime: false, // Used to hide the times until the doctor and date are selected

selectedTime: null, // Time that is selected on the form

// All timeslots available to begin with, queried later to determine which show

timeSlots: [

'09.00', '09.30', '10.00', '10.30', '11.00', '11.30', '13.00',

'13.30', '14.00', '14.30', '15.00', '15.30', '16.00', '16.30'

],

displayedTimeSlots: [], // Returns the queried array of available times to the patient

}

},

// Validations for form values

validations: {

chosenDoc: { required },

appointmentSelected: { required },

appointmentDate: { required },

additionalDetails: { required, maxLength: maxLength (150) }

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

// This query first gets the list of users whos role is doctor from the roles collection

db.collection("roles").where("role.doctor", "==", true).get().then(snap => {

// For each doctor document store the doc.id, which is the same ID used for the users colection doc.id, into doctorID

snap.forEach(doc => {

var doctorID = doc.id

// Then query the users collection with the doctorID to find corresponding record based on same ID

// so that the v-select for doctor can access the name of the doctor

db.collection("users").doc(doctorID).get().then(doc => {

var docID = doc.id

var doctorFirstName = doc.data().firstname

var doctorSurname = doc.data().surname

this.doctors.push({

name: doctorFirstName + " " + doctorSurname,

value: docID

})

})

})

})

},

methods: {

// When the cancel button is pressed, close the dialog, reset any validation errors and rest any input fields

cancel () {

this.dialog = false

this.$v.$reset()

this.appointmentSelected = null

this.chosenDoc = null

this.appointmentDate = null

this.additionalDetails = null

this.selectedTime = null

this.snackbar = null

},

// Get tomorrrows date so that the calendar can be limited and users can only book an appointment from tomorrow onwards

getTomorrowsDate () {

let tomorrow = new Date ()

tomorrow.setDate(tomorrow.getDate() + 1)

return tomorrow.toISOString()

},

// Get a date that is 3 months into the future from todays date to limit calendar

getLatestDate () {

let latest = new Date ()

latest.setMonth(latest.getMonth() + 3)

return latest.toISOString()

},

// This method is to get the appointment times that are available and to prevent double booking

getAvailableTimes () {

// Check if appointment date and chosen doctor isn't null

if(this.appointmentDate != null && this.chosenDoc != null) {

// Query the appointments collection to check for appointments where the date and doctor are the same

db.collection("appointments").where("appointmentDate", "==", this.appointmentDate).where("doctorID", "==", this.chosenDoc).get().then(snap => {

// Create an array of appointments to convert firebase doc to a standard doc

var appointments = []

snap.forEach(doc => appointments.push(doc.data()))

// set displayedtimeslots to the filtered version of timeslots, where any appointments that are pre existing are filtered out

this.displayedTimeSlots = this.timeSlots.filter(time => {

return appointments.find(item => item.appointmentTime == time) == null

})

if(this.displayedTimeSlots.length == 0){

this.triggerSnackbar("There are no times available for this date!", "error") // Display if there are no times left

}

else{

this.showSelectTime = true // Then display the times available to the user

}

})

}else {

this.showSelectTime = false // Times are hidden if there are no times available

}

},

// If the user selects a date and then changes it the appointments must be requeried

onDateChanged () {

this.showSelectTime = false

db.collection("appointments").where("appointmentDate", "==", this.appointmentDate)

.where("patientID", "==", this.currentUser).get().then(snap => {

if(snap.docs.length == 0){

// If the user doesn't already have an appointment on this date, then get available times for that day

this.getAvailableTimes()

}

else{

this.showSelectTime = false

this.triggerSnackbar("You have already booked an appointment on this day!", "error")

}

})

},

// If the user selects a doctor and then changes doctors during the form process, appointments must be requeried to check if

// date has already been booked on

onDropdownChanged(value) {

db.collection("appointments").where("appointmentDate", "==", this.appointmentDate)

.where("patientID", "==", this.currentUser).get().then(snap => {

if(snap.docs.length == 0){

// If no prior appointments on this day, restore chosenDoc value and get available times for that day

this.showSelectTime = false

this.chosenDoc = value

this.getAvailableTimes()

}

else{

// If date booked on, error and don't show times

this.triggerSnackbar("You have already booked an appointment on this day!", "error")

this.showSelectTime = false

}

})

},

// Triggers the snackbar with the passed message and colour of the message

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// This is the book appointment method, which fires when the book appointment button is pressed

bookAppointment () {

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

// Access the appointments collection, and where the date is equal to chosen appointment date and the patient is the user

db.collection("appointments").where("appointmentDate", "==", this.appointmentDate)

.where("patientID", "==", this.currentUser).get().then(snap => {

// Check if the date has already got an appointment for this user, if it doesm't proceed

if(snap.docs.length == 0){

// Store the form data

var document = {

patientID: this.currentUser,

firstname: this.$store.state.userProfile.firstname,

surname: this.$store.state.userProfile.surname,

dob: this.$store.state.userProfile.date,

email: this.$store.state.userProfile.email,

ppsn: this.$store.state.userProfile.ppsn,

mobile: this.$store.state.userProfile.mobile,

doctorID: this.chosenDoc,

appointmentType: this.appointmentSelected,

appointmentDate: this.appointmentDate,

appointmentTime: this.selectedTime,

appointmentDetails: this.additionalDetails

}

// There was already an appointment on this day

}else{

this.triggerSnackbar("You Have Already Booked An Appointment On This Day!", "error")

this.showSelectTime = false

}

// The appointment is successfully stored, passing the var document details and setting them

// in a new record in the appointments collection

this.triggerSnackbar("Appointment Was Successfully Booked!", "success")

db.collection("appointments").doc().set(document).then(() => {

// Reset any form error messages and inputs upon completion of booking

this.$v.$reset()

this.appointmentSelected = null

this.chosenDoc = null

this.appointmentDate = null

this.additionalDetails = null

this.selectedTime = null

}).then(() => {

// Select times must be hidden and the book appointment dialog closed

this.showSelectTime = !this.showSelectTime

this.dialog = false

})

}

)}

else{

this.triggerSnackbar("There Are Error In This Form", "error")

}

}

}

}

</script>

## View Component - ViewAppointments.vue

<template>

<v-row><v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-tabs v-model="tab">

<!-- Cycles through tab names -->

<v-tab v-for="tab in tabs" :key="tab.tabName">

{{ tab.tabName }}

</v-tab>

</v-tabs>

<v-tabs-items v-model="tab">

<!-- UPCOMING APPOINTMENTS FILTER HERE -->

<v-tab-item>

<!-- For each appointment print out an appointment card with data corresponding to the appointmentID -->

<v-card outlined class="mt-3" v-for="appointment in appointments" :key="appointment.id">

<v-card-title class="primary lighten-2 white--text">

<v-spacer></v-spacer>

<!-- Cancel button for the appointments -->

<v-icon right @click="toggleCancelAppointment(appointment.id)">fa-trash</v-icon>

</v-card-title>

<v-card-text class="mt-6">

<v-row class="mt-n8">

<v-col cols="12" md="4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-id-card</v-icon>

Appointment ID</v-list-item-title>

<h3>{{ appointment.id }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clinic-medical</v-icon>

Type</v-list-item-title>

<h3>{{ appointment.appointmentType }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-user-md</v-icon>

Doctor</v-list-item-title>

<h3>{{ appointment.doctorName }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-calendar</v-icon>

Date</v-list-item-title>

<h3>{{ appointment.appointmentDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clock</v-icon>

Time</v-list-item-title>

<h3>{{ appointment.appointmentTime }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Details</v-list-item-title>

<h3>{{ appointment.appointmentDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-tab-item>

<!-- OLD APPOINTMENTS FILTER HERE -->

<v-tab-item>

<!-- For each old appointment print out an appointment card with data corresponding to the appointmentID -->

<v-card class="mt-3" v-for="appointment in oldAppointments" :key="appointment.id" outlined>

<v-card-title class="primary lighten-2 white--text">

<v-spacer></v-spacer>

<!-- Cancel button for the appointments -->

<v-icon right @click="toggleCancelAppointment(appointment.id)">fa-trash</v-icon>

</v-card-title>

<v-card-text class="mt-6">

<v-row class="mt-n8">

<v-col cols="12" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-id-card</v-icon>

Appointment ID</v-list-item-title>

<h3>{{ appointment.id }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clinic-medical</v-icon>

Type</v-list-item-title>

<h3>{{ appointment.appointmentType }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-user-md</v-icon>

Doctor</v-list-item-title>

<h3>{{ appointment.doctorName }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-calendar</v-icon>

Date</v-list-item-title>

<h3>{{ appointment.appointmentDate }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clock</v-icon>

Time</v-list-item-title>

<h3>{{ appointment.appointmentTime }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Details</v-list-item-title>

<h3>{{ appointment.appointmentDetails }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-tab-item>

</v-tabs-items>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-col></v-row>

</template>

<script> // Importing; mapState for profile details to populate personal details, auth and db for queries

import { mapState } from 'vuex'

import {auth, db} from '../../firebase'

export default {

computed: {

...mapState(['userProfile']),

},

data() {

return {

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

// Tabs for upcoming and past

tab: null,

tabs: [

{ tabName: 'Upcoming' },

{ tabName: 'Past' },

],

// every appointment is stored here so they can then be called by appointment.propertyName

appointments: [],

oldAppointments: [],

}

},

created() {

var currentDate = new Date() //Get the current date

//date.get year returns the amount of years since 1900, using getFullYear instead

var currentDateStart = new Date(currentDate.getFullYear(), currentDate.getMonth(), currentDate.getDate());

var currentUser = auth.currentUser.uid // Set current user to the currently logged in user

// Query the db to identify any appointments connected to that user, order by date and then time, onSnapshot listens for real time updates

db.collection("appointments").where("patientID", "==", currentUser).orderBy("appointmentDate").orderBy("appointmentTime").onSnapshot(snap => {

let appointment = snap.docChanges()

appointment.forEach(async (appointment) => {

// realtime add document to appointments list

if(appointment.type == "added"){

let booked = appointment.doc.data()

booked.id = appointment.doc.id

// Get the doctors name by querying the users collection with the doctorID from appointments collection

let doctorSnapshot = await db.collection("users").doc(booked.doctorID).get();

doctorSnapshot = doctorSnapshot.data();

booked.doctorName = doctorSnapshot.firstname + ' ' + doctorSnapshot.surname;

// Checking booked time against current time

var checkAppointmentDates = new Date(booked.appointmentDate)

if(checkAppointmentDates < currentDateStart){

this.oldAppointments.push(booked)

}

else{

this.appointments.push(booked)

