INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

Coeliac Disease Symptom Tracker

Design Document

Student: Niamh Coleman (C00205225)

Supervisor: Joseph Kehoe

Date: 14/12/2018

Contents

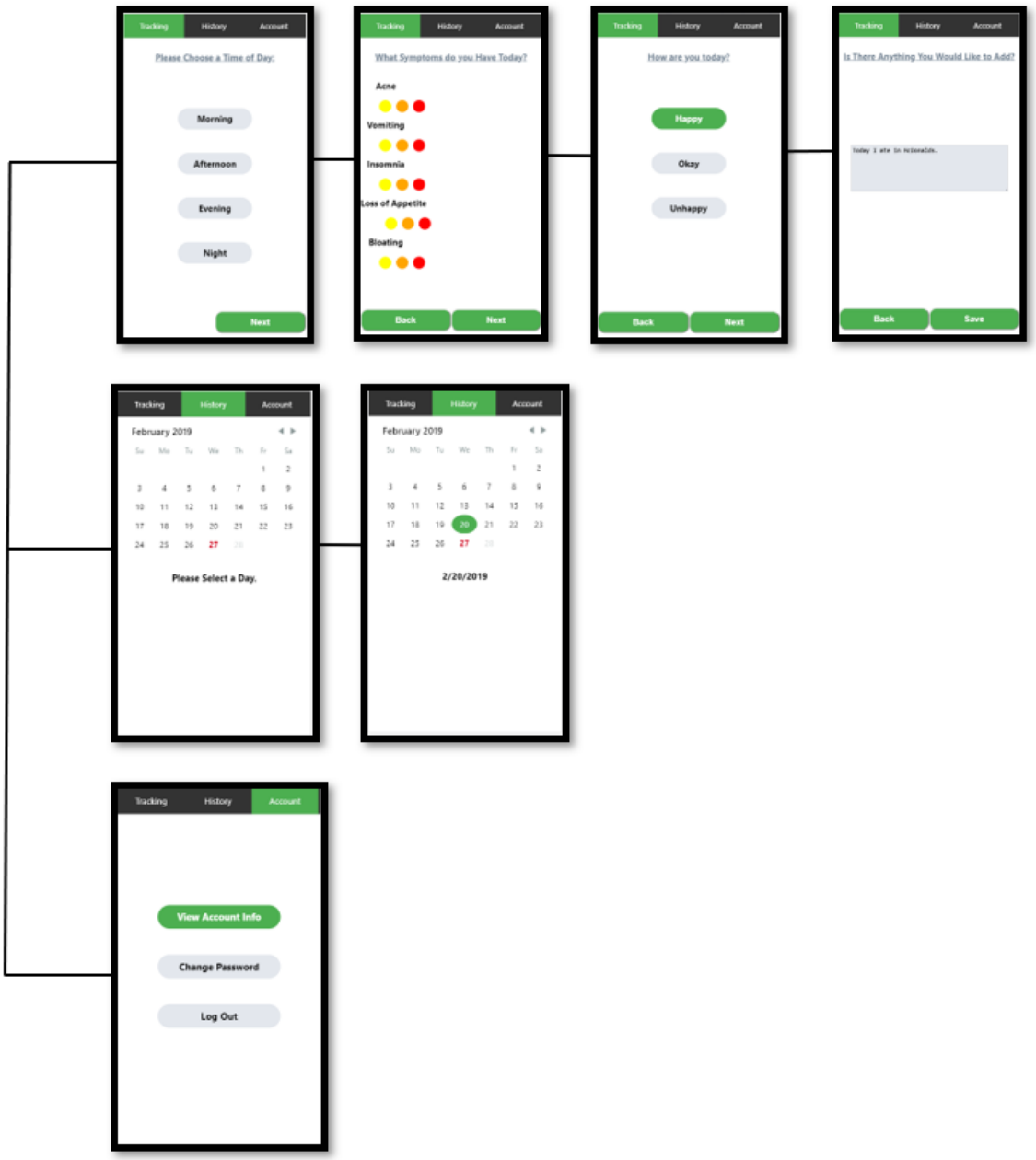
Introduction

After completion of this document, the reader will have a thorough understanding of the work required to create the application in full. This document will equip the reader to develop the application decribed herein, with a thorough understanding of the internal architecture and user interface layout. The learning outcomes of this document will be covered in three sections; front-end, database and server. The front-end section will cover the user interface layout and screen navigation of the application. The database section will cover an overview of the entire databse and the subsection of it that is being used by this project. The server section of the document will contain the code for the functions that the application needs to carry out against the database. This information will enable the reader to develop this application in its entirety. Previous documentation for this project has defined the main functionality and use cases of this application. The application will allow users to track their physical symptoms and emotions relating to coeliac disease. This application will provide information that is crucial to their continued wellbeing by allowing the individual to keep track of the symptoms that they possess and the severity of those symptoms. The application will be cross-platform and will consider ease-of-use to be a top priority.

