ELDER-CARE

A Progressive Web Application (PWA) to Assist the Elderly

Student Name: Craig Lawlor

Student Number: C00184465

Supervisor: Hisain Elshaafi

Submission Date: 12/4/2019

Technical Manual

BSc (Hons) in Software Development

INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

Table of Contents

1. Introduction

The purpose of this document is to display all of the code for the Progressive Web App. The code will be presented in a structured way looking at the main application files first before looking at the individual components. The components are made up of multiple files which will be grouped together.

This will be followed by the screenshots of the database structure with example data included. The code used for the schema of the collections will also be included in this section.

The complete application code can be found on my Github at the following link: https://github.com/craigtupac-96/elder-care-pwa

2. Application Code

2.1 Main Application Code

2.1.1 angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "elder-care-pwa": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "app",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/elder-care-pwa",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets",
              "src/manifest.json"
            ],
            "styles": [
              "src/styles.css",
              "node_modules/bootstrap/dist/css/bootstrap.min.css"
            ],
            "scripts": [
              "node_modules/jquery/dist/jquery.min.js",
              "node_modules/bootstrap/dist/js/bootstrap.min.js"
            ]
          },
          "configurations": {
            "production": {
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
                  "with": "src/environments/environment.prod.ts"
```

```
          }
        ],
        "optimization": true,
        "outputHashing": "all",
        "sourceMap": false,
        "extractCss": true,
        "namedChunks": false,
        "aot": true,
        "extractLicenses": true,
        "vendorChunk": false,
        "buildOptimizer": true,
        "budgets": [
          {
            "type": "initial",
            "maximumWarning": "2mb",
            "maximumError": "5mb"
          }
        ],
        "serviceWorker": true
      }
    }
  },
  "serve": {
    "builder": "@angular-devkit/build-angular:dev-server",
    "options": {
      "browserTarget": "elder-care-pwa:build"
    },
    "configurations": {
      "production": {
        "browserTarget": "elder-care-pwa:build:production"
      }
    }
  },
  "extract-i18n": {
    "builder": "@angular-devkit/build-angular:extract-i18n",
    "options": {
      "browserTarget": "elder-care-pwa:build"
    }
  },
  "test": {
    "builder": "@angular-devkit/build-angular:karma",
    "options": {
      "main": "src/test.ts",
      "polyfills": "src/polyfills.ts",
      "tsConfig": "src/tsconfig.spec.json",
      "karmaConfig": "src/karma.conf.js",
      "styles": [
        "src/styles.css"
      ],
      "scripts": [],
```

```
          "assets": [
            "src/favicon.ico",
            "src/assets",
            "src/manifest.json"
          ]
        }
      },
      "lint": {
        "builder": "@angular-devkit/build-angular:tslint",
        "options": {
          "tsConfig": [
            "src/tsconfig.app.json",
            "src/tsconfig.spec.json"
          ],
          "exclude": [
            "**/node_modules/**"
          ]
        }
      }
    }
  },
  "elder-care-pwa-e2e": {
    "root": "e2e/",
    "projectType": "application",
    "prefix": "",
    "architect": {
      "e2e": {
        "builder": "@angular-devkit/build-angular:protractor",
        "options": {
          "protractorConfig": "e2e/protractor.conf.js",
          "devServerTarget": "elder-care-pwa:serve"
        },
        "configurations": {
          "production": {
            "devServerTarget": "elder-care-pwa:serve:production"
          }
        }
      },
      "lint": {
        "builder": "@angular-devkit/build-angular:tslint",
        "options": {
          "tsConfig": "e2e/tsconfig.e2e.json",
          "exclude": [
            "**/node_modules/**"
          ]
        }
      }
    }
  },
```

```
  "defaultProject": "elder-care-pwa"
}
```

2.1.2 package.json

```
{
  "name": "elder-care-pwa",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "start:prod": "ng build --prod && lite-server --baseDir dist/elder-care-pwa",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~7.1.0",
    "@angular/common": "~7.1.0",
    "@angular/compiler": "~7.1.0",
    "@angular/core": "~7.1.0",
    "@angular/forms": "~7.1.0",
    "@angular/platform-browser": "~7.1.0",
    "@angular/platform-browser-dynamic": "~7.1.0",
    "@angular/pwa": "^0.12.1",
    "@angular/router": "~7.1.0",
    "@angular/service-worker": "~7.1.0",
    "angular2": "^2.0.0-beta.21",
    "angularfire2": "^5.1.2",
    "bootstrap": "^4.2.1",
    "core-js": "^2.5.4",
    "firebase": "^5.9.0",
    "jquery": "^3.3.1",
    "popper": "^1.0.1",
    "rxjs": "~6.3.3",
    "rxjs-compat": "^6.4.0",
    "tslib": "^1.9.0",
    "zone.js": "~0.8.26"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^0.13.8",
    "@angular/cli": "~7.1.3",
    "@angular/compiler-cli": "^7.2.12",
    "@angular/language-service": "~7.1.0",
    "@types/jasmine": "~2.8.8",
    "@types/jasminewd2": "~2.0.3",
    "@types/node": "~8.9.4",
    "codelyzer": "~4.5.0",
```

```
    "jasmine-core": "~2.99.1",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~3.1.1",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~2.0.1",
    "karma-jasmine": "~1.1.2",
    "karma-jasmine-html-reporter": "^0.2.2",
    "lite-server": "^2.4.0",
    "protractor": "~5.4.0",
    "ts-node": "~7.0.0",
    "tslint": "~5.11.0",
    "typescript": "~3.1.6"
  }
}
```

2.1.3 index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ElderCarePwa</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/icons/icon-128x128.png">
  <link href="https://fonts.googleapis.com/css?family=Julius+Sans+One" rel="stylesheet">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
        integrity="sha384-50oBUHEmvpQ+1lW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf"
crossorigin="anonymous">

  <link rel="manifest" href="manifest.json">
  <meta name="theme-color" content="#1976d2">
</head>
<body>
  <app-root></app-root>
  <noscript>Please enable JavaScript to continue using this application.</noscript>
</body>
</html>
```

2.1.4 style.css

```
html, body{
  background-color: #333;
  margin: 0;
  padding: 0;
}

form{
```

```
  padding: 1.5rem;
}

.headings{
  font-family: 'Julius Sans One', sans-serif;
  font-weight: normal;
  letter-spacing: 3.5px;
}

#heading2 {
  color: #12aa95;
  font-size: 1.9rem;
  text-align: center;
}

#heading3 {
  color: #12aa95;
  font-size: 1.4rem;
}

.main-content{
  background-color: #edfffc;
  margin: 1rem;
  padding:2rem;
  box-shadow: 2px 2px 8px 1px grey;
}

@media (max-width: 600px) {
  .main-content {
    width: 95%;
  }
}

button[type="submit"]{
  margin-top: 1rem;
}

.formFooterText{
  text-align: center;
}

.formFooterLink{
  color: #06BD89;
}
```

2.1.5 manifest.json

```
{
  "name": "elder-care-pwa",
```

```
  "short_name": "elder-care-pwa",
  "theme_color": "#12aa95",
  "background_color": "#ffffff",
  "display": "standalone",
  "scope": "/",
  "start_url": "/",
  "icons": [
    {
      "src": "assets/icons/icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-96x96.png",
      "sizes": "96x96",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-152x152.png",
      "sizes": "152x152",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-384x384.png",
      "sizes": "384x384",
      "type": "image/png"
    },
    {
      "src": "assets/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

2.1.6 environment.ts

```
export const environment = {
  production: true,
  firebase: {
    apiKey: 'AIzaSyD5kP5_tKVhaSx_d99NPMQ2-QPB-W4_pXA',
    authDomain: 'elder-care-pwa.firebaseapp.com',
    databaseURL: 'https://elder-care-pwa.firebaseio.com',
    projectId: 'elder-care-pwa',
    storageBucket: 'elder-care-pwa.appspot.com',
    messagingSenderId: '433319684787'
  }
};
```

2.2 App Component Code

2.2.1 app.component.html

```
<app-nav></app-nav>

<router-outlet></router-outlet>

<app-footer></app-footer>
```

2.2.2 app.component.ts

```
import {Component} from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'elder-care-pwa';
}
```

2.2.3 app.component.spec.ts

```
import { TestBed, async } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
```

```
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'elder-care-pwa'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('elder-care-pwa');
  });

  it('should render title in a h1 tag', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('Welcome to elder-care-pwa!');
  });
});
```

2.2.4 app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { AngularFireModule } from '@angular/fire';
import { AngularFirestoreModule } from '@angular/fire/firestore';
import { AngularFireAuthModule} from '@angular/fire/auth';
import { AngularFireDatabaseModule } from '@angular/fire/database';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ServiceWorkerModule } from '@angular/service-worker';
import { environment } from '../environments/environment';
import { NavComponent } from './nav/nav.component';
```

```
import { FooterComponent } from './footer/footer.component';
import { HomeComponent } from './home/home.component';
import { SignupComponent } from './auth/signup/signup.component';
import { LoginComponent } from './auth/login/login.component';
import { AppointmentsComponent } from './appointments/appointments.component';
import { DiaryComponent } from './diary/diary.component';
import { AuthService } from './auth/auth.service';
import { DetailsComponent } from './auth/details/details.component';
import { EmergencyContactsComponent } from './emergency-contacts/emergency-contacts.component';
import { UpdateDetailsComponent } from './update-details/update-details.component';
import { ToDoComponent } from './to-do/to-do.component';
import { MedicationComponent } from './medication/medication.component';
import { DashAppointmentsComponent } from './dash-appointments/dash-appointments.component';
import { DashDiaryEntryComponent } from './dash-diary-entry/dash-diary-entry.component';
import { DashGroupDiaryEntryComponent } from './dash-group-diary-entry/dash-group-diary-entry.component';

@NgModule({
  declarations: [
    AppComponent,
    NavComponent,
    FooterComponent,
    HomeComponent,
    SignupComponent,
    LoginComponent,
    AppointmentsComponent,
    DiaryComponent,
    DetailsComponent,
    EmergencyContactsComponent,
    UpdateDetailsComponent,
    ToDoComponent,
    MedicationComponent,
    DashAppointmentsComponent,
    DashDiaryEntryComponent,
    DashGroupDiaryEntryComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ServiceWorkerModule.register('ngsw-worker.js', { enabled: environment.production }),
    FormsModule,
    ReactiveFormsModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFirestoreModule,
    AngularFireAuthModule,
    AngularFireDatabaseModule
  ],
  providers: [AuthService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

2.2.5 app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './home/home.component';
import { SignupComponent } from './auth/signup/signup.component';
import { LoginComponent } from './auth/login/login.component';
import { AppointmentsComponent } from './appointments/appointments.component';
import { DiaryComponent } from './diary/diary.component';
import { EmergencyContactsComponent } from './emergency-contacts/emergency-contacts.component';
import { AuthGuard } from './auth/auth.guard';
import { DetailsComponent } from './auth/details/details.component';
import { UpdateDetailsComponent } from './update-details/update-details.component';
import { MedicationComponent } from './medication/medication.component';

const routes: Routes = [
  { path: '', component: HomeComponent, canActivate: [AuthGuard] },
  { path: 'signup', component: SignupComponent },
  { path: 'login', component: LoginComponent },
  { path: 'details', component: DetailsComponent, canActivate: [AuthGuard] },
  { path: 'appointments', component: AppointmentsComponent, canActivate: [AuthGuard] },
  { path: 'diary', component: DiaryComponent, canActivate: [AuthGuard] },
  { path: 'emergency-contacts', component: EmergencyContactsComponent, canActivate: [AuthGuard] },
  { path: 'update-details', component: UpdateDetailsComponent, canActivate: [AuthGuard] },
  { path: 'medication', component: MedicationComponent, canActivate: [AuthGuard] }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: [AuthGuard] // normally done in app.module.ts but guards are ok to do here
})
export class AppRoutingModule { }
```

2.3 Auth Component Code

2.3.1 auth.service.ts

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { Subject } from 'rxjs/Subject';
import { AngularFireAuth } from '@angular/fire/auth';
import { AngularFirestore, AngularFirestoreDocument } from '@angular/fire/firestore';
import { AuthData } from './auth-data.model';
```

```
import { Observable, of } from 'rxjs';
import { User } from './user.model';
import { FullEmailList } from './lists.model';
import { switchMap } from 'rxjs/operators';

@Injectable()
export class AuthService {
  authChange = new Subject<boolean>();
  private isAuthenticated = false;
  user: Observable<User>;

  constructor(private router: Router, private aFireAuth: AngularFireAuth, private aFirestore: AngularFirestore) {
    this.user = this.aFireAuth.authState.pipe(
      switchMap(user => {
        if (user) {
          return this.aFirestore.doc('users/' + user.uid).valueChanges();
        } else {
          return of(null);
        }
      }));
  }

  registerUser(authData: AuthData) {
    this.aFireAuth.auth
      .createUserWithEmailAndPassword(authData.email, authData.password)
      .then(result => {
        console.log(result);
        this.setInitialUserDoc(result.user);
        this.addToEmailList(result.user);
        this.authSuccess();
      })
      .catch(error => {
        console.log(error);
      });
  }

  // the initial user document is set with the uid and email stored in firebase authentication
  private setInitialUserDoc(user) {
    const userRef: AngularFirestoreDocument<User> = this.aFirestore.doc('users/' + user.uid);
    const data: User = {
      uid: user.uid,
      email: user.email,
      detailsComplete: false // detailsComplete set to false to force redirection to the details form
    };
    userRef.set(data);
  }

  private addToEmailList(user) {
    const emailListRef: AngularFirestoreDocument<FullEmailList> = this.aFirestore.doc('fullEmailList/' + user.email);
    const data: FullEmailList = {
```

```
      email: user.email
    };
    emailListRef.set(data);
  }

  login(authData: AuthData) {
    this.aFireAuth.auth
      .signInWithEmailAndPassword(authData.email, authData.password)
      .then(result => {
        console.log(result);
        this.authSuccess();
      })
      .catch(error => {
        console.log(error);
      });
  }

  logout() {
    this.isAuthenticated = false;
    this.authChange.next(false);
    this.router.navigate(['/login']);
  }

  isAuth() {
    return this.isAuthenticated;
  }

  private authSuccess() {
    this.isAuthenticated = true;
    this.authChange.next(true);
    this.router.navigate(['/']);
  }

}
```

2.3.2 auth.guard.ts

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
Router} from '@angular/router';
import { AuthService } from './auth.service';
import {Observable} from 'rxjs';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/take';
import { AngularFirestore } from '@angular/fire/firestore';
import { AngularFireAuth } from '@angular/fire/auth';
import * as firebase from 'firebase';
import { tap, map, take } from 'rxjs/operators';
```

```
@Injectable()
export class AuthGuard implements CanActivate {
  userId: string;
  myBool:  boolean;

  constructor(private authS: AuthService, private router: Router, private aFireAuth: AngularFireAuth,
              private aFirestore: AngularFirestore) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> | boolean  {

    return this.authS.user.pipe(
      take(1),
      map(user => !!(user)), // <-- map to boolean
      tap(loggedIn => {
        if (!loggedIn) {
          // If not logged in redirect to login screen
          return this.router.navigate(['/login']);
        } else {
          this.userId = firebase.auth().currentUser.uid;
          this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
            .get().then(querySnap => {
            querySnap.forEach( (doc) => {
              // Assign value in detailsComplete in firestore (for the user currently logged) in to myBool
              this.myBool = doc.data().detailsComplete;
            });
          }).then( event => {
            // using then to wait until the above query completes and then carrying out checks via if statements
            if (!this.authS.isAuth()) {
              // user is not authenticated, redirect to login
              return this.router.navigate(['/login']);
            } else if (!this.myBool) {
              // details are not set, redirect to details screen
              return this.router.navigate(['/details']);
            }
          }
        );
        }
      })
    );
  }
}
```

2.3.3 auth-data.model.ts

```
export interface AuthData {
  email: string;
  password: string;
}
```

2.3.4 lists.model.ts

```
import {firestore} from 'firebase';

export interface FullEmailList {
  email: string;
}

export interface PrimaryEmailList {
  email: string;
}

export interface AssistantRecord {
  uid: string;
  firstName: string;
  lastName: string;
  phoneNumber: string;
}

export interface ToDoList {
  record: string;
  checked: boolean;
}

export interface MedicationList {
  name: string;
  dosage: string;
  frequency: string;
  specialInstructions: string;
}

export interface DiaryEntries {
  startDate: firestore.Timestamp;
  entry: string;
  lastEdit: firestore.Timestamp;
}

export interface GroupDiaryEntries {
  startDate: firestore.Timestamp;
  entry: string;
  displayName: string;
  lastEdit: firestore.Timestamp;
}

export interface Appointment {
  dateOnly: firestore.Timestamp;
  timeOnly: firestore.Timestamp;
  monthOnly: firestore.Timestamp;
  title: string;
  description: string;
```

```
  location: string;
}
```

2.3.5 user.model.ts

```
import {AssistantRecord} from './lists.model';

export interface Role {
  type: string;
}

export interface User {
  uid: string;
  email: string;
  primaryEmail?: string;
  username?: string;
  firstName?: string;
  lastName?: string;
  phoneNumber?: string;
  detailsComplete?: boolean;
  roles?: Role;
  assistantRecords?: AssistantRecord[];
}
```

2.3.6 Login Component Code

2.3.6.1 login.component.html

```
<div class="main-content container" id="loginScreen">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>

  <form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Email</label>
        <input type="email" class="form-control" placeholder="Enter email" formControlName="email">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Password</label>
        <input type="password" class="form-control" placeholder="Enter password" formControlName="password">
      </div>
    </div>
    <div class="text-center">
      <button type="submit" name="login" class="btn login-btn" >Login</button>
```

```
    </div>
  </form>

  <hr>

  <p class="formFooterText">Don't have an account? <a class="formFooterLink" routerLink="/signup">Sign
Up!</a></p>

</div>
```

2.3.6.2 login.component.css

```
.login-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

#loginScreen{
  margin-top: 8rem;
  margin-bottom: 14rem;
}

label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}
```

2.3.6.3 login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  title = 'Login';
  loginForm: FormGroup;

  constructor(private authService: AuthService) { }

  ngOnInit() {
```

```
    this.loginForm = new FormGroup({
      email: new FormControl('', {
        validators: [Validators.required, Validators.email]
      }),
      password: new FormControl('', { validators: [Validators.required] })
    });
  }

  onSubmit() {
    this.authService.login({
      email: this.loginForm.value.email,
      password: this.loginForm.value.password
    });
  }
}
```

2.3.7 Signup Component Code

2.3.7.1 signup.component.html

```
<div class="main-content container" id="signupScreen">
    <h2 class="headings" id="heading2">{{ title }}</h2>
    <hr>