}

}

else if(appointment.type == "removed"){

let deleted = appointment.doc.data()

console.log("The following record has been removed ", deleted)

}

//Take in two params to compare, iterates over the array

var appointmentSort = (a,b) => {

let aDate = new Date(a.appointmentDate)

let bDate = new Date(b.appointmentDate)

return aDate.getTime() - bDate.getTime()

}

// using the.sort method to sort the values in the string, to ensure that if a new appointment

// is added that it is populated in the correct appointment order in upcoming appointments

this.appointments.sort(appointmentSort)

this.oldAppointments.sort(appointmentSort)

})

})

},

methods: {

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// Append a 0 to month or date of number is less than or equals 9 to match to appointmentDates

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

// Delete appointments in real time, takes the id of the appointment and filters it out of the corresponding array

toggleCancelAppointment (id) {

// Can only cancel appointments if not the day of appointment

db.collection("appointments").doc(id).get().then(doc => {

let appointmentToCancel = doc.data()

let today = new Date ()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

// Check to see if appointment is scheduled for today, only allow deletes not on same day

if(appointmentToCancel.appointmentDate == formattedDate){

this.triggerSnackbar("You Cannot Delete Appointment on the Day of Appointment!", "error")

}

else{

// Upcoming Appointments

db.collection("appointments").doc(id).delete().then(() => {

this.appointments = this.appointments.filter(appointment => {

return appointment.id != id

})

})

// Past Appointments

db.collection("appointments").doc(id).delete().then(() => {

this.oldAppointments = this.oldAppointments.filter(appointment => {

return appointment.id !=id

})

})

this.triggerSnackbar("Appointment Has Been Successfully Deleted!", "success")

}

})

},

}

}

</script>

## DoctorAppointment.vue

<template>

<v-container>

<DoctorNavbar />

<!-- Import the Calendar for the doctors appointments -->

<v-card id="AppointmentCalendar" flat>

<Calendar />

</v-card>

</v-container>

</template>

<script>

import DoctorNavbar from '../components/Navbars/DoctorNavbar'

import Calendar from "../components/Calendar"

export default {

components: {

DoctorNavbar,

Calendar

}

}

</script>

## Appointment Calendar Component – Calendar.vue

<template>

<v-row class="fill-height">

<v-col>

<!-- Contains the top toolbar with the today, prev, next btn options and a menu that switches the calendar view-->

<v-sheet height="64">

<v-toolbar flat>

<v-btn outlined color="primary" @click="setToday">Today</v-btn>

<v-btn fab text small color="primary" @click="prev">

<v-icon small >mdi-chevron-left</v-icon>

</v-btn>

<v-btn fab text small color="primary" @click="next">

<v-icon small>mdi-chevron-right</v-icon>

</v-btn>

<v-toolbar-title v-if="$refs.calendar">{{ $refs.calendar.title }}</v-toolbar-title>

<v-spacer></v-spacer>

<v-menu bottom right>

<template v-slot:activator="{ on, attrs }">

<v-btn

outlined

color="primary"

v-bind="attrs"

v-on="on"

>

<span>{{ typeToLabel[type] }}</span>

<v-icon right>mdi-menu-down</v-icon>

</v-btn>

</template>

<v-list>

<v-list-item @click="type = 'day'">

<v-list-item-title>Day</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'week'">

<v-list-item-title>Week</v-list-item-title>

</v-list-item>

<v-list-item @click="type = 'month'">

<v-list-item-title>Month</v-list-item-title>

</v-list-item>

</v-list>

</v-menu>

</v-toolbar>

</v-sheet>

<v-sheet height="500">

<v-calendar

ref="calendar"

color="primary"

:type="type"

v-model="focus"

:events="events"

@click:date="viewDay"

@click:event="showEvent"

>

</v-calendar>

<v-menu

v-model="selectedOpen"

:close-on-content-click="false"

:activator="selectedElement"

offset-x

>

<v-card

color="grey lighten-4"

min-width="350px"

flat

>

<v-toolbar class="primary"

:color="selectedEvent.color"

dark

>

<v-toolbar-title class="primary white--text" v-html="selectedEvent.name"></v-toolbar-title>

<v-spacer></v-spacer>

<v-btn

text

color="white"

@click="selectedOpen = false"

>

Cancel

</v-btn>

</v-toolbar>

<v-card-text>

<v-row class="mt-n8">

<v-col cols="12" md="12" class="mt-4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-clinic-medical</v-icon>

Appointment Type</v-list-item-title>

<h3 v-html="selectedEvent.type"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="12" md="12">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Details</v-list-item-title>

<h3 v-html="selectedEvent.details"></h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-menu>

</v-sheet>

</v-col>

</v-row>

</template>

<script>

import { auth, db } from '../firebase'

export default {

data() {

return {

focus: '',

type: 'month',

typeToLabel: {

month: 'Month',

week: 'Week',

day: 'Day',

},

selectedEvent: {},

selectedElement: null,

selectedOpen: false,

events: [],

}

},

mounted() {

this.getAppointments()

},

methods: {

getAppointments () {

// Set current user to the currently logged in user

var currentUser = auth.currentUser.uid

db.collection("appointments").where("doctorID", "==", currentUser).onSnapshot( snap => {

let appointment = snap.docChanges()

appointment.forEach(appointment => {

let record = appointment.doc.data()

let time = record.appointmentTime.replace(".", ":")

let endTime = time.substr(0, time.length -2)

let endTimeMinutes = time.substr(time.length -2, 2)

endTime = endTime + (parseInt(endTimeMinutes) + 25)

let event = {

type: record.appointmentType,

name: record.firstname + ' ' + record.surname,

details : record.appointmentDetails,

start: record.appointmentDate + " " + time,

end: record.appointmentDate + " " + endTime

}

this.events.push(event)

})

})

},

viewDay ({ date }) {

this.focus = date

this.type = 'day'

},

setToday () {

this.focus = ''

},

prev () {

this.$refs.calendar.prev()

},

next () {

this.$refs.calendar.next()

},

showEvent ({ nativeEvent, event }) {

const open = () => {

this.selectedEvent = event

this.selectedElement = nativeEvent.target

setTimeout(() => this.selectedOpen = true, 10)

}

if (this.selectedOpen) {

this.selectedOpen = false

setTimeout(open, 10)

} else {

open()

}

nativeEvent.stopPropagation()

},

},

}

</script>

# Prescriptions

## Prescriptions.vue

<template>

<v-container>

<Navbar />

<v-row><v-col cols="12">

<!-- Full screen dialog overlay for forms -->

<v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">

<template v-slot:activator="{ on, attrs }">

<!-- Viewing of appointments occurs here -->

<v-card flat>

<v-card-title class="primary lighten-1 white--text">Prescriptions

<v-spacer></v-spacer>

<v-btn

class="mx-2"

fab small

dark color="primary darken-1"

v-bind="attrs"

v-on="on"

>

<v-icon>fa-plus</v-icon>

</v-btn>

</v-card-title>

<!-- View Prescriptions Component imported -->

<ViewPrescriptions />

</v-card>

</template>

<!-- Book appointment starts here -->

<v-card>

<v-card-title class="primary white--text">

Request Prescription

<v-spacer></v-spacer>

<v-btn class="mr-4" icon dark @click="cancel()">

<v-icon class="mx-2" fab dark color="white--text darken-1 ">fa-window-close</v-icon>

<span>Cancel</span>

</v-btn>

</v-card-title>

<v-divider class="mx4"></v-divider>

<v-form @submit.prevent="requestPrescription">

<v-card-text>

<!-- No Gender Defined Card -->

<v-card v-if="noGenderView">

<v-card-title></v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="6" lg="6">

<h2 class="overline black--text ml-4" justify="center">You Have Not Selected Your Gender In Medical Record!</h2>

</v-col>

<v-col cols="12" md="6" lg="6">

<router-link to="/medicalRecord" tag="button">

<v-btn class="primary white--text">Go To Medical Record</v-btn>

</router-link>

</v-col>

</v-row>

</v-card-text>

</v-card>

<v-row>

<!-- Female Treatments -->

<v-col cols="12" s="6" sm="6" md="6" lg="6" v-if="showFemaleT">

<v-select

label="Female Treatments"

v-model="chosenOption"

:items="femaleTreatments"

outlined

@change="setTreatmentOption(chosenOption)"

></v-select>

</v-col>

<!-- Male Treatments -->

<v-col cols="12" s="6" sm="6" md="6" lg="6" v-if="showMaleT">

<v-select

label="Male Treatments"

v-model="chosenOption"

:items="maleTreatments"

outlined

@change="setTreatmentOption(chosenOption)"

></v-select>

</v-col>

<!-- Select a doctor to request prescription from -->

<v-col cols="12" s="6" sm="6" md="6" lg="6">

<v-select

name="selectDoctor"

v-model="chosenDoc"

:items="doctors"

item-text="name"

item-value="value"

label="Select a Doctor"

outlined

:error-messages="selectDocError"

@input="$v.chosenDoc.$touch()"

@blur="$v.chosenDoc.$touch()"

@change="onDropdownChanged($event)"

></v-select>

</v-col>

<v-col cols="12">

<!-- Asthma -->

<v-card v-if="showAsthma">

<v-card-title class="primary lighten-1 white--text">Asthma Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" sm="6" md="6" lg="6">

<v-subheader class="overline ml-n5">How Long With Asthma?</v-subheader>

<v-select

label="Select a Length of Time\*"

v-model="asthma.asthmaLength"

:items="asthmaLengthList"

outlined

:error-messages="asthmaLengthError"

@click="$v.asthma.asthmaLength.$touch()"

@blur="$v.asthma.asthmaLength.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Have You Ever Required Oral Steroids?</v-subheader>

<v-select

label="Select\*"

v-model="asthma.asthmaSteroids"

:items="asthmaSteroidList"

outlined

:error-messages="asthmaSteroidsError"

@click="$v.asthma.asthmaSteroids.$touch()"

@blur="$v.asthma.asthmaSteroids.$touch()"

></v-select>

</v-col>

<v-col cols="12" sm="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Asthma Severity\*</v-subheader>

<v-btn-toggle

v-model="asthma.asthmaSeverity"

color="primary"

group

:error-messages="asthmaSeverityError"

@click="$v.asthma.asthmaSeverity.$touch()"

@blur="$v.asthma.asthmaSeverity.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Mild Intermittent">Mild Intermittent</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="Mild Persistent">Mild Persistent</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="Moderate Persistent">Moderate Persistent</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="Severe Persistent">Severe Persistent</v-btn>

</v-btn-toggle>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Adrenaline -->

<v-card v-if="showAdrenaline">

<v-card-title class="primary lighten-1 white--text">Adrenaline Pen Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Have You Been Diagnosed With An Anaphylactic Allergy?</v-subheader>