Front-end

Screen-flow diagram

The application follows a simple navigation system where all pages can be accessed via the tabs at the top of the screen. This navigation style is used due to how easy it is for a user to intuitively use the application. The simple user interface allows the user to carry out functions quickly. Due to the nature of the application, this is extremely important. Being able to quickly navigate and carry out objectives within the application helps encourage the user to use the application consistently and often.



The tab bar will always be present at the top of the screen (figure 1). This menu allows the user to intuitively access all pages without having to navigate a complex screen access layout.

The application utilises a menu bar which will always be present at the bottom of the screen (figure 1). This menu allows the user to easily access all pages without having to navigate a complex screen access layout. The current page open will be indicated by a green background colour. This approach is familiar to all users and requires minimum user training.

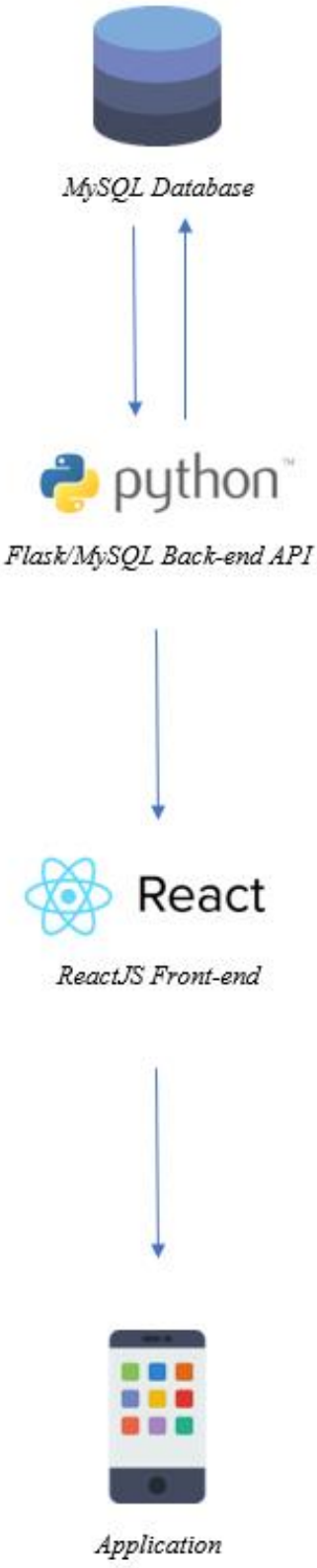
Figure 1.

Database

A database is an organised collection of data with the purpose of storage and retrieval of information. In this application, the database will serve the purpose of storing information regarding users and their symptoms and emotions. This will be done by creating multiple tables within a database and using the combined information stored there to provide the functionality required by the application.