    <form [formGroup]="signupForm" (ngSubmit)="signupForm.valid && onSubmit()">

      <div class="form-row justify-content-center">
        <div class="form-group col-md-4 col-sm-10" [ngClass]="{'has-error': signupForm.get('email').errors &&
        (signupForm.get('email').touched || signupForm.get('email').dirty)}">
          <label>Email</label>
          <input type="email" class="form-control" placeholder="Enter email" formControlName="email">
        </div>
      </div>

      <div class="form-row justify-content-center">
          <div class="form-group col-md-4 col-sm-10">
            <span class="alert alert-danger" *ngIf="signupForm.get('email').errors &&
            (signupForm.get('email').touched || signupForm.get('email').dirty)">
              <span *ngIf="signupForm.get('email').errors.required">
                Email is required
              </span>
              <span *ngIf="signupForm.get('email').errors.email">
                Must be a valid email address
              </span>
            </span>
          </div>
      </div>
```

```
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Password</label>
        <input type="password" name="password" class="form-control" placeholder="Enter password"
formControlName="password" required
              [ngClass]="{'has-error': signupForm.get('password').errors &&
      (signupForm.get('password').touched || signupForm.get('password').dirty)}">
      </div>
    </div>

    <div class="form-row justify-content-center">
      <div class="form-group col-md-7 col-sm-10">
        <span class="alert alert-danger" *ngIf="signupForm.get('password').errors &&
        (signupForm.get('password').touched || signupForm.get('password').dirty)">
          <span *ngIf="signupForm.get('password').errors.required">
            Password is required
          </span>
          <span *ngIf="signupForm.get('password').errors.pattern">
            Must be at least 8 characters long, contain 1 uppercase letter & 1 number
          </span>
        </span>
      </div>
    </div>

    <div class="text-center">
      <button type="submit" class="btn submit-btn">Sign Up</button>
    </div>

  </form>

  <hr>

  <p class="formFooterText">Already have an account? <a class="formFooterLink" routerLink="/login">Log
In!</a></p>

</div>
```

2.3.7.2 signup.component.css

```
.button-label{
  color: white;
}

.is-invalid {
  color: red;
}

label{
```

```
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}

.form-btn {
  color: #FFFFFF;
  background-color: #7BDBC0;
  border-color: #FFFFFF;
}

.form-btn:hover,
.form-btn:focus,
.form-btn:active,
.form-btn.active,
.open .dropdown-toggle.form-btn {
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.submit-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

#signupScreen{
  margin-top: 8rem;
  margin-bottom: 12rem;
}
```

2.3.7.3 signup.component.ts

```
import {Component, OnInit} from '@angular/core';
import {FormGroup, Validators, FormBuilder} from '@angular/forms';
import { AuthService } from '../auth.service';
import { AngularFireAuth } from '@angular/fire/auth';
import {AbstractControl} from 'angular2/common';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css'],
})
export class SignupComponent implements OnInit {
  title = 'Sign Up';
  signupForm: FormGroup;
```

```
  submitted = false;
  match: boolean;

  constructor(private authS: AuthService, private aFireAuth: AngularFireAuth, public fb: FormBuilder) {
  }

  ngOnInit() {
    this.signupForm = this.fb.group({
      'email': ['', [
        Validators.email,
        Validators.required
        ]
      ],
      'password': ['', [
        Validators.pattern('^(?=.*[0-9])(?=.*[a-zA-Z])([a-zA-Z0-9]+)$'),
        Validators.minLength(8),
        Validators.required
        ]
      ]
    });
  }

  get email() { return this.signupForm.get('email'); }
  get password() { return this.signupForm.get('password'); }

  signUp() {
    return this.authS.registerUser({
      email: this.email.value,
      password: this.password.value,
    });
  }

  onSubmit() {
    this.submitted = true;
    if (this.signupForm.invalid) {
      alert('Form submission not complete');
      return;
    } else {
      this.signUp();
    }
  }

}
```

2.3.8 Details Component Code

2.3.8.1 details.component.html

```
<div class="main-content container" id="detailsScreen">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>

  <form [formGroup]="detailForm" (ngSubmit)="setDetails()">

    <div class="form-row justify-content-center">
      <div class="form-group col-md-2 col-sm-2">
        <label>Acount Type</label>
        <select class="form-control" formControlName="roleType" (click)="checkType()">
          <option value="" disabled selected>Select...</option>
          <option *ngFor="let role of roles" [value]="role.type">
            {{role.type}}
          </option>
        </select>
      </div>
      <div class="form-group col-md-4 col-sm-10" [hidden]="roleType.value!='Assistant'">
        <label>Family Email</label>
        <input type="email" [id]="primaryExists ? 'borderSuccess' : 'borderDanger'" class="form-control"
               placeholder="Enter family email" formControlName="primaryEmail" (keyup)="checkPrimaryEmail()">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-2 col-sm-2">
        <!--Spacing for now-->
      </div>
      <div class="form-group col-md-4 col-sm-10">
      <p id="success" *ngIf="primaryExists && primaryEmail.value">
        {{primaryEmail.value}} exists
      </p>
      <p id="danger" *ngIf="!primaryExists && primaryEmail.value">
        {{primaryEmail.value}} doesn't exist
      </p>
      </div>
    </div>

    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>First Name</label>
        <input type="text" class="form-control" placeholder="Enter first name" formControlName="firstName"
               [ngClass]="{'has-error': detailForm.get('firstName').errors &&
      (detailForm.get('firstName').touched || detailForm.get('firstName').dirty)}">
      </div>
      <div class="form-group col-md-4 col-sm-10">
```

```
    <label>Last Name</label>
    <input type="text" class="form-control" placeholder="Enter last name" formControlName="lastName"
          [ngClass]="{'has-error': detailForm.get('lastName').errors &&
    (detailForm.get('lastName').touched || detailForm.get('lastName').dirty)}">
   </div>
  </div>

  <div class="form-row justify-content-center">
   <div class="form-group col-md-7 col-sm-10">
       <span class="alert alert-danger" *ngIf="detailForm.get('firstName').errors &&
       (detailForm.get('firstName').touched || detailForm.get('firstName').dirty)">
         <span *ngIf="detailForm.get('firstName').errors.required">
          First Name is required
         </span>
       </span>
   </div>
   <div class="form-group col-md-7 col-sm-10">
       <span class="alert alert-danger" *ngIf="detailForm.get('lastName').errors &&
       (detailForm.get('lastName').touched || detailForm.get('lastName').dirty)">
         <span *ngIf="detailForm.get('lastName').errors.required">
          Last Name is required
         </span>
       </span>
   </div>
  </div>



  <div class="form-row justify-content-center">
   <div class="form-group col-md-4 col-sm-10">
    <label>Phone Number</label>
    <input type="text" class="form-control" placeholder="Enter phone number" formControlName="phoneNumber"
          [ngClass]="{'has-error': detailForm.get('phoneNumber').errors &&
    (detailForm.get('phoneNumber').touched || detailForm.get('phoneNumber').dirty)}">
   </div>
  </div>

  <div class="form-row justify-content-center">
   <div class="form-group col-md-7 col-sm-10">
       <span class="alert alert-danger" *ngIf="detailForm.get('phoneNumber').errors &&
       (detailForm.get('phoneNumber').touched || detailForm.get('phoneNumber').dirty)">
         <span *ngIf="detailForm.get('phoneNumber').errors.required">
          Phone number is required
         </span>
       </span>
   </div>
  </div>
```

```
    <div class="text-center">
      <button type="submit" class="btn submit-btn" [disabled]="!canSubmit()">Submit</button>
    </div>

  </form>

</div>
```

2.3.8.2 details.component.css

```
.button-label{
  color: white;
}

label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}

.form-btn {
  color: #FFFFFF;
  background-color: #7BDBC0;
  border-color: #FFFFFF;
}

.form-btn:hover,
.form-btn:focus,
.form-btn:active,
.form-btn.active,
.open .dropdown-toggle.form-btn {
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.submit-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

#detailsScreen{
  margin-top: 4rem;
  margin-bottom: 12rem;
}
```

```
#borderSuccess{
  background-color: #d7ffd3;
}

#borderDanger{
  background-color: #ffd6d6;
}

#success{
  color: green;
  font-weight: bold;
}

#danger{
  color: red;
  font-weight: bold;
}
```

2.3.8.3 details.component.ts

```
import {Component, Injectable, OnInit} from '@angular/core';
import { AuthService } from '../auth.service';
import { AngularFireAuth } from '@angular/fire/auth';
import { AngularFirestore, AngularFirestoreDocument, AngularFirestoreCollection } from '@angular/fire/firestore';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {Router} from '@angular/router';
import {Role, User} from '../user.model';
import {AssistantRecord, PrimaryEmailList} from '../lists.model';
import * as firebase from 'firebase';
import {Observable} from 'rxjs';

@Component({
  selector: 'app-details',
  templateUrl: './details.component.html',
  styleUrls: ['./details.component.css']
})
export class DetailsComponent implements OnInit {
  title = 'Complete details to continue';
  detailForm: FormGroup;
  userId: string;
  roles: Role[];
  selectedType: string;
  priEmail: Observable<PrimaryEmailList[]>;
  primaryEmailList: AngularFirestoreCollection<PrimaryEmailList>;
  primaryExists: boolean;
  primaryEmailString: string;
  primaryUidString: string;
  submit: boolean;

  constructor( private authS: AuthService, private aFireAuth: AngularFireAuth,
```

```
        private aFirestore: AngularFirestore, public fb: FormBuilder, private router: Router ) {
  this.aFireAuth.authState.subscribe( user => {
    if (user) {
      this.userId = user.uid;
    }
  });
}

ngOnInit() {
  this.roles = [
    {type: 'Primary'},
    {type: 'Assistant'}
  ],
  this.detailForm = this.fb.group({
    'roleType': ['', Validators.required],
    'firstName': ['', Validators.required],
    'lastName': ['', Validators.required],
    'phoneNumber': ['', Validators.required],
    'primaryEmail': ['']
  });
}

get firstName() { return this.detailForm.get('firstName'); }
get lastName() { return this.detailForm.get('lastName'); }
get phoneNumber() { return this.detailForm.get('phoneNumber'); }
get roleType() { return this.detailForm.get('roleType'); }
get primaryEmail() { return this.detailForm.get('primaryEmail'); }

setDetails() {
  const data = {
    firstName: this.firstName.value,
    lastName: this.lastName.value,
    phoneNumber: this.phoneNumber.value,
    role: this.roleType.value,
    detailsComplete: true
  };
  this.aFirestore.doc('users/' + this.userId).set(data, {merge: true});
  // check if it is an assistant account or primary account
  this.ifAssistantAccount();
  this.ifPrimaryAccount();
}

checkPrimaryEmail() {
  // get the list of primary emails from firestore
  this.primaryEmailList = this.aFirestore.collection('primaryEmailList', ref =>
    ref.where('email', '==', this.detailForm.value.primaryEmail));
  this.priEmail = this.primaryEmailList.snapshotChanges().map(actions => {
    return actions.map(action => {
      const data = action.payload.doc.data();
      // assign the result email to primaryString
```

```
      this.primaryEmailString = action.payload.doc.data().email;
      return {email: data.email};
    });
  });
  // assign true to primaryExists if length > 0, otherwise false
  this.priEmail.subscribe(snapshot => {
    if (snapshot.length === 0) {
      this.primaryExists = false;
    } else {
      this.primaryExists = true;
    }
  });
  return this.primaryExists;
}

private ifAssistantAccount() {
  // if assistant account add the primary email address to their details
  if (this.checkType() === 'Assistant') {
    const additionalData = {
      primaryEmail: this.primaryEmail.value
    };
    this.addAssitantDetailsToPrimaryAcc();
    this.aFirestore.doc('users/' + this.userId).set(additionalData, {merge: true});
    this.router.navigate(['/']);
  }
}

private ifPrimaryAccount() {
  if (this.checkType() === 'Primary') {
    // this.addToPrimaryEmailList();
    const emailListRef: AngularFirestoreDocument<PrimaryEmailList> = this.aFirestore.doc(
      'primaryEmailList/' + firebase.auth().currentUser.email);
    const emailData: PrimaryEmailList = {
      email: firebase.auth().currentUser.email
    };
    emailListRef.set(emailData);
    // create the contact list for the primary user, saving their own details for reference
    const userRef: AngularFirestoreCollection<AssistantRecord> = this.aFirestore.collection('users/' + this.userId +
'/contactList');
    const recordData: AssistantRecord = {
      uid: this.userId,
      firstName: this.firstName.value,
      lastName: this.lastName.value,
      phoneNumber: this.phoneNumber.value,
    };
    userRef.add(recordData);
  }
  this.router.navigate(['/']);
}
```

```
  private addAssitantDetailsToPrimaryAcc() {
    // Get the uid for the primary account
    this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
      .get().then(querySnap => {
        querySnap.forEach( (doc) => {
          this.primaryUidString = doc.data().uid;
        });
      }).then( event => {
        const assistantData: AssistantRecord = {
          uid: firebase.auth().currentUser.uid,
          firstName: this.firstName.value,
          lastName: this.lastName.value,
          phoneNumber: this.phoneNumber.value,
        };
        // add assistants contact details to the primary contact list
        return this.aFirestore.collection('users/' + this.primaryUidString + '/contactList').add(assistantData);
    });
  }

  checkType() {
    this.selectedType = this.detailForm.value.roleType;
    return this.selectedType;
  }

  canSubmit() {
    this.submit = false;
    if (this.selectedType === 'Primary') {
      this.submit = true;
    }
    if (this.primaryExists ) {
      this.submit = true;
    }
    return this.submit;
  }
}
```

2.4 Nav Component Code

2.4.1 nav.component.html

```
<nav class="navbar navbar-expand-lg navbar-dark">
  <a class="navbar-brand" routerLink="/">{{ title }}</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
```

```
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item" *ngIf="isAuth">
        <a class="nav-link" routerLink="/">Dashboard</a>
      </li>
      <li class="nav-item" *ngIf="isAuth">
        <a class="nav-link" routerLink="/appointments">Appointments & Events</a>
      </li>
      <li class="nav-item" *ngIf="isAuth">
        <a class="nav-link" routerLink="/diary">Diary</a>
      </li>
      <li class="nav-item" *ngIf="isAuth">
        <a class="nav-link" routerLink="/medication">Medication</a>
      </li>
      <li class="nav-item" *ngIf="!isAuth">
        <a class="nav-link" routerLink="/signup">Sign up</a>
      </li>
      <li class="nav-item" *ngIf="!isAuth">
        <a class="nav-link" routerLink="/login">Login</a>
      </li>
      <li class="nav-item dropdown" *ngIf="isAuth">
        <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true"
          aria-expanded="false">{{displayName}}</a>
        <div class="dropdown-menu dropdown-menu-right" id="dropName">
          <a class="dropdown-item" routerLink="/update-details">Update Details</a>
          <a class="dropdown-item" routerLink="/emergency-contacts">Emergency Contacts</a>
          <a class="dropdown-item" routerLink="/medication">Medication</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" (click)="onLogout()">Logout</a>
        </div>
      </li>
    </ul>
  </div>
</nav>
```

2.4.2 nav.component.css

```
.navbar{
  background-color: #12aa95;
  font-family: 'Julius Sans One', sans-serif;
  box-shadow: 2px 2px 8px 1px grey;
}

.navbar li a{
  color: white;
  margin-right: 2rem;
  text-align: center;
}
```

```
.navbar li a:hover{
  background-color: white;
  color: #12aa95;
  border-radius: .2rem;
}

.navbar-brand {
  color: white;
  font-size: 1.9rem;
  font-weight: normal;/*removes bold*/
  letter-spacing: 3.5px;/*spaces out leters nicely*/
}

a {
  color: #aaa;
  font-weight: 200;
}

.dropdown-toggle{
  border: 1px solid white;
  border-radius: .2rem;
}

.dropdown-menu{
  background-color: #edfffc;
}

#dropName a {
  color: #12aa95;
}

#dropName a:hover {
  background-color: #12aa95;
  color: white;
}
```

2.4.3 nav.component.ts

```
import {Component, OnInit, OnDestroy} from '@angular/core';
import { AuthService } from '../auth/auth.service';
import { AngularFireAuth } from '@angular/fire/auth';
import { Subscription } from 'rxjs/Subscription';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';

@Component({
  selector: 'app-nav',
```

```
  templateUrl: './nav.component.html',
  styleUrls: ['./nav.component.css'],
})
export class NavComponent implements OnInit, OnDestroy {
  // private title: string;
  title = 'Elder-Care';
  isAuth = false;
  authSubscription: Subscription;
  userId: string;
  displayName: string;
  // accountType: string;

  constructor(private authS: AuthService,  private aFireAuth: AngularFireAuth,
  private aFirestore: AngularFirestore) {
  }

  ngOnInit() {
    this.authSubscription = this.authS.authChange.subscribe(authStatus => {
      this.isAuth = authStatus;
      if (this.isAuth) {
        this.userId = firebase.auth().currentUser.uid;
        // get user information
        this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
          .get().then(querySnap => {
          querySnap.forEach( (doc) => {
            this.displayName = doc.data().firstName + ' ' + doc.data().lastName;
            if (this.displayName === 'undefined undefined') {
              this.displayName = 'More';
            }
          });
        }).then( event => {
        });
      }
    });
  }

  onLogout() {
    this.authS.logout();
  }

  ngOnDestroy() {
    this.authSubscription.unsubscribe();
  }

}
```

2.5 Home Component Code

2.5.1 home.component.html

```
<div class="main-content container">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <div class="dash-wrap">
    <app-dash-appointments></app-dash-appointments>
  <!--Only show this component if the user is primary-->
  <app-to-do *ngIf="isPrimary()"></app-to-do>
  <app-dash-diary-entry *ngIf="isPrimary()"></app-dash-diary-entry>
  <app-dash-group-diary-entry *ngIf="!isPrimary()"></app-dash-group-diary-entry>
  </div>

</div>
```

2.5.2 home.component.css

```
.main-content{
  background-color: unset;
  margin: 0 auto;
  padding: 0;
  box-shadow: none;
}

.dash-wrap {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  margin-bottom: 4rem;
}

@media (max-width: 600px) {
  .main-content {
    width: 96%;
    margin-left: 0;
    margin-right: 0;
  }
  .dash-wrap {
    display: flex;
    flex-direction: column;
  }
}

