INSTITUTE of TECHNOLOGY CARLOW
Institiúid Teicneolaíochta Cheatharlach

migraineHub

# Design Document

**Student Name:** Michelle Bolger

**Student Number:** C00242743

**Supervisor:** Dr. Chris Staff

**Submission Date:** 25/4/2022

# Abstract

The inspiration behind this project was to discover more information about migraines, their triggers and symptoms and also to discover information about tracking migraines and the benefits of keeping a journal of migraines.

The purpose of the “migraineHub” app is to develop a cross platform mobile application that will allow people who suffer from migraines to record their migraines (the frequency, type of migraine, pain scale, medications, triggers and symptoms).

Users will also be able to view statistic about their past migraines and email this information to themselves for future reference or for review by their doctor/neurologist.

This design document will provide information about the tools and technologies used to create the migraineHub application. Screen designs are also provided to showcase the user interface of the application.

# Table of Contents

# Table of Figures

# 1. Introduction

The purpose of this document is to outline and discuss the design features of the “migraineHub” application.

This document will provide information about the applications architecture and the design pattern implemented.

The User Interface (UI) and User Experience (UX) will also be discussed. System sequence diagrams will be provided to illustrate how certain tasks are carried out between the users and the system. A class diagram will also be provided to show the systems structure in detail.

A layout of the database will also be provided to showcase how the data is planned to be stored.

# 2. System Architecture

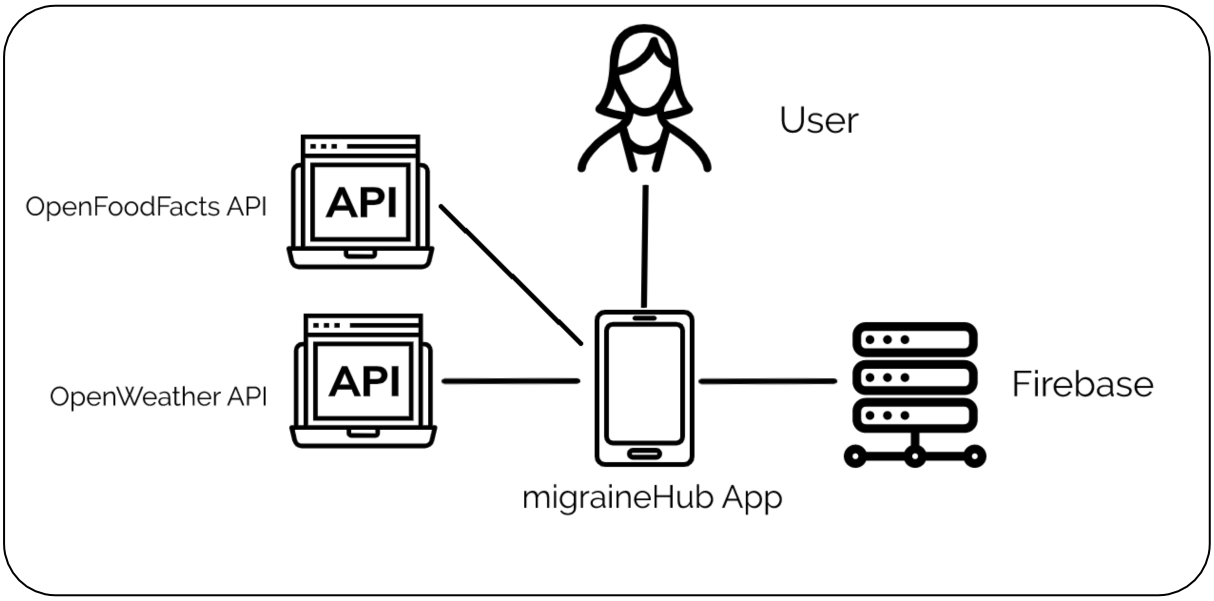


**Figure 1 - System Architecture**

The migraineHub application will be developed using Xamarin.Forms. The front end will be developed using XAML while the backend will be written in C#.

Firebase Realtime Database will be the database for the application. The OpenFoodFacts API and the OpenWeather API will both be used to provide the application with data.

# 3. Tools & Technologies

## 3.1 Xamarin Forms

Xamarin.Forms was chosen as the User Interface (UI) framework to develop this application. This framework was chosen for its capabilities to build application from a shared codebase with the UI code written in (XAML) and the back-end code written in C#.

## 3.2 Firebase

Google's Firebase Realtime Database has been chosen as the database for this project. Firebase is a cloud based non Structured Query Language (NoSQL) database which allows users data to be stored and synced in realtime.

This database was chosen as it is optimised for offline use ensuring that if a user loses connection then their information will be stored locally on their device and when a connection is re-established their data will automatically be synchronised.

Firebase also provides functionalities such as analytics and crash reports allowing developers to focus on their user's needs.

## 3.3 API

The OpenFoodFacts API will be used to provide information about food items that will be scanned by members. OpenFoodFacts is a non-profit project which provides a free database regularly updated by volunteers around the world.

The OpenWeather API will be used to provide up to date weather information about a location a member has entered. This API provides current weather data collected from a number of different sources such as global and local weather stations.

The Xamarin.Essentials API will be used to allow members to send e-mails with their migraine information. This API opens a default email application and will include a specified subject, body and recipients.

# 4. Design Pattern

## 4.1 Model-View-ViewModel (MVVM)

The Model-View-ViewModel (MVVM) pattern consists of separate layers.

- **Model**: This layer will consist of non-visual classes that contain data. The “Model” layer is independent of the “ViewModel” layer. This layer will contain the logic for the application.
- **ViewModel**: This layer binds data between “View” and “Model”. This layer acts like a bridge to execute commands. “ViewModel” knows about the “Model” layer but not the “View” later.
- **View**: This layer will contain the User Interface (UI) elements. This will be the layer that the user/member will interact with. The “View” layer “contains behaviours, events, and data-bindings that ultimately require knowledge of the underlying “Model” and “ViewModel”” [1].

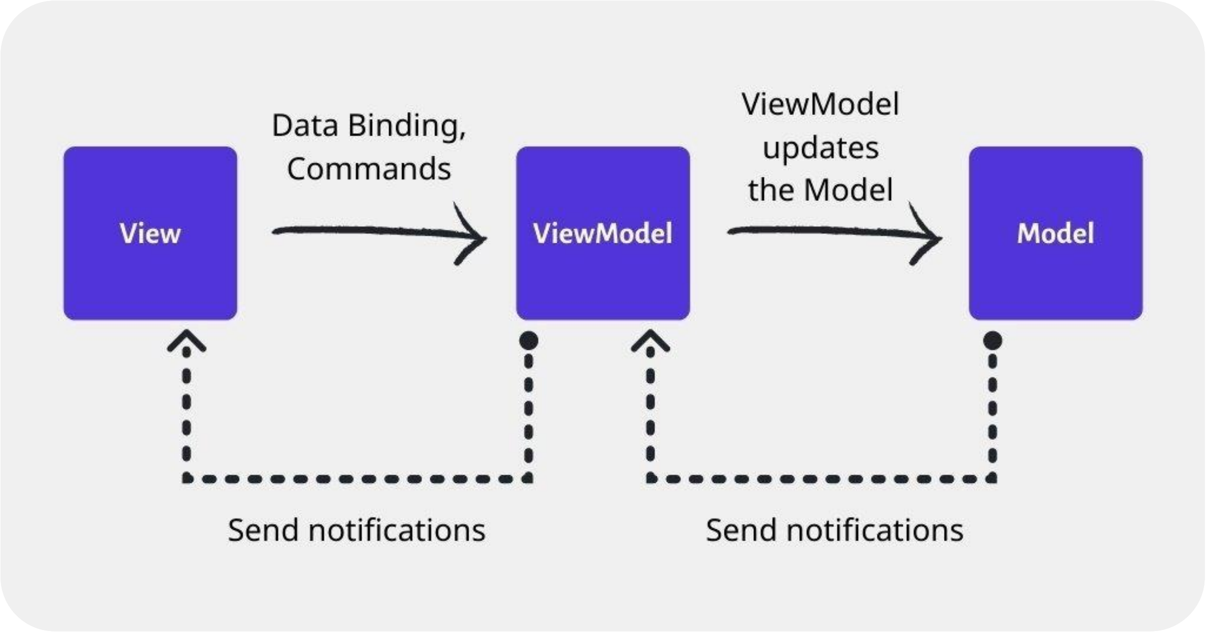


**Figure 2 - MVVM Design Pattern. Source: https://www.serkanseker.com/xamarin-forms-mvvm-pattern/**

The advantages of utilising the MVVM pattern is the testability is increased, because the UI and the C# code are separate from each other the testing will be easier to carry out. Code written using the MVVM pattern is also easier to maintain, an example of this would be if changes are made within the “Model” layer these do not affect the “View” layer.

A disadvantage of the MVVM pattern would be that the amount of code will increase because of all the connecting layers although as previously stated it will be easier to maintain.

