Solution Design Document

MyMediCare

**Callum McLaughlin**

24678805

Games Programming – Mobile Apps

25/02/2021

Contents

[1 Introduction 3](#_Toc67019353)

[1.1 Architectural Model 3](#_Toc67019354)

[1.1.1 Wireless Internet Architecture 3](#_Toc67019355)

[1.1.2 Smart Client Architecture 3](#_Toc67019356)

[1.2 Android Architecture with MVP or MVVM 4](#_Toc67019357)

[1.2.1 Model View Presenter (MVP) 4](#_Toc67019358)

[1.2.2 Model View View Model (MVVM) 4](#_Toc67019359)

[1.2.3 Design Choice 4](#_Toc67019360)

[1.3 Mobile Hardware/Platform Options 5](#_Toc67019361)

[2 Functional Specification 6](#_Toc67019362)

[2.1 Use Case Design 6](#_Toc67019363)

[2.2 Functional and Non-Functional Requirements 7](#_Toc67019364)

[2.2.1 Non-Functional Requirements 7](#_Toc67019365)

[2.2.2 Functional Requirements 8](#_Toc67019366)

[2.3 Sequence Diagrams 9](#_Toc67019367)

[2.3.1 Login Process 9](#_Toc67019368)

[2.3.2 Update Details Process 10](#_Toc67019369)

[2.4 Flowcharts 11](#_Toc67019370)

[2.4.1 Login Process/Application Start-up Process 11](#_Toc67019371)

[2.5 GUI UI/UX Design 12](#_Toc67019372)

[2.5.1 Login Screen 12](#_Toc67019373)

[2.5.2 Main Screen/Home Screen 13](#_Toc67019374)

[2.5.3 Enter Patient Details 14](#_Toc67019375)

[2.5.4 Edit Patient Details 15](#_Toc67019376)

[2.5.5 Send Message to GP 16](#_Toc67019377)

[2.5.6 Customise App 17](#_Toc67019378)

[3 Solution Design 18](#_Toc67019379)

[3.1 Class Diagram 18](#_Toc67019380)

[3.2 Architectural Design 18](#_Toc67019381)

[3.3 Interface Design 19](#_Toc67019382)

[4 Testing 20](#_Toc67019383)

[5 Bibliography 24](#_Toc67019384)

# Introduction

The objective is to create an easy to use and fit for purpose mobile application that will enable GPs to assess conditions and to perform basic health checks remotely, using the information of the patients on the application.

The application will be used in General Practices by General Practitioners and it will enable them to remotely assess conditions and ailments over the phone whilst having access to patient data on the application. The application will also be used by patients who will be able to sign in/register and add their health details to the app when registering with the GP. The users will be able to update their details at any time should they need to. It has been stated in the client requirements that most of the patients make use of old mobile phones so its safe to assume that a large number of the patients may be elderly and therefore not as computer literate as others, they should be accommodated and kept in mind when designing the application. The user will be able to customise the design of the application in order to give them the most comfortable experience while using the application.

The group may be having problems with having up to date information on their patients, without an application like the one being suggested, the GP may not have up-to-date details on most of their patients until their next visit which may lead to the doctors having to update the records during appointment time which would eventually add up and could even cause delays to appointments.

By having an application that allows the patients to remotely update their details and health records easily and without any assistance from the doctors themselves or the GP receptionists, this makes for a significantly more efficient way of managing and keeping patients details as up to date as possible. Having up-to-date patient records also allows the doctors to be able to do their job better when trying to diagnose patients.

# 1.1 Architectural Model

There are a number of architectural models available for use and I will talk briefly about each before identifying and justifying which I have selected. When selecting the architecture, there are a number of factors that must be considered such as who the application users are, the device type, connectivity, data, and security just to name a few.

### Wireless Internet Architecture

Wireless Internet Architecture (Web App Architecture) uses the internet to connect mobile devices and tablets to the web server and subsequently the data sources through the base station through the use of HTTP. Use of this architecture comes with a number of benefits such as up to data availability, only one code base to maintain, minimal to no software deployment, familiar user interface and good security as the data is store on a server and not on the user device, minimising breaches. There are also a number of disadvantages to this architecture such as the user interface being too simple in some respects, application performance can be a problem especially on older devices and the biggest disadvantage being that the app would not be on the app store.

### Smart Client Architecture

Smart Client Applications give us many useful features for the end users. This architecture provides a rich interface with persistent data storage, allowing the user to access data even when they are not connected to the network as the data is stored remotely on their device, making smart client applications suitable for a variety of large-scale applications. As mentioned, the main advantage of smart client is the “always available data” due to the persistent data storage. It also has great performance and is very cost effective. The downsides to this architecture come when the application needs to be deployed and integrated into the business, mobile viruses also become a problem and development can be complex as multiple coding languages and techniques are usually needed as well as multiple development cycles. (eTutorials, 2020)

## Android Architecture with MVP or MVVM

“The Android default templates encourage the creation of large activities or fragments” (Vogel, 2017). These elements usually have both business and UI logic which makes testing and maintenance of the Android applications harder. There are two popular architecture choices within the Android community that allow for improved testability. The two most popular choices are:

* Model View Presenter (MVP)
* Model View View Presenter (MVVP)

### Model View Presenter (MVP)

The Model view presenter architecture allows for improved application architecture which in turn allows increased testability. The MVP architecture separates the datamodel from a view through a presenter. The view in MVP contains a visual part for the application and only contains the UI and it has no knowledge of the data that is displayed. The presenter triggers the business logic and tells the view when to update and transforms data from the model to update the view when necessary. The Model contains a data provider and the code to fetch and update the data. This section of MVP updates the database or if using a webserver, it communicates with that.