#heading2 {
  margin-top: 1rem;
```

```
  color: #12aa95;
  font-size: 1.9rem;
  text-align: center;
}
```

2.5.3 home.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../auth/auth.service';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  title = 'Dashboard';
  userId: string;
  accountType: string;

  constructor(public auth: AuthService, private aFirestore: AngularFirestore) {
    this.userId = firebase.auth().currentUser.uid;
    this.aFirestore.firestore.collection('users/').where('uid', '==', this.userId)
      .get().then(querySnap => {
        querySnap.forEach((doc) => {
          this.title = doc.data().firstName + "'s Dashboard";
          this.accountType = doc.data().role;
        });
    });
  }

  ngOnInit() {
  }

  isPrimary() {
    if (this.accountType === 'Primary') {
      return true;
    } else {
      return false;
    }
  }

}
```

2.6 Footer Component Code

2.6.1 footer.component.html

```
<footer class="footer">
  <div class="container pull-left">
    <div class="row">
      <div class="col left-col">
        <div class="dateBox">
          {{ currentDate | date:"fullDate" }}
        </div>

      </div>
      <div class="col-6 middle-col">
        Elder-Care &copy; 2019
      </div>
      <div class="col right-col" *ngIf="checkAuth()">
        <a id="footerLink" routerLink="/emergency-contacts">Emergency Contacts</a>
      </div>
    </div>
    <div class="row"></div>
  </div>
</footer>
```

2.6.2 footer.component.css

```
.footer {
  background-color: #333;
  padding: 2rem 0 1rem 0;
  box-shadow: 2px -8px 8px -1px grey;
}

.left-col{
  color: white;
}

.dateBox{
  background-color: #12aa95;
  text-align: center;
  border-radius: 2px;
}

.middle-col{
  font-family: 'Julius Sans One', sans-serif;
  text-align: center;
  color: white;
```

```
}

.right-col{
  text-align: right;
  color: #12aa95;
}

#footerLink{
  text-align: right;
  color: #12aa95;
  text-decoration: none;
}

#footerLink:hover{
  color: white;
}
```

2.6.3 footer.component.ts

```
import { Component, OnInit } from '@angular/core';
import * as firebase from 'firebase';
import { AngularFirestore } from '@angular/fire/firestore';
import {AuthService} from '../auth/auth.service';

@Component({
  selector: 'app-footer',
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.css']
})
export class FooterComponent implements OnInit {
  currentDate = Date.now();
  userId: string;

  constructor(private authS: AuthService, private aFirestore: AngularFirestore) { }

  ngOnInit() {
    if (this.authS.isAuth()) {
      this.userId = firebase.auth().currentUser.uid;
    }
  }

  checkAuth() {
    if (this.authS.isAuth()) {
      return true;
    } else {
      return false;
    }
  }

}
```

2.7 Appointments Component Code

2.7.1 appointments.component.html

```
<div class="main-content">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>

  <div class="text-right">
    <button type="button" class="btn add-btn" *ngIf="selected == 'appointmentsHome'" (click)="selected='addApp'">
      Add Appointment/Event</button>
  </div>

  <form [formGroup]="addAppForm" (ngSubmit)="addAppointment()" *ngIf="selected == 'addApp'">
    <p class="text-center">Required fields are indicated with a *</p>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Title*</label>
        <input type="text" class="form-control" placeholder="Enter title" formControlName="appTitle">
      </div>
      <div class="form-group col-md-3 col-sm-6">
        <label>Date*</label>
        <input type="date" class="form-control" placeholder="" formControlName="date">
      </div>
      <div class="form-group col-md-1 col-sm-3">
        <label>Time (24hr)</label>
        <input type="time" class="form-control" placeholder="" formControlName="time">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Description</label>
        <input type="text" class="form-control" placeholder="Enter description" formControlName="description">
      </div>
      <div class="form-group col-md-4 col-sm-10">
        <label>Location</label>
        <input type="text" class="form-control" placeholder="Enter location" formControlName="location">
      </div>
    </div>
    <div class="text-center">
      <button type="submit" class="btn submit-btn">Submit</button>
    </div>
  </form>
```

```
<form [formGroup]="updateAppForm" (ngSubmit)="updateAppointment()" *ngIf="selected == 'update'">
  <p class="text-center">Only enter the fields you wish to change</p>
  <div class="form-row justify-content-center">
    <div class="form-group col-md-4 col-sm-10">
      <label>Title</label>
      <input type="text" class="form-control" formControlName="uAppTitle">
    </div>
    <div class="form-group col-md-3 col-sm-6">
      <label>Date</label>
      <input type="date" class="form-control" formControlName="uDate">
    </div>
    <div class="form-group col-md-1 col-sm-3">
      <label>Time (24hr)</label>
      <input type="time" class="form-control" formControlName="uTime">
    </div>
  </div>
  <div class="form-row justify-content-center">
    <div class="form-group col-md-4 col-sm-10">
      <label>Description</label>
      <input type="text" class="form-control" formControlName="uDescription">
    </div>
    <div class="form-group col-md-4 col-sm-10">
      <label>Location</label>
      <input type="text" class="form-control" formControlName="uLocation">
    </div>
  </div>
  <div class="text-center">
    <button type="submit" class="btn submit-btn">Submit</button>
  </div>
</form>

<div class="text-center" *ngIf="selected == 'appointmentsHome'">
  <div class="btn-group flex-wrap" role="group" aria-label="Months">
    <button type="button" class="btn btn-month" (click)="changeMonth(1)">Jan</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(2)">Feb</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(3)">Mar</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(4)">Apr</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(5)">May</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(6)">Jun</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(7)">Jul</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(8)">Aug</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(9)">Sep</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(10)">Oct</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(11)">Nov</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(12)">Dec</button>
    <button type="button" class="btn btn-month" (click)="changeMonth(13)">All</button>
  </div>
  <h3>{{monthTitle}}</h3>
</div>
```

```
 <div class="appointment-wrap" *ngIf="selected == 'appointmentsHome'">
  <div class="appointment" *ngFor="let a of monthSelection">
   <div class ="appointmentHead">
    {{a.dateOnly | date:'fullDate'}}
   </div>
   <table class="table appointmentBody">
    <tr>
     <td class="title" colspan="2">{{a.title}}</td>
    </tr>
    <tr>
     <td>Time:</td>
     <td>{{a.timeOnly}}</td>
    </tr>
    <tr>
     <td>Desc:</td>
     <td>{{a.description}}</td>
    </tr>
    <tr>
     <td>Location</td>
     <td>{{a.location}}</td>
    </tr>
    <tr class="appointmentFoot">
     <td align="center"><i class="far fa-edit" (click)="showUpdate(a)"></i></td>
     <td align="center"><i class="far fa-times-circle" (click)="deleteAppointment(a)"></i></td>
    </tr>
   </table>
  </div>
 </div>



</div>
```

2.7.2 appointments.component.css

```
.appointment-wrap {
  margin-top: 1rem;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
}

.appointment {
  width: 17rem;
  background-color: white;
  margin: 1rem;
  box-shadow: 2px 2px 8px 1px grey;
}
```

```
.appointment:hover {
  box-shadow: none;
}

.appointmentHead {
  padding: 1rem;
  color: white;
  background-color: #333;
}

.appointmentHead:hover {
  background-color: #5a5a5a;
}

.table {
  background-color: #cbfff0;
  padding: 1rem;
}

.title {
  padding: 1rem;
  color: #09aa95;
  font-size: 1.3rem;
}

@media (max-width: 600px) {
  .main-content {
    width: 96%;
    margin-left: 0;
    margin-right: 0;
  }
  .appointment {
    width: 100%;
    margin-left: 0;
  }
  .appointment-wrap {
    display: flex;
    flex-direction: column;
  }
}

label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}
```

```
.submit-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.add-btn {
  color: #FFFFFF;
  background-color: #12aa95;
  border-color: #FFFFFF;
}

.add-btn:hover {
  color: #FFFFFF;
  background-color: #13b7a0;
  border-color: #FFFFFF;
}

.text-center{
  color: #db5655;
  margin-bottom: 1rem;
}

input[type="date"]::-webkit-clear-button, input[type="time"]::-webkit-clear-button {
  display: none;
}

input[type="date"]::-webkit-inner-spin-button {
  display: none;
}

input[type="date"]::-webkit-calendar-picker-indicator {
  color: #12aa95;
}

input[type="date"]-webkit-datetime-edit-text {
  color: #aaaaaa;
}

input[type=date], input[type=time] {
  color: #aaaaaa;
}

.btn-month {
  color: white;
  background-color: #333;
  border-color: #FFFFFF;
  margin-top: 1.5rem;
}
```

```
.btn-month:hover {
  background-color: #525252;
}

.btn-group {
  margin: 0 auto;
}

h3 {
  margin-top: 2rem;
  color: black;
  font-family: 'Julius Sans One', sans-serif;
  font-size: 1.6rem;
}

.appointmentFoot {
  background-color: white;
  margin: 0;
}

.far {
  cursor: pointer;
}
```

2.7.3 appointments.component.ts

```
import { Component, OnInit } from '@angular/core';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import * as firebase from 'firebase';
import { AngularFirestore } from '@angular/fire/firestore';
import { Appointment } from '../auth/lists.model';
import { firestore } from 'firebase';
import { DatePipe } from '@angular/common';

@Component({
  selector: 'app-appointments',
  templateUrl: './appointments.component.html',
  styleUrls: ['./appointments.component.css'],
  providers: [DatePipe]
})
export class AppointmentsComponent implements OnInit {
  title = 'Appointments & Events';
  monthTitle = 'All Appointments & Events';
  role: string;
  userId: string;
  selected: string;
  addAppForm: FormGroup;
  updateAppForm: FormGroup;
  accountType: string;
  primaryEmailString: string;
```

```
primaryUidString: string;
monthOnly;
uMonthOnly;
dateOnly;
timeOnly;
monthSelection: any[] = [];
allAppointments: any[] = [];
janAppointments: any[] = [];
febAppointments: any[] = [];
marAppointments: any[] = [];
aprAppointments: any[] = [];
mayAppointments: any[] = [];
junAppointments: any[] = [];
julAppointments: any[] = [];
augAppointments: any[] = [];
sepAppointments: any[] = [];
octAppointments: any[] = [];
novAppointments: any[] = [];
decAppointments: any[] = [];
existingTitle;
existingDescription;
existingLocation;
existingTime;
existingDateOnly;
existingMonthOnly;
existingData;

constructor(private aFirestore: AngularFirestore, public fb: FormBuilder, private datePipe: DatePipe) { }

ngOnInit() {
  this.userId = firebase.auth().currentUser.uid;
  this.checkAccountType();
  this.selected = 'appointmentsHome';
  this.monthSelection = this.allAppointments;
  this.addAppForm = this.fb.group({
    'date': ['', Validators.required],
    'time': [''],
    'appTitle': ['', Validators.required],
    'description': [''],
    'location': ['']
  });
  this.updateAppForm = this.fb.group({
    'uDate': [''],
    'uTime': [''],
    'uAppTitle': [''],
    'uDescription': [''],
    'uLocation': ['']
  });
}
```

```
get date() { return this.addAppForm.get('date'); }
get time() { return this.addAppForm.get('time'); }
get appTitle() { return this.addAppForm.get('appTitle'); }
get description() { return this.addAppForm.get('description'); }
get location() { return this.addAppForm.get('location'); }

get uDate() { return this.updateAppForm.get('uDate'); }
get uTime() { return this.updateAppForm.get('uTime'); }
get uAppTitle() { return this.updateAppForm.get('uAppTitle'); }
get uDescription() { return this.updateAppForm.get('uDescription'); }
get uLocation() { return this.updateAppForm.get('uLocation'); }

checkAccountType() {
  this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.accountType = user.data().role;
    });
  }).then(event => {
    if (this.accountType === 'Primary') {
      this.primaryUidString = firebase.auth().currentUser.uid;
      this.getAppointments(this.primaryUidString);
    } else {
      this.getPrimaryUid();
    }
  });
}

getPrimaryUid() {
  this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.primaryEmailString = user.data().primaryEmail;
    });
  }).then( event => {
    this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
      .get().then(querySnap => {
      querySnap.forEach( (doc) => {
        this.getAppointments(doc.data().uid);
      });
    });
  });
}

getAppointments(str) {
  this.primaryUidString = str;
  this.getAllApppointments(this.primaryUidString);
  this.getJanApppointments(this.primaryUidString);
  this.getFebApppointments(this.primaryUidString);
  this.getMarApppointments(this.primaryUidString);
```

```
  this.getAprApppointments(this.primaryUidString);
  this.getMayApppointments(this.primaryUidString);
  this.getJunApppointments(this.primaryUidString);
  this.getJulApppointments(this.primaryUidString);
  this.getAugApppointments(this.primaryUidString);
  this.getSepApppointments(this.primaryUidString);
  this.getOctApppointments(this.primaryUidString);
  this.getNovApppointments(this.primaryUidString);
  this.getDecApppointments(this.primaryUidString);
}

addAppointment() {
  this.monthOnly = new Date(this.date.value).getMonth() + 1;
  const data: Appointment = {
    dateOnly: this.date.value,
    timeOnly: this.time.value,
    monthOnly: this.monthOnly,
    title: this.appTitle.value,
    description: this.description.value,
    location: this.location.value,
  };
  this.aFirestore.collection('users/' + this.primaryUidString + '/appointments').add(data);
  this.selected = 'appointmentsHome';
  this.addAppForm.reset();
  this.pushNewAppointment(data);
}

// for live changes before the next initialisation
pushNewAppointment(data) {
  this.allAppointments.push(data);
  this.sortArray(this.allAppointments);
  switch (data.monthOnly) {
    case 1: {
      this.janAppointments.push(data);
      this.sortArray(this.janAppointments);
      break;
    }
    case 2: {
      this.febAppointments.push(data);
      this.sortArray(this.febAppointments);
      break;
    }
    case 3: {
      this.marAppointments.push(data);
      this.sortArray(this.marAppointments);
      break;
    }
    case 4: {
      this.aprAppointments.push(data);
      this.sortArray(this.aprAppointments);
```

```
      break;
    }
    case 5: {
      this.mayAppointments.push(data);
      this.sortArray(this.mayAppointments);
      break;
    }
    case 6: {
      this.junAppointments.push(data);
      this.sortArray(this.junAppointments);
      break;
    }
    case 7: {
      this.julAppointments.push(data);
      this.sortArray(this.julAppointments);
      break;
    }
    case 8: {
      this.augAppointments.push(data);
      this.sortArray(this.augAppointments);
      break;
    }
    case 9: {
      this.sepAppointments.push(data);
      this.sortArray(this.sepAppointments);
      break;
    }
    case 10: {
      this.octAppointments.push(data);
      this.sortArray(this.octAppointments);
      break;
    }
    case 11: {
      this.novAppointments.push(data);
      this.sortArray(this.novAppointments);
      break;
    }
    case 12: {
      this.decAppointments.push(data);
      this.sortArray(this.decAppointments);
      break;
    }
    default: {
      break;
    }
  }
}

deleteAppointment(data) {
  if (confirm('Are you sure you want to delete this appointment?')) {
```

```
    this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/appointments')
      .where('title', '==', data.title).get()
      .then(querySnap => {querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
          if (doc.data().dateOnly === data.dateOnly) {
            this.aFirestore.firestore.doc('users/' + this.primaryUidString + '/appointments/' + doc.id).delete();
            this.removeAppointment(data);
          }
        }
      });
      });
  }
}

// for live changes before the next initialisation
removeAppointment(data) {
  switch (data.monthOnly) {
    case 1: {
      const janIndex: number = this.janAppointments.indexOf(data);
      this.janAppointments.splice(janIndex, 1);
      break;
    }
    case 2: {
      const febIndex: number = this.febAppointments.indexOf(data);
      this.febAppointments.splice(febIndex, 1);
      break;
    }
    case 3: {
      const marIndex: number = this.marAppointments.indexOf(data);
      this.marAppointments.splice(marIndex, 1);
      break;
    }
    case 4: {
      const aprIndex: number = this.aprAppointments.indexOf(data);
      this.aprAppointments.splice(aprIndex, 1);
      break;
    }
    case 5: {
      const mayIndex: number = this.mayAppointments.indexOf(data);
      this.mayAppointments.splice(mayIndex, 1);
      break;
    }
    case 6: {
      const junIndex: number = this.junAppointments.indexOf(data);
      this.junAppointments.splice(junIndex, 1);
      break;
    }
    case 7: {
      const julIndex: number = this.julAppointments.indexOf(data);
      this.julAppointments.splice(julIndex, 1);
```

```
      break;
    }
    case 8: {
      const augIndex: number = this.augAppointments.indexOf(data);
      this.augAppointments.splice(augIndex, 1);
      break;
    }
    case 9: {
      const sepIndex: number = this.sepAppointments.indexOf(data);
      this.sepAppointments.splice(sepIndex, 1);
      break;
    }
    case 10: {
      const octIndex: number = this.octAppointments.indexOf(data);
      this.octAppointments.splice(octIndex, 1);
      break;
    }
    case 11: {
      const novIndex: number = this.novAppointments.indexOf(data);
      this.novAppointments.splice(novIndex, 1);
      break;
    }
    case 12: {
      const decIndex: number = this.decAppointments.indexOf(data);
      this.decAppointments.splice(decIndex, 1);
      break;
    }
    default: {
      break;
    }
  }
  const index: number = this.allAppointments.indexOf(data);
  this.allAppointments.splice(index, 1);
}

updateAppointment() {
  this.removeAppointment(this.existingData);
  this.uMonthOnly = new Date(this.uDate.value).getMonth() + 1;
  const editData = {
    dateOnly: this.uDate.value,
    timeOnly: this.uTime.value,
    monthOnly: this.uMonthOnly,
    title: this.uAppTitle.value,
    description: this.uDescription.value,
    location: this.uLocation.value,
  };
  this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/appointments').where(
    'title', '==', this.existingTitle).get().then(querySnap => {
      querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
```

```
        if (doc.data().dateOnly === this.existingDateOnly) {
          this.aFirestore.doc('users/' + this.primaryUidString + '/appointments/' + doc.id).set(editData, {merge: true});
          this.pushNewAppointment(editData);
        }
      }
    });
  });
  this.selected = 'appointmentsHome';
}

getAllApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').get().then(query => {
    query.forEach(doc => {
      this.allAppointments.push(doc.data());
      return this.sortArray(this.allAppointments);
    });
  });
}

getJanApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    1).get().then(query => {
    query.forEach(doc => {
      this.janAppointments.push(doc.data());
      return this.sortArray(this.janAppointments);
    });
  });
}

getFebApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    2).get().then(query => {
    query.forEach(doc => {
      this.febAppointments.push(doc.data());
      return this.sortArray(this.febAppointments);
    });
  });
}

getMarApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    3).get().then(query => {
    query.forEach(doc => {
      this.marAppointments.push(doc.data());
      return this.sortArray(this.marAppointments);
    });
  });
}

getAprApppointments(primaryId) {
```

```
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    4).get().then(query => {
    query.forEach(doc => {
      this.aprAppointments.push(doc.data());
      return this.sortArray(this.aprAppointments);
    });
  });
}

getMayApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    5).get().then(query => {
    query.forEach(doc => {
      this.mayAppointments.push(doc.data());
      return this.sortArray(this.mayAppointments);
    });
  });
}

getJunApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    6).get().then(query => {
    query.forEach(doc => {
      this.junAppointments.push(doc.data());
      return this.sortArray(this.junAppointments);
    });
  });
}

getJulApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    7).get().then(query => {
    query.forEach(doc => {
      this.julAppointments.push(doc.data());
      return this.sortArray(this.julAppointments);
    });
  });
}

getAugApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    8).get().then(query => {
    query.forEach(doc => {
      this.augAppointments.push(doc.data());
      return this.sortArray(this.augAppointments);
    });
  });
}

getSepApppointments(primaryId) {
```

```
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    9).get().then(query => {
    query.forEach(doc => {
      this.sepAppointments.push(doc.data());
      return this.sortArray(this.sepAppointments);
    });
  });
}

getOctApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    10).get().then(query => {
    query.forEach(doc => {
      this.octAppointments.push(doc.data());
      return this.sortArray(this.octAppointments);
    });
  });
}

getNovApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    11).get().then(query => {
    query.forEach(doc => {
      this.novAppointments.push(doc.data());
      return this.sortArray(this.novAppointments);
    });
  });
}

getDecApppointments(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('monthOnly', '==',
    12).get().then(query => {
    query.forEach(doc => {
      this.decAppointments.push(doc.data());
      return this.sortArray(this.decAppointments);
    });
  });
}

sortArray(array) {
  array.sort((a: any, b: any) => {
    if (a.dateOnly < b.dateOnly) {
      return -1;
    } else if (a.dateOnly > b.dateOnly) {
      return 1;
    } else if (a.dateOnly === b.dateOnly) {
        if (a.timeOnly < b.timeOnly) {
          return -1;
        } else {
          return 1;
```

```
      }
    } else {
      return 0;
    }
  });
  return array;
}

changeMonth(data) {
  switch (data) {
    case 1: {
      this.monthSelection = this.janAppointments;
      this.monthTitle = 'Appointments & Events in January';
      break;
    }
    case 2: {
      this.monthSelection = this.febAppointments;
      this.monthTitle = 'Appointments & Events in February';
      break;
    }
    case 3: {
      this.monthSelection = this.marAppointments;
      this.monthTitle = 'Appointments & Events in March';
      break;
    }
    case 4: {
      this.monthSelection = this.aprAppointments;
      this.monthTitle = 'Appointments & Events in April';
      break;
    }
    case 5: {
      this.monthSelection = this.mayAppointments;
      this.monthTitle = 'Appointments & Events in May';
      break;
    }
    case 6: {
      this.monthSelection = this.junAppointments;
      this.monthTitle = 'Appointments & Events in June';
      break;
    }
    case 7: {
      this.monthSelection = this.julAppointments;
      this.monthTitle = 'Appointments & Events in July';
      break;
    }
    case 8: {
      this.monthSelection = this.augAppointments;
      this.monthTitle = 'Appointments & Events in August';
      break;
    }
```

```
    case 9: {
      this.monthSelection = this.sepAppointments;
      this.monthTitle = 'Appointments & Events in September';
      break;
    }
    case 10: {
      this.monthSelection = this.octAppointments;
      this.monthTitle = 'Appointments & Events in October';
      break;
    }
    case 11: {
      this.monthSelection = this.novAppointments;
      this.monthTitle = 'Appointments & Events in November';
      break;
    }
    case 12: {
      this.monthSelection = this.decAppointments;
      this.monthTitle = 'Appointments & Events in December' ;
      break;
    }
    case 13: {
      this.monthSelection = this.allAppointments;
      this.monthTitle = 'All Appointments & Events';
      break;
    }
    default: {
      break;
    }
  }
}

showUpdate(data) {
  this.existingData = data;
  this.updateAppForm.get('uAppTitle').setValue(data.title);
  this.updateAppForm.get('uDate').setValue(data.dateOnly);
  this.updateAppForm.get('uTime').setValue(data.timeOnly);
  this.updateAppForm.get('uDescription').setValue(data.description);
  this.updateAppForm.get('uLocation').setValue(data.location);
  this.selected = 'update';
  this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/appointments').where(
    'title', '==', data.title).get().then(querySnap => {
    querySnap.forEach((doc) => {
      if (doc.id !== 'undefined') {
        if (doc.data().dateOnly === data.dateOnly) {
          this.existingTitle = doc.data().title;
          this.existingDescription = doc.data().description;
          this.existingLocation = doc.data().location;
          this.existingTime = doc.data().timeOnly;
          this.existingDateOnly = doc.data().dateOnly;
          this.existingMonthOnly = doc.data().monthOnly;
```

```
        }
      }
    });
  });
}
```