# 5. Sequence Diagrams

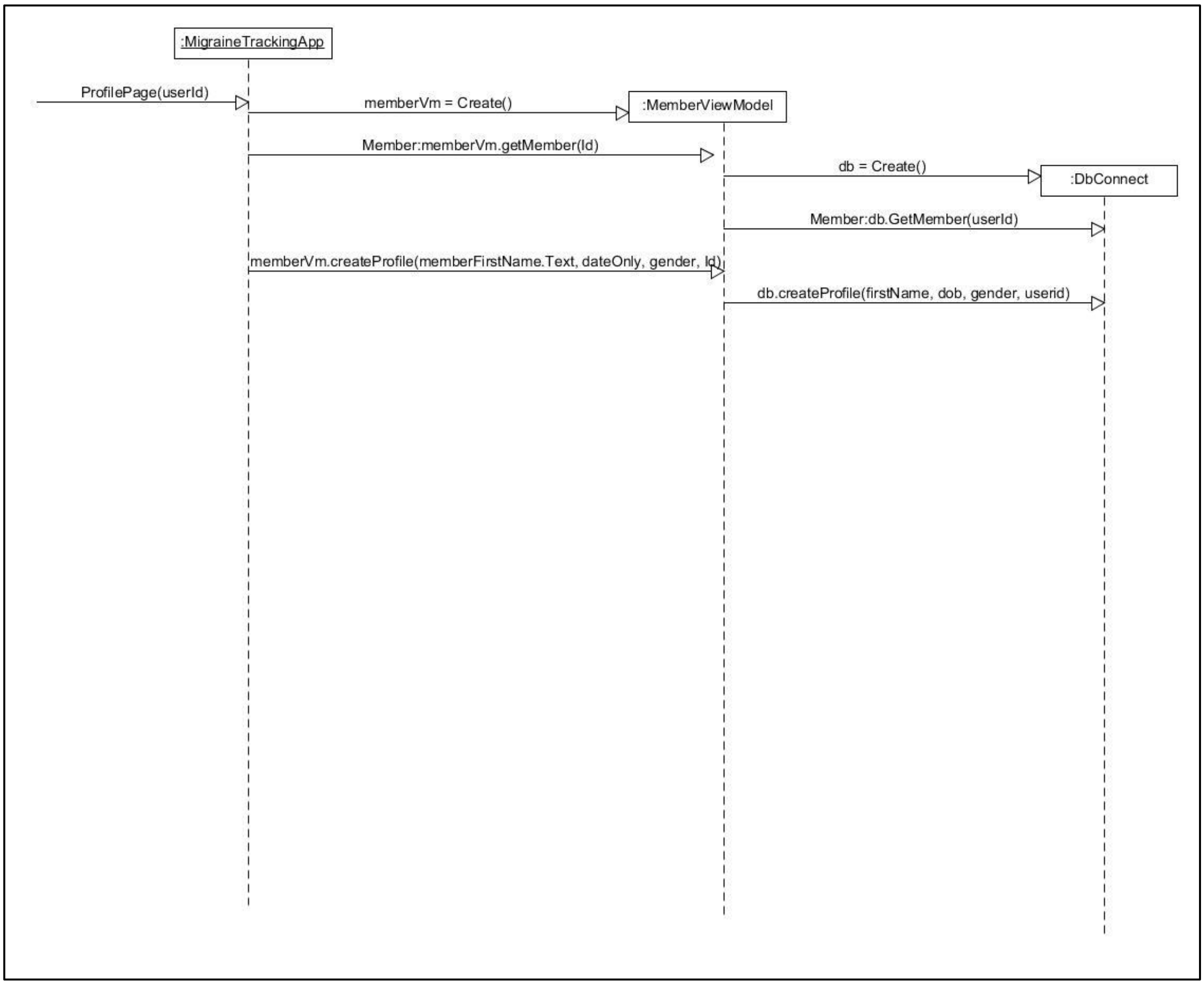

:MigraineTrackingApp
ProfilePage(userId)
memberVm = Create()
:MemberViewModel
Member:memberVm.getMember(Id)
db = Create()
:DbConnect
Member:db.GetMember(userId)
memberVm.createProfile(memberFirstName.Text, dateOnly, gender, Id)
db.createProfile(firstName, dob, gender, userid)

**Figure 3 - Create Profile Sequence Diagram**

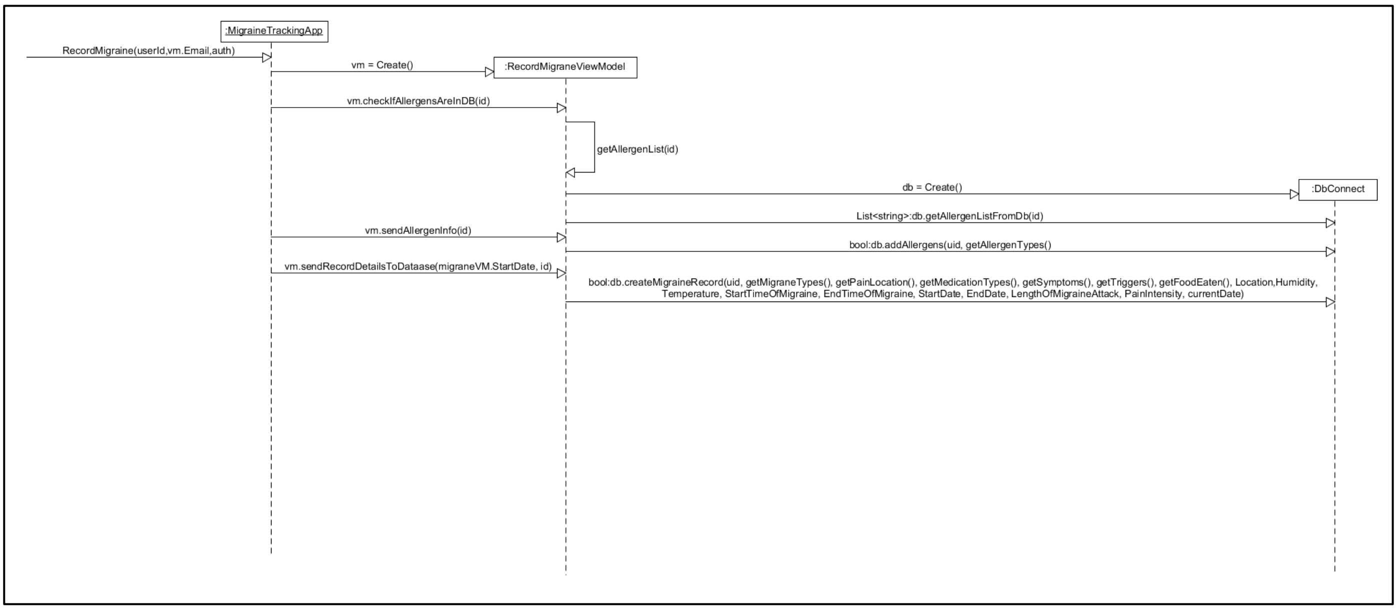

:MigraineTrackingApp
RecordMigraine(userId,vm.Email,auth)
vm = Create()
:RecordMigraneViewModel
vm.checkIfAllergensAreInDB(id)
getAllergenList(id)
db = Create()
:DbConnect
List<string>:db.getAllergenListFromDb(id)
vm.sendAllergenInfo(id)
bool:db.addAllergens(uid, getAllergenTypes()
vm.sendRecordDetailsToDataase(migraneVM.StartDate, id)
bool:db.createMigraineRecord(uid, getMigraneTypes(), getPainLocation(), getMedicationTypes(), getSymptoms(), getTriggers(), getFoodEaten(), Location,Humidity, Temperature, StartTimeOfMigraine, EndTimeOfMigraine, StartDate, EndDate, LengthOfMigraineAttack, PainIntensity, currentDate)