### Model View View Model (MVVM)

The Model View View Model design pattern is also known as Model View Binder. There are three core elements in the MVVM pattern: the model, the view, and the view model. Each serves a unique purpose. The view is responsible for defining the structure, layout, and appearance of what the user sees on screen. The view model implements the properties and commands to which the view can bind data to and lets the view know of any state changes through the change notification events. Finally, the model classes are non-visual classes that will encapsulate the applications data.

### Design Choice

When creating the MyMediCare application I will use the MVVM design pattern and the main reason for this is that the MVP architecture tends to have longer code that makes it harder to test the application. I want to be able to test my application thoroughly when it is completed to ensure there are no major flaws, which is my reasoning for choosing the MVVM architecture for my application design.

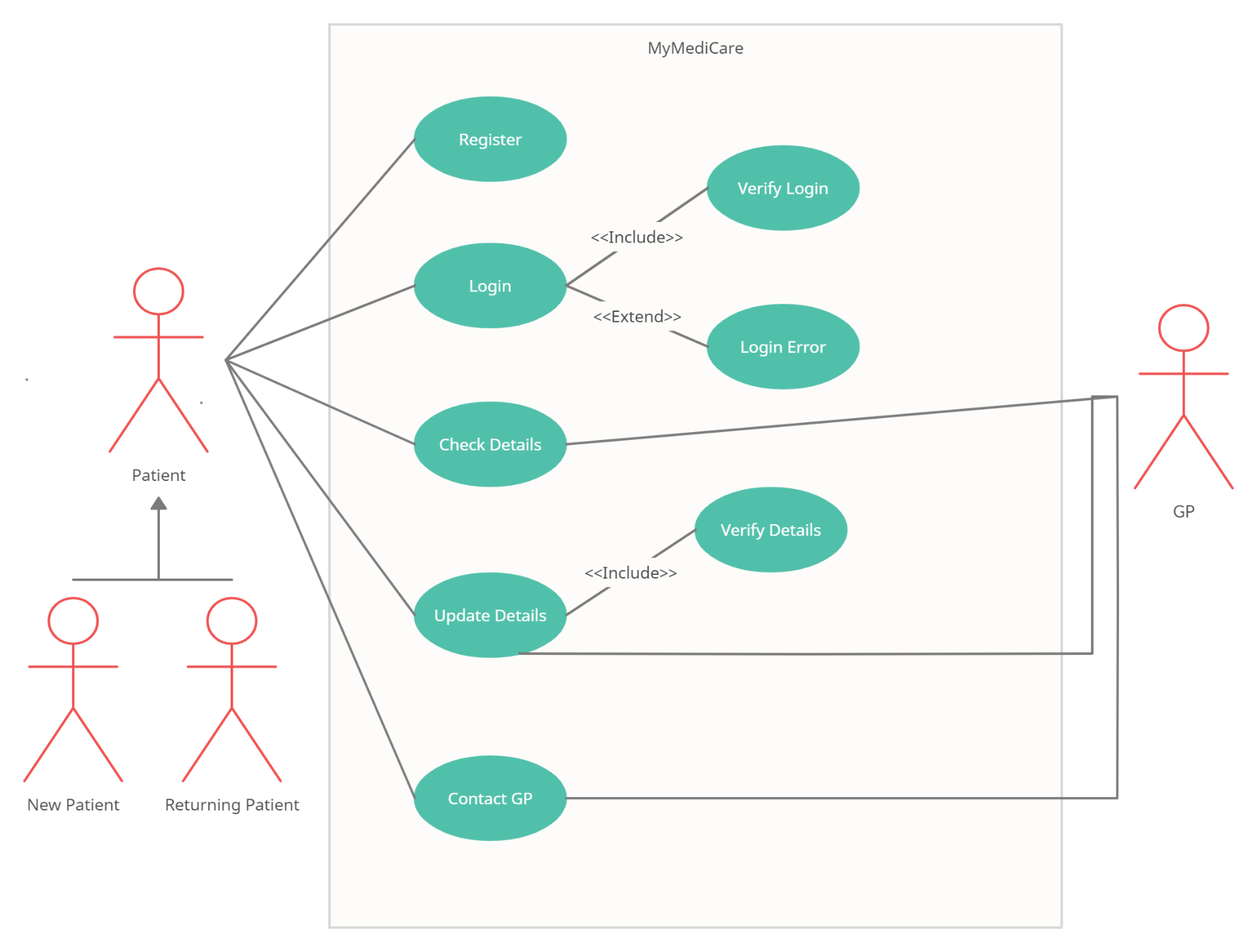
## Mobile Hardware/Platform Options

There are a number of mobile frameworks available for developing mobile apps that could be used to create the MyMediCare application. The one that will be used to create this application will be Android Studio. When building the application there is also two ways of building the UI, programmatically in java or using XML. For this application build, XML will be used as it is generally better. The reason for building the application on android is because android OS powers all types of mobile devices and it is an open-source platform.

