

|  |
| --- |
| BSHBIS2 / BSHC2 |
| Requirements Specification (RS) |
| “Dr. Connect” Mobile Application |

Matthew Byrne

x17138744 – BSHC2

Ben Carroll

x17501726 – BSHC2

Dylan Murphy

x17506166 – BSHBIS2

Piyush Sharma

x17342356 – BSHBIS2

Keith Mahony

x17521139 – BSHC2

Document Control

Revision History:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Scope of Activity** | **Prepared** | **Reviewed** | **Approved** |
| 17.02.19 | 1 | FINAL | Y | Y | Y |
| 15.02.19 | 2 | Update | Y | N | Y |

Distribution List:

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Version** |
| Dr.Connect Specification | Lisa Murphy | FINAL 17.02.19 |
| Dr.Connect Specification | Enda Stafford | FINAL 17.02.19 |
|  |  |  |
|  |  |  |
|  |  |  |

Related Documents:

|  |  |
| --- | --- |
| **Title** | **Comments** |
|  |  |
|  |  |

Table of Contents

1. Introduction

* 1. Purpose
  2. Project Scope

2. User Requirements Definition

3. Requirements Specification

3.1. Use Case System Diagram

4. Functional Requirements

4.1.1. Requirement 1 <Log In Registration>

4.1.2. Requirement 2 <Prescription Order>

4.1.3. Requirement 2 <Appointment System >

4.1.4. Requirement 2 <Notifier>

4.1.5. Requirement 2 <NFC Transfer>

4.2. Non-Functional Requirements

4.2.1. Performance / Response Time Requirement

4.2.2. Availability Requirement

4.2.3. Recover Requirement

4.2.4. Robustness Requirement

4.2.5. Security Requirement

4.2.6. Reliability Requirement

4.2.7. Maintainability Requirement

4.2.8. Portability Requirement

4.2.9. Extendibility Requirement

5. G.U.I.

6. System Architecture.

7. System Evolution.

8. Summary.

1. Introduction
   1. **Purpose**

The purpose of the document is to set out the requirements for the development of “DR.Connect” android app. “DR. Connect” will be a very beneficial and efficient app for patients and Doctors/GPs. It is made for the government health department and private hospital/GP corporations.

The intended customers of “DR.Connect” are patients with sickness and a consistent patient.

* 1. **Project Scope**

The scope of the project is to develop an app that makes a patient day-to-day life more convenient.

This app will prevent patients from carrying around a Files/Documents, it will also allow for the ordering Prescription/Booking appointments from home and will act as a notifier for appointments and handling of prescription.

The application also considers a reliable access to past prescription and other information.