**Figure 4 - Create Migraine Record Sequence Diagram**

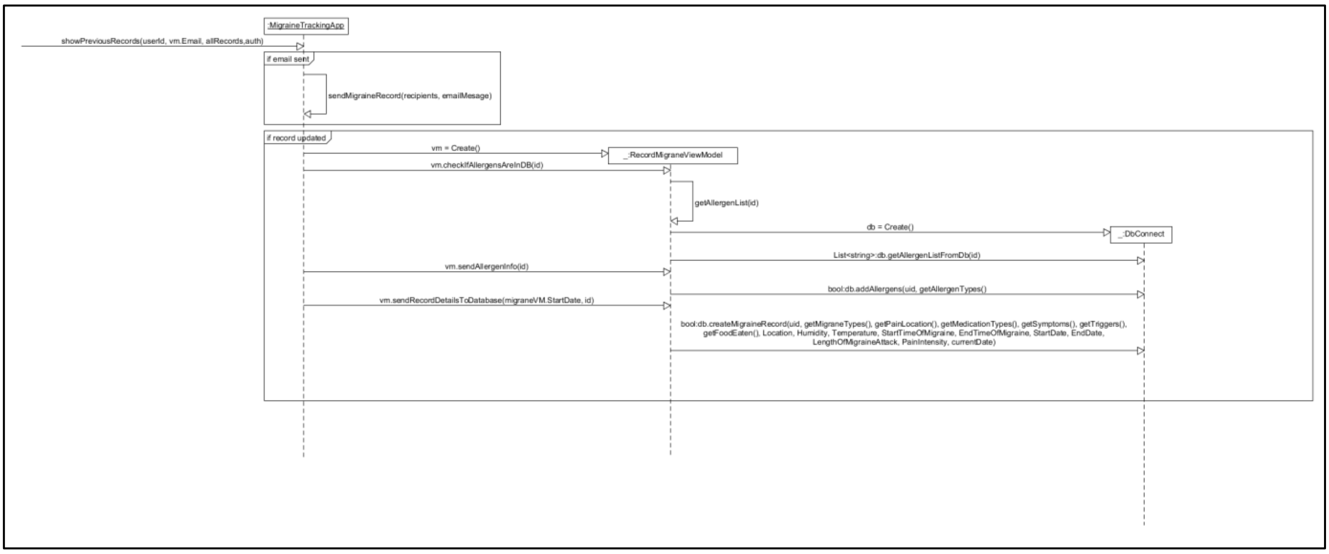

:MigraineTrackingApp
showPreviousRecords(userId, vm.Email, allRecords,auth)
if email sent
sendMigraineRecord(recipients, emailMesage)
if record updated
vm = Create()
_:RecordMigraneViewModel
vm.checkIfAllergensAreInDB(id)
getAllergenList(id)
db = Create()
_:DbConnect
List<string>:db.getAllergenListFromDb(id)
vm.sendAllergenInfo(id)
bool:db.addAllergens(uid, getAllergenTypes()
vm.sendRecordDetailsToDatabase(migraneVM.StartDate, id)
bool:db.createMigraineRecord(uid, getMigraneTypes(), getPainLocation(), getMedicationTypes(), getSymptoms(), getTriggers(), getFoodEaten(), Location, Humidity, Temperature, StartTimeOfMigraine, EndTimeOfMigraine, StartDate, EndDate, LengthOfMigraineAttack, PainIntensity, currentDate)

**Figure 5 - Show Previous Records Sequence Diagram**

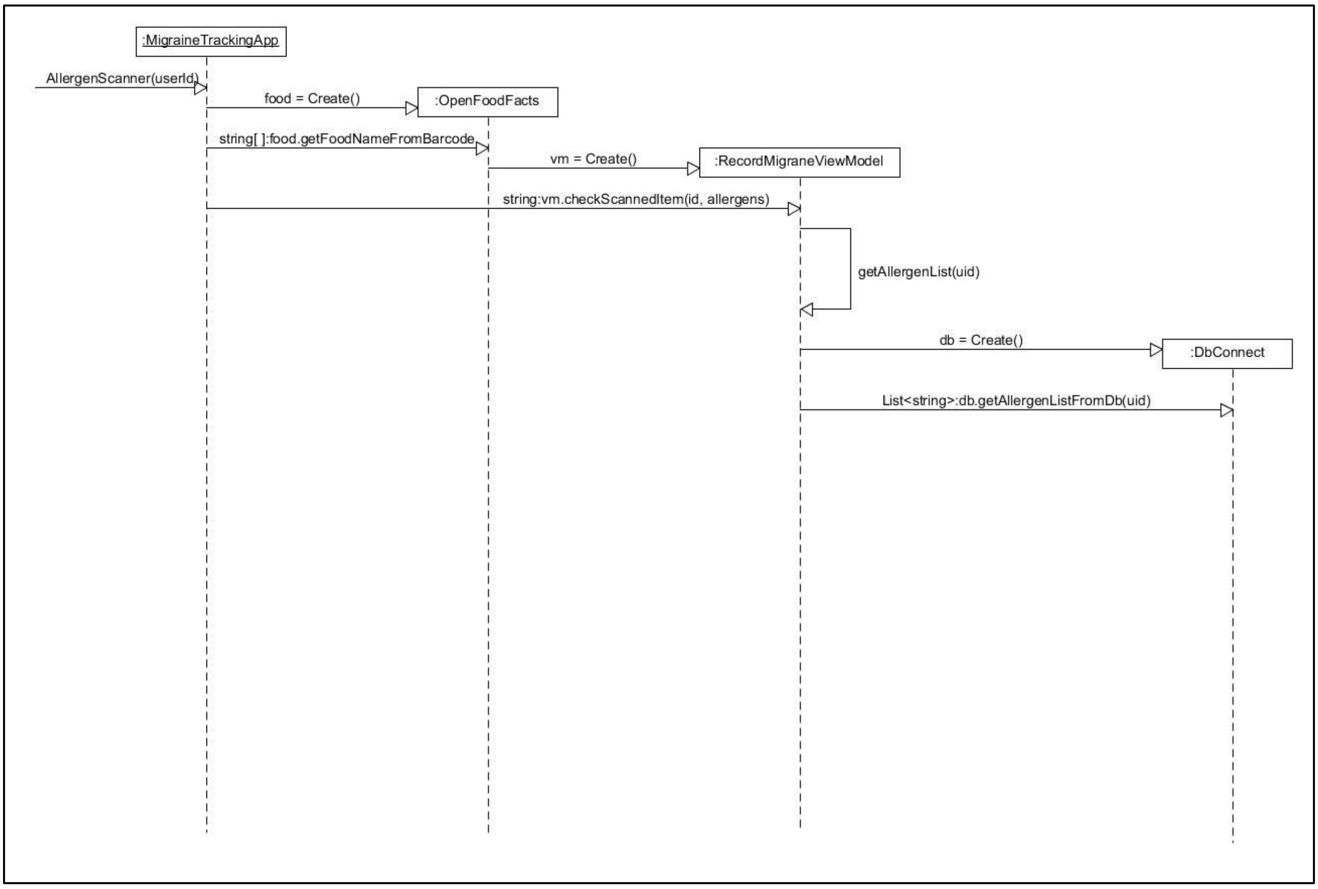


**Figure 6 - Allergen Scanner Sequence Diagram**

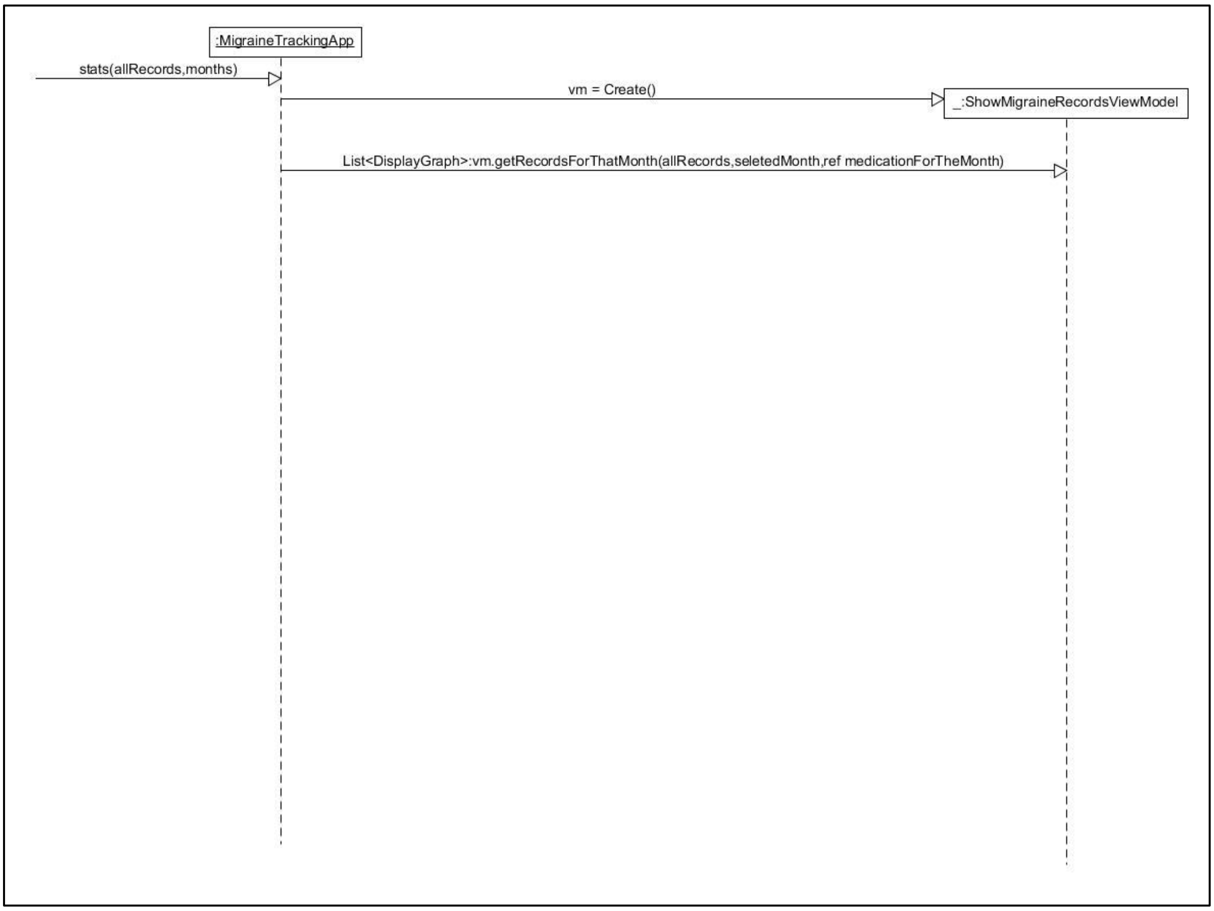

:MigraineTrackingApp
stats(allRecords,months)
vm = Create()
_:ShowMigraineRecordsViewModel
List<DisplayGraph>:vm.getRecordsForThatMonth(allRecords,seletedMonth,ref medicationForTheMonth)