<v-btn-toggle

v-model="adrenaline.adrenalineDiagnosis"

color="primary"

group

:error-messages="adrenalineDiagnosisError"

@click="$v.adrenaline.adrenalineDiagnosis.$touch()"

@blur="$v.adrenaline.adrenalineDiagnosis.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Are You Trained To Use An Adrenaline Pen?</v-subheader>

<v-btn-toggle

v-model="adrenaline.adrenalinetrained"

color="primary"

group

:error-messages="adrenalinetrainedError"

@click="$v.adrenaline.adrenalinetrained.$touch()"

@blur="$v.adrenaline.adrenalinetrained.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Are You Aware Of The Potential Symptoms of An Anaphylactic Reaction?</v-subheader>

<v-btn-toggle

v-model="adrenaline.adrenalinesymptoms"

color="primary"

group

:error-messages="adrenalineSymptomsError"

@click="$v.adrenaline.adrenalinesymptoms.$touch()"

@blur="$v.adrenaline.adrenalinesymptoms.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Contraception -->

<v-card v-if="showContraception">

<v-card-title class="primary lighten-1 white--text">Contraception Pill & Patch</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">Choose a contraceptive</v-subheader>

<v-select

label="Contraceptive Type"

v-model="contraceptives.contraceptiveType"

:items="contraceptiveList"

outlined

:error-messages="contraceptiveTypeError"

@input="$v.contraceptives.contraceptiveType.$touch()"

@blur="$v.contraceptives.contraceptiveType.$touch()"

>

</v-select>

</v-col>

<v-col cols="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Regulated Periods?</v-subheader>

<v-btn-toggle

v-model="contraceptives.periodsRegular"

color="primary"

group

:error-messages="periodsRegularError"

@click="$v.contraceptives.periodsRegular.$touch()"

@blur="$v.contraceptives.periodsRegular.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Previous Usage?</v-subheader>

<v-btn-toggle

v-model="contraceptives.previoulyTaken"

color="primary"

group

:error-messages="previoulyTakenError"

@input="$v.contraceptives.previoulyTaken.$touch()"

@blur="$v.contraceptives.previoulyTaken.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Any Side Effects?</v-subheader>

<v-btn-toggle

v-model="contraceptives.sideEffects"

color="primary"

group

:error-messages="sideEffectsError"

@input="$v.contraceptives.sideEffects.$touch()"

@blur="$v.contraceptives.sideEffects.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="12" md="6" lg="6">

<v-textarea

label="Describe side effects"

v-model="contraceptives.sideEffectDescription"

outlined

:error-messages="sideEffectDescriptionError"

@input="$v.contraceptives.sideEffectDescription.$touch()"

@blur="$v.contraceptives.sideEffectDescription.$touch()"

></v-textarea>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Thrush -->

<v-card v-if="showThrush">

<v-card-title class="primary lighten-1 white--text">Thrush Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Have You Had Any Severe Lower Abdominal Pain In The Past 48 Hours?</v-subheader>

<v-btn-toggle

v-model="thrush.thrushAbdoPain"

color="primary"

group

:error-messages="thrushAbdoPainError"

@click="$v.thrush.thrushAbdoPain.$touch()"

@blur="$v.thrush.thrushAbdoPain.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Are There Any Blisters Or Rashes On Or Around The Vaginal Skin?</v-subheader>

<v-btn-toggle

v-model="thrush.thrushSkinIssues"

color="primary"

group

:error-messages="thrushSkinIssuesError"

@click="$v.thrush.thrushSkinIssues.$touch()"

@blur="$v.thrush.thrushSkinIssues.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

<v-col cols="12" md="12" lg="12">

<v-subheader class="overline ml-n5">Are There Any Blisters Or Rashes On Or Around The Vaginal Skin?</v-subheader>

<v-btn-toggle

v-model="thrush.thrushUrinary"

color="primary"

group

:error-messages="thrushUrinaryError"

@click="$v.thrush.thrushUrinary.$touch()"

@blur="$v.thrush.thrushUrinary.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- ErecDys -->

<v-card v-if="showErecDys">

<v-card-title class="primary lighten-1 white--text">Erectile Dysfunction Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">Choose a EDT Type</v-subheader>

<v-select

label="Treatment Type"

v-model="ereDys.ereDysType"

:items="ereDysList"

outlined

:error-messages="ereDysTypeError"

@input="$v.ereDys.ereDysType.$touch()"

@blur="$v.ereDys.ereDysType.$touch()"

>

</v-select>

</v-col>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">Select A Dosage</v-subheader>

<v-select

label="Treatment Type"

v-model="ereDys.ereDysDosage"

:items="ereDysDosageList"

outlined

:error-messages="ereDysDosageError"

@input="$v.ereDys.ereDysDosage.$touch()"

@blur="$v.ereDys.ereDysDosage.$touch()"

>

</v-select>

</v-col>

<v-col cols="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Previous Usage?</v-subheader>

<v-btn-toggle

v-model="ereDys.ereDysPrevious"

color="primary"

group

:error-messages="ereDysPreviousError"

@click="$v.ereDys.ereDysPrevious.$touch()"

@blur="$v.ereDys.ereDysPrevious.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="Yes">Yes</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="No">No</v-btn>

</v-btn-toggle>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- PreEj -->

<v-card v-if="showPreE">

<v-card-title class="primary lighten-1 white--text">Premature Ejaculation Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">How Long Have You Been Suffering From PE?</v-subheader>

<v-select

label="Time Sexually Active"

v-model="pe.peDuration"

:items="peDurationList"

outlined

:error-messages="peDurationError"

@input="$v.pe.peDuration.$touch()"

@blur="$v.pe.peDuration.$touch()"

>

</v-select>

</v-col>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">How Often Do You Experience PE?</v-subheader>

<v-select

label="When is it Experienced"

v-model="pe.peOften"

:items="peOftenList"

outlined

:error-messages="peOftenError"

@input="$v.pe.peOften.$touch()"

@blur="$v.pe.peOften.$touch()"

>

</v-select>

</v-col>

<v-col cols="12" md="6" lg="6">

<v-subheader class="overline ml-n5">When Does PE Occur?</v-subheader>

<v-select

label="When does Ejaculation Occur"

v-model="pe.peOccur"

:items="peOccurList"

outlined

:error-messages="peOccurError"

@input="$v.pe.peOccur.$touch()"

@blur="$v.pe.peOccur.$touch()"

>

</v-select>

</v-col>

<v-col cols="6" md="6" lg="6">

<v-subheader class="overline ml-n5">Choose A Medication</v-subheader>

<v-btn-toggle

v-model="pe.medication"

color="primary"

group

:error-messages="medicationError"

@input="$v.pe.medication.$touch()"

@blur="$v.pe.medication.$touch()"

>

<v-btn depressed x-large color="primary--text darken-1" value="EMLA">EMLA</v-btn>

<v-btn depressed x-large color="primary--text darken-1" value="PRILIGY">PRILIGY</v-btn>

</v-btn-toggle>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col>

</v-row>

<v-card-actions>

<v-row>

<v-btn

v-if="requestBtn"

type="submit"

block class="primary white--text"

@click.prevent="requestPrescription()">

<span>Request Prescription</span>

</v-btn>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-row>

</v-card-actions>

</v-card-text>

</v-form>

</v-card> <!-- End of book appointment form within dialog -->

</v-dialog>

</v-col></v-row>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

import ViewPrescriptions from '../components/Prescriptions/ViewPrescriptions'

import { auth, db } from '../firebase'

import { required } from "vuelidate/lib/validators"

export default {

components: {

Navbar,

ViewPrescriptions,

},

computed: {

// Validations

selectDocError () {

const errors = []

if(!this.$v.chosenDoc.$dirty) return errors

!this.$v.chosenDoc.required && errors.push('Please Select A Doctor')

return errors

},

// CONTRACEPTION

contraceptiveTypeError () {

const errors = []

if(!this.$v.contraceptives.contraceptiveType.$dirty) return errors

!this.$v.contraceptives.contraceptiveType.required && errors.push('Please Select A Contraceptive Type')

return errors

},

periodsRegularError () {

const errors = []

if(!this.$v.contraceptives.periodsRegular.$dirty) return errors

!this.$v.contraceptives.periodsRegular.required && errors.push('A Value Is Required')

return errors

},

previoulyTakenError () {

const errors = []

if(!this.$v.contraceptives.previoulyTaken.$dirty) return errors

!this.$v.contraceptives.previoulyTaken.required && errors.push('A Value Is Required')

return errors

},

sideEffectsError () {

const errors = []

if(!this.$v.contraceptives.sideEffects.$dirty) return errors

!this.$v.contraceptives.sideEffects.required && errors.push('A Value Is Required')

return errors

},

sideEffectDescriptionError () {

const errors = []

if(!this.$v.contraceptives.sideEffectDescription.$dirty) return errors

!this.$v.contraceptives.sideEffectDescription.required && errors.push('A Value Is Required')

return errors

},

// EREDYS

ereDysTypeError () {

const errors = []

if(!this.$v.ereDys.ereDysType.$dirty) return errors

!this.$v.ereDys.ereDysType.required && errors.push('Please Select A Form Of Treatment')

return errors

},

ereDysDosageError () {

const errors = []

if(!this.$v.ereDys.ereDysDosage.$dirty) return errors

!this.$v.ereDys.ereDysDosage.required && errors.push('Dosage Is Required')

return errors

},

ereDysPreviousError () {

const errors = []

if(!this.$v.ereDys.ereDysPrevious.$dirty) return errors

!this.$v.ereDys.ereDysPrevious.required && errors.push('A Value Is Required')

return errors

},

// Asthma

asthmaLengthError () {

const errors = []

if(!this.$v.asthma.asthmaLength.$dirty) return errors

!this.$v.asthma.asthmaLength.required && errors.push('Time With Asthma Is Required')

return errors

},

asthmaSteroidsError () {

const errors = []

if(!this.$v.asthma.asthmaSteroids.$dirty) return errors

!this.$v.asthma.asthmaSteroids.required && errors.push('Any Steroids Is Required')

return errors

},

asthmaSeverityError () {

const errors = []

if(!this.$v.asthma.asthmaSeverity.$dirty) return errors

!this.$v.asthma.asthmaSeverity.required && errors.push('Asthma Severity Is Required')

return errors

},

// Adrenaline

adrenalineDiagnosisError () {

const errors = []

if(!this.$v.adrenaline.adrenalineDiagnosis.$dirty) return errors

!this.$v.adrenaline.adrenalineDiagnosis.required && errors.push('Adrenaline Diagnosis Is Required')

return errors

},

adrenalinetrainedError () {

const errors = []