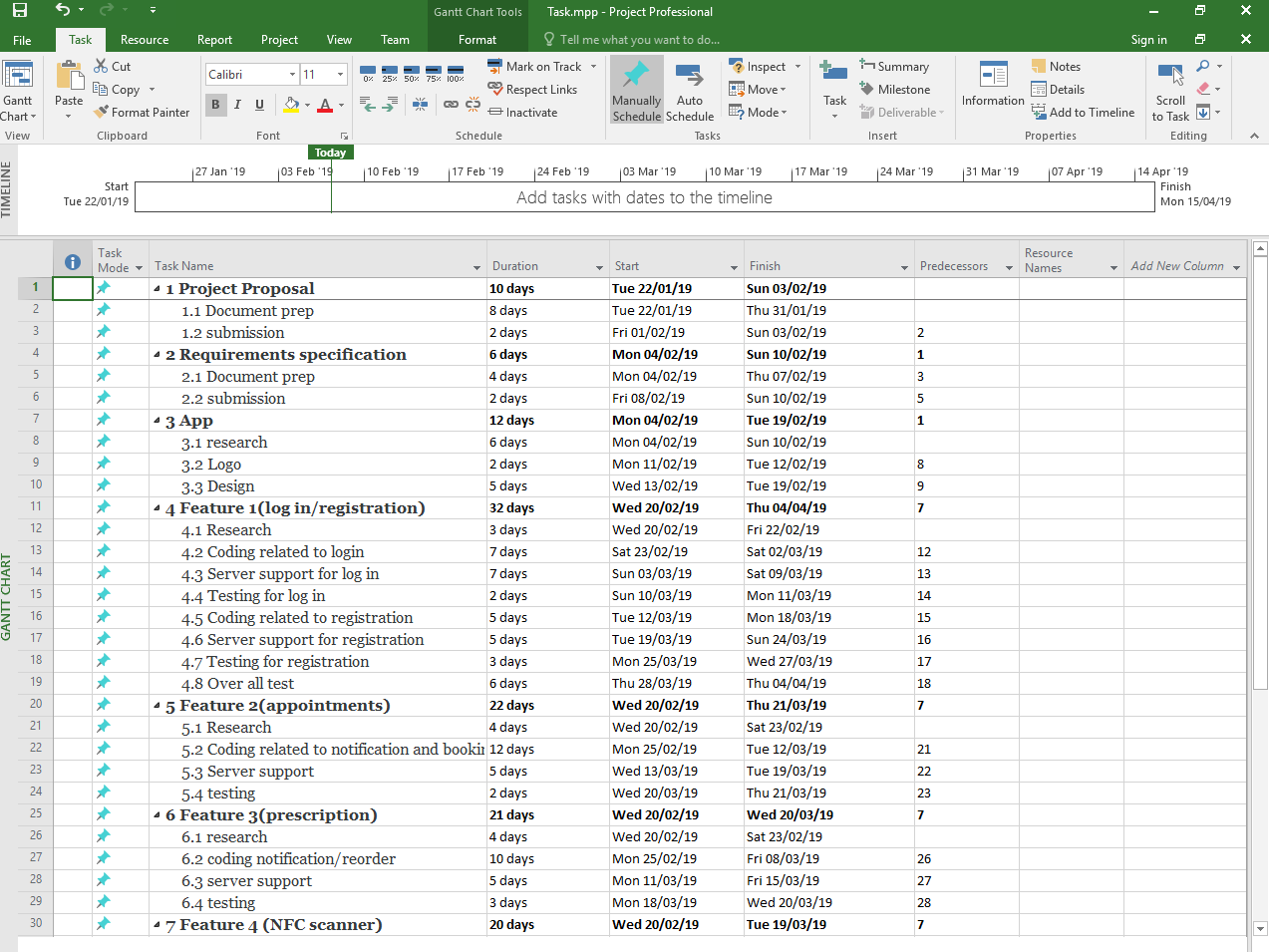
It will also provide other minor services for the Patient / Doctor/ GP’s.

The system will obtain:

* A login/registration system for both patients and doctors
* NFC transmission for easy access of data
* A prescription ordering system
* A notifier for appointments and prescription taking/finishing dates
* An appointment system that will allow the patient and doctors to book appointments with the doctor/patient of their choice.

To prevent scope creep and any project delays we created a schedule on Microsoft project, a Gantt chart to monitor and help project progression.

The purchase of an NFC kit was required to satisfy one of the main features. The use of Android studios as the app developer and firebase as the server are very important and helpful for the project.



1. User Requirements

The set of objectives / requirements for this project are to allow patient/doctors/GPs to access information about the patient. It will also provide efficient services such as scheduling appointments and a database for patient information. It will include the easy transmission of information using NFC, an easier method of booking appointments and ordering/tracking of prescription. Doctors will be allowed to change and update patient profiles without hassle of printing new documents and editing past logs. All together it will solve problems that patients face such as going to stores far away to get prescribed medication, carrying around reports and the tracking of patient health.

1. Requirements Specification

The Requirements Specification List will be broken down below and listed in **Ranked Order.**