**Figure 7 - Migraine Stats Sequence Diagram**

# 6. Class Diagram

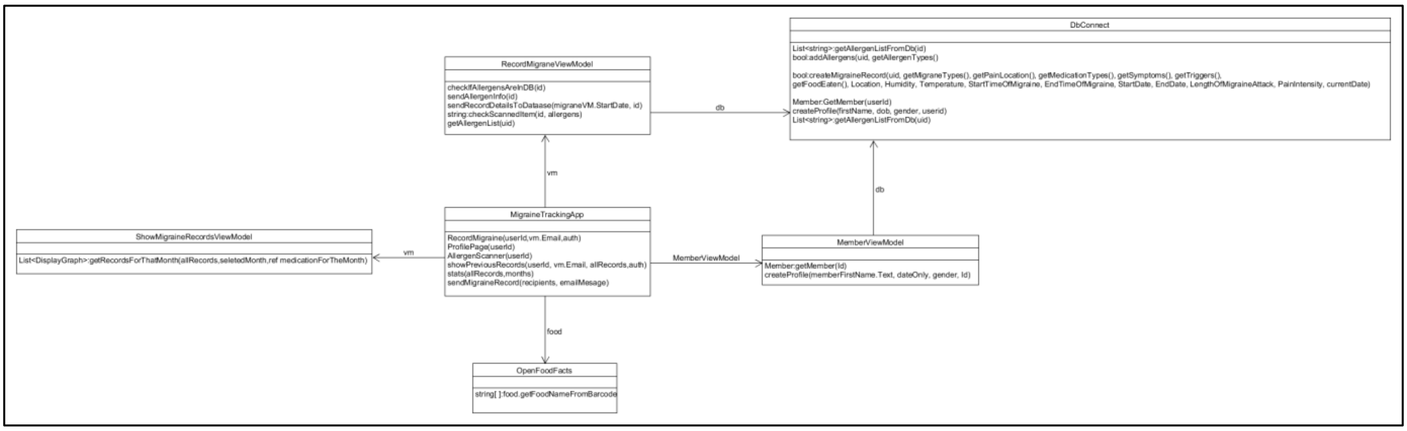
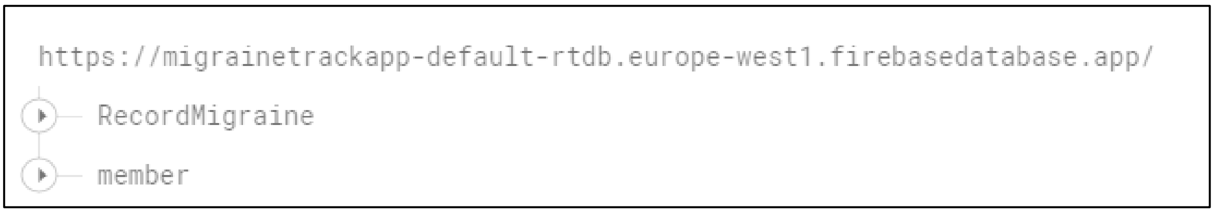

DbConnect
List<string>:getAllergenListFromDb(id)
bool:addAllergens(uid, getAllergenTypes()
bool:createMigraineRecord(uid, getMigraneTypes(), getPainLocation(), getMedicationTypes(), getSymptoms(), getTriggers(),
getFoodEaten(), Location, Humidity, Temperature, StartTimeOfMigraine, EndTimeOfMigraine, StartDate, EndDate, LengthOfMigraineAttack, PainIntensity, currentDate)
Member:GetMember(userId)
createProfile(firstName, dob, gender, userid)
List<string>:getAllergenListFromDb(uid)
RecordMigraneViewModel
checkIfAllergensAreInDB(id)
sendAllergenInfo(id)
sendRecordDetailsToDataase(migraneVM.StartDate, id)
string:checkScannedItem(id, allergens)
getAllergenList(uid)
db
vm
db
ShowMigraineRecordsViewModel
List<DisplayGraph>:getRecordsForThatMonth(allRecords,seletedMonth,ref medicationForTheMonth)
vm
MigraineTrackingApp
RecordMigraine(userId,vm.Email,auth)
ProfilePage(userId)
AllergenScanner(userId)
showPreviousRecords(userId, vm.Email, allRecords,auth)
stats(allRecords,months)
sendMigraineRecord(recipients, emailMesage)
MemberViewModel
MemberViewModel
Member:getMember(Id)
createProfile(memberFirstName.Text, dateOnly, gender, Id)
food
OpenFoodFacts
string[ ]:food.getFoodNameFromBarcode