if(!this.$v.adrenaline.adrenalinetrained.$dirty) return errors

!this.$v.adrenaline.adrenalinetrained.required && errors.push('Adrenaline Trained Is Required')

return errors

},

adrenalineSymptomsError () {

const errors = []

if(!this.$v.adrenaline.adrenalinesymptoms.$dirty) return errors

!this.$v.adrenaline.adrenalinesymptoms.required && errors.push('Adrenaline Symptoms Are Required')

return errors

},

// PE

peDurationError () {

const errors = []

if(!this.$v.pe.peDuration.$dirty) return errors

!this.$v.pe.peDuration.required && errors.push('PE Duration Is Required')

return errors

},

peOftenError () {

const errors = []

if(!this.$v.pe.peOften.$dirty) return errors

!this.$v.pe.peOften.required && errors.push('How Often Is Required')

return errors

},

peOccurError () {

const errors = []

if(!this.$v.pe.peOccur.$dirty) return errors

!this.$v.pe.peOccur.required && errors.push('PE Occurence Is Required')

return errors

},

medicationError () {

const errors = []

if(!this.$v.pe.medication.$dirty) return errors

!this.$v.pe.medication.required && errors.push('PE Medication Is Required')

return errors

},

// Thrush

thrushAbdoPainError () {

const errors = []

if(!this.$v.thrush.thrushAbdoPain.$dirty) return errors

!this.$v.thrush.thrushAbdoPain.required && errors.push('Thrush Abdominal Pain Confirmation Is Required')

return errors

},

thrushSkinIssuesError () {

const errors = []

if(!this.$v.thrush.thrushSkinIssues.$dirty) return errors

!this.$v.thrush.thrushSkinIssues.required && errors.push('Skin Condition Confirmation Is Required')

return errors

},

thrushUrinaryError () {

const errors = []

if(!this.$v.thrush.thrushUrinary.$dirty) return errors

!this.$v.thrush.thrushUrinary.required && errors.push('Urinary Condition Confirmation Is Required')

return errors

},

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

this.setGenderOption ()

// This query first gets the list of users whos role is doctor from the roles collection

db.collection("roles").where("role.doctor", "==", true).get().then(snap => {

// For each doctor document store the doc.id, which is the same ID used for the users colection doc.id, into doctorID

snap.forEach(doc => {

var doctorID = doc.id

// Then query the users collection with the doctorID to find corresponding record based on same ID

// so that the v-select for doctor can access the name of the doctor

db.collection("users").doc(doctorID).get().then(doc => {

var docID = doc.id

var doctorFirstName = doc.data().firstname

var doctorSurname = doc.data().surname

this.doctors.push({

name: doctorFirstName + " " + doctorSurname,

value: docID

})

})

})

})

},

data() {

return {

// Snackbar properties

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

dialog: false, // Dialog for overall dialog panel

doctors: [], // Doctors array contains a list of doctors

chosenDoc: null, // Chosen Doc stores the value of selected doctor, derived of onDropdownChanged()

noGenderView: false, // Show if no gender is defined

genderOption: null,

name: null,

requestBtn: false,

showFemaleT: false, // show female treatments

showMaleT: false, // show male treatments

status: "Pending Review",

chosenOption: null,

femaleTreatments: [

{ text: 'Adrenaline Pen Treatment', value: 'Adrenaline Pen Treatment'},

{ text: 'Asthma Treatment', value: 'Asthma Treatment'},

{ text:'Contraceptive Pill & Patch', value: 'Contraception' },

{ text:'Thrush Treatment', value: 'Thrush Treatment' },

],

maleTreatments: [

{ text: 'Adrenaline Pen Treatment', value: 'Adrenaline Pen Treatment'},

{ text: 'Asthma Treatment', value: 'Asthma Treatment'},

{ text: 'Erectile Dysfunction Treatment', value: 'Erectile Dysfunction Treatment' },

{ text: 'Premature Ejaculation Treatment', value: 'Premature Ejaculation Treatment' },

],

// Treatment forms

showAdrenaline: false,

showAsthma: false,

showContraception: false,

showThrush: false,

showErecDys: false,

showPreE: false,

// Contraception Card

contraceptives: {

contraceptiveType: null,

periodsRegular: null,

previoulyTaken: null,

sideEffects: null,

sideEffectDescription: null,

},

contraceptiveList: [

{text: 'Azalia', value: "Azalia" },

{text: 'Cerazette', value: "Cerazette" },

{text: 'Cilest', value: "Cilest" },

{text: 'Elvina', value: "Elvina" },

{text: 'Elvinette', value: "Elvinette" },

{text: 'Evra Patches', value: "Evra Patches" },

{text: 'Freedonel', value: "Freedonel" },

{text: 'Leonore', value: "Leonore" },

{text: 'Microlite', value: "Microlite" },

{text: 'Yasmin', value: "Yasmin" },

{text: 'Zoely ', value: "Zoely " },

],

// EreDys

ereDys: {

ereDysType: null,

ereDysDosage: null,

ereDysPrevious: null,

},

ereDysList: [

{text: 'Viagra', value: 'Viagra'},

{text: 'Sildenafil', value: 'Sildenafil'},

{text: 'Cialis', value: 'Cialis'},

{text: 'Levitra', value: 'Levitra'},

],

ereDysDosageList: [

{text: '2.5 MG', value: '2.5 MG'},

{text: '5 MG', value: '5 MG'},

{text: '25 MG', value: '25 MG'},

{text: '50 MG', value: '50 MG'},

{text: '100 MG', value: '100 MG'},

{text: 'Unsure', value: 'Unsure'},

],

// Asthma

asthma: {

asthmaLength: null,

asthmaSteroids: null,

asthmaSeverity: null,

},

asthmaLengthList: [

{text:'Less Than 6 Months', value:'Less Than 6 Months'},

{text:'Over A Year', value:'Over A Year'},

{text:'Over 5 Years', value:'Over 5 Years'},

{text:'Since Birth', value:'Since Birth'}

],

asthmaSteroidList: [

{text:'Never', value: 'Never'},

{text:'In The Last 6 Months', value: 'In The Last 6 Months'},

{text:'In The Last 12 Months', value: 'In The Last 12 Months'},

{text:'Over A Year Ago', value: 'Over A Year Ago'},

],

// Adrenaline

adrenaline: {

adrenalineDiagnosis: null,

adrenalinetrained: null,

adrenalinesymptoms: null,

},

// PE

pe: {

peDuration: null,

peOften: null,

peOccur: null,

medication: null,

},

peDurationList: [

{text: 'Less Than 3 Months', valude: 'Less Than 3 Months'},

{text: 'Less Than 6 Months', valude: 'Less Than 6 Months'},

{text: 'Less Than 1 Year', valude: 'Less Than 1 Year'},

{text: 'Longer Than 5 Years', valude: 'Longer Than 5 Years'},

{text: 'Indefinetly', valude: 'Indefinetly'},

],

peOftenList: [

{text: 'During Sex', value: 'During Sex'},

{text: 'Only With New Partner', value: 'During Sex With New Partner'},

],

peOccurList: [

{text: 'During Foreplay', value: 'During Foreplay'},

{text: 'While Attempting Penetration', value: 'While Attempting Penetration'},

{text: 'Just After Penetration', value: 'Just After Penetration'},

{text: 'Less Than 2 Minutes After Penetration', value: 'Less Than 2 Minutes After Penetration'},

{text: 'More Than 2 Minutes After Penetration', value: 'More Than 2 Minutes After Penetration'},

],

// Thrush

thrush: {

thrushAbdoPain: null,

thrushSkinIssues: null,

thrushUrinary: null,

},

}

},

validations: {

chosenDoc: { required },

contraceptives: {

contraceptiveType: { required },

periodsRegular: { required },

previoulyTaken: { required },

sideEffects: { required },

sideEffectDescription: { required },

},

ereDys: {

ereDysType: { required },

ereDysDosage: { required },

ereDysPrevious: { required },

},

asthma: {

asthmaLength: { required },

asthmaSteroids: { required },

asthmaSeverity: { required },

},

adrenaline: {

adrenalineDiagnosis: { required },

adrenalinetrained: { required },

adrenalinesymptoms: { required },

},

pe: {

peDuration: { required },

peOften: { required },

peOccur: { required },

medication: { required },

},

thrush: {

thrushAbdoPain: { required },

thrushSkinIssues: { required},

thrushUrinary: { required },

},

},

methods: {

// Triggers the snackbar with the passed message and colour of the message

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// Set the visible treatments v-select for the user

setGenderOption () {

db.collection("users").doc(this.currentUser).get().then(( snap => {

this.genderOption = snap.data().gender // Return the gender of the patient

this.name = snap.data().firstname + " " + snap.data().surname

if(this.genderOption == 'Male'){

this.hideAllTreatments()

this.requestBtn = true

this.showFemaleT = false

this.showMaleT = true

this.chosenOption = null

}

else if(this.genderOption == 'Female'){

this.requestBtn = true

this.hideAllTreatments()

this.showMaleT= false

this.showFemaleT = true

this.chosenOption = null

}

else{

// No gender defined, show redirect button for user to set their gender in

// medical record

this.requestBtn = false

this.noGenderView = true

}

}))

},

// Hide All Treatment Forms

hideAllTreatments () {

this.showAdrenaline = false,

this.showAsthma = false,

this.showContraception = false,

this.showThrush = false,

this.showErecDys = false,

this.showPreE = false

},

//Doctor is changed

onDropdownChanged(value) {

db.collection("prescriptions").where("requestedDate", "==", this.getTodaysDate())

.where("patientID", "==", this.currentUser).get().then(() => {

this.chosenDoc = value

this.hideAllTreatments()

this.clearForms()

this.setTreatmentOption()

})

},

// Set the Visible Form Depending on Treatment Selected,

//Unless Pending Prescription Already

setTreatmentOption () {

this.clearForms()

if (this.chosenOption == 'Adrenaline Pen Treatment' && this.chosenDoc != null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Adrenaline Pen Treatment!", "error")

}

})

this.showAdrenaline = show

})

}

else if(this.chosenOption == 'Asthma Treatment' && this.chosenDoc !=null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Asthma Treatment!", "error")

}

})

this.showAsthma = show

})

}

else if(this.chosenOption == 'Contraception' && this.chosenDoc !=null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Contraceptive Pill & Patch!", "error")

}

})

this.showContraception = show

})

}

else if(this.chosenOption == 'Thrush Treatment' && this.chosenDoc !=null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Thrush Treatment!", "error")

}

})

this.showThrush = show

})

}

else if(this.chosenOption == 'Erectile Dysfunction Treatment' && this.chosenDoc !=null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Erectile Dysfunction Treatment!", "error")

}

})

this.showErecDys = show

})

}

else if(this.chosenOption == 'Premature Ejaculation Treatment' && this.chosenDoc !=null){

this.hideAllTreatments()

db.collection("prescriptions").where("patientID", "==", this.currentUser).get().then(snap => {

let prescriptions = snap.docChanges()

let show = true

prescriptions.forEach(prescriptions => {

let presDoc = prescriptions.doc.data()

if(presDoc.chosenType == this.chosenOption && presDoc.status == this.status){

show = false

this.triggerSnackbar("You Have Already Made A Prescription Request For Premature Ejaculation Treatment!", "error")

}

})

this.showPreE = show

})

}

else{

this.hideAllTreatments()

console.log("Treatment & Doctor Must Be Selected!")