System Architecture

The system is comprised of a MySQL database, a Python MySQL application programming interface and a ReactJS user interface. These technologies were chosen by the leaders of the *Erasmus+* project. The database is common among all students involved with the *Erasmus+* project. The API is written using Python MySQL. An API is a set of operations that can be used by the application to carry out its primary functions. The API contains functions that add, delete, update and read from the database and return that information for use by the API. The user interface uses the API by making calls and requests and receiving responses in JSON format.


MySQL Database
python
Flask/MySQL Back-end API
React
ReactJS Front-end
Application

Database Layout

The database is being shared among all the students involved in the *Erasmus+* project. The following figure (figure 2) is the subset of the database that is used in this project. Figure 3 illustrates the database in its entirety.

Figure 2.

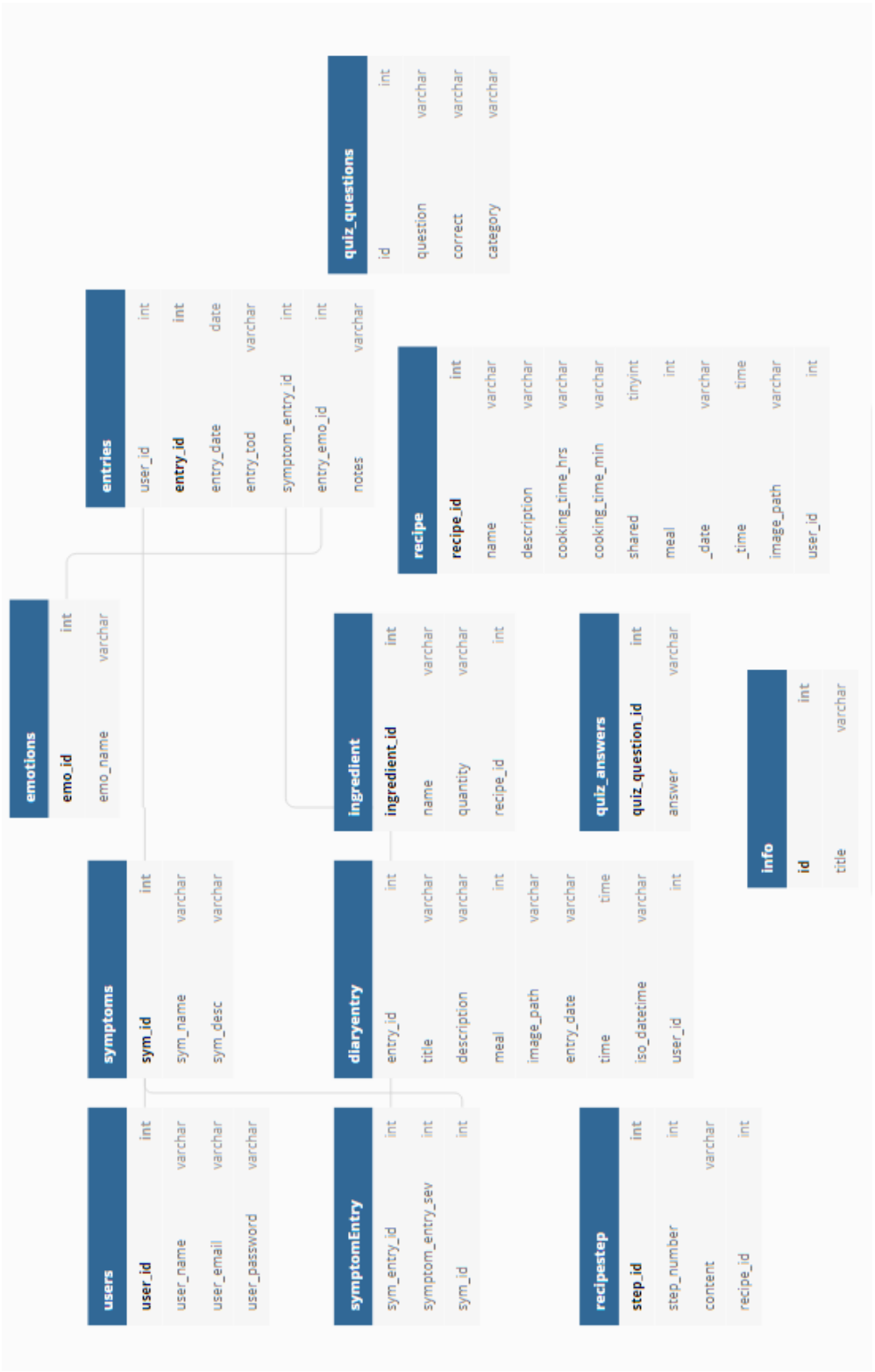

emotions
emo_id int
emo_name varchar
entries
user_id int
entry_id int
entry_date date
entry_tod varchar
symptom_entry_id int
entry_emo_id int
notes varchar
quiz_questions
id int
question varchar
correct varchar
category varchar
recipe
recipe_id int
name varchar
description varchar
cooking_time_hrs varchar
cooking_time_min varchar
shared tinyint
meal int
_date varchar
_time time
image_path varchar
user_id int
symptoms
sym_id int
sym_name varchar
sym_desc varchar
ingredient
ingredient_id int
name varchar
quantity varchar
recipe_id int
quiz_answers
quiz_question_id int
answer varchar
info
id int
title varchar
users
user_id int
user_name varchar
user_email varchar
user_password varchar
diaryentry
entry_id int
title varchar
description varchar
meal int
image_path varchar
entry_date varchar
time time
iso_datetime varchar
user_id int
symptomEntry
sym_entry_id int
symptom_entry_sev int
sym_id int
recipestep
step_id int
step_number int
content varchar
recipe_id int

Figure 3.

Server

Database Tables

Users Table