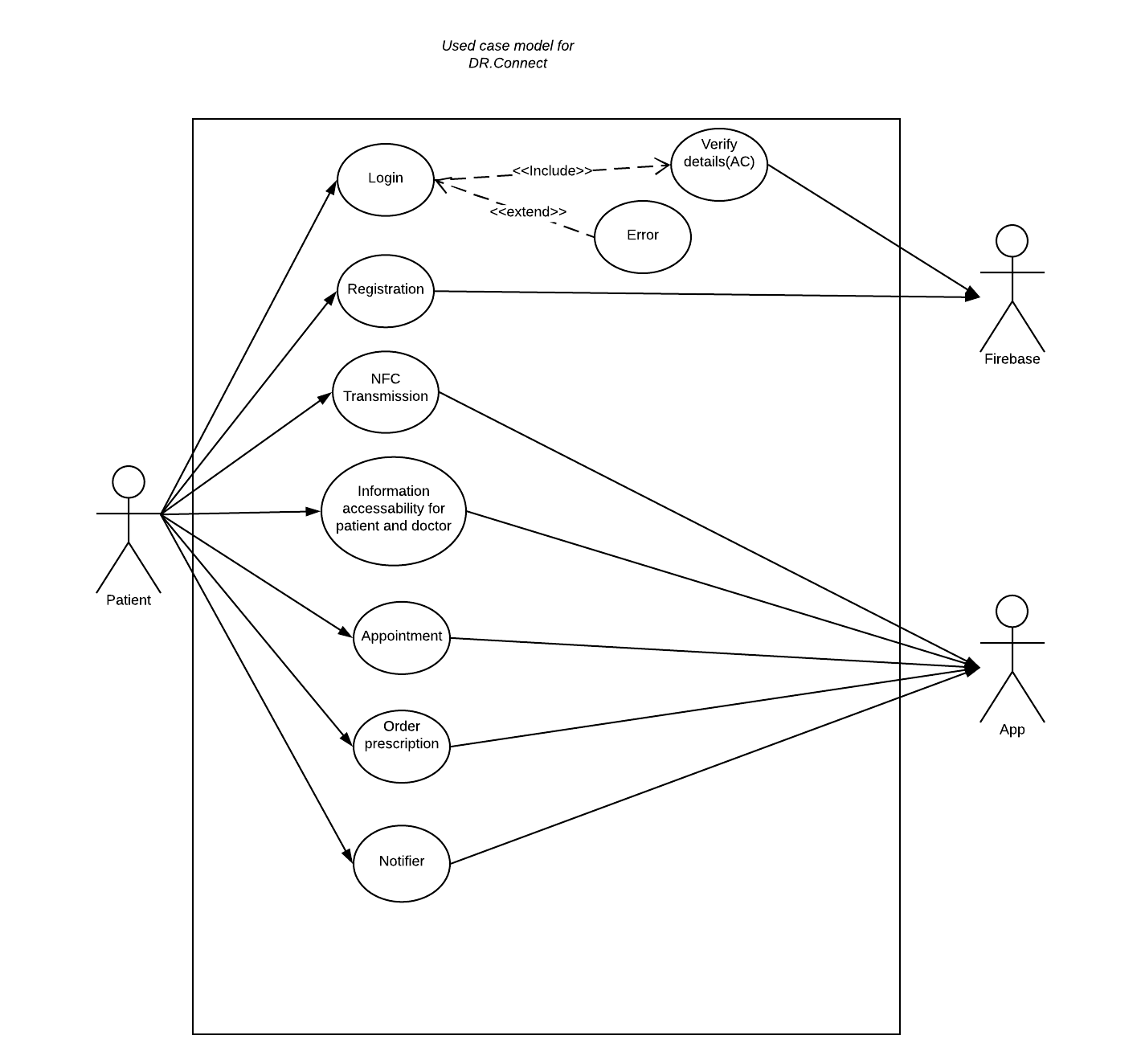
These features were discussed and talked about it within the team in the realm of possibility and simply put can it be feasible and important to the application.

We discussed many requirements that were vast and had a range of features to bring to the table, and in the end decided to lay it down to five features that can potentially built up and bring functionality once the application gets off the ground.

Below are these features with an overall Use Case Diagram for visual aspect.