}

},

// Append a 0 to month or date of number is less than or equals 9 to match to appointmentDates

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

getTodaysDate () {

let today = new Date()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

return formattedDate

},

// Cancel the page

cancel () {

// this.$v.$reset()

this.dialog = false

this.hideAllTreatments()

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.snackbar = null

},

clearForms () {

this.$v.$reset()

// Clear contraceptive

this.contraceptives.contraceptiveType = null

this.contraceptives.periodsRegular = null

this.contraceptives.previoulyTaken = null

this.contraceptives.sideEffects = null

this.contraceptives.sideEffectDescription = null

// Clear EDT

this.ereDys.ereDysType = null

this.ereDys.ereDysDosage = null

this.ereDys.ereDysPrevious = null

// Clear Asthma

this.asthma.asthmaLength = null

this.asthma.asthmaSteroids = null

this.asthma.asthmaSeverity = null

// Clear Adrenaline

this.adrenaline.adrenalineDiagnosis = null

this.adrenaline.adrenalinetrained = null

this.adrenaline.adrenalinesymptoms = null

// Clear PE

this.pe.peDuration = null

this.pe.peOften = null

this.pe.peOccur = null

this.pe.medication = null

// Clear Thrush

this.thrush.thrushAbdoPain = null

this.thrush.thrushSkinIssues = null,

this.thrush.thrushUrinary = null

},

// Submit request form

requestPrescription () {

if (this.chosenOption == 'Adrenaline Pen Treatment'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.adrenaline.$anyDirty

this.errors = this.$v.adrenaline.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription ={

patientName: this.name,

patientID: this.currentUser,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

adrenalineDiagnosis: this.adrenaline.adrenalineDiagnosis,

trained: this.adrenaline.adrenalinetrained,

recogniseSymptoms: this.adrenaline.adrenalinesymptoms,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else if(this.chosenOption == 'Asthma Treatment'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.asthma.$anyDirty

this.errors = this.$v.asthma.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription ={

patientID: this.currentUser,

patientName: this.name,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

asthmaLength: this.asthma.asthmaLength,

anySteroids: this.asthma.asthmaSteroids,

asthmaSeverity: this.asthma.asthmaSeverity,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else if(this.chosenOption == 'Contraception'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.contraceptives.$anyDirty

this.errors = this.$v.contraceptives.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription ={

patientName: this.name,

patientID: this.currentUser,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

contraception: this.contraceptives.contraceptiveType,

periodRegulation: this.contraceptives.periodsRegular,

previousUsage: this.contraceptives.previoulyTaken,

sideEffects: this.contraceptives.sideEffects,

effectsDescription: this.contraceptives.sideEffectDescription,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else if(this.chosenOption == 'Thrush Treatment'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.thrush.$anyDirty

this.errors = this.$v.thrush.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription ={

patientName: this.name,

patientID: this.currentUser,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

abdominalPain: this.thrush.thrushAbdoPain,

skinIssues: this.thrush.thrushSkinIssues,

urinaryIssue: this.thrush.thrushUrinary,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else if(this.chosenOption == 'Erectile Dysfunction Treatment'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.ereDys.$anyDirty

this.errors = this.$v.ereDys.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription = {

patientName: this.name,

patientID: this.currentUser,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

eDTreatment: this.ereDys.ereDysType,

dosage: this.ereDys.ereDysDosage,

previousUsage: this.ereDys.ereDysPrevious,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else if(this.chosenOption == 'Premature Ejaculation Treatment'){

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.pe.$anyDirty

this.errors = this.$v.pe.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

let addPrescription = {

patientName: this.name,

patientID: this.currentUser,

doctor: this.chosenDoc,

dateRequested: this.getTodaysDate(),

chosenType: this.chosenOption,

status: this.status,

durationWith: this.pe.peDuration,

howOften: this.pe.peOften,

occurs: this.pe.peOccur,

medication: this.pe.medication,

}

this.triggerSnackbar("Request Has Been Submitted", "success")

db.collection("prescriptions").doc().set(addPrescription).then(()=>{

this.clearForms()

this.chosenOption = null

this.chosenDoc = null

this.hideAllTreatments()

// Close the form

this.dialog = false

}).catch(error => {

console.log("Prescription Error ", error)

this.triggerSnackbar("There Were Errors With The Form!", "error")

})

}

else{

this.triggerSnackbar("Could Not Submit Precription Request, Missing Form Data!", "error")

}

}

else{

this.hideAllTreatments()

this.clearForms()

this.triggerSnackbar("You Must Select A Treatment!", "error")

}

},

},

}

</script>

## ViewPrescriptions.vue

<template>

<v-row><v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-tabs v-model="tab">

<v-tab v-for="tab in tabs" :key="tab.tabName">

{{ tab.tabName }}

</v-tab>

</v-tabs>

<v-tabs-items v-model="tab">

<!-- Requested -->

<v-tab-item>

<v-card outlined class="mt-3" v-for="prescription in requestedPrescriptions" :key="prescription.id">

<v-card-title class="primary lighten-2 white--text">

<v-spacer></v-spacer>

<!-- Cancel button for pending prescriptions -->

<v-icon right @click="cancelPrescription(prescription.id)">fa-trash</v-icon>

</v-card-title>

<v-card-text class="mt-6">

<v-row class="mt-n8">

<v-col cols="12" md="4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-id-card</v-icon>

Prescription ID</v-list-item-title>

<h3>{{ prescription.id }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-prescription</v-icon>

Type</v-list-item-title>

<h3>{{ prescription.chosenType }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-calendar</v-icon>

Date Requested</v-list-item-title>

<h3>{{ prescription.dateRequested }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-medical-alt</v-icon>

Status</v-list-item-title>

<h3>{{ prescription.status }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-tab-item>

<!-- Accepted -->

<v-tab-item>

<v-card outlined class="mt-3" v-for="prescription in acceptedPrescriptions" :key="prescription.id">

<v-card-title class="primary lighten-2 white--text"></v-card-title>

<v-card-text class="mt-6">

<v-row class="mt-n8">

<v-col cols="12" md="4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-id-card</v-icon>

Prescription ID</v-list-item-title>

<h3>{{ prescription.id }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-prescription</v-icon>

Type</v-list-item-title>

<h3>{{ prescription.chosenType }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-calendar</v-icon>

Date Requested</v-list-item-title>

<h3>{{ prescription.dateRequested }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-medical-alt</v-icon>

Status</v-list-item-title>

<h3>{{ prescription.status }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Prescription</v-list-item-title>

<img :src="prescription.image" alt="prescriptionImage" />

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-tab-item>

<!-- Denied -->

<v-tab-item>

<v-card outlined class="mt-3" v-for="prescription in deniedPrescriptions" :key="prescription.id">

<v-card-title class="primary lighten-2 white--text"></v-card-title>

<v-card-text class="mt-6">

<v-row class="mt-n8">

<v-col cols="12" md="4">

<v-list >

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-id-card</v-icon>

Prescription ID</v-list-item-title>

<h3>{{ prescription.id }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-prescription</v-icon>

Type</v-list-item-title>

<h3>{{ prescription.chosenType }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-calendar</v-icon>

Date Requested</v-list-item-title>

<h3>{{ prescription.dateRequested }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-file-medical-alt</v-icon>

Status</v-list-item-title>

<h3>{{ prescription.status }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

<v-col cols="6" md="4">

<v-list>

<v-list-item>

<v-list-item-content>

<v-list-item-title class="overline grey--text mb-4">

<v-icon>fa-info</v-icon>

Reason</v-list-item-title>

<h3>{{ prescription.reason }}</h3>

</v-list-item-content>

</v-list-item>

</v-list>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-tab-item>

</v-tabs-items>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-col></v-row>

</template>

<script>

import {auth, db, storage} from '../../firebase'

export default {

created() {

this.currentUser = auth.currentUser.uid // Get current users uid

// In prescriptions, where patientID is the same as the current users, access

db.collection("prescriptions").where("patientID", "==", this.currentUser).orderBy("dateRequested").onSnapshot(snap => {

let prescriptions = snap.docChanges()

prescriptions.forEach(async (prescriptions) => {

// if any prescriptions are added

if(prescriptions.type == "added"){

let prescription = prescriptions.doc.data()

prescription.id = prescriptions.doc.id

// pending goes into pending array

if(prescription.status == "Pending Review"){

this.requestedPrescriptions.push(prescription)

}

// if accepted grab both the record and the uploaded prescription file

// and push to the accepted prescriptions array

else if(prescription.status == "Accepted"){

// Get prescription file

var pathReference = storage.ref("prescriptions/" + prescription.id)

pathReference.getDownloadURL().then(url => {

prescription.image = url

this.acceptedPrescriptions.push(prescription)

}).catch(error => {

console.log(error)

})

}

// denied goes into denied array

else if (prescription.status == "Denied"){

this.deniedPrescriptions.push(prescription)

}

else{

console.log("Internal Prescription Error")

}

}

})

})

},

data() {

return {

currentUser: null,

// Snackbar

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

// Tabs for requested, accepted and denied

tab: null,

tabs: [

{ tabName: 'Requested' },

{ tabName: 'Accepted' },

{ tabName: 'Denied' },

],

requestedPrescriptions: [],

acceptedPrescriptions: [],

deniedPrescriptions: [],

}

},

methods: {

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// Filter any prescriptions from pending array when deleted

cancelPrescription (id) {

db.collection("prescriptions").doc(id).delete().then(() => {

this.requestedPrescriptions = this.requestedPrescriptions.filter(prescription => {

return prescription.id !=id

})

})

this.triggerSnackbar("Prescription Has Been Successfully Deleted!", "success")

},

}

}

</script>

## DoctorPrescription.vue

<template>

<v-container>

<DoctorNavbar />

<v-row><v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">

<template v-slot:activator="{ on, attrs }">

<!-- Patient Prescription Search Bar -->

<v-card outlined>

<v-card-title class="text-uppercase grey--text text--darken-1">Prescriptions

<v-spacer></v-spacer>

<v-text-field

v-model="search"

append-icon="fa-search"

label="Search"

single-line

hide-details

></v-text-field>

</v-card-title>

</v-card>

<!-- Card for the Data Table -->

<v-card outlined>

<v-data-table

:headers="headers"