The users table will contain information on all users registered with the application. Users will be added to the users table upon registering with the application. The table will contain four columns; user ID, user name, email address and password. SQL statements are required for the following actions:

- Initial creation of the users table.
- Adding a new user to the table upon user registration.
- Change user password.
- Return account information about a specific user.

Field Name	Field Type	Field Description
user_id	INT (6)	A **unique** identification number that serves as primary key.
user_name	VARCHAR(10)	This field contains the name of the user entered at registration.
user_email	VARCHAR (50)	This **unique** field contains the email address entered by the user at registration.
user_password	VARCHAR (50)	This field contains the password entered by the user at registration.

Table Creation:

CREATE TABLE users (

user_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,

user_name VARCHAR(10),

user_email VARCHAR(50) UNIQUE,

user_password VARCHAR(50)

);

Adding a User (Registration Use Case):

INSERT INTO users (user_name, user_email, user_password)

VALUES (%name, %email, %password);

Change User Password (Change Password Use Case):

UPDATE users SET user_pass = %newpass WHERE user_id = %user_id;

Return User Information (Account Information Use Case):

SELECT user_name, user_email FROM users WHERE user_id = %user_id;

Symptoms Table

The symptoms table contains information on all symptoms. Users may not add new symptoms to the table. If there is a popular symptom that does not exist in the database, it will be added to the database by the developers. The table will contain three columns; a symptom ID, symptom name and a description of the symptom. SQL statements are required for the following actions:

- Initial creation of the table.
- Adding a symptom to the table.
- Retrieving all symptom names for use by the application.

Field Name	Field Type	Field Description
sym_id	INT (6)	A **unique** identification number that serves as primary key.
sym_name	VARCHAR (20)	This **unique** field contains the name of the symptom.
sym_desc	VARCHAR (50)	This field contains a brief description of the symptom.