2.8 Dashboard Appointments Component Code

2.8.1 dash-appointments.component.html

```
<div class="dash-appoint-content container">
  <h3 class="headings" id="heading3">{{ title }}</h3>

  <div class="day-wrap">

    <div class="day">
      <div class ="dayHead">
        Yesterday
      </div>
      <div class="appointmentBody" *ngFor="let y of yesterday">
          <h4>{{y.title}}</h4>
        <table class="table table-sm">
          <tr *ngIf="y.timeOnly"><td>Time:</td> <td>{{y.timeOnly}}</td></tr>
          <tr *ngIf="y.description"><td>Desc:</td> <td>{{y.description}}</td></tr>
          <tr *ngIf="y.location"><td>Location</td> <td>{{y.location}}</td></tr>
        </table>
      </div>
    </div>

    <div class="day">
      <div id="today" class ="dayHead">
        Today
      </div>
      <div class="appointmentBody" *ngFor="let t of today">
        <h4>{{t.title}}</h4>
        <table class="table table-sm">
          <tr *ngIf="t.timeOnly"><td>Time:</td> <td>{{t.timeOnly}}</td></tr>
          <tr *ngIf="t.description"><td>Desc:</td> <td>{{t.description}}</td></tr>
          <tr *ngIf="t.location"><td>Location</td> <td>{{t.location}}</td></tr>
        </table>
      </div>
    </div>

    <div class="day">
```

```
    <div class ="dayHead">
      {{plus1 | date:'fullDate'}}
    </div>
    <div class="appointmentBody" *ngFor="let tp1 of todayPlus1">
      <h4>{{tp1.title}}</h4>
      <table class="table table-sm">
        <tr *ngIf="tp1.timeOnly"><td>Time:</td> <td>{{tp1.timeOnly}}</td></tr>
        <tr *ngIf="tp1.description"><td>Desc:</td> <td>{{tp1.description}}</td></tr>
        <tr *ngIf="tp1.location"><td>Location</td> <td>{{tp1.location}}</td></tr>
      </table>
    </div>
  </div>

  <div class="day">
    <div class ="dayHead">
      {{plus2 | date:'fullDate'}}
    </div>
    <div class="appointmentBody" *ngFor="let tp2 of todayPlus2">
      <h4>{{tp2.title}}</h4>
      <table class="table table-sm">
        <tr *ngIf="tp2.timeOnly"><td>Time:</td> <td>{{tp2.timeOnly}}</td></tr>
        <tr *ngIf="tp2.description"><td>Desc:</td> <td>{{tp2.description}}</td></tr>
        <tr *ngIf="tp2.location"><td>Location</td> <td>{{tp2.location}}</td></tr>
      </table>
    </div>
  </div>

  <div class="day">
    <div class ="dayHead">
      {{plus3 | date:'fullDate'}}
    </div>
    <div class="appointmentBody" *ngFor="let tp3 of todayPlus3">
      <h4>{{tp3.title}}</h4>
      <table class="table table-sm">
        <tr *ngIf="tp3.timeOnly"><td>Time:</td> <td>{{tp3.timeOnly}}</td></tr>
        <tr *ngIf="tp3.description"><td>Desc:</td> <td>{{tp3.description}}</td></tr>
        <tr *ngIf="tp3.location"><td>Location</td> <td>{{tp3.location}}</td></tr>
      </table>
    </div>
  </div>

 </div>

 <div class="text-right">
   <button type="button" class="btn add-btn" (click)="goToAppointments()">
     View more</button>
 </div>

</div>
```

2.8.2 dash-appointments.component.css

```
.dash-appoint-content{
  margin: 1rem;
  padding: 1.5rem;
  box-shadow: 2px 2px 8px 1px grey;
  background-color: #edfffc;
}

.day-wrap {
  margin-top: 2rem;
  display: flex;
  flex-direction: row;
}

.day {
  background-color: white;
  margin-left: 1rem;
  margin-bottom: 1rem;
  flex: 1;
  border: 1px solid #9C9C9C;
}

@media (max-width: 600px) {
  .dash-appoint-content {
    width: 100%;
    margin-left: 0;
    margin-right: 0;
  }
  .day-wrap {
    display: flex;
    flex-direction: column;
  }
  .day {
    margin-left: 0;
  }
}

#heading3 {
  color: #12aa95;
  font-size: 1.4rem;
  text-align: center;
}

.dayHead {
  padding: 0.5rem;
  color: white;
  background-color: #333;
  font-family: 'Julius Sans One', sans-serif;
  font-weight: normal;
```

```
  font-size: 1.1rem;
  text-align: center;
  border: 1px solid #9C9C9C;
}

#today {
  background-color: #12aa95;
}

.appointmentBody {
  padding: 1rem;
}

.table {
  border-collapse: collapse;
}

h4 {
  color: #12aa95;
  font-size: 1.3rem;
}

.add-btn{
  color: white;
  background-color: #333;
  border-color: #FFFFFF;
  margin-top: 1.5rem;
}

.add-btn:hover{
  background-color: #606060;
}
```

2.8.3 dash-appointments.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';
import { DatePipe } from '@angular/common';

@Component({
  selector: 'app-dash-appointments',
  templateUrl: './dash-appointments.component.html',
  styleUrls: ['./dash-appointments.component.css'],
  providers: [DatePipe]
})
export class DashAppointmentsComponent implements OnInit {
  title = 'Appointments & Events';
  userId: string;
```

```
currentDate;
plus1;
plus2;
plus3;
yesterday: any[] = [];
today: any[] = [];
todayPlus1: any[] = [];
todayPlus2: any[] = [];
todayPlus3: any[] = [];
yesterdayDate;
todayDate;
accountType: string;
primaryUidString: string;
primaryEmailString: string;

constructor(private router: Router, private aFirestore: AngularFirestore, private datePipe: DatePipe) { }

ngOnInit() {
  this.userId = firebase.auth().currentUser.uid;
  this.currentDate = new Date(Date.now());
  this.todayDate = this.datePipe.transform(this.currentDate, 'yyyy-MM-dd');
  this.checkAccountType();
  this.setTitleDates();
}

checkAccountType() {
  this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.accountType = user.data().role;
    });
  }).then(event => {
    if (this.accountType === 'Primary') {
      this.primaryUidString = firebase.auth().currentUser.uid;
      this.getFiveDay(this.primaryUidString);
    } else {
      this.getPrimaryUid();
    }
  });
}

getPrimaryUid() {
  this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.primaryEmailString = user.data().primaryEmail;
    });
  }).then( event => {
    this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
      .get().then(querySnap => {
```

```
      querySnap.forEach( (doc) => {
        this.getFiveDay(doc.data().uid);
      });
    });
  });
}

getFiveDay(str) {
  this.getYesterday(str);
  this.getToday(str);
  this.getTodayPlus1(str);
  this.getTodayPlus2(str);
  this.getTodayPlus3(str);
}

getYesterday(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('dateOnly', '==',
    this.yesterdayDate).get().then(query => {
     query.forEach(doc => {
       this.yesterday.push(doc.data());
       return this.sortArray(this.yesterday);
     });
  });
}

getToday(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('dateOnly', '==',
    this.todayDate).get().then(query => {
    query.forEach(doc => {
      this.today.push(doc.data());
      return this.sortArray(this.today);
    });
  });
}

getTodayPlus1(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('dateOnly', '==',
    this.plus1).get().then(query => {
    query.forEach(doc => {
      this.todayPlus1.push(doc.data());
      return this.sortArray(this.todayPlus1);
    });
  });
}

getTodayPlus2(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('dateOnly', '==',
    this.plus2).get().then(query => {
    query.forEach(doc => {
      this.todayPlus2.push(doc.data());
```

```
      return this.sortArray(this.todayPlus2);
    });
  });
}

getTodayPlus3(primaryId) {
  this.aFirestore.firestore.collection('users/' + primaryId + '/appointments').where('dateOnly', '==',
    this.plus3).get().then(query => {
    query.forEach(doc => {
      this.todayPlus3.push(doc.data());
      return this.sortArray(this.todayPlus3);
    });
  });
}

setTitleDates() {
  this.yesterdayDate = new Date();
  this.yesterdayDate.setDate( this.yesterdayDate.getDate() - 1 );
  this.yesterdayDate = this.datePipe.transform(this.yesterdayDate, 'yyyy-MM-dd');

  this.plus1 = new Date();
  this.plus1.setDate( this.plus1.getDate() + 1 );
  this.plus1 = this.datePipe.transform(this.plus1, 'yyyy-MM-dd');

  this.plus2 = new Date();
  this.plus2.setDate( this.plus2.getDate() + 2 );
  this.plus2 = this.datePipe.transform(this.plus2, 'yyyy-MM-dd');

  this.plus3 = new Date();
  this.plus3.setDate( this.plus3.getDate() + 3 );
  this.plus3 = this.datePipe.transform(this.plus3, 'yyyy-MM-dd');
}

// this function will sort the above arrays for display purposes
sortArray(array) {
  array.sort((a: any, b: any) => {
    if (a.date < b.date) {
      return -1;
    } else if (a.date > b.date) {
      return 1;
    } else {
      return 0;
    }
  });
  return array;
}

goToAppointments() {
  this.router.navigate(['/appointments']);
}
```

```
}
```

2.9 Dashboard Diary Entry Component Code

2.9.1 dash-diary-entry.component.html

```
<div class="dash-diary container">
  <h3 class="headings" id="heading3">{{ title }}</h3>

  <form [formGroup]="diaryCreateForm" (ngSubmit)="diaryEntry()">

    <label>Diary Entry</label>
    <div class="input-group">
      <textarea class="form-control textArea" aria-label="With textarea" rows="5" formControlName="textArea">

      </textarea>
    </div>
    <div class="text-right">
      <button type="submit" class="btn submit-btn">Add Entry</button>
    </div>

  </form>

</div>
```

2.9.2 dash-diary-entry.component.css

```
.dash-diary {
  width: 37rem;
  margin: 1rem;
  padding: 1.5rem;
  box-shadow: 2px 2px 8px 1px grey;
  background-color: #edfffc;
}

@media (max-width: 600px) {
  .dash-diary {
    width: 100%;
    margin-left: 0;
    margin-right: 0;
  }
}

#heading3 {
  color: #12aa95;
```

```
  font-size: 1.4rem;
  text-align: center;
}

label{
  color: #12aa95;
}

.submit-btn{
  color: white;
  background-color: #333;
  border-color: #FFFFFF;
  margin-bottom: 2rem;
}
```

2.9.3 dash-diary-entry.component.ts

```
import { Component, OnInit } from '@angular/core';
import { DiaryEntries } from '../auth/lists.model';
import * as firebase from 'firebase';
import { AngularFirestore } from '@angular/fire/firestore';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {Router} from '@angular/router';

@Component({
  selector: 'app-dash-diary-entry',
  templateUrl: './dash-diary-entry.component.html',
  styleUrls: ['./dash-diary-entry.component.css']
})
export class DashDiaryEntryComponent implements OnInit {
  title = 'Add diary entry';
  diaryCreateForm: FormGroup;

  constructor(private aFirestore: AngularFirestore, public fb: FormBuilder, private router: Router) { }

  ngOnInit() {
    this.diaryCreateForm = this.fb.group({
      'textArea': ['', Validators.maxLength(150)]
    });
  }

  get addEntryText() { return this.diaryCreateForm.get('textArea'); }

  diaryEntry() {
      const addData: DiaryEntries = {
        startDate: firebase.firestore.Timestamp.now(),
        entry: this.addEntryText.value,
        lastEdit: firebase.firestore.Timestamp.now()
      };
```

```
    this.aFirestore.collection('users/' + firebase.auth().currentUser.uid + '/diary').add(addData);
   return this.router.navigate(['/diary']);
  }

}
```

2.10 Dashboard Group Diary Entry Component Code

2.10.1 dash-group-diary-entry.component.html

```
<div class="dash-diary container">
  <h3 class="headings" id="heading3">{{ title }}</h3>

  <form [formGroup]="diaryGroupCreateForm" (ngSubmit)="diaryGroupEntry()">

    <label>Diary Entry</label>
    <div class="input-group">
      <textarea class="form-control textArea" aria-label="With textarea" rows="5" formControlName="textArea">

      </textarea>
    </div>
    <div class="text-right">
      <button type="submit" class="btn submit-btn">Add Entry</button>
    </div>

  </form>

</div>
```

2.10.2 dash-group-diary-entry.component.css

```
.dash-diary {
  width: 37rem;
  margin: 1rem;
  padding: 1.5rem;
  box-shadow: 2px 2px 8px 1px grey;
  background-color: #edfffc;
}

@media (max-width: 600px) {
  .dash-diary {
    width: 100%;
    margin-left: 0;
    margin-right: 0;
  }
```

```
}

#heading3 {
  color: #12aa95;
  font-size: 1.4rem;
  text-align: center;
}

label{
  color: #12aa95;
}

.submit-btn{
  color: white;
  background-color: #333;
  border-color: #FFFFFF;
  margin-bottom: 2rem;
}
```

2.10.3 dash-group-diary-entry.component.ts

```
import { Component, OnInit } from '@angular/core';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import { AngularFirestore } from '@angular/fire/firestore';
import {Router} from '@angular/router';
import {DiaryEntries, GroupDiaryEntries} from '../auth/lists.model';
import * as firebase from 'firebase';