:items="prescriptionsPending"

:search="search"

:items-per-page="10"

class="elevation-1"

>

<template v-slot:[`item.actions`]="{ item }">

<v-col cols="12" s="12" sm="12" md="12" lg="12">

<v-icon

class="mr-2"

:key="item.id"

:items="prescriptionsPending"

v-bind="attrs"

v-on="on"

@click="storeID(item.id)"

>fa-file-alt

</v-icon>

</v-col>

</template>

</v-data-table>

</v-card>

</template>

<!-- Opens info pertaining to the request -->

<v-card>

<v-card-title class="primary white--text">

<v-btn class="ml-0" icon dark @click="close()">

<v-icon class="ml-0" fab dark color="white--text darken-1 ">fa-long-arrow-alt-left</v-icon>

</v-btn>

<span class="ml-10">Review Prescription</span>

</v-card-title>

<v-row><v-col cols="12" md="12" lg="12">

<!-- Contraception Request -->

<v-card flat v-if="showContraception">

<v-card-title class="mb-4 primary lighten-2 white--text">Contraceptive Pill & Patches Request</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Contraceptive Type"

v-model="contraceptiveType"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Regular Periods"

v-model="period"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Previous Usage"

v-model="previousUsage"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Side Effects"

v-model="sideEffects"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="12" md="8" lg="8">

<v-text-field

label="Side Effects Description"

v-model="description"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Thrush Request -->

<v-card flat v-if="showThrush">

<v-card-title class="mb-4 primary lighten-2 white--text">Thrush Treatment Request</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Severe Abdominal Pain"

v-model="abdominalPain"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Blisters/Rashes on Skin Around Vagina"

v-model="skinIssues"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Pain Urinating/Increase Urinary Frequency"

v-model="urinaryIssue"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Adrenaline Request -->

<v-card flat v-if="showAdrenaline">

<v-card-title class="mb-4 primary lighten-2 white--text">Adrenaline Pen Treatment Request</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Diagnosed with an Anaphylactic Allergy by a Doctor"

v-model="diagnosed"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Trained to use a Prefilled Adrenaline Injectable Pen"

v-model="trained"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Recognises Signs & Symptoms of an Anaphylactic Reaction"

v-model="recogniseSymptoms"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Asthma Request -->

<v-card flat v-if="showAsthma">

<v-card-title class="mb-4 primary lighten-2 white--text">Asthma Treatment Request</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Asthma Condition"

v-model="asthmaCondition"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Asthma Severity"

v-model="asthmaSeverity"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Have Oral Steroids ever been Required"

v-model="steroids"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- PreE Request -->

<v-card flat v-if="showPreE">

<v-card-title class="mb-4 primary lighten-2 white--text">Premature Ejaculation Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Time Sexually Active"

v-model="durationWith"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="When is it Experienced"

v-model="often"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="When does Ejaculation Occur"

v-model="occurs"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Medication Requested"

v-model="medicationRequest"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- ErecDys Request -->

<v-card flat v-if="showErecDys">

<v-card-title class="mb-4 primary lighten-2 white--text">Erectile Dysfunction Treatment</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Medication Requested"

v-model="eDTreatment"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Dosage Requested"

v-model="eDDosage"

outlined

readonly

></v-text-field>

</v-col>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-text-field

label="Any Prior Usage"

v-model="eDpreviousUsage"

outlined

readonly

></v-text-field>

</v-col>

</v-row>

</v-card-text>

</v-card>

</v-col></v-row>

<!-- FOR DOCTOR USE TO RESPOND -->

<v-card flat>

<v-card-title>

<v-divider></v-divider>

</v-card-title>

<v-card-text>

<v-row>

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4">

<v-select

label="Select Response"

v-model="response"

:items="responses"

outlined

@change="responseChanges(response)"

:error-messages="responseError"

@input="$v.response.$touch()"

@blur="$v.response.$touch()"

></v-select>

</v-col>

<!-- If accepted, display upload box -->

<v-col class="mt-n3" cols="12" sm="6" md="4" lg="4" v-if="showUpload">

<v-file-input

type="file"

v-model="prescriptionFile"

label="Upload prescription"

outlined

></v-file-input>

</v-col>

<!-- If denying request display response text area -->

<v-col class="mt-n3" cols="12" sm="6" md="8" lg="8" v-if="showDenied">

<v-textarea

label="Reason"

v-model="reasonDenied"

outlined

counter="250"

maxlength="250"

></v-textarea>

</v-col>

</v-row>

</v-card-text>

<v-card-actions>

<v-col cols="12">

<v-btn

type="submit"

block class="primary white--text"

@click.prevent="resolveRequest()">

<span>Resolve Request</span>

</v-btn>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-col>

</v-card-actions>

</v-card>

</v-card>

</v-dialog>

</v-col></v-row>

</v-container>

</template>

<script>

import DoctorNavbar from '../components/Navbars/DoctorNavbar'

import { auth, db, storage } from '../firebase'

import { required } from "vuelidate/lib/validators"

export default {

components: {

DoctorNavbar,

},

computed: {

responseError () {

const errors = []

if(!this.$v.response.$dirty) return errors

!this.$v.response.required && errors.push('A Response Is Required')

return errors

},

},

created() {

this.currentUser = auth.currentUser.uid // Get current user uid

// Access prescription requests pertaining to the current doctor and order by date request and patient name

db.collection("prescriptions").where("doctor", "==", this.currentUser).orderBy("dateRequested").orderBy("patientName").onSnapshot(snap => {

let prescriptionToHandle = snap.docChanges()

prescriptionToHandle.forEach(prescriptionToHandle => {

if(prescriptionToHandle.type == "added"){

let prescription = prescriptionToHandle.doc.data()

prescription.id = prescriptionToHandle.doc.id

//Only show prescriptions of status pending to the doctor

if(prescription.status == "Pending Review"){

let patient = {

id: prescription.id,

name: prescription.patientName,

prescription: prescription.chosenType,

requestDate: prescription.dateRequested,

}

this.prescriptionsPending.push(patient)

}

else if (prescription.status == "Accepted"){

this.prescriptionsAccepted.push(prescription)

}

else if(prescription.status == "Denied"){

this.prescriptionsDenied.push(prescription)

}

else{

console.log("Internal prescription error")

}

}

else{

console.log("Prescription handling error")

}

})

})

},

data() {

return {

currentUser: null,

// Snackbar properties

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

prescriptionRecord: null, // Store the patient ID when doctor clicks on actions

search: '', // Search bar

dialog: false,

headers: [

{ text: 'Patient Name', value: 'name' },

{ text: 'Prescription', value: 'prescription' },

{ text: 'Date Requested', value: 'requestDate' },

{ text: 'Actions', value: 'actions', sortable: false }

],

prescriptionsPending: [],

prescriptionsAccepted: [],

prescriptionsDenied: [],

// Doctor Form Section

response: null,

responses: [

{ text: 'Accepted', value: 'Accepted' },

{ text: 'Denied', value: 'Denied' }

],

showDenied: false,

reasonDenied: null,

showUpload: false,

prescriptionFile: null,

prescriptionUpload: "none",

// Contraceptive Details

showContraception: false,

contraceptiveType: null,

period: null,

previousUsage: null,

sideEffects: null,

description: null,

// Thrush Details

showThrush: false,

abdominalPain: null,

skinIssues: null,

urinaryIssue: null,

// Adrenaline Details

showAdrenaline: false,

diagnosed: null,

trained: null,

recogniseSymptoms: null,

// Asthma Details

showAsthma: false,

asthmaCondition: null,

asthmaSeverity: null,

steroids: null,

// PreE Details

showPreE: false,

durationWith: null,

often: null,

occurs: null,

medicationRequest: null,

// ErecDys Details

showErecDys: false,

eDTreatment: null,

eDDosage: null,

eDpreviousUsage: null,

}

},

validations:{

response: { required }

},

methods: {

// Triggers the snackbar with the passed message and colour of the message

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

close () {

this.dialog = false

this.response = null,

this.reasonDenied = null

},

hideAllTreatments () {

this.showAdrenaline = false,

this.showAsthma = false,

this.showContraception = false,

this.showThrush = false,

this.showErecDys = false,

this.showPreE = false

},

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

getTodaysDate () {

let today = new Date()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

return formattedDate

},

// Access the record associated with the user by the ID of the prescription to populate their information into variables

storeID (id) {

this.prescriptionRecord = id

db.collection("prescriptions").doc(this.prescriptionRecord).onSnapshot(snap => {

let prescriptionInfo = snap.data()

prescriptionInfo.id = snap.id

// determine which type of prescription request and populate

// the appropriate fields

if (prescriptionInfo.chosenType == "Contraception"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showContraception = true

// Populate any info for this form

this.contraceptiveType = prescriptionInfo.contraception

this.period = prescriptionInfo.periodRegulation

this.previousUsage = prescriptionInfo.previousUsage

this.sideEffects = prescriptionInfo.sideEffects

this.description = prescriptionInfo.effectsDescription

}

else if (prescriptionInfo.chosenType == "Thrush Treatment"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showThrush = true

// Populate any info for this form

this.abdominalPain = prescriptionInfo.abdominalPain

this.skinIssues = prescriptionInfo.skinIssues

this.urinaryIssue = prescriptionInfo.urinaryIssue

}

else if (prescriptionInfo.chosenType == "Adrenaline Pen Treatment"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showAdrenaline = true

// Populate any info for this form

this.diagnosed = prescriptionInfo.adrenalineDiagnosis

this.trained = prescriptionInfo.trained

this.recogniseSymptoms = prescriptionInfo.recogniseSymptoms

}

else if (prescriptionInfo.chosenType == "Asthma Treatment"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showAsthma = true

// Populate any info for this form

this.asthmaCondition = prescriptionInfo.asthmaLength

this.asthmaSeverity = prescriptionInfo.asthmaSeverity

this.steroids = prescriptionInfo.anySteroids

}

else if (prescriptionInfo.chosenType == "Premature Ejaculation Treatment"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showPreE = true

// Populate any info for this form

this.durationWith = prescriptionInfo.durationWith

this.often = prescriptionInfo.howOften

this.occurs = prescriptionInfo.occurs

this.medicationRequest = prescriptionInfo.medication

}

else if (prescriptionInfo.chosenType == "Erectile Dysfunction Treatment"){

// Hide all forms before showing specific one

this.hideAllTreatments()

this.showErecDys = true

// Populate any info for this form

this.eDTreatment = prescriptionInfo.eDTreatment

this.eDDosage = prescriptionInfo.dosage

this.eDpreviousUsage = prescriptionInfo.previousUsage

}

else{

console.log("Treatment not recognised!")