Table Creation:

```
CREATE TABLE symptoms (
sym_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
sym_name VARCHAR(20) UNIQUE,
sym_desc VARCHAR(50)
);
```

Adding a New Symptom (Logging Symptoms Use Case):

```
INSERT INTO symptoms (sym_name, sym_desc)
VALUES (%example, %example);
```

Retrieving all Symptom Names (Creating User Interface):

```
SELECT sym_name FROM symptoms;
```

Emotions Table

The emotions table contains information on all emotions in the database. Users may not add new emotions to the table. If there is a popular emotion that does not exist in the database, it will be added to the database by the developers. The table will contain three columns; an ID, emotion name and a description of the emotion. SQL statements are required for the following actions:

- Initial creation of the table.
- Adding an emotion to the table.
- Retrieving all emotion names for use by the application.

Field Name	Field Type	Field Description
emo_id	INT (6)	A **unique** identification number that serves as primary key.
emo _name	VARCHAR (20)	This **unique** field contains the name of the emotion.

Table Creation:

CREATE TABLE emotions (

emo_id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,

emo_name VARCHAR(20) UNIQUE

);

Adding a New Emotion (Creating User Interface):

INSERT INTO emotions (emo _name)

VALUES (%example);

Retrieving all Emotion Names (Creating User Interface):

SELECT emotion _name FROM symptoms;

Entries Table

The entries table contains information on all entries by a user in the database. The table will contain five rows; a user ID, a datetime, a list of the symptoms entered, a list of the emotions entered, and any notes added by the user. SQL statements are required for the following actions:

- Initial creation of the table.
- Logging an entry in the table.
- Retrieving an entry from the table.

Field Name	Field Type	Field Description
user_id	INT (6)	The ID associated with the user making the entry.
entry_id	INT (6)	The **unique** ID associated with the specifics of the entry (auto increment).
entry_date	DATETIME()	The datetime when the entry was submitted by the user.
entry_tod	VARCHAR(10)	Can consist of any one of the following: morning, afternoon. evening, night.
symptom_entry_id	INT(6)	The ID associated with the specifics of the symptom(s) entered by the user.
entry_emo_id	INT(6)	The ID associated with the specifics of the emotion(s) entered by the user.
notes	VARCHAR (25)	Any notes entered by the user associated with the current entry.

Table Creation:

```
CREATE TABLE entries (
user_id INT(6),
entry_id INT(6) AUTO_INCREMENT PRIMARY KEY,
entry_date DATE,
entry_tod VARCHAR(10),
symptom_entry_id INT(6),
entry_emo_id INT(6),
notes VARCHAR (25)
);
```

Adding a New Entry (Logging Symptoms Use Case):

Firstly, add the entry to the entries table:

INSERT into entries (user_id, entry_date, entry_tod, symptom_entry_id, entry_emo_id,notes)

VALUES (%user_id, %entry_date, %entry_tod, %symptom_entry_id, %entry_emo_id, %notes);

Secondly, add each symptom to the symptomEntry table:

For row in symptomsList:

INSERT INTO symtomEntry (sym_entry_id, sym_entry_sev, sym_id)

VALUES (%sym_entry_id, %sym_entry_sev, %sym_id);

Selecting an Entry (View History Use Case):

SELECT * FROM symptoms WHERE userID = $id AND entryDate = $dateChosen AND entryTime = $timeChosen;

symptomEntry Table

The entries table contains information on all symptoms entered by a user in an entry, as well as their corresponding severity.

Field Name	Field Type	Field Description
sym_entry_id	INT (6)	Value corresponds with the symptom_entry_id in the **Entries** table.
symptom_entry_sev	INT (6)	This field contains the severity of the symptom entry i.e. low, moderate, severe.
sym _id	INT (6)	Value corresponds with a symptom ID in the **Symptoms** table.

Table Creation:

CREATE TABLE symptomEntry (

sym_entry_id INT(6),

symptom_entry_sev INT (6),

sym_id INT(6)

);

Adding a New Entry (Logging Symptoms Use Case):

For row in symptomsList:

INSERT INTO symtomEntry (sym_entry_id, sym_entry_sev, sym_id)

VALUES (%sym_entry_id, %sym_entry_sev, %sym_id);