**Figure 8 - Class Diagram**

# 7. Database Layout

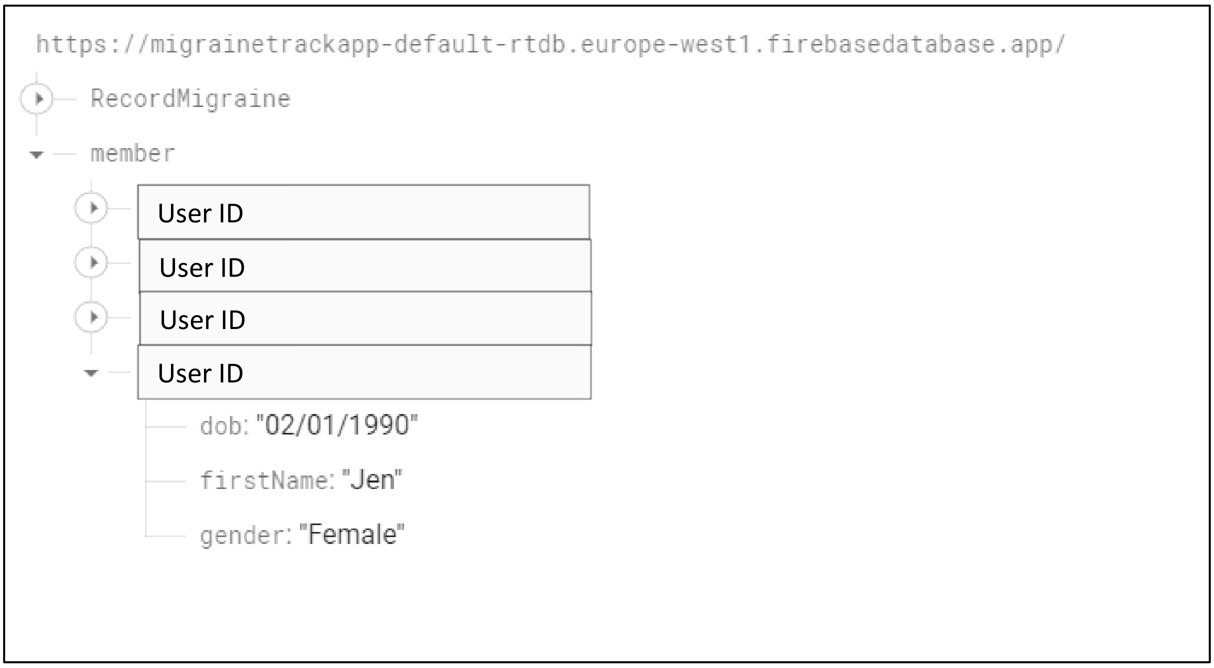


https://migrainetrackapp-default-rtdb.europe-west1.firebasedatabase.app/

RecordMigraine

member

**Figure 9 - Database Layout**



https://migrainetrackapp-default-rtdb.europe-west1.firebasedatabase.app/

RecordMigraine

member

User ID

User ID

User ID

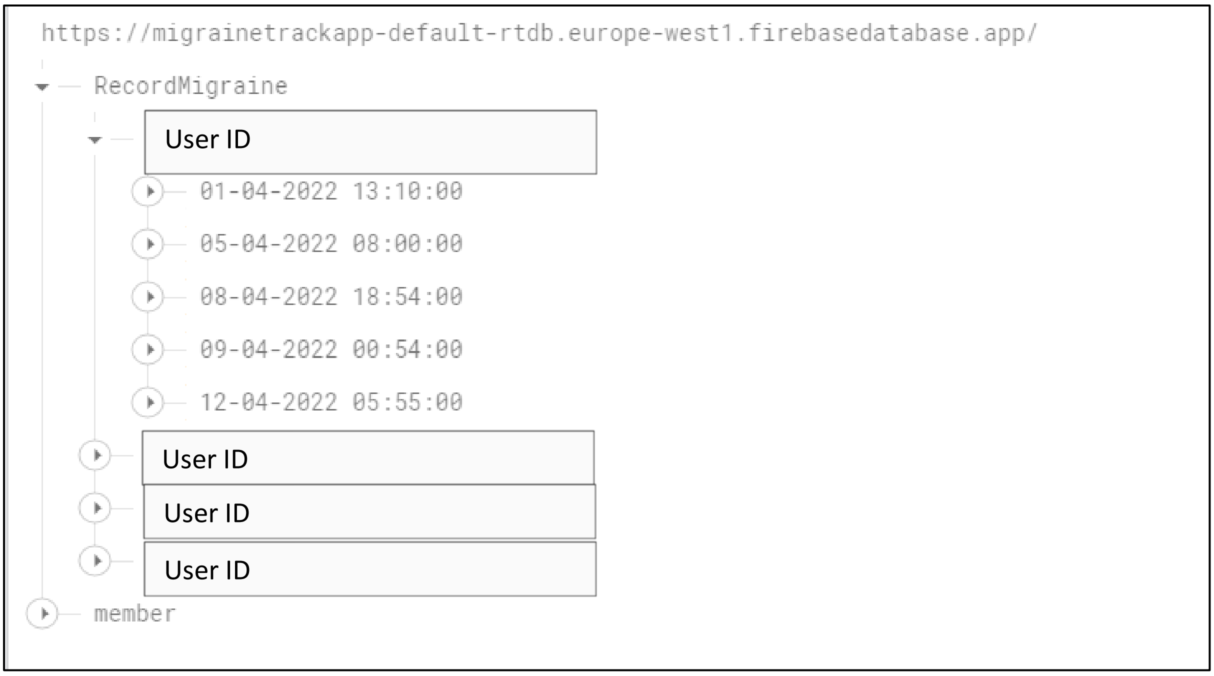
User ID

dob: "02/01/1990"

firstName: "Jen"

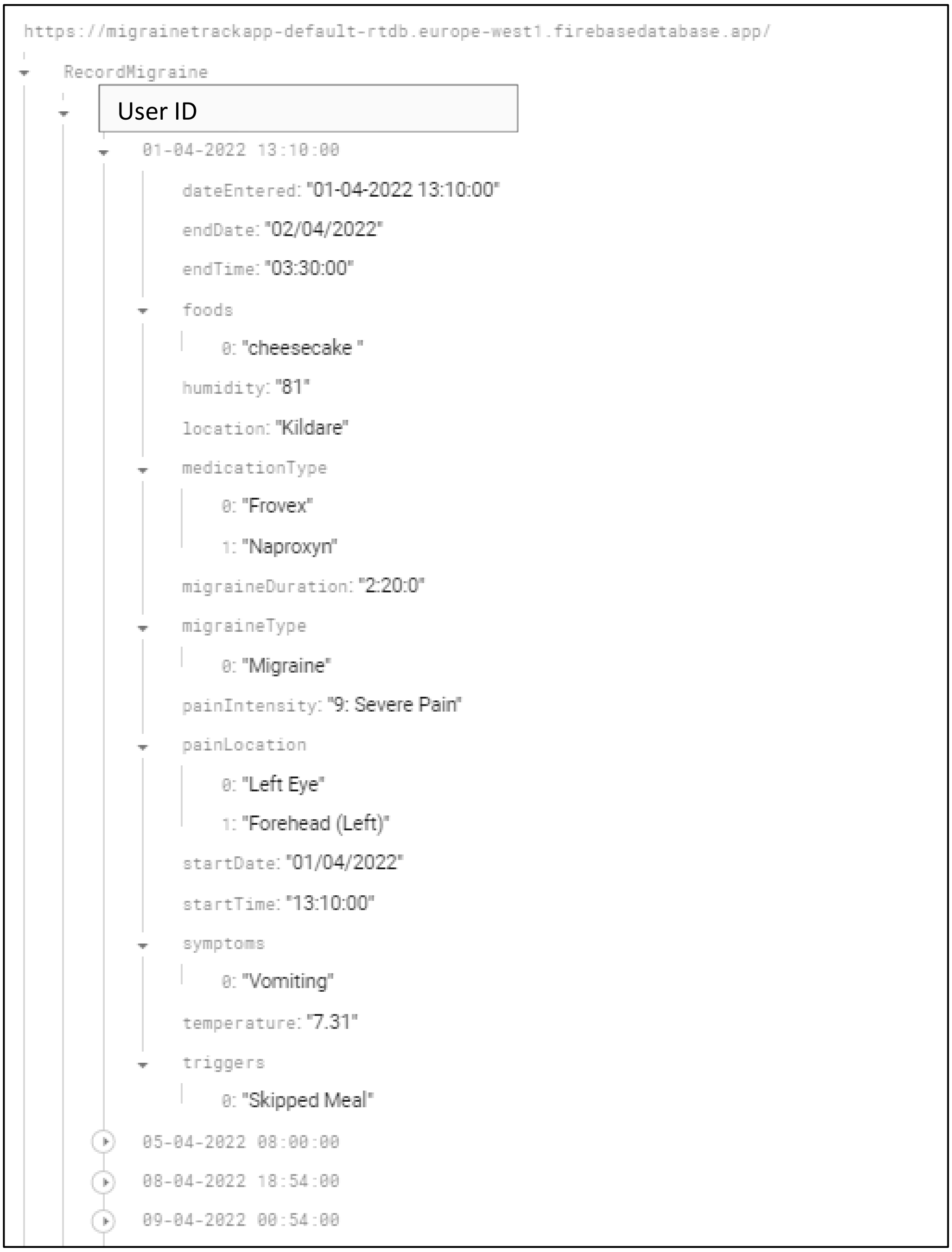
gender: "Female"

**Figure 10 - Database Layout (member)**



```
https://migrainetrackapp-default-rtdb.europe-west1.firebasedatabase.app/
RecordMigraine
    User ID
        01-04-2022 13:10:00
        05-04-2022 08:00:00
        08-04-2022 18:54:00
        09-04-2022 00:54:00
        12-04-2022 05:55:00
    User ID
    User ID
    User ID
member
```

**Figure 11 - Database Layout (RecordMigraine)**



```
https://migrainetrackapp-default-rtdb.europe-west1.firebasedatabase.app/
  RecordMigraine
    User ID
      01-04-2022 13:10:00
        dateEntered: "01-04-2022 13:10:00"
        endDate: "02/04/2022"
        endTime: "03:30:00"
        foods
          0: "cheesecake "
        humidity: "81"
        location: "Kildare"
        medicationType
          0: "Frovex"
          1: "Naproxyn"
        migraineDuration: "2:20:0"
        migraineType
          0: "Migraine"
        painIntensity: "9: Severe Pain"
        painLocation
          0: "Left Eye"
          1: "Forehead (Left)"
        startDate: "01/04/2022"
        startTime: "13:10:00"
        symptoms
          0: "Vomiting"
        temperature: "7.31"
        triggers
          0: "Skipped Meal"
      05-04-2022 08:00:00
      08-04-2022 18:54:00
      09-04-2022 00:54:00
```

**Figure 12 - Database Layout (RecordMigraine) expanded**

# 8. UI/UX Design

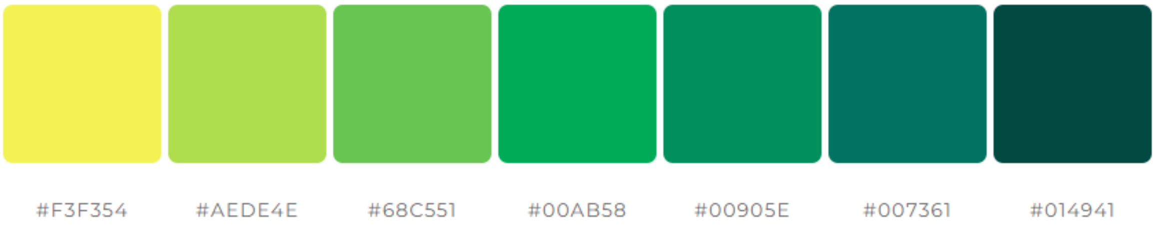


**Figure 13 - Application Colour Scheme**

According to Maurice B. Vincent, a migraine “disrupts vision more than any other motor or sensory function”[2]. This is one of the reasons the colour palette shown above has been chosen for the Migraine Tracking Application.