The application could also be built for iOS. This could be done using Xamarin which is an Open-Source mobile application platform for .NET. Xamarin allows for development of a single mobile application which can then be deployed onto all platforms including iOS, android, and windows. Xamarin does this by building separate versions of the application at the same time, so if you start building the application in the android files, it translates the equivalent of what you have created in those files and put it into the iOS and windows files. Although not everything can be transferred Xamarin has up to 90% code transference meaning development for multiple platforms is much easier this way as you would only need to fix or add 10% or the code to the iOS version. (Microsoft, 2021)

# Functional Specification

## Use Case Design



## Functional and Non-Functional Requirements

### Non-Functional Requirements

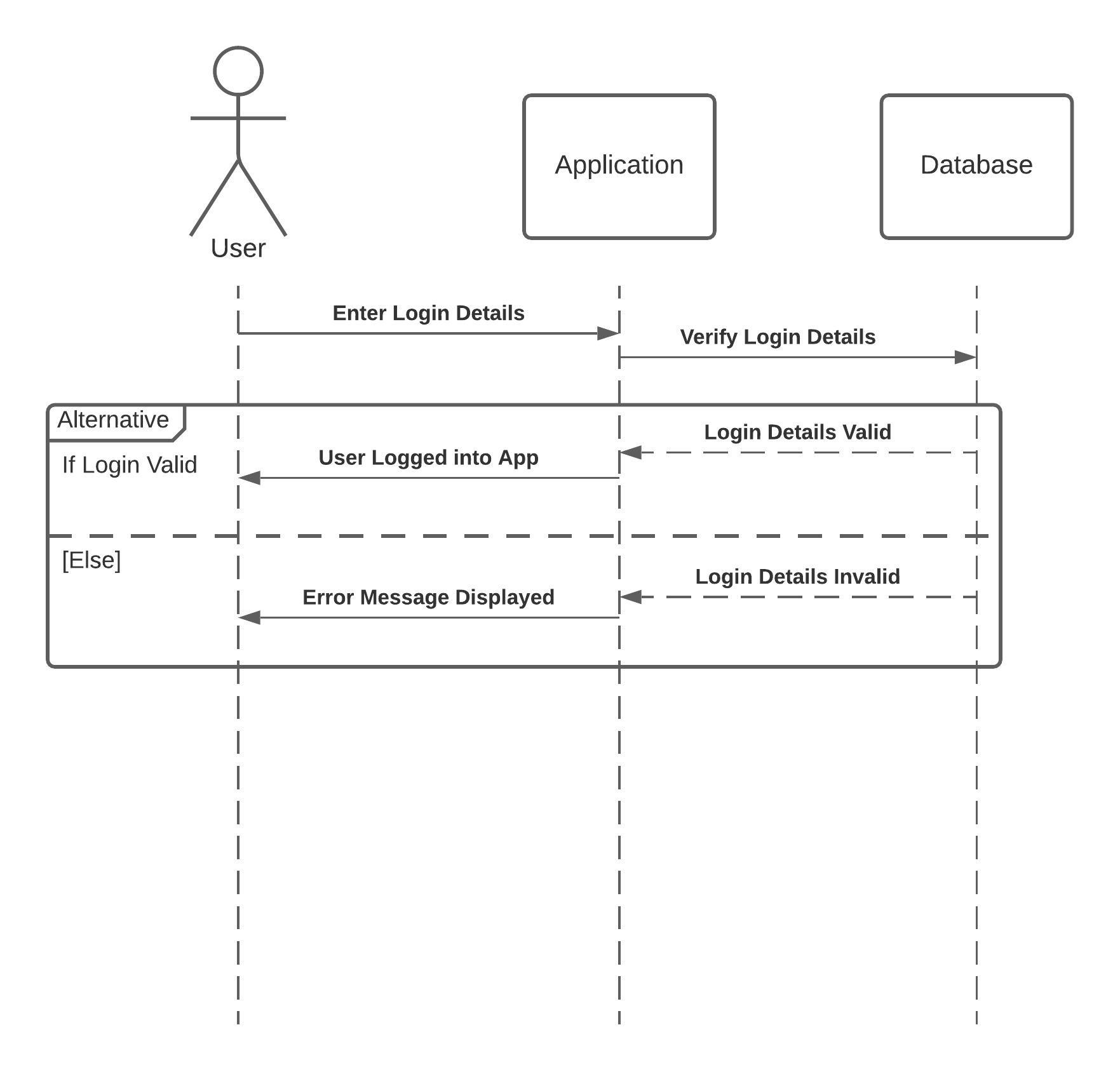
|  |  |  |
| --- | --- | --- |
| **No** | **Description** | **Category**  **(E, I or D)** |
| *R01* | *The system must allow new users to register on the application/create an account.* | *E* |
| *R02* | *The system must prompt new users to create an account on first use* | *E* |
| *R03* | *Registered users must login with their username and password.* | *E* |
| *R04* | *Registered users must be able to change their password.* | *E* |
| *R05* | *Registered users must be able to update their medical details.* | *E* |
| *R06* | *Users should be able to contact their GP via Text through the app.* | *I* |
| *R07* | *GP should be able to access details of registered patients* | *E* |
| *R08* | *User should be able to customise the background/foreground colour* | *D* |
| *R09* | *User should be able to customise the font size* | *D* |
| *R10* | *System should classify patient details into one of three categories. Normal, Low Risk and High Risk.* | *E* |
| *R11* | *System should be able to obtain readings automatically and remotely from patients’ device* | *E* |
| *R12* | *System should also be able to interact with address book in order to obtain GP contact details and send SMS or emails* | *D* |
| *R13* | *The app should follow the design guidelines using common UI patterns and recognisable icons* | *D* |