this.hideAllTreatments()

}

})

},

clearForm () {

this.hideAllTreatments()

this.showDenied = false

this.showUpload = false

this.response = null

this.reasonDenied = null

this.prescriptionFile = null

this.dialog = false

},

// Used to hide or display the denied response

// and the upload prescription box

responseChanges (response) {

if(response == "Denied"){

this.showUpload = false

this.showDenied = true

}

else{

this.showDenied = false

this.showUpload = true

}

},

// Resolve the prescription request

resolveRequest () {

this.$v.$touch() // used to check the state of the form fields

this.formTouched = !this.$v.$anyDirty

this.errors = this.$v.$anyError

// If the form does not have any errors or each individual field has no invalid data

if (this.errors === false && this.formTouched === false){

db.collection("prescriptions").doc(this.prescriptionRecord).get().then(() => {

if(this.response == "Accepted"){

var file = this.prescriptionFile // Get the file

// Create the storage reference

const storageRef = storage.ref("prescriptions/" + this.prescriptionRecord)

storageRef.put(file) // Store the file using the storage reference

this.triggerSnackbar("Response Submitted", "success")

db.collection("prescriptions").doc(this.prescriptionRecord).update({

status: this.response,

dateAccepted: this.getTodaysDate()

}).then(() => {

this.clearForm()

// Clear the prescription from list of pending prescriptions

this.prescriptionsPending = this.prescriptionsPending.filter(prescription => {

return prescription.id != this.prescriptionRecord

})

})

}

else if(this.response == "Denied"){

this.triggerSnackbar("Response Submitted", "success")

db.collection("prescriptions").doc(this.prescriptionRecord).update({

status: this.response,

reason: this.reasonDenied

}).then(() =>{

this.clearForm()

// Clear the prescription from list of pending prescriptions

this.prescriptionsPending = this.prescriptionsPending.filter(prescription => {

return prescription.id != this.prescriptionRecord

})

})

}

else{

console.log("We got problems")

}

})

}

else{

this.triggerSnackbar("Please Complete Form Before Submission", "error")

}

},

},

}

</script>

# Chat

## Chat.vue

<template>

<v-container>

<Navbar />

<v-card v-if="preMessageView">

<v-card-title></v-card-title>

<v-card-text>

<v-row>

<v-col cols="12" md="6" lg="6">

<h2 class="overline black--text ml-4" justify="center">You do not have any Online Appointments!</h2>

</v-col>

<v-col cols="12" md="6" lg="6">

<router-link to="/appointments" tag="button">

<v-btn class="primary white--text">Create An Appointment</v-btn>

</router-link>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Card for displaying the message area and delete room option -->

<v-card v-if="chatRoom">

<v-card-title>End Session

<v-spacer></v-spacer>

<v-btn @click="destroyRoom()"><span>Delete</span></v-btn>

</v-card-title>

</v-card>

<!-- Chat Card css gets called to display messages in correct format -->

<v-card v-if="chatRoom" class="mt-6 mb-9">

<v-card-title class="primary lighten-1 white--text">Online Consultation</v-card-title>

<v-container>

<v-card outlined class="messages">

<v-card-text class="chat-message" v-for="(message, index) in messages" :key="index">

<div class="right-bubble" v-if="message.name === userName ">

<div class="subtitle-1">{{ message.name }}&nbsp;</div>

<span class="green-text">{{ message.text }}</span>

</div>

<div class="left-bubble right-align" v-if="message.name !== userName">

<div class="subtitle-1">{{ message.name }}&nbsp;</div>

<span class="green-text">{{ message.text }}</span>

</div>

</v-card-text>

</v-card>

<v-card flat class="mt-4">

<v-card-text>

<v-row>

<!-- Area to type messages -->

<v-col cols="12" sm="12" md="6">

<v-textarea

rows="1"

auto-grow

v-model="message"

outlined

:error-messages="messageErrors"

@input="$v.message.$touch()"

@blur="$v.message.$touch()"

></v-textarea>

</v-col>

<!-- Button to send messages -->

<v-col cols="12" sm="12" md="2">

<v-btn

class="primary white--text ml-6"

block

@click="sendMessage()"

>

Send

</v-btn>

</v-col>

<v-spacer></v-spacer>

</v-row>

</v-card-text>

</v-card>

</v-container>

</v-card>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-container>

</template>

<script>

import Navbar from '../components/Navbars/Navbar'

// import { auth, db, fieldValue, storage } from '../firebase'

import { auth, db, fieldValue } from '../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

components: {

Navbar,

},

computed: {

// Validation of message box

messageErrors () {

const errors = []

if(!this.$v.message.$dirty) return errors

!this.$v.message.required && errors.push('Enter A Message Before Sending')

return errors

},

},

created() {

this.currentUser = auth.currentUser.uid // Get current users ID

this.today = this.getTodaysDate(this.today) // get today

// Get current users name

db.collection("users").doc(this.currentUser).get().then(doc => {

let user = doc.data()

this.userName = user.firstname + ' ' + user.surname

})

// Get appointments pertaining to the current patient

db.collection("appointments").where("patientID", "==", this.currentUser).onSnapshot(snap => {

let appointment = snap.docChanges()

appointment.forEach(appointment => {

let doctor = appointment.doc.data()

doctor.id = doctor.doctorID

// Get appointments for todays date to get the specific doctor ID

if(doctor.appointmentDate == this.today){

this.chosenDoctor = doctor.id

this.triggerSnackbar("Room Is Active", "success")

// Where doc and patient IDS in same room

db.collection("rooms").where("doctorID", "==", this.chosenDoctor).where("patientID", "==", this.currentUser).onSnapshot(snap => {

// Check room for updates

let rooms = snap.docChanges()

rooms.forEach(rooms => {

let room = rooms.doc.data()

room.id = rooms.doc.id

// Room session starts when a room is added

if(rooms.type == "added"){

this.triggerSnackbar("Room Session Has Begun!")

}

// if the room is deleted, the session has ended

else if(rooms.type == "removed"){

this.triggerSnackbar("Room Session Has Ended!")

// this.roomID = null

this.chatRoom = false

this.preMessageView = true

}

else{

this.triggerSnackbar("New Message")

}

})

this.loadMessages() // Load in messages for this chat

})

}

// If no appointments no chat

else if (doctor.appointmentDate != this.today){

this.snackbar = null

// Appointment is not for today

this.message = null

this.chatRoom = false

this.preMessageView = true

}

else{

this.message = null

this.chatRoom = false

this.preMessageView = true

}

})

})

},

data() {

return {

currentUser: null,

userName: null, // name of logged in user

today: null, // todays date

// snackbar info

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

preMessageView: true, // visibility of preChat

chatRoom: false, // visibilit of chat card

roomID: null, // room where doc and patient share

chosenDoctor: null, // doc of appointment

message: null, // message box message

messages: [], // stores all messages from room

}

},

validations: {

message: { required },

},

methods: {

// Append a 0 to month or date of number is less than or equals 9 to match to appointmentDates

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

getTodaysDate () {

let today = new Date()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

return formattedDate

},

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

// Allows user to delete the room in the rooms collection

destroyRoom () {

console.log("THE ROOM", this.roomID)

if(this.roomID == null){

this.message = null

this.chatRoom = false

this.preMessageView = true

}

else{

db.collection("rooms").doc(this.roomID).delete().then(() => {

this.triggerSnackbar("Room Has Been Deleted!", "success")

}).then(()=> {

this.message = null

this.chatRoom = false

this.preMessageView = true

})

}

},

// Loads in all messages pertaining to the session

loadMessages () {

// Load all rooms where DoctorID is the same as the user

db.collection("rooms").where("patientID", "==", this.currentUser).onSnapshot(snap => {

let rooms = snap.docChanges()

// This gets the data of each doc connected to the user

rooms.forEach(rooms => {

let room = rooms.doc.data()

this.messages = room.message

if(rooms.type == "added"){

// Sorting can only occur once their are messages to be sorted

if(this.messages != null) {

this.preMessageView = false

this.messages.sort((a, b) => {

if(a.timestamp > b.timestamp) return 1

if(a.timestamp < b.timestamp) return -1

})

// Only show the last 50 messages

let len = this.messages.length

let numMessages = this.messages.length < 50 ? this.messages.length : 50

this.messages = this.messages.slice(len - numMessages, len)

this.chatRoom = true

}

else{

this.chatRoom = false

this.preMessageView = true

}

}

else if(rooms.type == "removed"){

// this.roomID = null

this.chatRoom = false

this.preMessageView = true

}

})

})

},

sendMessage () {

db.collection("appointments").where("patientID", "==", this.currentUser).onSnapshot(snap => {

let appointment = snap.docChanges()

appointment.forEach(appointment => {

let doctor = appointment.doc.data()

doctor.id = doctor.doctorID

//Get appointments for todays date to get the specific doctor ID

if(doctor.appointmentDate == this.today){

this.chosenDoctor = doctor.id // Grab the ID of the doctor of appointment

// Where doctor and patient share the same room with their IDs

db.collection("rooms").where("doctorID", "==", this.chosenDoctor).where("patientID", "==", this.currentUser).get().then(snap => {

snap.forEach(doc =>{

let room = doc.id

this.roomID = room

})

if(this.roomID != null){

this.$v.$touch()

this.formTouched = !this.$v.message.$anyDirty

this.errors = this.$v.message.$anyError

// Stop users from entering blank messages

if(this.errors === false && this.formTouched === false){

let messageToSend = this.message

var addMessage = {

patientId: this.currentUser,

name: this.userName,

text: messageToSend,

timestamp: this.timestamp = new Date()

}

var messageSaved = {

message: fieldValue.arrayUnion(addMessage)

}

db.collection("rooms").doc(this.roomID).update(messageSaved).then(() => {

this.message = null

this.$v.$reset()

})

}

}

else{

this.message = null

this.chatRoom = false

this.preMessageView = true

}

})

}

})

})

},

},

}

</script>

<style>

.container {

border-radius: 5px;

}

.chat h2{

font-size: 2.6em;

margin-bottom: 40px;

}

.chat span {

font-size: 1.4em;

}

.chat .time {

display: block;

font-size: 0.8em;

}

.messages {

max-height: 400px;

overflow: auto;

}

.chat-message {

min-height: 40px;

word-wrap: break-word;

margin-bottom: 10px;

}

.right-bubble {

background: #dcf8c6;

width:50%;

margin-left: 50%;

padding: 5px 10px 10px;