The colours chosen have decreased brightness which will allow the application to be easy on the user’s eyes and in turn avoid unwanted eye strain. The colours chosen also avoid the phenomena known as “colour vibration” where certain colours when next to each other appear to blur together and glow.

According to Harvard Medical School researchers at the Beth Israel Deaconess Medical Centre, “exposing migraine sufferers to a narrow band of green light significantly reduces photophobia and can reduce headache severity” [3]. The study also found that over 80% of migraine attacks are exacerbated by light sensitivity for this reason the screens within the application that may be used during an attack have a darker colour palette, other screens which will be mainly used when a migraine is not affecting the user do not adhere to a darker palette but do follow a green colour scheme to alleviate eye stress.

# 9. Prototype Screens

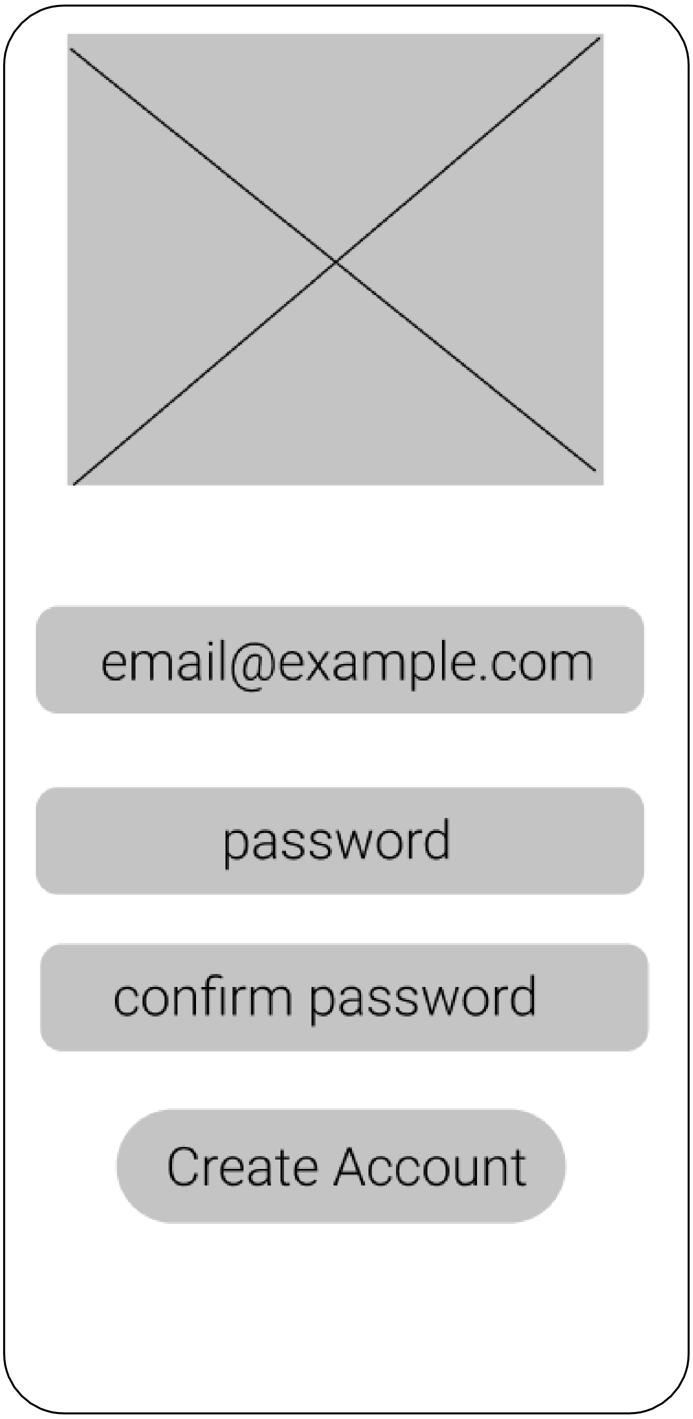


**Figure 14 - Create Account Prototype Screen**

This is the "Create Account" screen. Users will access this screen from the main page.

On this screen users can enter their email and password. Users can then press the "Create Account" button to create an account. They will then be brought to the home screen.

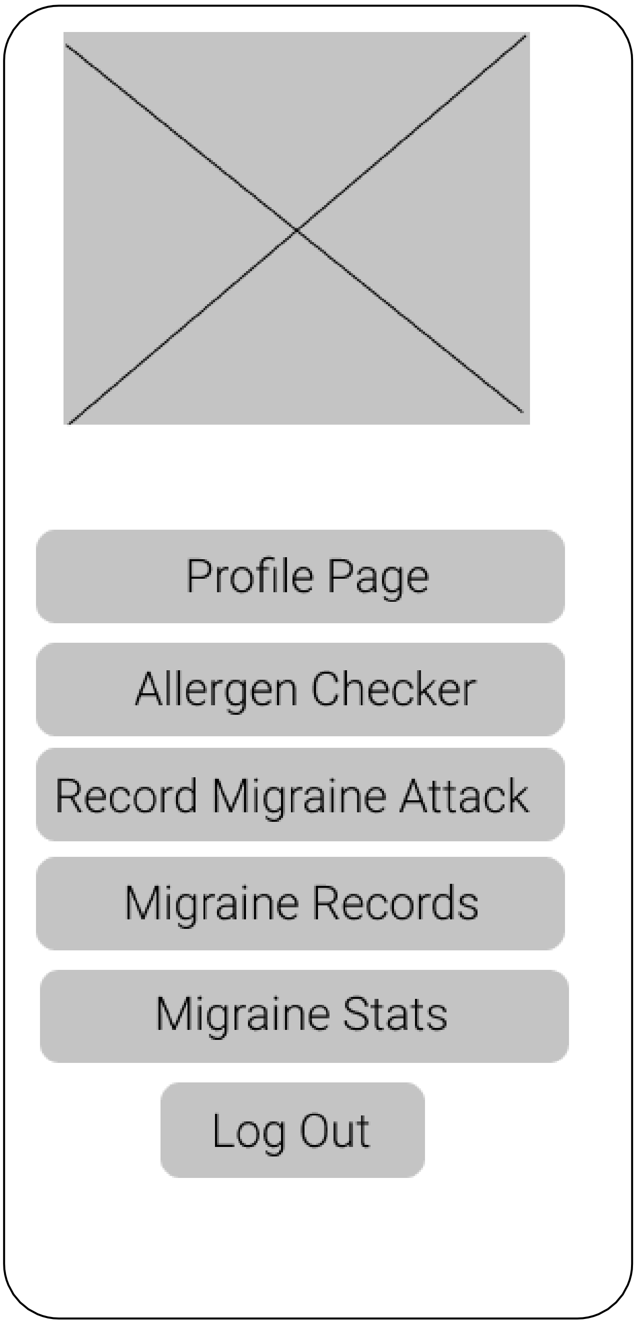


**Figure 15 - Home Screen Prototype Screen**

This is the home screen. On this screen members can navigate to other screens within the application.