### Functional Requirements

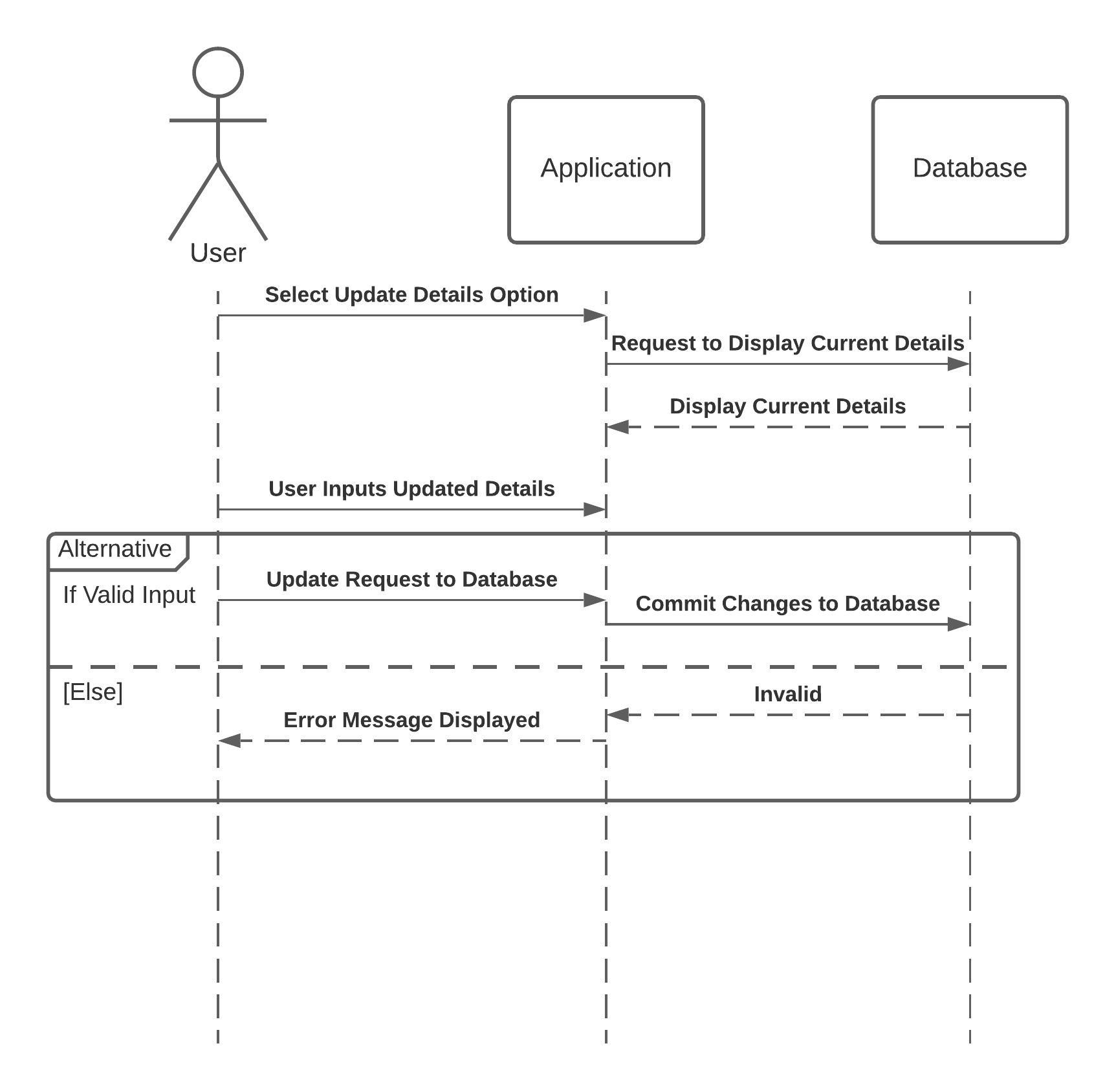
|  |  |  |
| --- | --- | --- |
| **No** | **Description** | **Category**  **(E, I or D)** |
| *R01* | *The system should have a welcome screen upon start-up* | *D* |
| *R02* | *The system shall store patient data persistently* | *E* |
| *R03* | *The system shall allow for new patient accounts to be created* | *E* |
| *R04* | *The system shall allow registered patients to update/edit their details and medical information* | *E* |
| *R05* | *The system shall display the patient details for the GP* | *E* |
| *R06* | *The system shall allow patients to contact their GP using the application* | *I* |
| *R07* | *The system shall place the patient into one of three categories depending on their health information. Normal, Low Risk and High Risk.* | *E* |
| *R08* | *The system shall allow the user to customise their application through a settings menu* | *D* |
| *R09* | *The System shall only allow GPs to view the details of all patients in their practice* | *E* |
| *R10* | *The System should be able to verify login details of users by checking the entered details against the database* | *E* |
| *R11* | *The system should allow users to reset their password, updating it in the database* | *E* |
| *R12* | *The system should allow users to delete their accounts, removing them from the database* | *I* |
| *R13* | *The system should allow the user to log out* | *E* |
| *R14* | *The system should have a patient view and GP view, allowing the GP to view all patient details but only allowing patients to see their own* | *E* |