@Component({
  selector: 'app-dash-group-diary-entry',
  templateUrl: './dash-group-diary-entry.component.html',
  styleUrls: ['./dash-group-diary-entry.component.css']
})
export class DashGroupDiaryEntryComponent implements OnInit {
  title = 'Add Group Diary Entry';
  diaryGroupCreateForm: FormGroup;
  displayName: string;
  fName: string;
  lName: string;
  primaryEmailString: string;

  constructor(private aFirestore: AngularFirestore, public fb: FormBuilder, private router: Router) { }

  ngOnInit() {
    this.diaryGroupCreateForm = this.fb.group({
      'textArea': ['', Validators.maxLength(150)]
    });
  }

  get addEntryText() { return this.diaryGroupCreateForm.get('textArea'); }
```

```
 diaryGroupEntry() {
   this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
     .get().then(querySnap => {
     querySnap.forEach((user) => {
       this.fName = user.data().firstName;
       this.lName = user.data().lastName;
     });
   }).then(event => {
     this.title = 'Group Diary';
     this.displayName = this.fName + ' ' + this.lName;
     this.getPrimaryUid();
     });
 }

 private getPrimaryUid() {
   // query assistant account to get primary email address, then query primary account using this email address to get
primary uid
   this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
     .get().then(querySnap => {
     querySnap.forEach((user) => {
       this.primaryEmailString = user.data().primaryEmail;
     });
   }).then(event => {
     this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
       .get().then(querySnap => {
       querySnap.forEach( (doc) => {
         const addGroupData: GroupDiaryEntries = {
           startDate: firebase.firestore.Timestamp.now(),
           entry: this.addEntryText.value,
           displayName: this.displayName,
           lastEdit: firebase.firestore.Timestamp.now()
         };
         this.aFirestore.collection('users/' + doc.data().uid + '/groupDiary').add(addGroupData);
         return this.router.navigate(['/diary']);
       });
     });
   });
 }
}
```

2.11 Diary Component Code

2.11.1 diary.component.html

```
<div class="main-content">
```

```
<h2 class="headings" id="heading2">{{ title }}</h2>
<hr>

<div class="text-right">
  <button type="button" class="btn submit-btn" *ngIf="selected == 'diaryHome'" (click)="showCreate()">
    Create Diary Entry</button>
</div>

<div *ngIf="selected == 'diaryHome'">
  <div class="entry"  *ngFor="let d of diary">
    <div class="entryHead">
      {{d.startDate.toDate() | date:"fullDate"}}
    </div>
    <div class="entryBody">
      <p id="entryBody">{{d.entry}}</p>
    </div>
    <div class="entryFoot">
      <div class="d-flex">
        <div class="mr-auto p-2"><p id="entryFoot">Last Edit: <b>{{d.lastEdit.toDate() |
date:"medium"}}</b></p></div>
        <div class="p-2"><i class="far fa-edit" (click)="showEdit(d)"></i></div>
        <div class="p-2"><i class="far fa-trash-alt" (click)="deleteEntry(d)"></i></div>
      </div>
    </div>
  </div>
</div>

<form [formGroup]="diaryCreateForm" (ngSubmit)="diaryEntry()" *ngIf="selected == 'diaryCreate'">

  <label>Diary Entry</label>
  <div class="input-group">
    <textarea class="form-control textArea" aria-label="With textarea" rows="5" formControlName="textArea">

    </textarea>
  </div>
  <div class="text-right">
    <button type="submit" class="btn submit-btn">Add Entry</button>
  </div>

</form>

<form [formGroup]="diaryEditForm" (ngSubmit)="diaryEdit()" *ngIf="selected == 'editEntry'">

  <label>Edit Entry</label>
  <div class="input-group">
    <textarea class="form-control textArea" aria-label="With textarea" rows="5" formControlName="editTextArea">

    </textarea>
  </div>
  <div class="text-right">
```

```
    <button type="submit" class="btn submit-btn">Submit Changes</button>
   </div>

  </form>

  <div class="text-right">
   <button type="button" class="btn submit-btn" *ngIf="selected == 'groupDiaryHome'" (click)="showCreate()">
    Create Diary Entry</button>
  </div>

  <div *ngIf="selected == 'groupDiaryHome'">
   <div class="entry"  *ngFor="let gd of groupDiary">
    <div class="entryHead">
     <div class="d-flex">
      <div class="mr-auto p-2">{{gd.startDate.toDate() | date:"fullDate"}}</div>
      <div id="displayName" class="p-2">{{gd.displayName}}</div>
     </div>

    </div>
    <div class="entryBody">
     <p id="groupEntryBody">{{gd.entry}}</p>
    </div>
    <div class="entryFoot">
     <div class="d-flex">
      <div class="mr-auto p-2"><p id="groupEntryFoot">Last Edit: <b>{{gd.lastEdit.toDate() |
date:"medium"}}</b></p></div>
      <div class="p-2" *ngIf="canEdit(gd.displayName)"><i class="far fa-edit" (click)="showEdit(gd)"></i></div>
      <div class="p-2" *ngIf="canEdit(gd.displayName)"><i class="far fa-trash-alt"
(click)="deleteEntry(gd)"></i></div>
     </div>
    </div>
   </div>
  </div>

</div>
```

2.11.2 diary.component.css

```
.entry {
  border-top-left-radius: 0.5rem;
  border-top-right-radius: 0.5rem;
  box-shadow: 1px 1px 4px 1px grey;
  margin-bottom: 2rem;
}

@media (max-width: 600px) {
  .entry {
    width: 97%;
  }
```

```
}

.entryHead {
  padding-left: 1rem;
  color: white;
  background-color: #12aa95;
  font-family: 'Julius Sans One', sans-serif;
  font-weight: normal;
  letter-spacing: 3.5px;
  font-size: 1.1rem;
  border-top-left-radius: 0.5rem;
  border-top-right-radius: 0.5rem;
}

.entryBody {
  padding: 1rem;
  color: #12aa95;
  background-color: white;
}

#entryBody, #groupEntryBody {
  font-size: 1.2rem;
}

.entryFoot {
  padding-left: 1rem;
  padding-right: 1rem;
  background-color: white;
  border-top: 1px solid #12aa95;
}

#entryFoot, #groupEntryFoot {
  color: #aaaaaa;
}

label{
  color: #12aa95;
}

.submit-btn{
  color: white;
  background-color: #333;
  border-color: #FFFFFF;
  margin-bottom: 2rem;
}

.submit-btn:hover{
  background-color: #606060;
}
```

```
.far {
  cursor: pointer;
  font-size: 1.5rem;
  float: right;
}

.textArea {
  margin-bottom: 1rem;
}

#displayName {
  border-top-left-radius: 0.5rem;
  border-top-right-radius: 0.5rem;
  background-color: #edfffc;
  color: #12aa95;
  font-weight: bold;
  font-size: 1rem;
  letter-spacing: 1px;
  font-style: italic;
}
```

2.11.3 diary.component.ts

```
import { Component, OnInit } from '@angular/core';
import * as firebase from 'firebase';
import { AngularFirestore } from '@angular/fire/firestore';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { DiaryEntries, GroupDiaryEntries } from '../auth/lists.model';

@Component({
  selector: 'app-diary',
  templateUrl: './diary.component.html',
  styleUrls: ['./diary.component.css']
})
export class DiaryComponent implements OnInit {
  title = 'Diary';
  userId: string;
  fName: string;
  lName: string;
  displayName: string;
  accountType: string;
  primaryEmailString: string;
  primaryUidString: string;
  selected: string;
  diaryCreateForm: FormGroup;
  diaryEditForm: FormGroup;
  startDate: Date;
  diary: any[] = [];
  groupDiary: any[] = [];
  existingEntry: string;
```

```
exisitngStartDate: string;

constructor(private aFirestore: AngularFirestore, public fb: FormBuilder) {}

ngOnInit() {
  this.userId = firebase.auth().currentUser.uid;
  this.checkAccountType();
  this.diaryCreateForm = this.fb.group({
    textArea: ['', Validators.maxLength(150)]
  });
  this.diaryEditForm = this.fb.group({
    editTextArea: ['', Validators.maxLength(150)]
  });
}

get addEntryText() { return this.diaryCreateForm.get('textArea'); }
get editTextArea() { return this.diaryEditForm.get('editTextArea'); }

private checkAccountType() {
  this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.accountType = user.data().role;
      this.fName = user.data().firstName;
      this.lName = user.data().lastName;
    });
  }).then(event => {
    if (this.accountType === 'Primary') {
      this.primaryUidString = this.userId;
      this.title = this.fName + "'s Diary";
      this.getDiaryEntries(this.primaryUidString);
      this.selected = 'diaryHome';
    } else {
      this.title = 'Group Diary';
      this.displayName = this.fName + ' ' + this.lName;
      this.getPrimaryUid();
      this.selected = 'groupDiaryHome';
    }
  });
}

getDiaryEntries(primaryId) {
  this.aFirestore.collection('users/' + primaryId + '/diary', ref => ref.orderBy
  ('startDate', 'desc')).valueChanges()
    .subscribe(collection => {
      this.diary = collection;
    });
}

getGroupDiaryEntries(primaryId) {
```

```
    this.aFirestore.collection('users/' + primaryId + '/groupDiary', ref => ref.orderBy
    ('startDate', 'desc')).valueChanges().subscribe(collection => {
      this.groupDiary = collection;
    });
  }

  private getPrimaryUid() {
    // query assistant account to get primary email address, then query primary account using this email address to get
primary uid
    this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
      .get().then(querySnap => {
      querySnap.forEach((user) => {
        this.primaryEmailString = user.data().primaryEmail;
      });
    }).then(event => {
      this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
        .get().then(querySnap => {
        querySnap.forEach( (doc) => {
          this.getGroupDiaryEntries(doc.data().uid);
          this.primaryUidString = doc.data().uid;
        });
      });
    });
  }

  diaryEntry() {
    if (this.accountType === 'Primary') {
      const addData: DiaryEntries = {
        startDate: firebase.firestore.Timestamp.now(),
        entry: this.addEntryText.value,
        lastEdit: firebase.firestore.Timestamp.now()
      };
      this.aFirestore.collection('users/' + firebase.auth().currentUser.uid + '/diary').add(addData);
      this.selected = 'diaryHome';
    } else {
      const addGroupData: GroupDiaryEntries = {
        startDate: firebase.firestore.Timestamp.now(),
        entry: this.addEntryText.value,
        displayName: this.displayName,
        lastEdit: firebase.firestore.Timestamp.now()
      };
      this.aFirestore.collection('users/' + this.primaryUidString + '/groupDiary').add(addGroupData);
      this.selected = 'groupDiaryHome';
    }
  }

  diaryEdit() {
    if (this.accountType === 'Primary') {
      const editData = {
        entry: this.editTextArea.value,
```

```
      lastEdit: firebase.firestore.Timestamp.now()
    };
    this.aFirestore.firestore.collection('users/' + this.userId + '/diary').where(
     'startDate', '==', this.exisitngStartDate).get().then(querySnap => {
     querySnap.forEach((doc) => {
      if (doc.id !== 'undefined') {
       this.aFirestore.doc('users/' + this.userId + '/diary/' + doc.id).set(editData, {merge: true});
      }
     });
    });
    this.selected = 'diaryHome';
   } else {
    const editGroupData = {
     entry: this.editTextArea.value,
     lastEdit: firebase.firestore.Timestamp.now()
    };
    this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/groupDiary').where(
     'startDate', '==', this.exisitngStartDate).get().then(querySnap => {
     querySnap.forEach((doc) => {
      if (doc.id !== 'undefined') {
       this.aFirestore.doc('users/' + this.primaryUidString + '/groupDiary/' + doc.id).set(editGroupData, {merge:
true});
      }
     });
    });
    this.selected = 'groupDiaryHome';
   }
   this.diaryEditForm.reset();
  }

  deleteEntry(data) {
   if (confirm('Are you sure you want to delete this diary entry?')) {
    if (this.accountType === 'Primary') {
     this.aFirestore.firestore.collection('users/' + this.userId + '/diary').where(
      'startDate', '==', data.startDate).get().then(querySnap => {
      querySnap.forEach((doc) => {
       if (doc.id !== 'undefined') {
        this.aFirestore.firestore.doc('users/' + this.userId + '/diary/' + doc.id).delete();
       }
      });
     });
     this.selected = 'diaryHome';
    } else {
     this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/groupDiary').where(
      'startDate', '==', data.startDate).get().then(querySnap => {
      querySnap.forEach((doc) => {
       if (doc.id !== 'undefined') {
        this.aFirestore.firestore.doc('users/' + this.primaryUidString + '/groupDiary/' + doc.id).delete();
       }
      });
```

```
      });
      this.selected = 'groupDiaryHome';
    }
  }
}

showEdit(data) {
  this.diaryEditForm.get('editTextArea').setValue(data.entry); // this line is joyous
  this.selected = 'editEntry';
  if (this.accountType === 'Primary') {
    this.aFirestore.firestore.collection('users/' + this.userId + '/diary').where(
      'startDate', '==', data.startDate).get().then(querySnap => {
      querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
          this.existingEntry = doc.data().entry;
          this.exisitngStartDate = doc.data().startDate;
        }
      });
    });
  } else {
    this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/groupDiary').where(
      'startDate', '==', data.startDate).get().then(querySnap => {
      querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
          this.existingEntry = doc.data().entry;
          this.exisitngStartDate = doc.data().startDate;
        }
      });
    });
  }
}

showCreate() {
  this.selected = 'diaryCreate';
}

canEdit(formDisplayName) {
  if (this.displayName === formDisplayName) {
    return true;
  } else {
    return false;
  }
}

}
```

2.12 Emergency Contacts Component Code

2.12.1 emergency-contacts.component.html

```
<div class="main-content container" id="emergency-main">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>

  <div class="table-responsive table-striped" *ngIf="showContacts">
  <table class="table">
    <thead class="thead customHead">
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Phone Number</th>
    </thead>
    <tr *ngFor="let c of contacts">
      <td>{{c.firstName}}</td>
      <td>{{c.lastName}}</td>
      <td>{{c.phoneNumber}}</td>
    </tr>
  </table>
  </div>

  <form [formGroup]="addContactForm" (ngSubmit)="addToContacts()" *ngIf="!showContacts">
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>First Name</label>
        <input type="text" class="form-control" placeholder="Enter first name" formControlName="firstName">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Last Name</label>
        <input type="text" class="form-control" placeholder="Enter last name" formControlName="lastName">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Phone Number</label>
        <input type="text" class="form-control" placeholder="Enter phone number" formControlName="phoneNumber">
      </div>
    </div>
    <div class="text-center">
      <button type="submit" class="btn submit-btn emergency-btn">Submit</button>
    </div>
    <button  class="btn submit-btn cancel-btn" (click)="cancelButton()">Back to Contacts</button>
  </form>
```

```
  <button type="button" class="btn submit-btn emergency-btn" *ngIf="showContacts"
(click)="addContactButton()">Add Contact</button>
</div>
```

2.12.2 emergency-contacts.component.css

```
label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}

.emergency-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.cancel-btn{
  color: #FFFFFF;
  background-color: #db5655;
  border-color: #FFFFFF;
}

.customHead {
  background-color: #12aa95;
  color: white;
}

#emergency-main {
  margin-bottom: 16rem;
}
```

2.12.3 emergency-contacts.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {AssistantRecord} from '../auth/lists.model';

@Component({
  selector: 'app-emergency-contacts',
  templateUrl: './emergency-contacts.component.html',
  styleUrls: ['./emergency-contacts.component.css']
```

```
})
export class EmergencyContactsComponent implements OnInit {
  title = 'Emergency Contacts';
  userId: string;
  name: string;
  contacts: any[];
  showContacts: boolean;
  addContactForm: FormGroup;
  accountType: string;
  primaryEmailString: string;
  primaryUidString: string;

  constructor(private aFirestore: AngularFirestore, public fb: FormBuilder) { }

  ngOnInit() {
    this.showContacts = true;
    this.checkAccountType();
    this.addContactForm = this.fb.group({
      'firstName': ['', Validators.required],
      'lastName': ['', Validators.required],
      'phoneNumber': ['', Validators.required]
    });
  }

  get firstName() { return this.addContactForm.get('firstName'); }
  get lastName() { return this.addContactForm.get('lastName'); }
  get phoneNumber() { return this.addContactForm.get('phoneNumber'); }

  // this must be done for assistant also
  addToContacts() {
    this.showContacts = false;
    const contactData: AssistantRecord = {
      uid: null,
      firstName: this.firstName.value,
      lastName: this.lastName.value,
      phoneNumber: this.phoneNumber.value,
    };
    // add new contact to contact list
    this.showContacts = true;
    this.addContactForm.reset(); // clears the form fields for if the user wants to add another contact
    if (this.accountType === 'Primary') {
      return this.aFirestore.collection('users/' + firebase.auth().currentUser.uid + '/contactList').add(contactData);
    } else {
      return this.aFirestore.collection('users/' + this.primaryUidString + '/contactList').add(contactData);
    }
  }

  private checkAccountType() {
    this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
      .get().then(querySnap => {
```

```
    querySnap.forEach((user) => {
      this.accountType = user.data().role;
    });
  }).then(event => {
    if (this.accountType === 'Primary') {
      this.contactList(firebase.auth().currentUser.uid);
    } else {
      this.getPrimaryUid();
    }
  });
}

private contactList(primaryId) {
  this.aFirestore.collection('users/' + primaryId + '/contactList').valueChanges()
    .subscribe(contacts => {
      return this.contacts = contacts;
    });
}

private getPrimaryUid() {
  // query assistant account to get primary email address, then query primary account using this email address to get
primary uid
  this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.primaryEmailString = user.data().primaryEmail;
    });
  }).then( event => {
    this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
      .get().then(querySnap => {
      querySnap.forEach( (doc) => {
        this.contactList(doc.data().uid);
        this.primaryUidString = doc.data().uid;
        this.title = doc.data().firstName + "'s" + ' Emergency Contacts';
      });
    });
  });
}

addContactButton() {
  this.showContacts = false;
}

cancelButton() {
  this.showContacts = true;
}
}
```