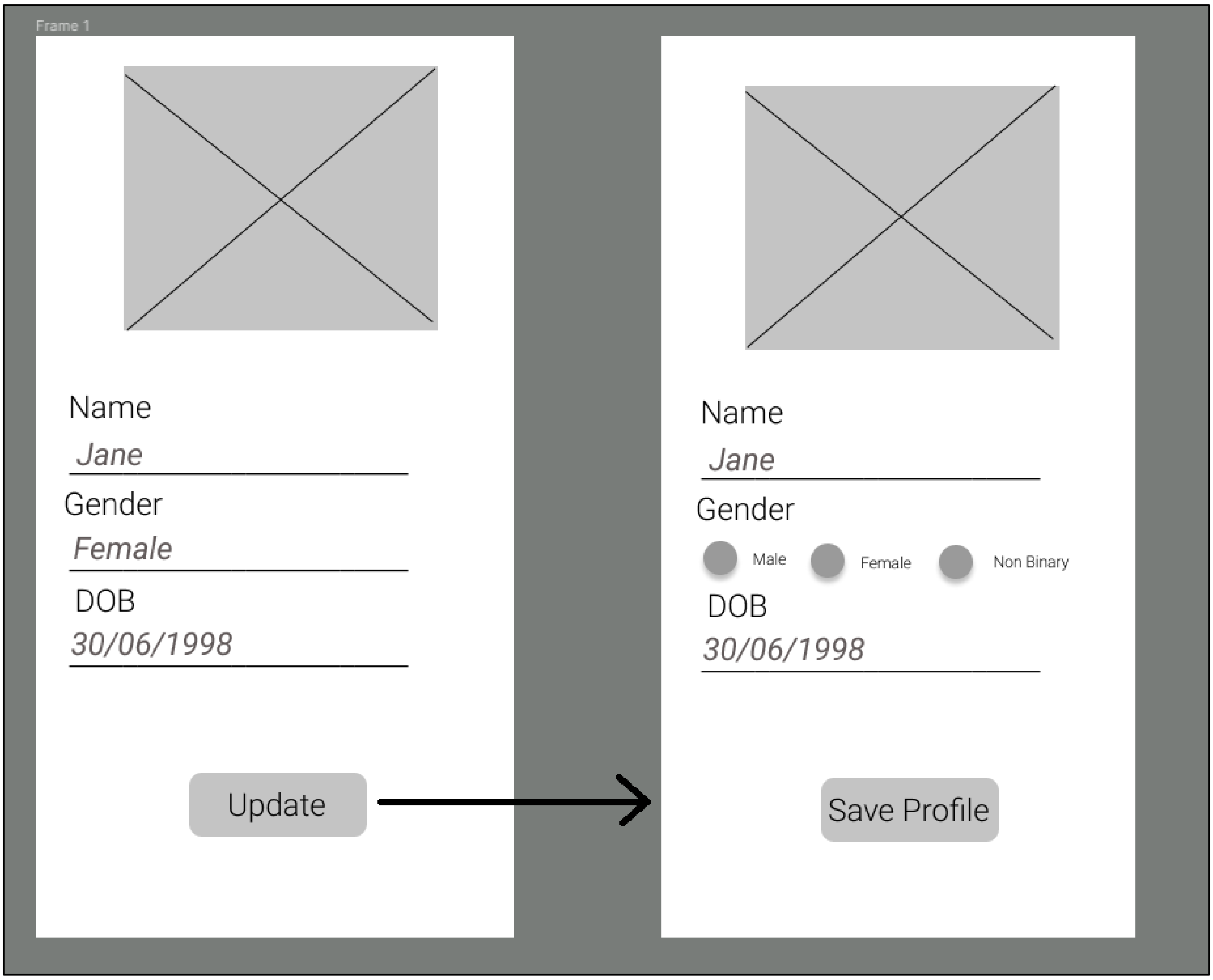


**Figure 16 - Profile & Update Profile Prototype Screens**

These are the profile and update profile screens. On the profile screen members can see their profile information they can also navigate to the update profile screen to update their information.

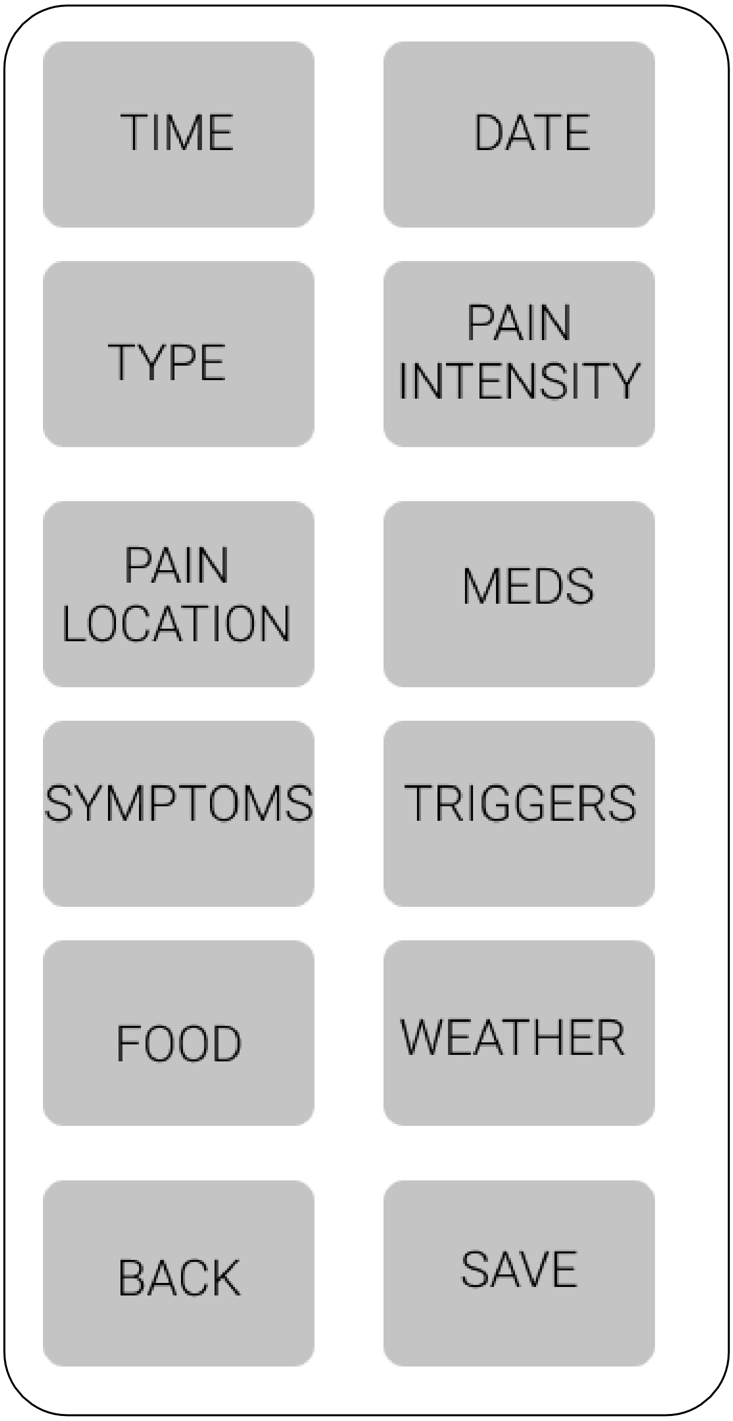


**Figure 17 - Record Migraine Screen Prototype**

This is the record migraine screen. On this screen members can navigate to different screens to record migraine information. Members can then save their migraine record.

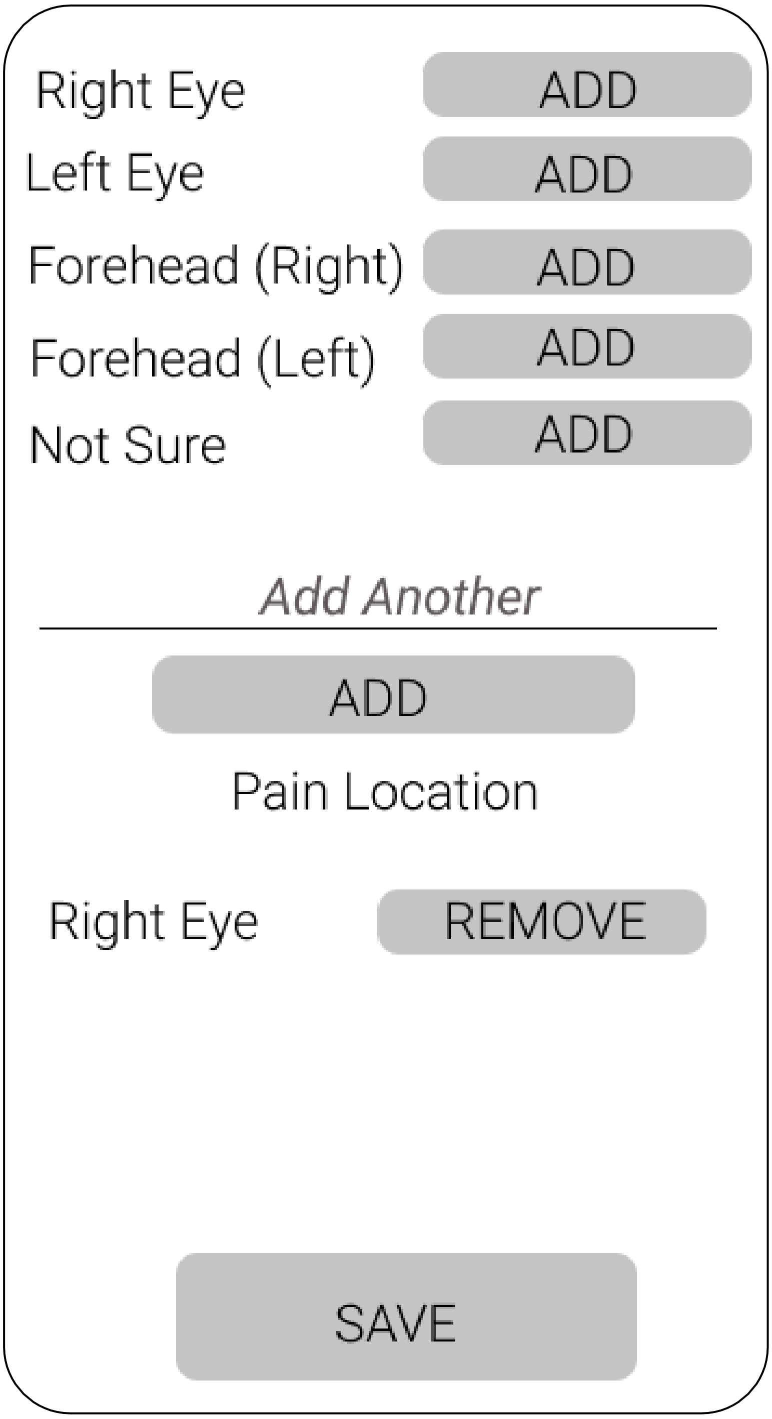


**Figure 18 - Add Pain Location Prototype Screen**

This is the record pain location screen (this will also be the layout for the record type, record pain intensity, record medication, record symptoms and record triggers screens).