## Sequence Diagrams

### Login Process

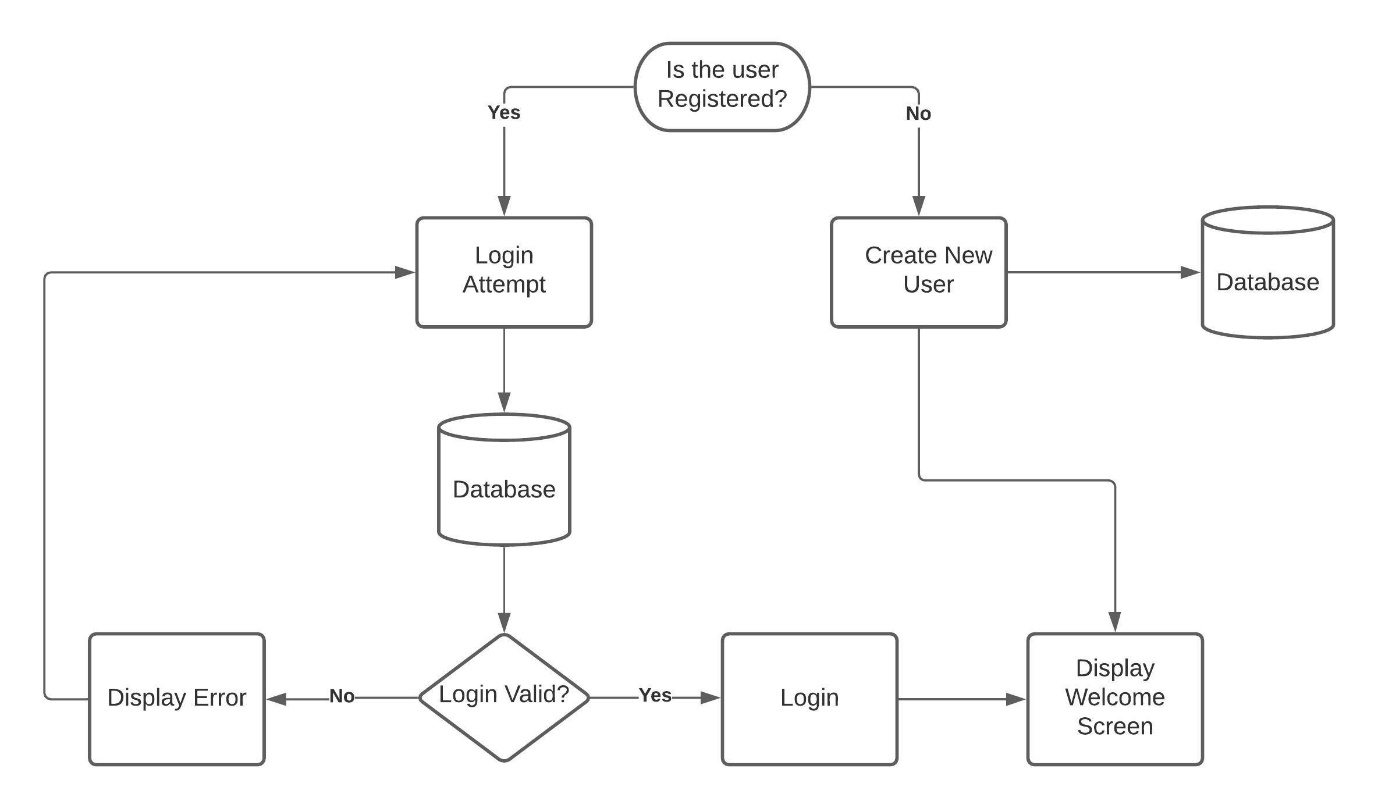


### Update Details Process



## Flowcharts

### Login Process/Application Start-up Process



## GUI UI/UX Design

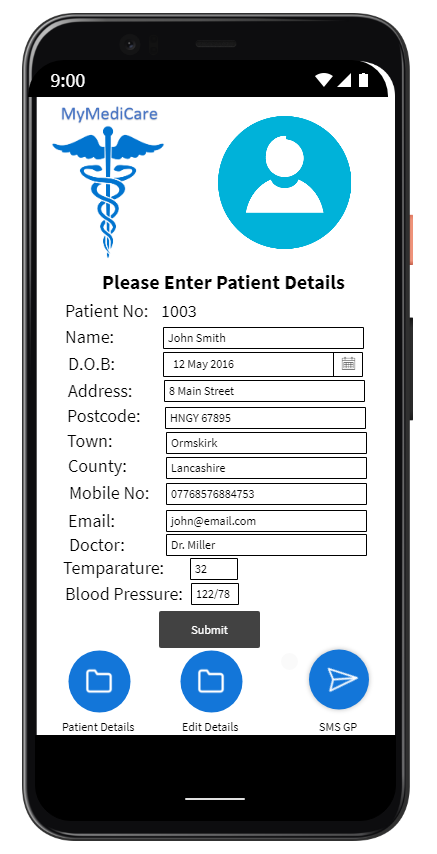
### Login Screen



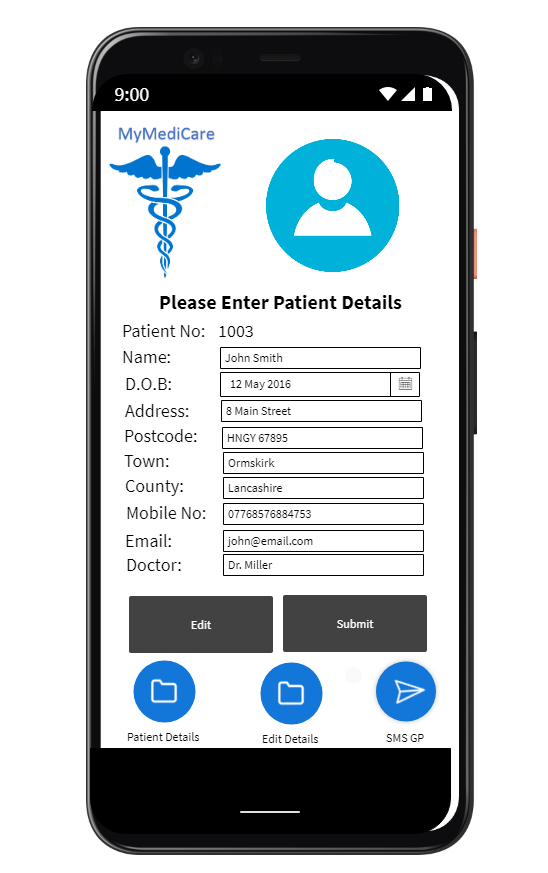
### Main Screen/Home Screen



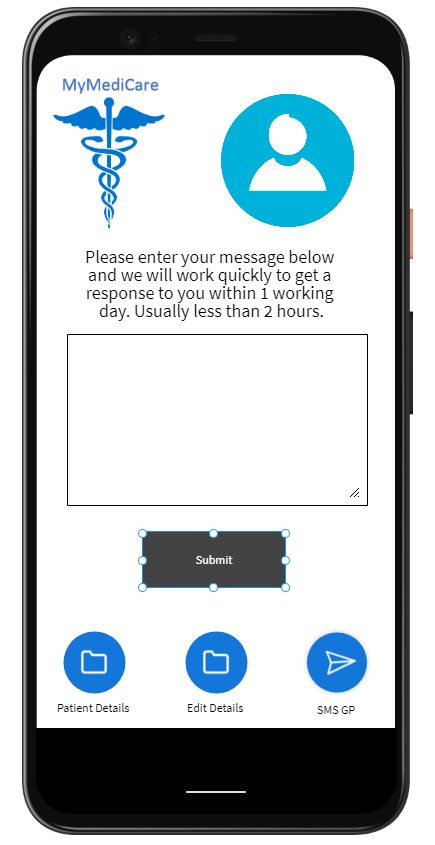
### Enter Patient Details



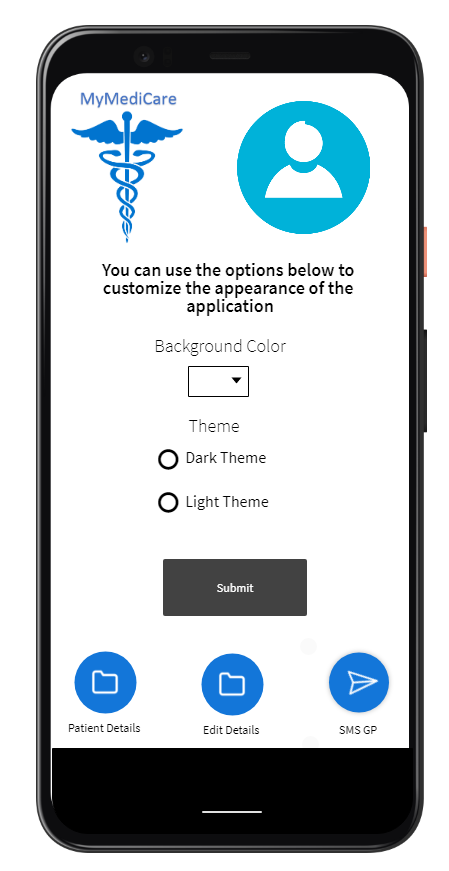
### Edit Patient Details



### Send Message to GP

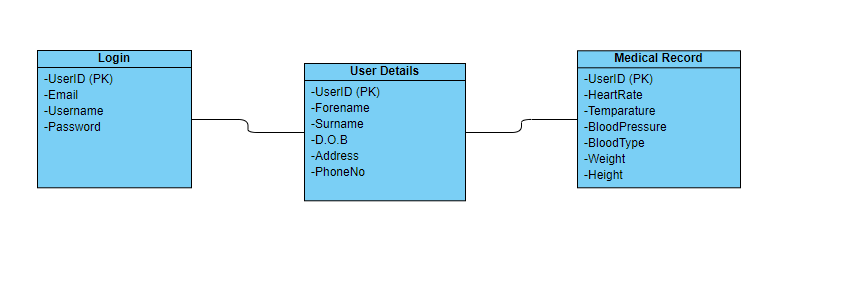


### Customise App

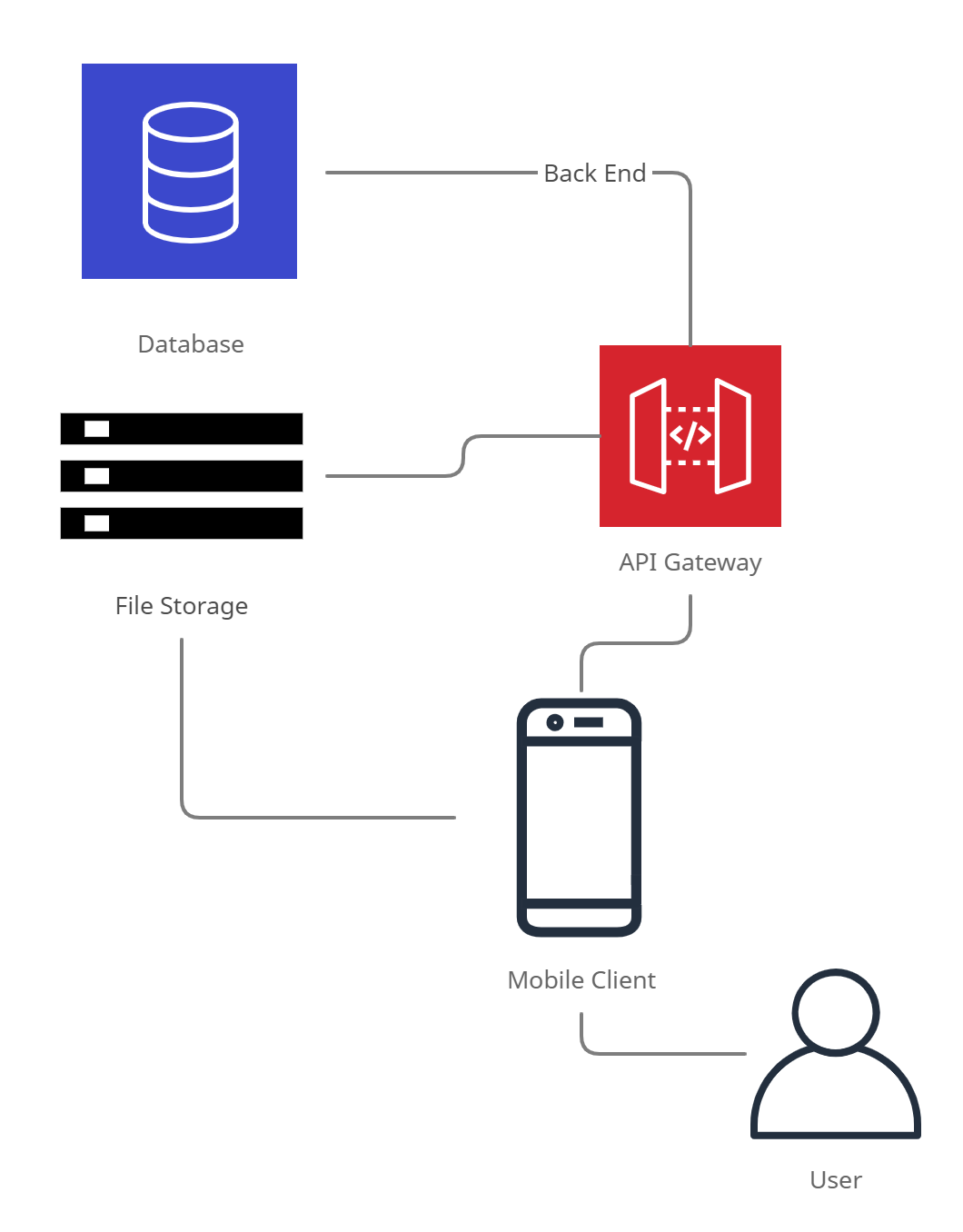


# Solution Design

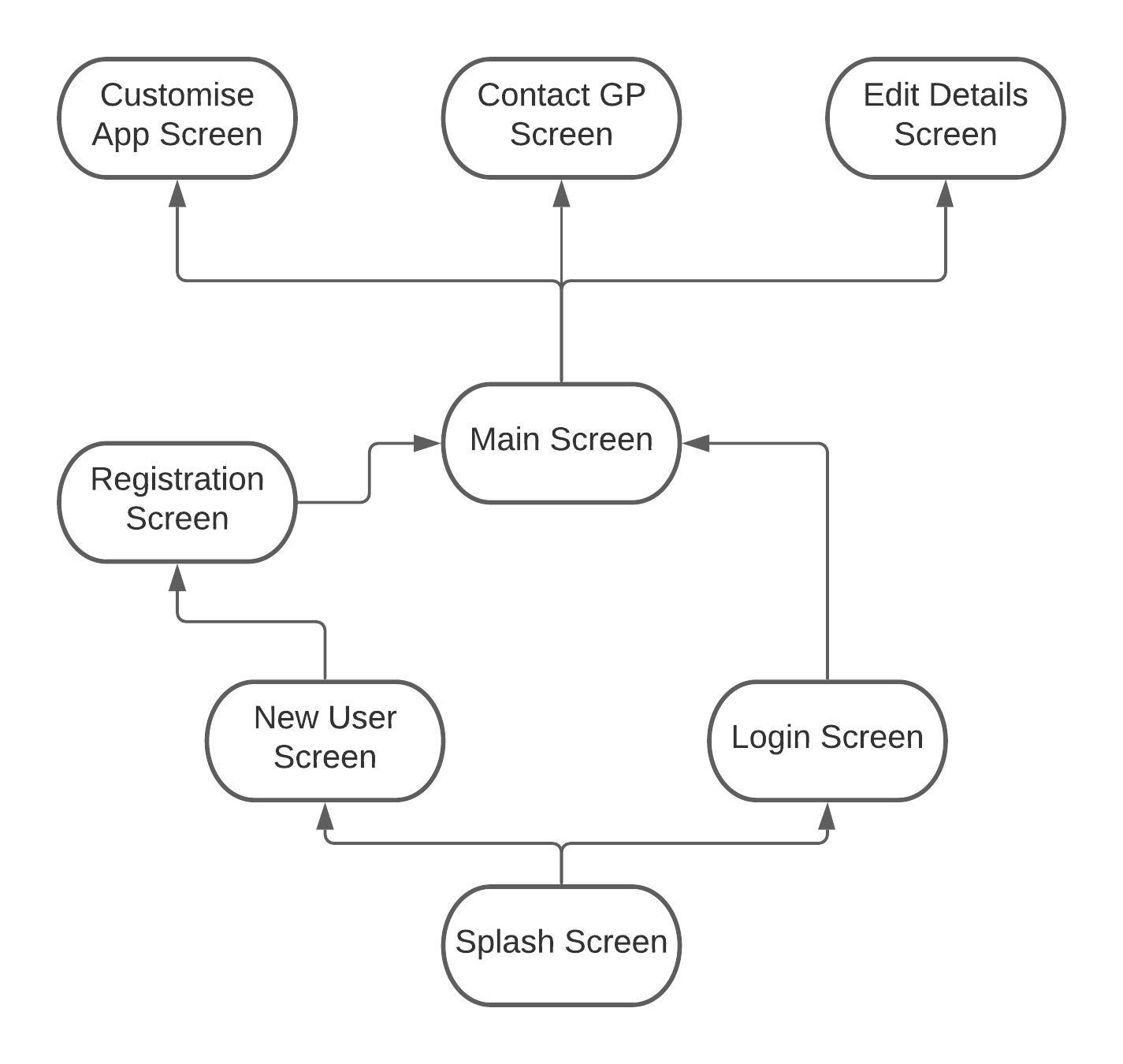
## Class Diagram



## Architectural Design



## Interface Design

**

# Testing

|  |  |
| --- | --- |
| **Use Case Title** | ***User Registration*** |
| **Pre-conditions** | User is not registered. |
| **Assumptions** | * Device is connected to the internet |
| **Success Criteria** | User can successfully register on the mobile app |
| **Failure Criteria** | User is unable to register on the app. |
| **User Role** | New App User |
| **Steps** | 1. User starts the app. user is presented with the welcome screen, and then the login screen (for instance). 2. User clicks on register new user and follows steps |