2.13 Medication Component Code

2.13.1 medication.component.html

```
<div class="main-content container" id="medication-main">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>

  <div class="table-responsive table-striped" *ngIf="selected == 'list'">
    <table class="table">
      <thead class="thead customHead">
      <th scope="col">Medication Name</th>
      <th scope="col">Dosage</th>
      <th scope="col">Frequency (per day)</th>
      <th scope="col">Special Instructions</th>
      <th></th>
      <th></th>
      </thead>
      <tr *ngFor="let m of medication">
        <td>{{m.name}}</td>
        <td>{{m.dosage}}</td>
        <td>{{m.frequency}}</td>
        <td>{{m.specialInstructions}}</td>
        <td><i class="far fa-edit" (click)="showUpdate(m)"></i></td>
        <td><i class="far fa-times-circle" (click)="deleteMedication(m)"></i></td> <!--Must alert here-->
      </tr>
    </table>
  </div>

  <form [formGroup]="medForm" (ngSubmit)="addMedication()" *ngIf="selected == 'add'">
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Medication Name</label>
        <input type="text" class="form-control" placeholder="Enter medication name" formControlName="name">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Dosage</label>
        <input type="text" class="form-control" placeholder="Enter dosage" formControlName="dosage">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Frequency (per day)</label>
        <input type="text" class="form-control" placeholder="Enter frequency" formControlName="frequency">
      </div>
```

```
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Special Instructions</label>
        <input type="text" class="form-control" placeholder="Optional" formControlName="special">
      </div>
    </div>
    <div class="text-center">
      <button type="submit" class="btn submit-btn">Add</button>
    </div>
  </form>

  <form [formGroup]="updateMedForm" (ngSubmit)="updateMedication()" *ngIf="selected == 'update'">
    <p class="text-center">* Only enter the fields you wish to change</p>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Medication Name</label>
        <input type="text" class="form-control" placeholder="{{existingName}}" formControlName="newName">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Dosage</label>
        <input type="text" class="form-control" placeholder="{{existingDosage}}" formControlName="newDosage">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Frequency (per day)</label>
        <input type="text" class="form-control" placeholder="{{exisitngFrequency}}"
formControlName="newFrequency">
      </div>
    </div>
    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Special Instructions</label>
        <input type="text" class="form-control" placeholder="{{existingSpeacialInstructions}}"
formControlName="newSpecial">
      </div>
    </div>
    <div class="text-center">
      <button type="submit" class="btn submit-btn">Update</button>
    </div>
  </form>

  <button type="button" class="btn submit-btn" *ngIf="selected == 'list'" (click)="showAdd()">Add Medication</button>
</div>
```

2.13.2 medication.component.css

```
.button-label{
  color: white;
}

label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}

.submit-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.far {
  cursor: pointer;
}

.customHead {
  background-color: #12aa95;
  color: white;
}

.text-center{
  color: #db5655;;
}

#medication-main {
  margin-bottom: 16rem;
}
```

2.13.3 medication.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import { MedicationList } from '../auth/lists.model';

@Component({
  selector: 'app-medication',
  templateUrl: './medication.component.html',
  styleUrls: ['./medication.component.css']
```

```
})
export class MedicationComponent implements OnInit {
  title = 'Medication';
  userId: string;
  medication: any[] = [];
  medForm: FormGroup;
  updateMedForm: FormGroup;
  selected: string;
  existingName: string;
  existingDosage: string;
  exisitngFrequency: string;
  existingSpeacialInstructions: string;
  updateName: string;
  updateDosage: string;
  updateFrequency: string;
  updateSpecialInstructions: string;
  accountType: string;
  primaryEmailString: string;
  primaryUidString: string;

  constructor(private aFirestore: AngularFirestore, public fb: FormBuilder) {}

  ngOnInit() {
    this.userId = firebase.auth().currentUser.uid;
    this.checkAccountType();
    // this.getMedication();
    this.medForm = this.fb.group({
      'name': ['', Validators.required],
      'dosage': ['', Validators.required],
      'frequency': ['', Validators.required],
      'special': ['']
    });
    this.updateMedForm = this.fb.group({
      'newName': [''],
      'newDosage': [''],
      'newFrequency': [''],
      'newSpecial': ['']
    });
    this.selected = 'list';
  }

  get name() { return this.medForm.get('name'); }
  get dosage() { return this.medForm.get('dosage'); }
  get frequency() { return this.medForm.get('frequency'); }
  get specialInstructions() { return this.medForm.get('special'); }
  get newName() { return this.updateMedForm.get('newName'); }
  get newDosage() { return this.updateMedForm.get('newDosage'); }
  get newFrequency() { return this.updateMedForm.get('newFrequency'); }
  get newSpecialInstructions() { return this.updateMedForm.get('newSpecial'); }
```

```
private checkAccountType() {
  this.aFirestore.firestore.collection('users').where('uid', '==', firebase.auth().currentUser.uid)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.accountType = user.data().role;
    });
  }).then(event => {
    if (this.accountType === 'Primary') {
      this.primaryUidString = this.userId;
      this.getMedication(this.primaryUidString);
    } else {
      this.getPrimaryUid();
    }
  });
}

getMedication(primaryId) {
  this.aFirestore.collection('users/' + primaryId + '/medication').valueChanges()
    .subscribe(collection => {
      return this.medication = collection;
    });
}

private getPrimaryUid() {
  // query assistant account to get primary email address, then query primary account using this email address to get
primary uid
  this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
    .get().then(querySnap => {
    querySnap.forEach((user) => {
      this.primaryEmailString = user.data().primaryEmail;
    });
  }).then( event => {
    this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
      .get().then(querySnap => {
      querySnap.forEach( (doc) => {
        this.getMedication(doc.data().uid);
        this.primaryUidString = doc.data().uid;
        this.title = doc.data().firstName + "'s" + ' Medication';
      });
    });
  });
}

addMedication() {
  const listData: MedicationList = {
    name: this.name.value,
    dosage: this.dosage.value,
    frequency: this.frequency.value,
    specialInstructions: this.specialInstructions.value
  };
```

```
  this.aFirestore.collection('users/' + this.primaryUidString + '/medication').add(listData);
  this.medForm.reset();
  this.showList();
}

deleteMedication(data) {
  if (confirm('Are you sure you want to delete this medication')) {
    this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/medication').where(
      'name', '==', data.name).get().then(querySnap => {
      querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
          this.aFirestore.firestore.doc('users/' + this.primaryUidString + '/medication/' + doc.id).delete();
        }
      });
    });
  }
}

updateMedication() {
  if (this.newName.value === '') {
    this.updateName = this.existingName;
  } else {
    this.updateName = this.newName.value;
  }
  if (this.newDosage.value === '') {
    this.updateDosage = this.existingDosage;
  } else {
    this.updateDosage = this.newDosage.value;
  }
  if (this.newFrequency.value === '') {
    this.updateFrequency = this.exisitngFrequency;
  } else {
    this.updateFrequency = this.newFrequency.value;
  }
  if (this.newSpecialInstructions.value === '') {
    this.updateSpecialInstructions = this.existingSpeacialInstructions;
  } else {
    this.updateSpecialInstructions = this.newSpecialInstructions.value;
  }
  const updateData: MedicationList = {
    name: this.updateName,
    dosage: this.updateDosage,
    frequency: this.updateFrequency,
    specialInstructions: this.updateSpecialInstructions
  };
  this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/medication').where(
    'name', '==', this.existingName).get().then(querySnap => {
    querySnap.forEach((doc) => {
      if (doc.id !== 'undefined') {
        this.aFirestore.doc('users/' + this.primaryUidString + '/medication/' + doc.id).set(updateData, {merge: true});
```

```
      }
    });
  });
  this.updateMedForm.reset();
  this.showList();
}

showAdd() {
  this.selected = 'add';
}

showList() {
  this.selected = 'list';
}

showUpdate(data) {
  this.selected = 'update';
  this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/medication').where(
    'name', '==', data.name).get().then(querySnap => {
    querySnap.forEach((doc) => {
      if (doc.id !== 'undefined') {
        this.existingName = doc.data().name;
        this.existingDosage = doc.data().dosage;
        this.exisitngFrequency = doc.data().frequency;
        this.existingSpeacialInstructions = doc.data().specialInstructions;
      }
    });
  });
}

}
```

2.14 To-Do Component Code

2.14.1 to-do.component.html

```
<div class="todo-content container">
  <h3 class="headings" id="heading3">{{ title }}</h3>

  <form class="input-group" [formGroup]="toDoForm" (ngSubmit)="addToList()">
    <input type="text" class="form-control" placeholder="" formControlName="record">
    <div class="input-group-append" (click)="addToList()">
      <i class="fas fa-plus"></i>
    </div>
  </form>

  <div class="table-responsive">
```

```
    <table class="table">
      <tr  class="toDoRow" valign="middle" *ngFor="let item of toDoList">
        <td *ngIf="!item.checked"><i class="checked far fa-circle" (click)="toggleCheck(item)"></i></td>
        <td *ngIf="item.checked"><i class="checked far fa-check-circle" (click)="toggleCheck(item)"></i></td>
        <td id="record">{{item.record}}</td>
        <td id="delete"><i class="far fa-times-circle" (click)="deleteRecord(item)"></i></td>
      </tr>
    </table>
  </div>

</div>
```

2.14.2 to-do.component.css

```
.todo-content{
  width: 30rem;
  margin: 1rem;
  padding: 1.5rem;
  box-shadow: 2px 2px 8px 1px grey;
  background-color: #edfffc;
}

@media (max-width: 600px) {
  .todo-content {
    width: 100%;
    margin-left: 0;
    margin-right: 0;
  }
}

#heading3 {
  color: #12aa95;
  font-size: 1.4rem;
  text-align: center;
}

.toDoRow:hover {
  background-color: white;
}

.far {
  font-size: 1.5rem;
  cursor: pointer;
}

.fa-plus {
  font-size: 2rem;
  cursor: pointer;
}
```

```
#delete {
   text-align: right;
}

.checked {
  text-align: left;
}

#record {
  color: #12aa95;
  text-align: left;
}
```

2.14.3 to-do.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFirestore } from '@angular/fire/firestore';
import * as firebase from 'firebase';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import {AssistantRecord, ToDoList} from '../auth/lists.model';

@Component({
  selector: 'app-to-do',
  templateUrl: './to-do.component.html',
  styleUrls: ['./to-do.component.css'],
})

export class ToDoComponent implements OnInit {
  title = 'To-Do List';
  userId: string;
  toDoList: any[] = [];
  toDoForm: FormGroup;
  isChecked: boolean;

  constructor(private aFirestore: AngularFirestore, public fb: FormBuilder) { }

  ngOnInit() {
    this.userId = firebase.auth().currentUser.uid;
    this.getList();
    this.toDoForm = this.fb.group({
      'record': ['', Validators.required]
    });
  }

  get record() { return this.toDoForm.get('record'); }

  getList() {
    this.aFirestore.collection('users/' + this.userId + '/toDoList').valueChanges()
      .subscribe(list => {
        return this.toDoList = list;
```

```
    });
  }

  addToList() {
    const listData: ToDoList = {
      record: this.record.value,
      checked: false
    };
    this.aFirestore.collection('users/' + this.userId + '/toDoList').add(listData);
    this.toDoForm.reset();
  }

  deleteRecord(data) {
    this.aFirestore.firestore.collection('users/' + this.userId + '/toDoList').where(
      'record', '==', data.record).get().then(querySnap => {
        querySnap.forEach((doc) => {
          if (doc.id !== 'undefined') {
            this.aFirestore.firestore.doc('users/' + this.userId + '/toDoList/' + doc.id).delete();
          }
        });
      });
  }

  toggleCheck(data) {
    this.isChecked = !this.isChecked;
    const checkData = {
      checked: this.isChecked
    };
    this.aFirestore.firestore.collection('users/' + this.userId + '/toDoList').where(
      'record', '==', data.record).get().then(querySnap => {
      querySnap.forEach((doc) => {
        if (doc.id !== 'undefined') {
          this.aFirestore.firestore.doc('users/' + this.userId + '/toDoList/' + doc.id).update(checkData);
        }
      });
    });
  }
}
```

2.15 Update Details Component Code

2.15.1 update-details.component.html

```
<div class="main-content container" id="detailsScreen">
  <h2 class="headings" id="heading2">{{ title }}</h2>
  <hr>
  <p class="text-center">* Only enter the fields you wish to change</p>
```

```
  <form [formGroup]="updateDetailsForm" (ngSubmit)="updateDetails()">

    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>First Name</label>
        <input type="text" class="form-control" placeholder="{{currentFirstName}}" formControlName="firstName">
      </div>
    </div>

    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Last Name</label>
        <input type="text" class="form-control" placeholder="{{currentLastName}}" formControlName="lastName">
      </div>
    </div>

    <div class="form-row justify-content-center">
      <div class="form-group col-md-4 col-sm-10">
        <label>Phone Number</label>
        <input type="text" class="form-control" placeholder="{{currentPhoneNumber}}"
formControlName="phoneNumber">
      </div>
    </div>

    <div class="text-center">
      <button type="submit" class="btn submit-btn">Submit</button>
    </div>

  </form>
</div>
```

2.15.2 update-details.component.css

```
label{
  color: #12aa95;
}

input::placeholder{
  color: #aaaaaa;
}

.submit-btn{
  color: #FFFFFF;
  background-color: #06BD89;
  border-color: #FFFFFF;
}

.text-center{
  color: #db5655;;
}
```

2.15.3 update-details.component.ts

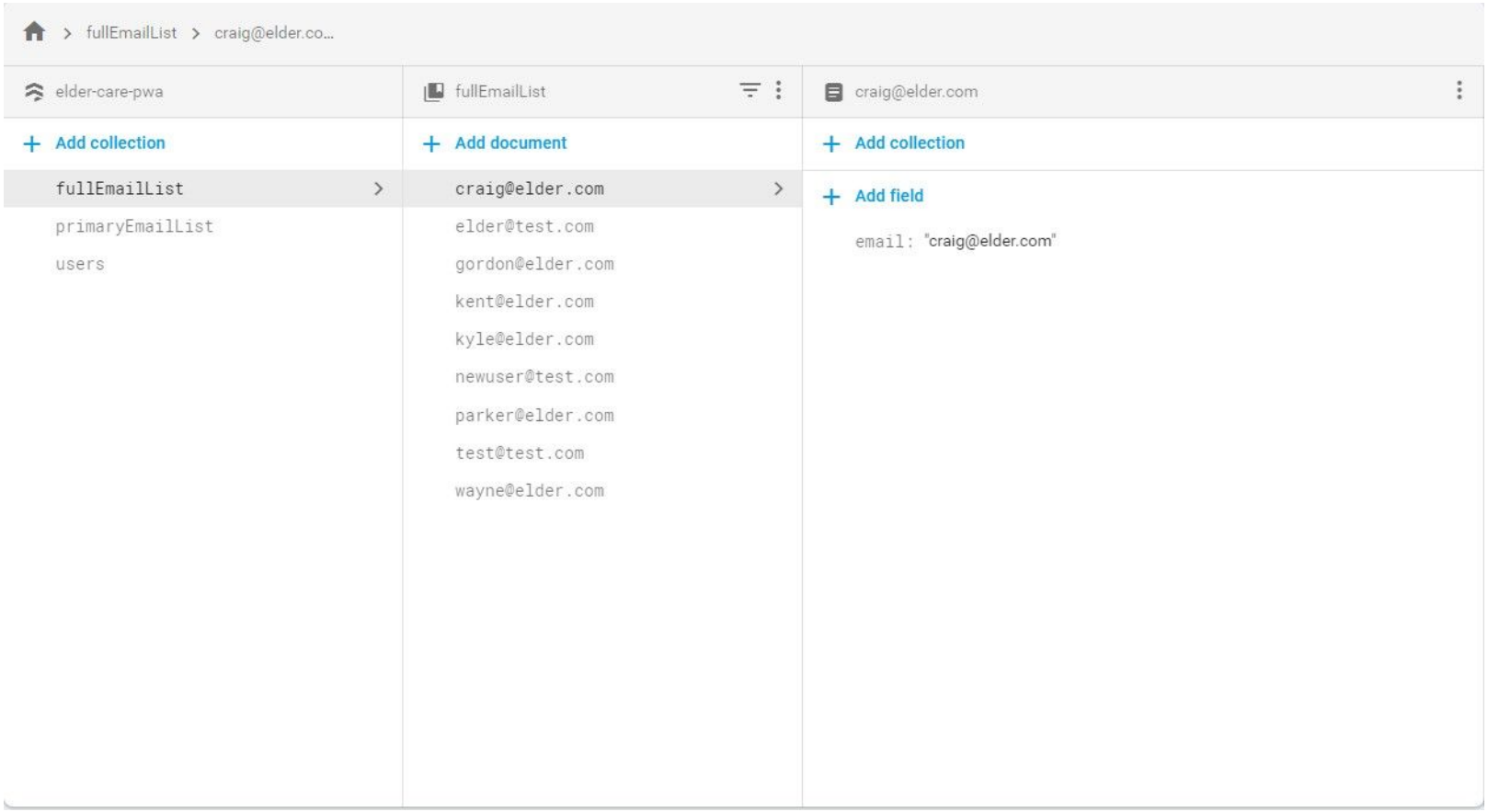
```
import { Component, OnInit} from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { AngularFireAuth } from '@angular/fire/auth';
import * as firebase from 'firebase';
import { AngularFirestore } from '@angular/fire/firestore';
import { Router } from '@angular/router';
import { AssistantRecord } from '../auth/lists.model';
import {AuthService} from '../auth/auth.service';

@Component({
  selector: 'app-update-details',
  templateUrl: './update-details.component.html',
  styleUrls: ['./update-details.component.css']
})
export class UpdateDetailsComponent implements OnInit {
  title = 'Update Details';
  userId: string;
  updateDetailsForm: FormGroup;
  accountType: string;
  currentFirstName: string;
  currentLastName: string;
  currentPhoneNumber: string;
  primaryEmailString: string;
  newFirstName: string;
  newLastName: string;
  newPhoneNumber: string;
  primaryUidString: string;

  constructor( private authS: AuthService, public fb: FormBuilder, private aFireAuth: AngularFireAuth, private
aFirestore: AngularFirestore,
            private router: Router ) { }

  ngOnInit() {
    this.userId = firebase.auth().currentUser.uid;
    this.aFirestore.firestore.collection('users').where('uid', '==', this.userId)
      .get().then(querySnap => {
      querySnap.forEach((user) => {
        // Assign value in detailsComplete in firestore (for the user currently logged) in to myBool
        this.currentFirstName = user.data().firstName;
        this.currentLastName = user.data().lastName;
        this.currentPhoneNumber = user.data().phoneNumber;
        this.accountType = user.data().role;
        if (this.accountType === 'Assistant') {
          this.primaryEmailString = user.data().primaryEmail;
        } else {
          this.primaryEmailString = user.data().email;
        }
```

```
    });
  }).then();
  this.updateDetailsForm = this.fb.group({
    'firstName': [''],
    'lastName': [''],
    'phoneNumber': ['']
  });
}

get formFirstName() { return this.updateDetailsForm.get('firstName'); }
get formLastName() { return this.updateDetailsForm.get('lastName'); }
get formPhoneNumber() { return this.updateDetailsForm.get('phoneNumber'); }

updateDetails() {
  // input is optional, check if the field was entered and assign appropriately
  if (this.formFirstName.value === '') {
    this.newFirstName = this.currentFirstName;
  } else {
    this.newFirstName = this.formFirstName.value;
  }
  if (this.formLastName.value === '') {
    this.newLastName = this.currentLastName;
  } else {
    this.newLastName = this.formLastName.value;
  }
  if (this.formPhoneNumber.value === '') {
    this.newPhoneNumber = this.currentPhoneNumber;
  } else {
    this.newPhoneNumber = this.formPhoneNumber.value;
  }
  const data = {
    firstName: this.newFirstName,
    lastName: this.newLastName,
    phoneNumber: this.newPhoneNumber,
  };
  this.aFirestore.doc('users/' + this.userId).set(data, {merge: true});
  this.updateContactDetails();
  this.updateDetailsForm.reset();
  return this.router.navigate(['/']); // work on alerts ***
}

private updateContactDetails() {
  this.aFirestore.firestore.collection('users').where('email', '==', this.primaryEmailString)
    .get().then(querySnap => {
    querySnap.forEach( (doc) => {
      this.primaryUidString = doc.data().uid;
    });
  }).then( event => {
        const assistantData: AssistantRecord = {
          uid: firebase.auth().currentUser.uid,
```

```
          firstName: this.newFirstName,
          lastName: this.newLastName,
          phoneNumber: this.newPhoneNumber,
        };
        this.getPathAndSetData(assistantData);
      });
  }

  private getPathAndSetData(data) {
    // query to get the right right contact, then get the document name to add to the path to set the contact list
    this.aFirestore.firestore.collection('users/' + this.primaryUidString + '/contactList').
      where('uid', '==', this.userId).get().then(querySnap => {
        querySnap.forEach((doc) => {
          if (doc.id !== 'undefined') {
            return this.aFirestore.doc('users/' + this.primaryUidString + '/contactList/' + doc.id)
              .set(data, {merge: true});
          }
        });
    });
  }
}
```