On this screen members can add the information about their pain locations, they can also add their own and remove whichever they want from the list before saving.

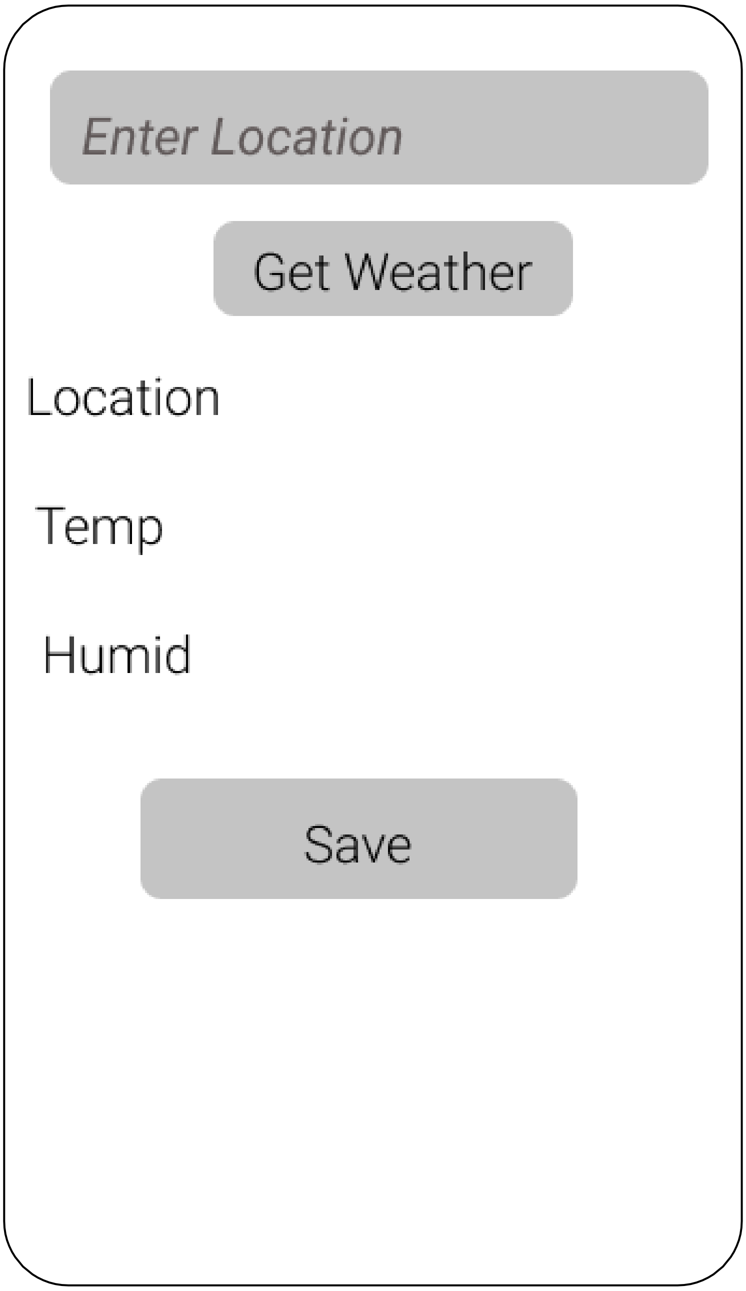


**Figure 19 - Record Weather Screen Prototype**

This is the record weather screen. This screen will be accessed by a member to record weather information.

The user enters a location and can save the information.

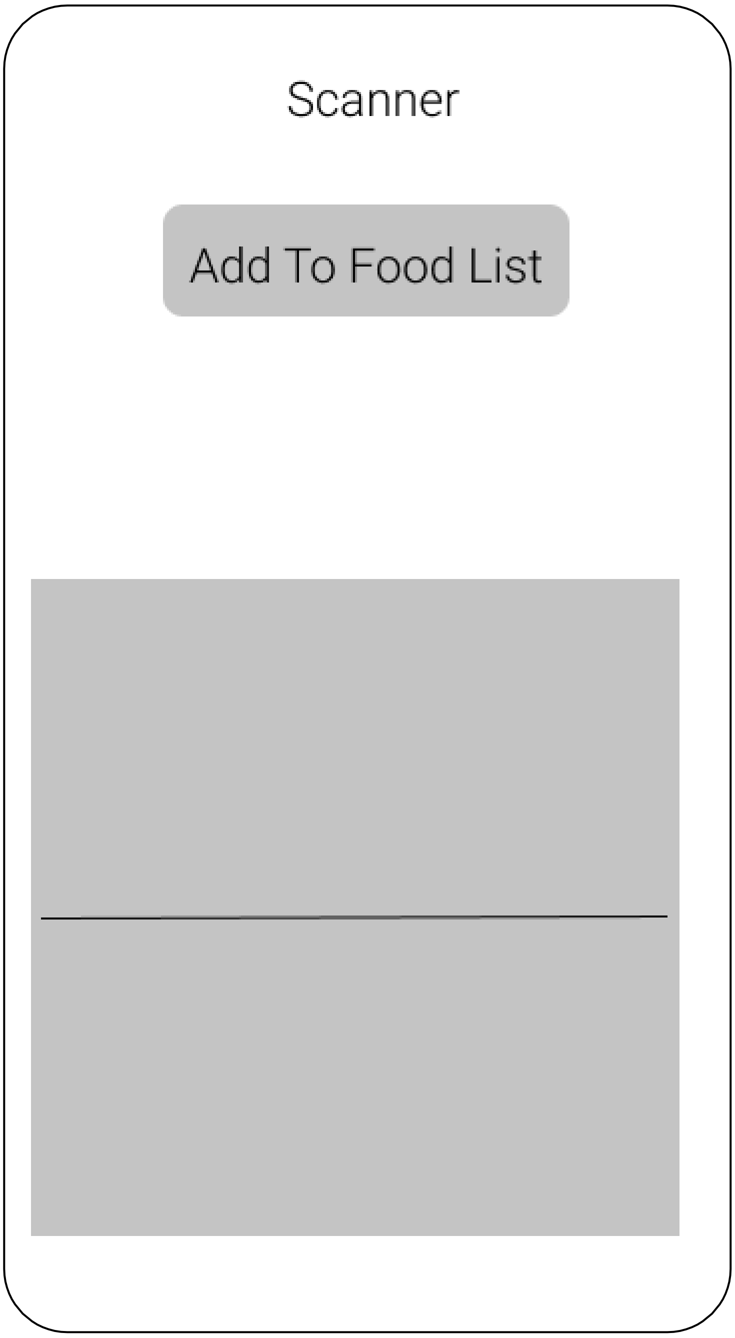


**Figure 20 - Barcode Scanner Prototype Screen**

This is the barcode scanner screen. This will be how to food item scanner and the allergen scanner will be displayed.

- **Food Scanner**: A member will navigate to the food scanner through the record migraine screen.
  The member will then use the camera to place a barcode within the line which will be displayed on the screen.
  When a barcode is detected the corresponding food item name will be displayed and the member can then add this item to their food list.

- **Allergen Scanner**: A member will navigate to the food scanner through the home page.
  The member will then use the camera to place a barcode within the line which will be displayed on the screen.
  When a barcode is detected the corresponding allergens will be displayed.

# 10. Conclusion

This document has successfully discussed the design features of the migraineHub application. The tools and technologies used have been discussed in detail as well as the design pattern being used.

Sequence and class diagrams have been provided to show the layout and interactions of the application.

The layout of the database has also been shown to provide a clear layout of how data will be stored.

The UI/UX has also been discussed in detail and reasons behind the UI/UX decisions have also been provided.

Finally, a number of prototype screens have also been provided to show the initial plan for the UI of the application.

# References

[1] Jeremy Likness, Model-View-ViewModel (MVVM) Explained (24/4/2014) Available at: https://www.wintellect.com/model-view-viewmodel-mvvm-explained/

[2] Maurice B Vincent, Vision and migraine (11/5/15), Available at: https://pubmed.ncbi.nlm.nih.gov/25758366/

[3] Jennifer Kritz, Green Light for Migraine Relief (17/5/16), Available at: https://hms.harvard.edu/news/green-light-migraine-relief