| **Post-conditions** | User details are stored within mobile app |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***User Details are stored*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | Details are stored on database |
| **Failure Criteria** | Details not stored on database |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User is presented with login screen. 3. User clicks on view details. 4. Users’ details are shown |
| **Post-conditions** | User details are stored within mobile app |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***User Details can be edited*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | Details are changed on database |
| **Failure Criteria** | Details are not changed on database |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User is presented with login screen. 3. User clicks on edit details. 4. User edits details and submits changes. 5. Users’ details are updated |
| **Post-conditions** | User details are stored within mobile app |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***User can send SMS to GP*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | Message is received by test device |
| **Failure Criteria** | Message is not received by test device |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User logs in 3. User selects the “SMS GP” button 4. User enters message and clicks send. 5. Message is received by test device |
| **Post-conditions** | GP/Test device phone number is stored on device/app |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***User customise theme*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | User can change theme from light to dark |
| **Failure Criteria** | User cannot change theme from light to dark |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User logs in 3. User selects settings button. 4. User selects dark theme option and clicks submit. 5. Application theme changes to dark |
| **Post-conditions** | Theme changes to dark |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***User customises background colour*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | User can change theme from light to dark |
| **Failure Criteria** | User cannot change theme from light to dark |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User logs in 3. User selects settings button. 4. User selects new colour for background. 5. Application theme changes colour |
| **Post-conditions** | Colour changes to selected colour |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

|  |  |
| --- | --- |