3. Database Layout - Cloud Firestore

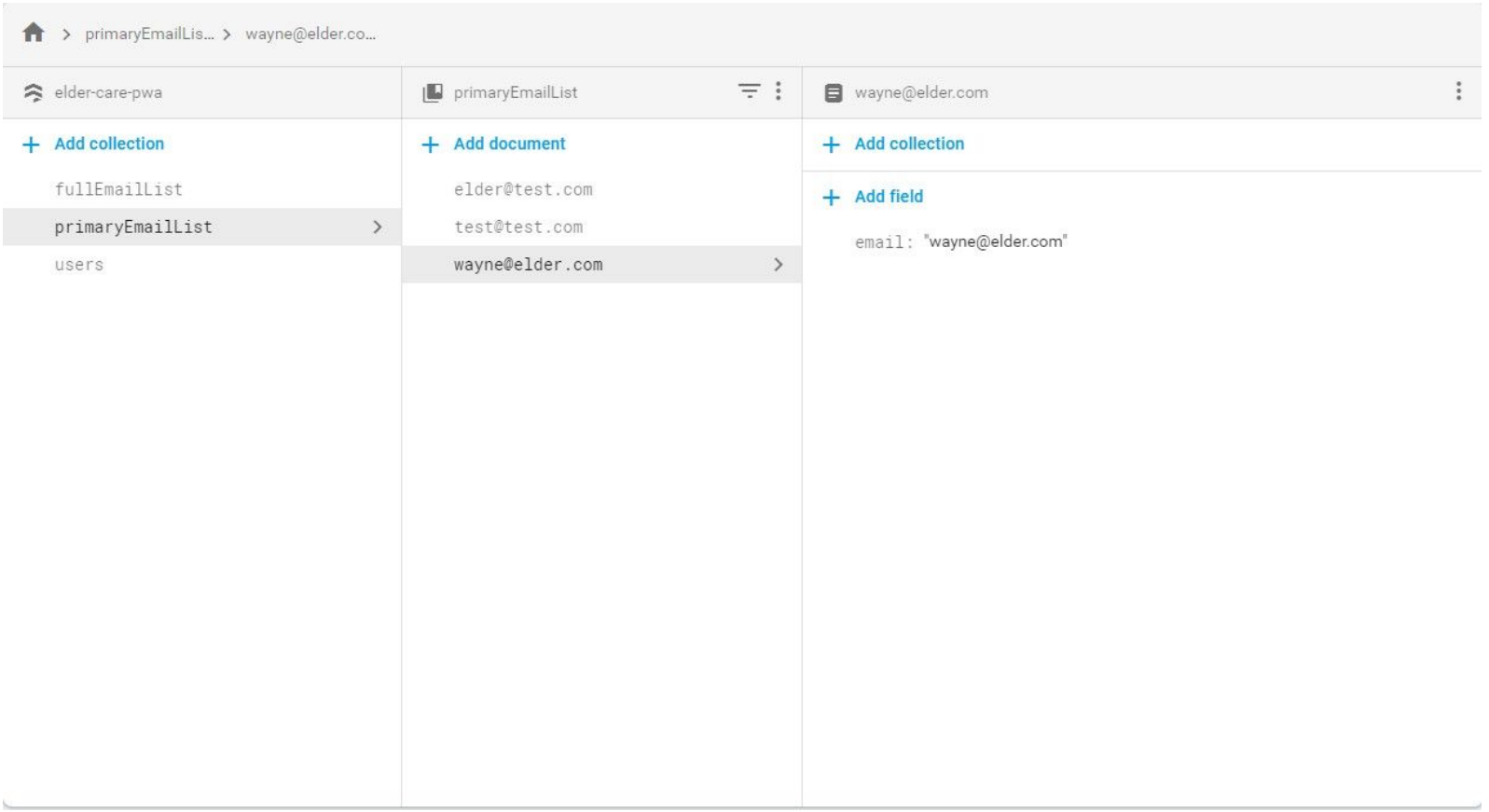
3.1 Full Email List



Code

```
export interface FullEmailList {
  email: string;
}
```

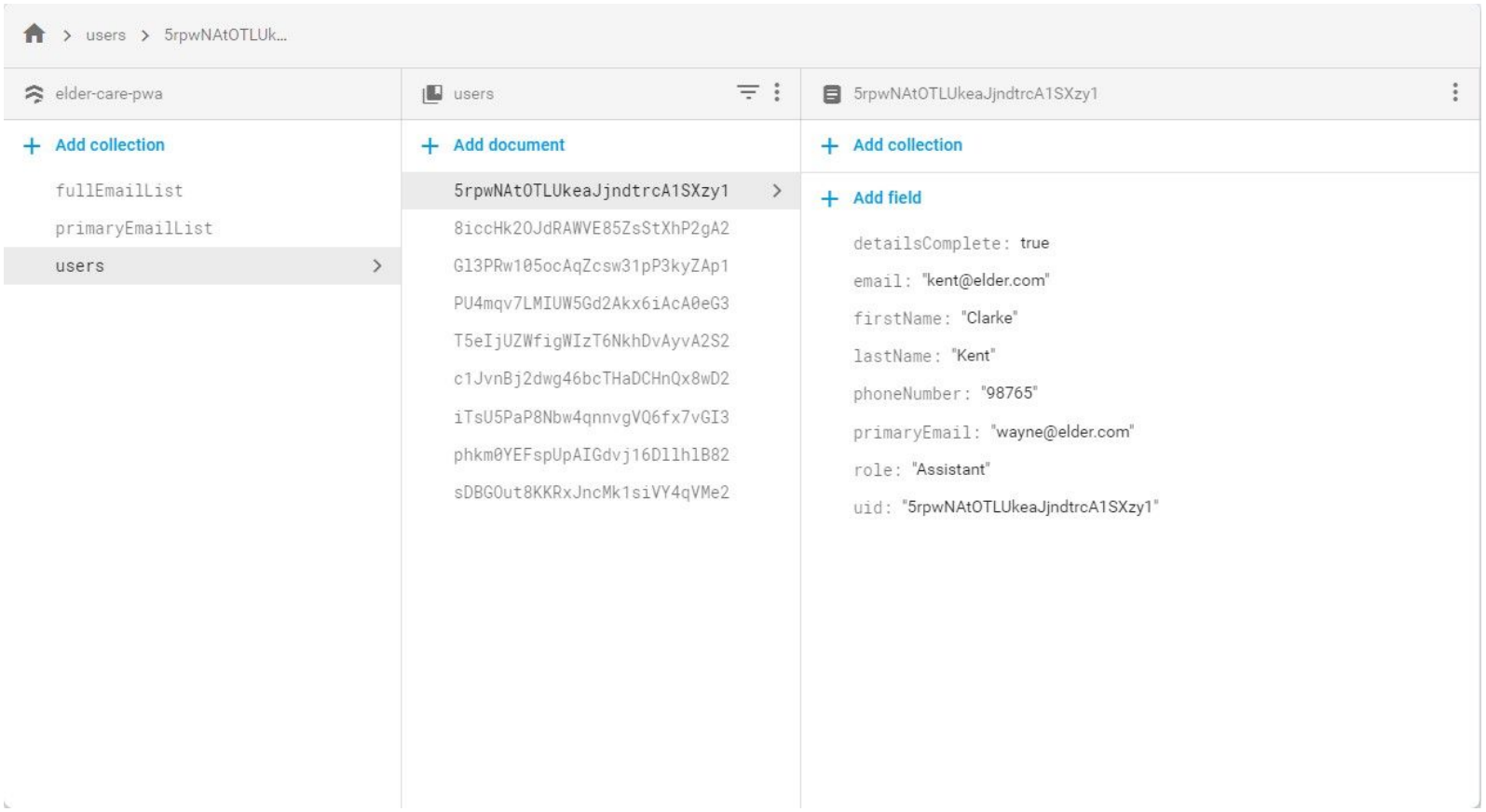
3.2 Primary Account Email List



Code

```
export interface PrimaryEmailList {
  email: string;
}
```

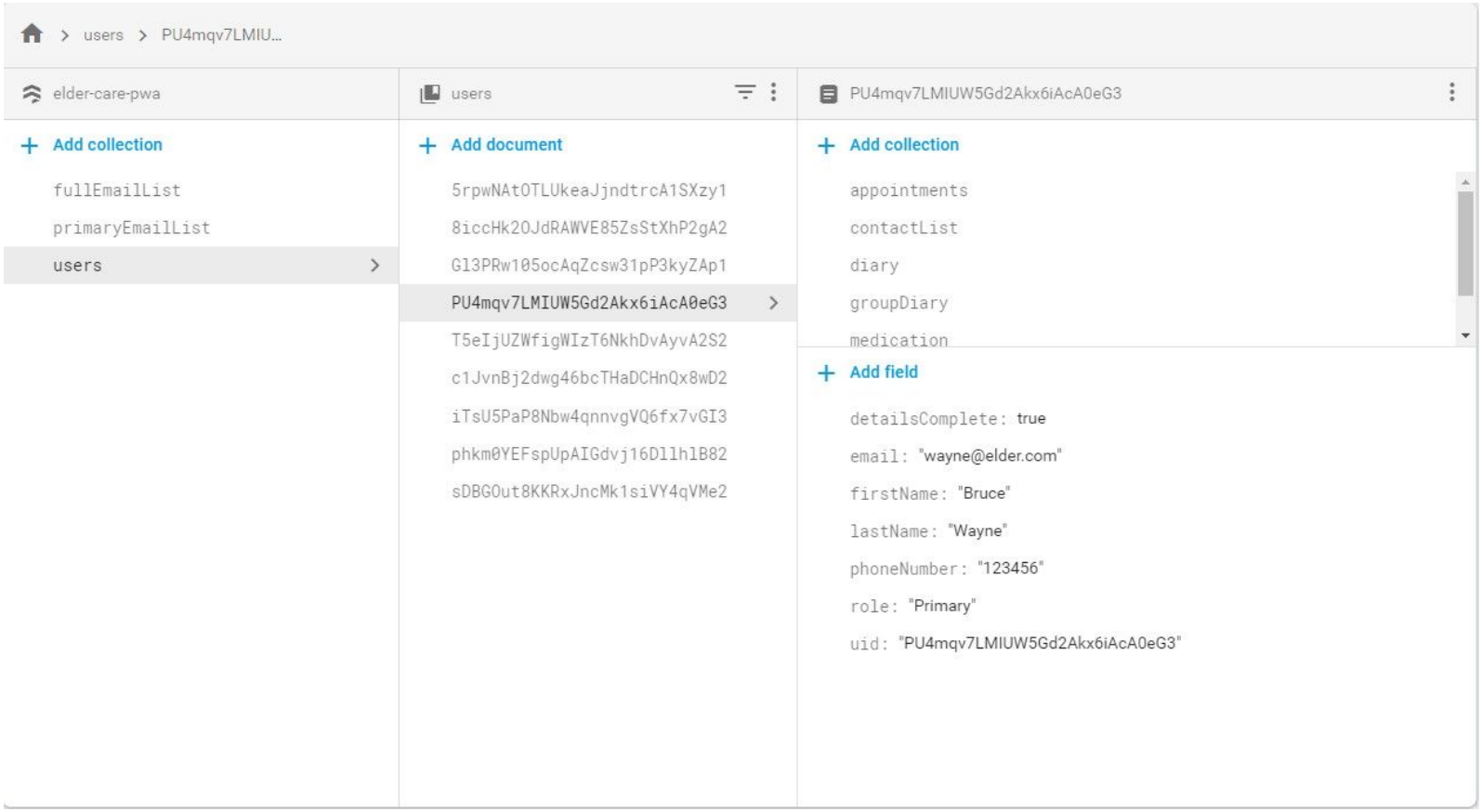
3.3 Assistant User



Code

```
export interface Role {
  type: string;
}

export interface User {
  uid: string;
  email: string;
  primaryEmail?: string;
  username?: string;
  firstName?: string;
  lastName?: string;
  phoneNumber?: string;
  detailsComplete?: boolean;
  roles?: Role;
  assistantRecords?: AssistantRecord[];
}
```

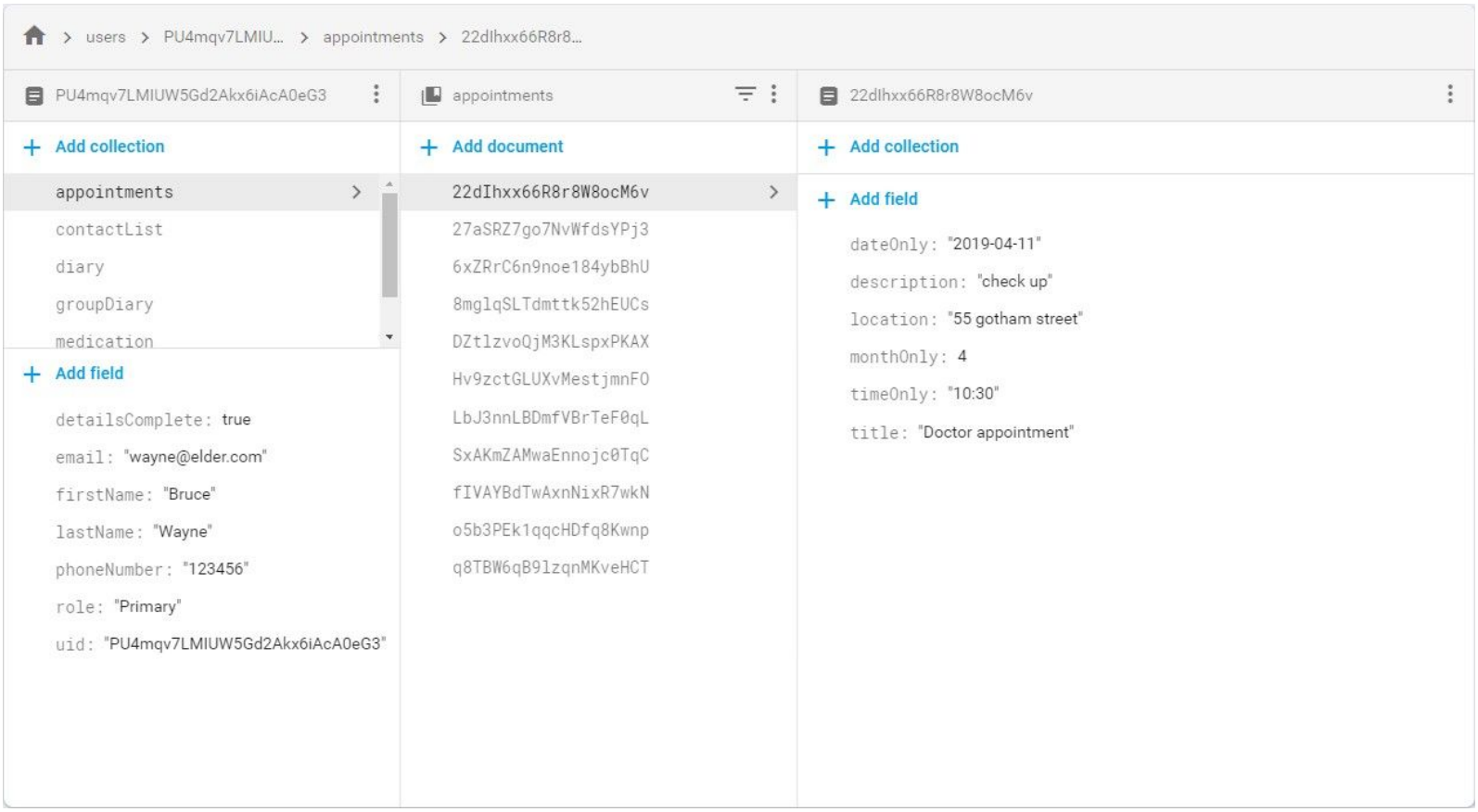
3.4 Elderly User



Code

```
export interface Role {
  type: string;
}

export interface User {
  uid: string;
  email: string;
  primaryEmail?: string;
  username?: string;
  firstName?: string;
  lastName?: string;
  phoneNumber?: string;
  detailsComplete?: boolean;
  roles?: Role;
  assistantRecords?: AssistantRecord[];
}
```