}

.left-bubble {

background: #efefef;

width:50%;

margin-right: 50%;

padding: 5px 10px 10px;

}

</style>

## DoctorRooms.vue

<template>

<v-container>

<DoctorNavbar />

<v-card v-if="joinRoom">

<v-card-title></v-card-title>

<v-card-text>

<v-row>

<!-- Allow doctor to view all patients with

appointments on this day -->

<v-col cols="12" sm="6" md="6" lg="6">

<v-select

label="Todays Patients"

:items="patients"

item-text="name"

item-value="value"

outlined

@change="onPatientChanged($event)"

></v-select>

</v-col>

<v-col cols="12" sm="4" md="4" lg="4">

<v-btn

class="primary white--text"

block

@click="createRoom()"

>

Join Room

</v-btn>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Card with close room button -->

<v-card v-if="closeRoom">

<v-card-text>

<v-row>

<v-col cols="12" sm="4" md="4" lg="4">

<v-btn

class="primary white--text"

block

@click="destroyRoom()"

>

Close Room

</v-btn>

</v-col>

</v-row>

</v-card-text>

</v-card>

<!-- Chat Card, css is used to display messages in correct format -->

<v-card v-if="chatRoom" class="mt-6 mb-9">

<v-card-title class="primary lighten-1 white--text">Online Consultation</v-card-title>

<v-container>

<v-card outlined class="messages">

<v-card-text class="chat-message" v-for="(message, index) in messages" :key="index">

<div class="right-bubble" v-if="message.name === userName ">

<div class="subtitle-1">{{ message.name }}&nbsp;</div>

<span class="green-text">{{ message.text }}</span>

</div>

<div class="left-bubble right-align" v-if="message.name !== userName">

<div class="subtitle-1">{{ message.name }}&nbsp;</div>

<span class="green-text">{{ message.text }}</span>

</div>

</v-card-text>

</v-card>

<v-card flat class="mt-6">

<v-card-text>

<v-row>

<!-- Area to type messages -->

<v-col cols="12" sm="12" md="6">

<v-textarea

rows="1"

auto-grow

v-model="message"

outlined

:error-messages="messageErrors"

@input="$v.message.$touch()"

@blur="$v.message.$touch()"

></v-textarea>

</v-col>

<!-- Button to send messages -->

<v-col cols="12" sm="12" md="2">

<v-btn

class="primary white--text ml-6"

block

@click="sendMessage()"

>

Send

</v-btn>

</v-col>

<v-spacer></v-spacer>

</v-row>

</v-card-text>

</v-card>

</v-container>

</v-card>

<v-snackbar

:color="color"

v-model="snackbar"

:timeout="timeout"

:multi-line="multiLine"

>{{ snackbarText }}

</v-snackbar>

</v-container>

</template>

<script>

import DoctorNavbar from '../components/Navbars/DoctorNavbar'

import { auth, db, fieldValue } from '../firebase'

import { required } from 'vuelidate/lib/validators'

export default {

components: {

DoctorNavbar,

},

computed: {

// Validation of message box

messageErrors () {

const errors = []

if(!this.$v.message.$dirty) return errors

!this.$v.message.required && errors.push('Enter A Message Before Sending')

return errors

},

},

created() {

this.currentUser = auth.currentUser.uid

this.today = this.getTodaysDate(this.today)

// Get Doctors name

db.collection("users").doc(this.currentUser).get().then(doc => {

let doctor = doc.data()

this.userName = 'Dr. ' + doctor.firstname + ' ' + doctor.surname

})

this.loadMessages() // Load in the messages for this chat

// Gets all the appointments for the doctor

db.collection("appointments").where("doctorID", "==", this.currentUser).onSnapshot( snap => {

let appointment = snap.docChanges()

appointment.forEach( appointment => {

let patient = appointment.doc.data()

patient.id = patient.patientID

// Push any patients into patients array,

// if they have an appointment on current day

if(patient.appointmentDate == this.today){

if(patient.appointmentType == "Online"){

this.patients.push({

name: patient.firstname + ' ' + patient.surname,

value: patient.id

})

}

}

})

})

},

data() {

return {

currentUser: null,

userName: null,

today: null,

snackbar: false,

color: null,

multiLine: true,

timeout: 5000,

snackbarText: "",

chatRoom: false,

joinRoom: true,

closeRoom: false,

patients: [], // Patient array containing list of patients for todays date

chosenPatient: null,

roomID: null,

timestamp: null,

message: null, // Message being sent

messages: [], // Contains all messages between doc & patient

}

},

validations: {

message: { required },

},

methods: {

// Append a 0 to month or date of number is less than or equals 9 to match to appointmentDates

appendLeadingZeroes(n){

if(n <= 9){

return "0" + n;

}

return n

},

getTodaysDate () {

let today = new Date()

let formattedDate = today.getFullYear() + "-" + this.appendLeadingZeroes(today.getMonth() + 1) + "-" + this.appendLeadingZeroes(today.getDate())

return formattedDate

},

triggerSnackbar (message, color) {

this.snackbarText = message,

this.color = color,

this.snackbar = true

},

onPatientChanged (value) {

// Set the ChosePatient to the value of the selected item

this.chosenPatient = value

console.log(this.chosenPatient)

},

// Create a room to chat

createRoom (){

if(this.chosenPatient == null){

this.triggerSnackbar("Select A Patient To Create A Room For", "error")

}

else{

// Check the rooms collection for any rooms with these two users

db.collection("rooms").where("patientID", "==", this.chosenPatient).where("doctorID", "==", this.currentUser).get().then(snap => {

snap.forEach(doc =>{

let room = doc.id

this.roomID = room

})

// There is no room and therefore one must be created

if(snap.docs.length == 0) {

// Create room collection here and add patientID and doctorID to the room for later reference

db.collection("rooms").add({

doctorID: this.currentUser,

patientID: this.chosenPatient,

}).then( () => {

this.joinRoom = false

this.closeRoom = true

this.chatRoom = true

}).catch(error => {

console.log("Error with Room Creation", error)

})

this.triggerSnackbar("New Room Created!", "success")

}

// There is already a room with chosenPatients and doctors/currentUsers ID

else{

this.triggerSnackbar("Chat Room Opened")

// Open the "chat" to this room as a card to test

this.joinRoom = false

this.closeRoom = true

this.chatRoom = true

this.loadMessages()

}

}).catch(error => {

console.log("Rooms error", error)

})

}

},

// End session and destroy room in rooms collection

destroyRoom () {

db.collection("rooms").doc(this.roomID).delete().then(() => {

this.triggerSnackbar("Room Has Been Deleted!", "success")

}).then(() => {

this.roomID = null

this.chatRoom = false

this.closeRoom = false

this.joinRoom = true

}).catch(error => {

console.log("Room Deletion Error", error)

})

},

sendMessage () {

// Get rooms where patient and doctor exist in the same

db.collection("rooms").where("patientID", "==", this.chosenPatient).where("doctorID", "==", this.currentUser).get().then(snap => {

snap.forEach(doc =>{

let room = doc.id

this.roomID = room

})

if(this.roomID != null){

this.$v.$touch()

this.formTouched = !this.$v.message.$anyDirty

this.errors = this.$v.message.$anyError

// Stop users from entering blank messages

if(this.errors === false && this.formTouched === false){

let messageToSend = this.message

var addMessage = {

doctorId: this.currentUser,

name: this.userName,

text: messageToSend,

timestamp: this.timestamp = new Date()

}

var messageSaved = {

message: fieldValue.arrayUnion(addMessage)

}

db.collection("rooms").doc(this.roomID).update(messageSaved).then(() => {

this.message = null

this.$v.$reset()

})

}

}

else{

this.message = null

this.chatRoom = false

this.joinRoom = true

}

})

},

loadMessages () {

// Load all rooms where DoctorID is the same as the user

db.collection("rooms").where("doctorID", "==", this.currentUser).onSnapshot(snap => {

let rooms = snap.docChanges()

// This gets the data of each doc connected to the user

rooms.forEach(rooms => {

let room = rooms.doc.data()

this.messages = room.message

if(rooms.type == "added"){

// Sorting can only occur once their are messages to be sorted

if(this.messages != null) {

this.messages.sort((a, b) => {

if(a.timestamp > b.timestamp) return 1

if(a.timestamp < b.timestamp) return -1

})

let len = this.messages.length

let numMessages = this.messages.length < 50 ? this.messages.length : 50

this.messages = this.messages.slice(len - numMessages, len)

this.chatRoom = true

}

}

else if(rooms.type == "removed"){

this.roomID = null

this.chatRoom = false

this.closeRoom = false

this.joinRoom = true

}

else{

this.triggerSnackbar("New Message")

}

})

})

},

},

}

</script>

<style>

.container {

border-radius: 5px;

}

.chat h2{

font-size: 2.6em;

margin-bottom: 40px;

}

.chat span {

font-size: 1.4em;

}

.chat .time {

display: block;

font-size: 0.8em;

}

.messages {

max-height: 400px;

overflow: auto;

}

.chat-message {

min-height: 40px;

word-wrap: break-word;

margin-bottom: 10px;

}

.right-bubble {

background: #dcf8c6;

width:50%;

margin-left: 50%;

padding: 5px 10px 10px;

}

.left-bubble {

background: #efefef;

width:50%;

margin-right: 50%;

padding: 5px 10px 10px;

}

</style>

# Plagiarism Declaration

Machine generated alternative text:
 
 
 
 
 
 
Work submitted for assessment which does not include this 
declaration will not be assessed. 
 
 
DECLARATION 
 
*I declare that all material in this submission e.g. thesis/essay/project/assignment is 
entirely my/our own work except where duly acknowledged. 
 
*I have cited the sources of all quotations, paraphrases, summaries of information, 
tables, diagrams or other material; including software and other electronic media in 
which intellectual property rights may reside.  
 
*I have provided a complete bibliography of all works and sources used in the 
preparation of this submission. 
 
*I understand that failure to comply with the Institute’s regulations governing 
plagiarism constitutes a serious offence.    
 
   
 
Student Name: (Printed) ____________________________________________ 
 
Student Number(s):  ____________________________________________ 
 
Signature(s):   ____________________________________________
   
 
Date:    ____________________________________________ 
 
 
 
 
------------------------------------------------------------------------------------------------------- 
Please note: 
a)
 
* Individual declaration is required by each student for joint projects. 
b)
 
Where projects are submitted electronically, students are required to type their name 
under signature. 
c)
 
The Institute regulations on plagarism are set out in Section 10 of Examination and 
Assessment Regulations published each year in the Student Handbook.
 
INSTITUTE
 
of
TECHNOLOGY
CARLOW


30/04/2020

Holly Gunning

C00214010