| **Use Case Title** | ***Check that all navigation buttons work*** |
| **Pre-conditions** | User has signed in |
| **Assumptions** | * Device is connected to the internet. * User is registered. * User is signed in |
| **Success Criteria** | Each button works as intended |
| **Failure Criteria** | Faults found in navigation buttons |
| **User Role** | Existing App User |
| **Steps** | 1. User starts the app. 2. User logs in 3. User selects settings button. 4. User selects view details button. 5. User selects edit details button |
| **Post-conditions** | Each button takes the user to the right page |
| **Test Status** | Pass or Fail |
| **Risks and constraints** | * Mobile device might not be connected to the internet. |

# Bibliography

eTutorials, 2020. *eTutorials.org.* [Online]   
Available at: http://etutorials.org/Mobile+devices/mobile+wireless+design/Part+Two+Building+Smart+Client+Applications/Chapter+7+Smart+Client+Overview/Smart+Client+Architecture/  
[Accessed 18 March 2021].

Microsoft, 2021. *dotnet.microsoft.com.* [Online]   
Available at: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin  
[Accessed 13 2 2021].

Vogel, L., 2017. *Vogella.* [Online]   
Available at: https://www.vogella.com/tutorials/AndroidArchitecture/article.html  
[Accessed 11 March 2021].