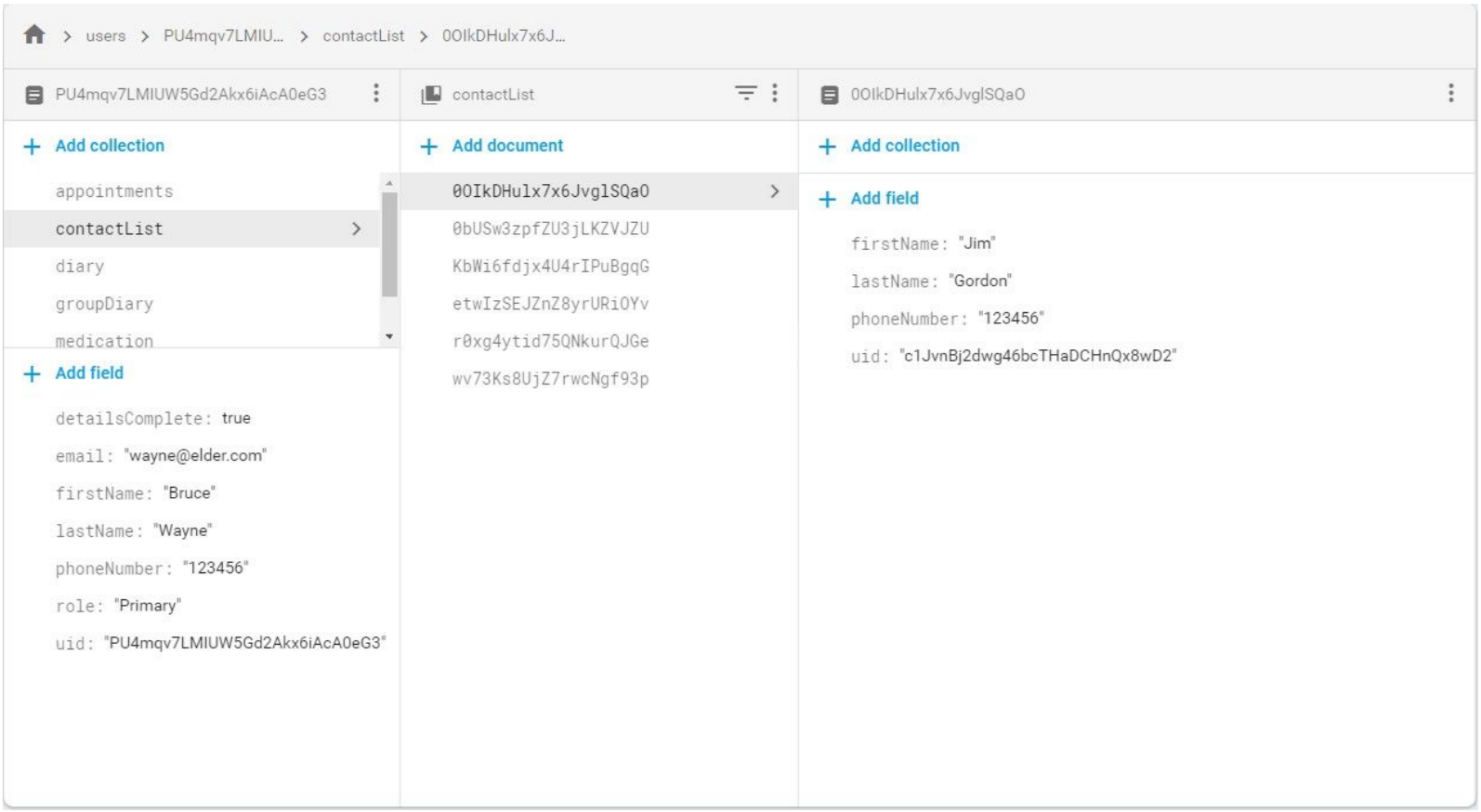
3.5 Appointments



Code

```
export interface Appointment {
  dateOnly: firestore.Timestamp;
  timeOnly: firestore.Timestamp;
  monthOnly: firestore.Timestamp;
  title: string;
  description: string;
  location: string;
}
```

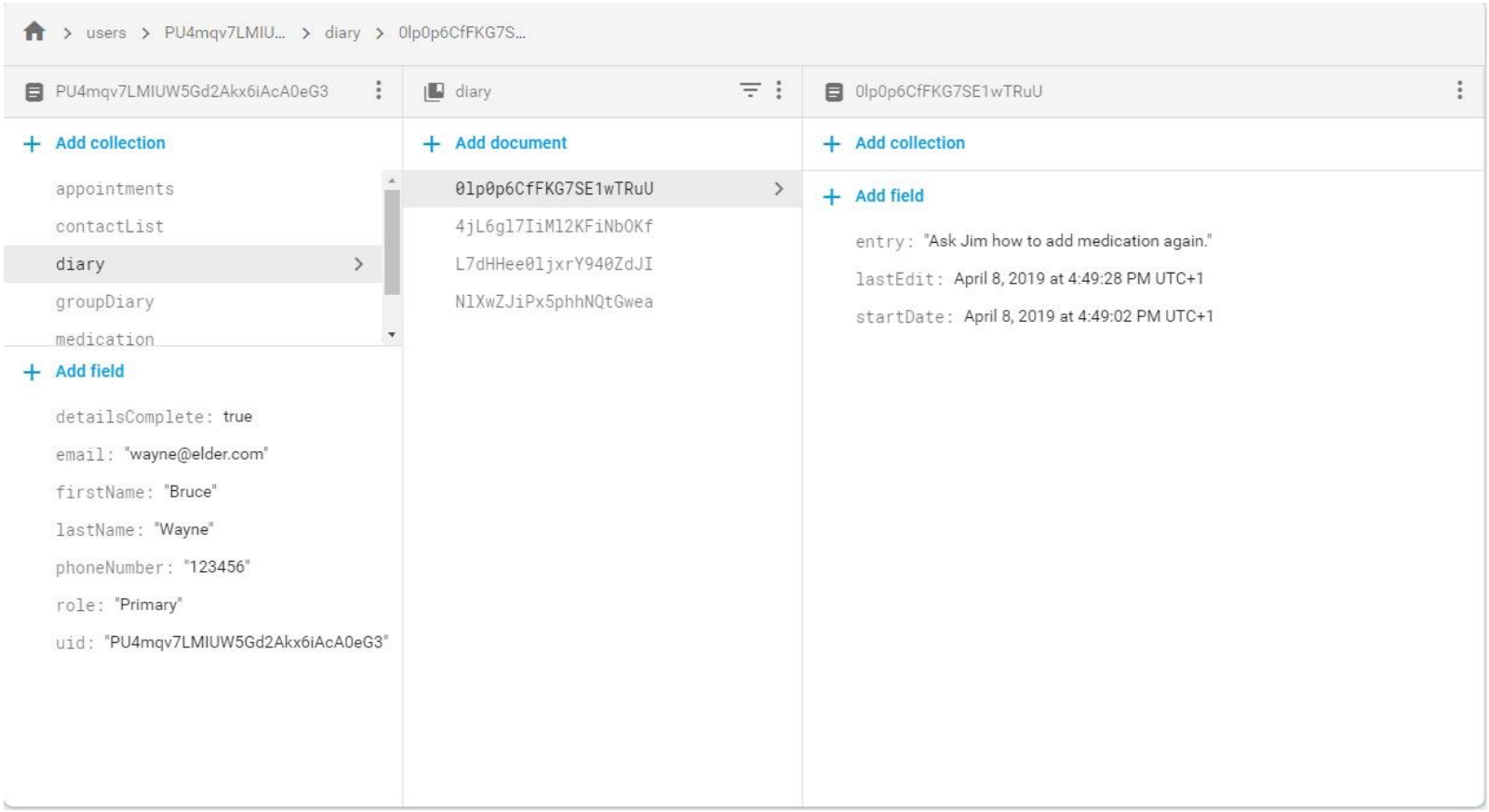
3.6 Contact List



Code

```
export interface AssistantRecord {
  uid: string;
  firstName: string;
  lastName: string;
  phoneNumber: string;
}
```

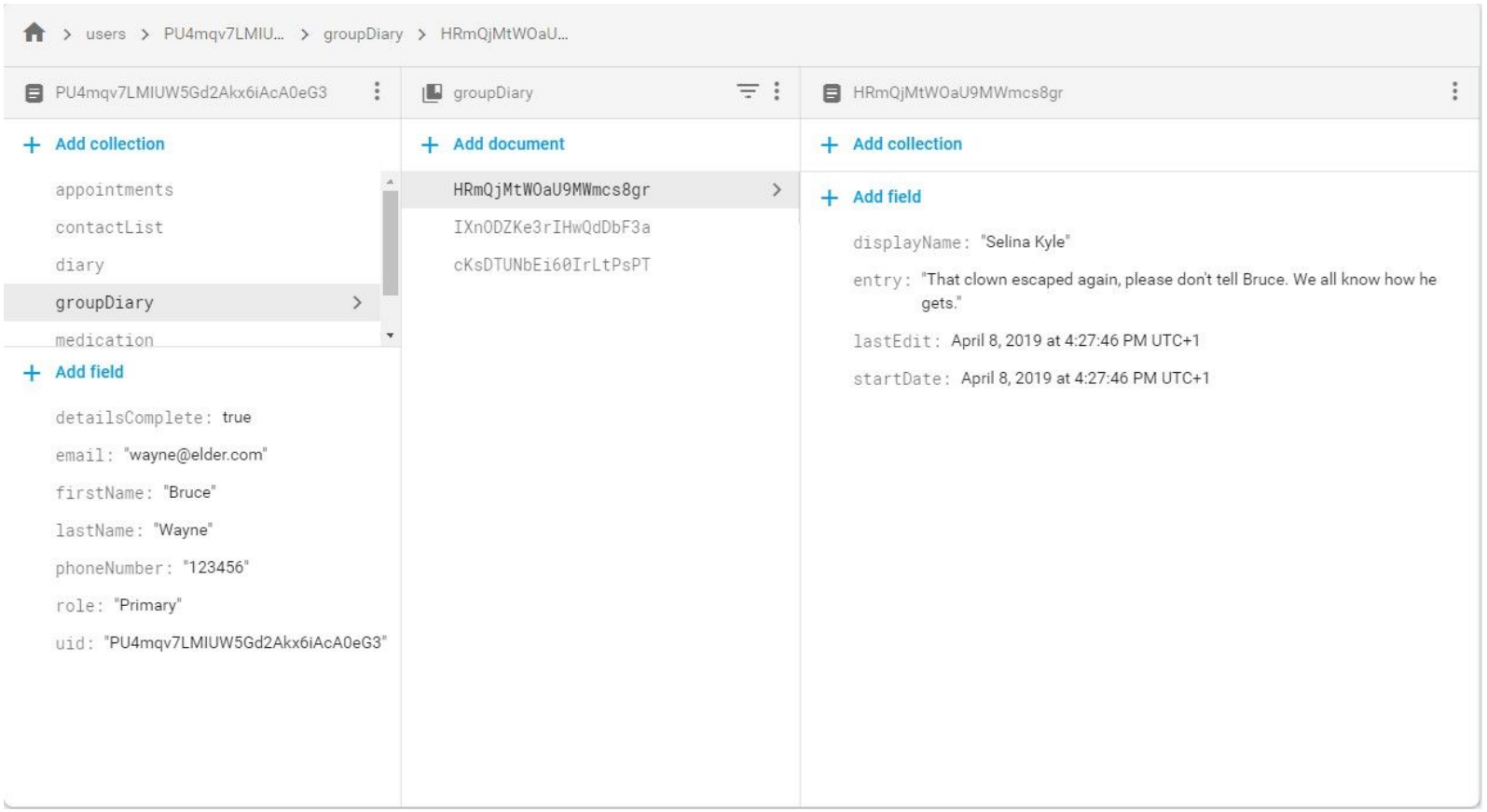
3.7 Diary



Code

```
export interface DiaryEntries {
  startDate: firestore.Timestamp;
  entry: string;
  lastEdit: firestore.Timestamp;
}
```

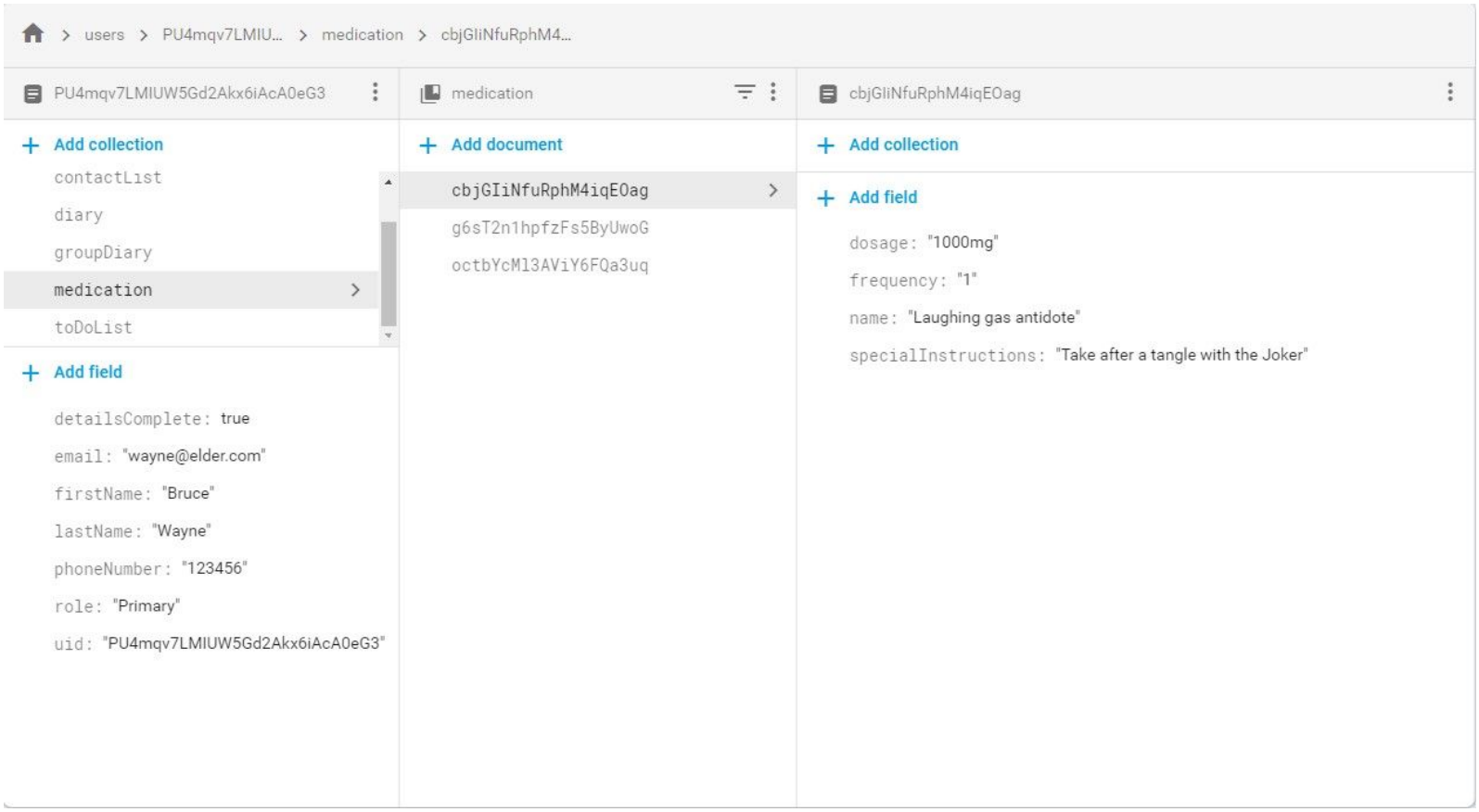
3.8 Group Diary



Code

```
export interface GroupDiaryEntries {
  startDate: firestore.Timestamp;
  entry: string;
  displayName: string;
  lastEdit: firestore.Timestamp;
}
```

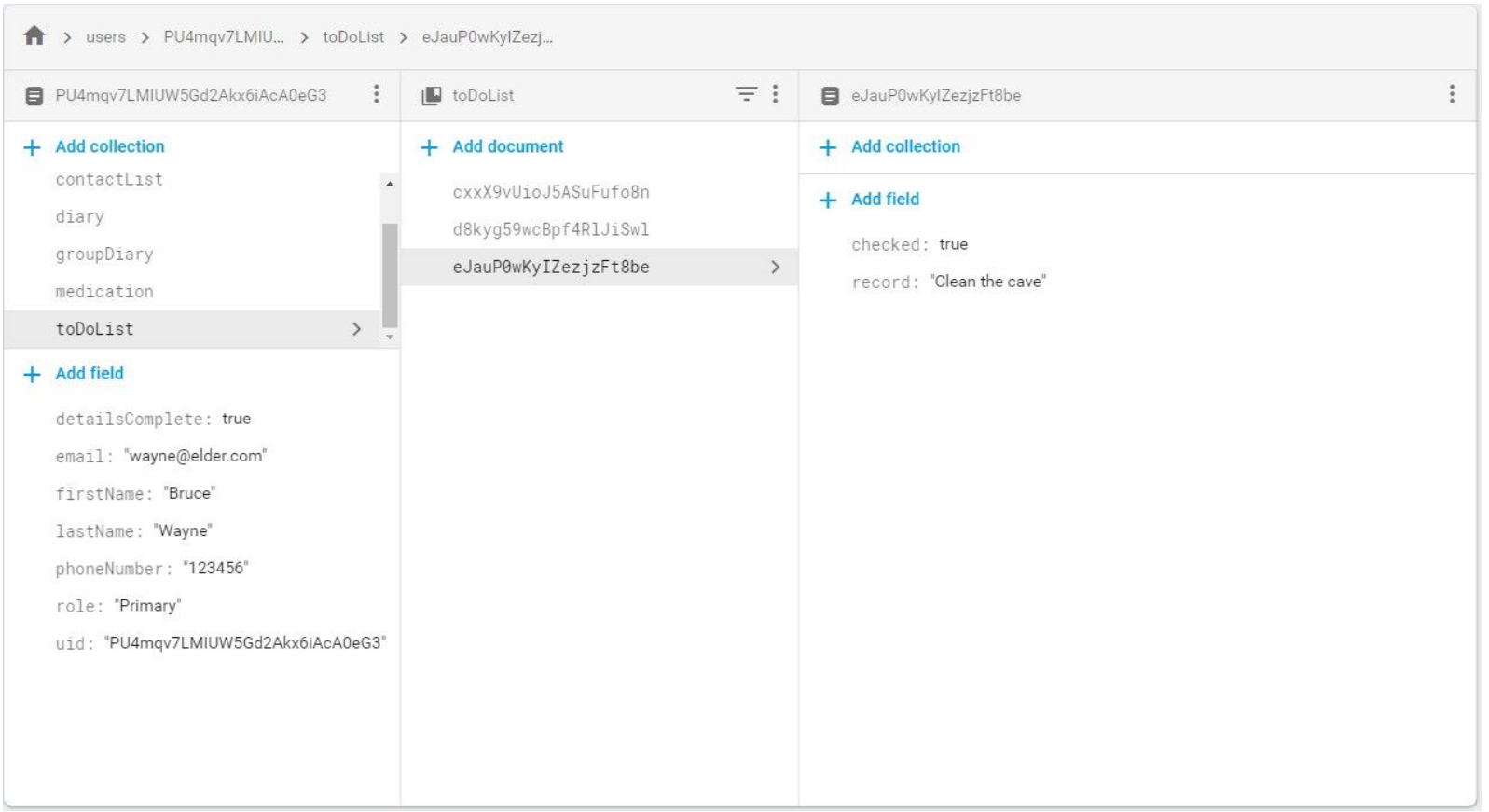
3.9 Medication



Code

```
export interface MedicationList {
  name: string;
  dosage: string;
  frequency: string;
  specialInstructions: string;
}
```

3.10 To-Do List



Code

```
export interface ToDoList {
  record: string;
  checked: boolean;
}
```