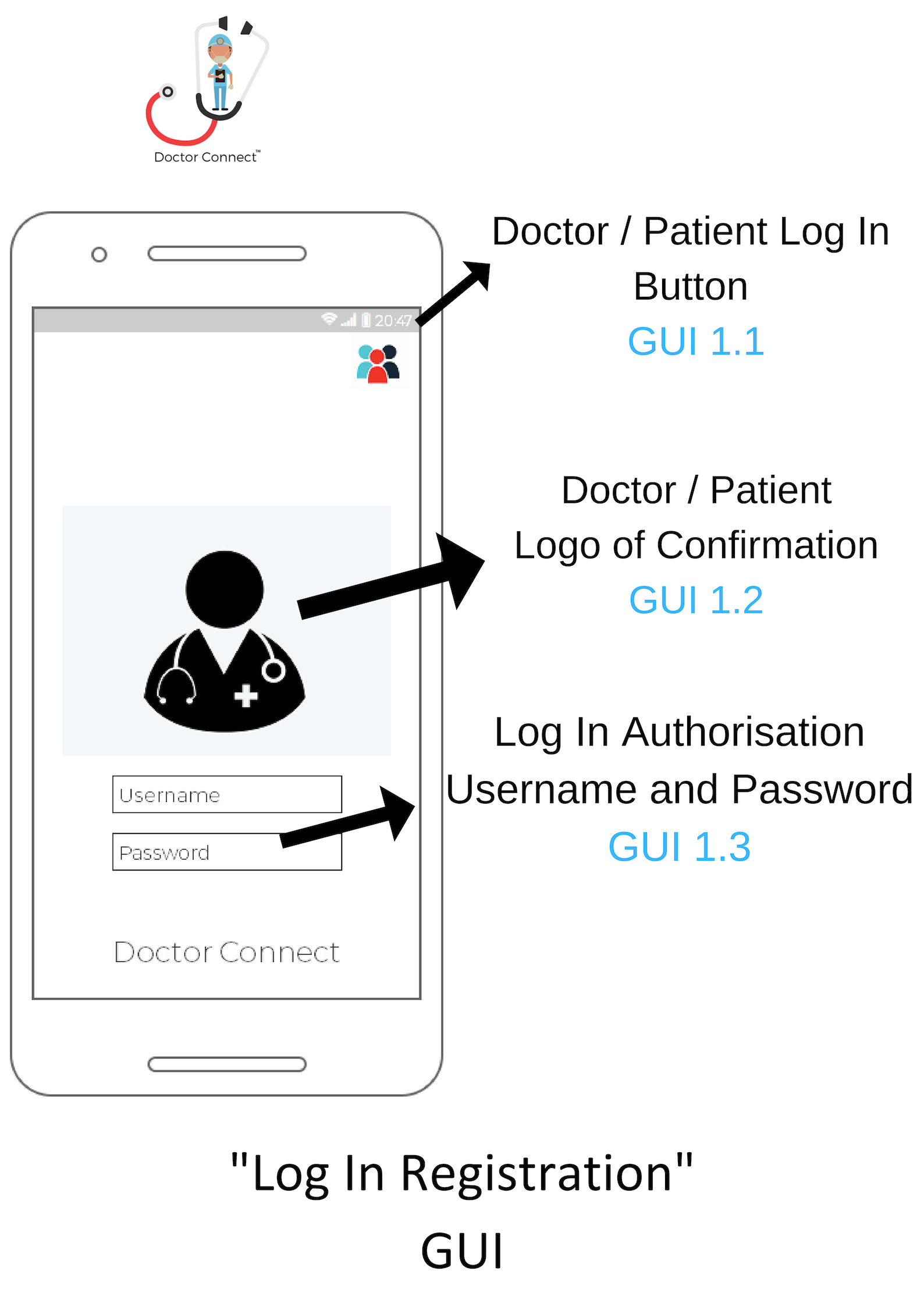


1. **Login / Authorization to the application from two ends User and Doctor. (including security)**
2. **Prescription Order System that can be access on both side that can view, add, delete and save.**
3. **Appointment System that notifies your doctor and lets you arrange face to face appointments.**
4. **A notify system that allows the user to know when appointment coming.**
5. **NFC technology that can pass on information to other users with confidentiality.**
   * 1. **Use Case Diagram**



1. Functional Requirements

4.1.1. Requirement 1 <Log In Registration>



* Description & Priority

This requirement will allow both patients and doctors to register for an account on Dr. Connect. It will also allow users with an existing account to log into the app. This is one of the highest priority requirements for Dr. Connect as we need to differentiate between doctors and patients as users for the difference in functions as well as for security reasons. It is also essential to the security of the app that we have a secure login page with unique usernames and strong passwords for every user, as the app deals with private information.

* Use Case

Log in / Registration

* Scope

The scope of this use case is to allow new users, both patients and doctors, to register for an account or log in if they have an existing account.

* Description

This use case describes the action of registering for an account or logging in to an existing account.

* Flow Description
* Precondition

The app is installed on the phone and running.

* Activation

This use case starts when a patient/doctor opens the app. (GUI 1.1)

* Main flow

1. The patient enters their username and password
2. The app checks these details through the database
3. The patient gains access to the main features of the app
4. The app displays a “doctor” button

* Alternate flow

A1: Doctor log in

1. The doctor presses the button located at the top right of the screen

 (GUI 1.1)

1. The doctor is brought to the doctor’s login page
2. The use case continues at position 2 of the main flow
3. The doctor gains access to the main features of the app
4. The app displays a “patients” button (GUI 1.2)

A2: Patient register

1. The patient clicks the registration button on the patient log in page
2. The patient enters a username and password for their account (GUI 1.3)
3. The app checks with the database that the username isn’t taken
4. The app checks with the database that the chosen password meets the password requirements of the app
5. The app enters the users details to the database thus registering them on the app
6. The use case continues at position 3 of the main flow

A3: Doctor register

1. The use case continues at position 2 of the A1 flow
2. The doctor clicks the registration button on the doctor log in page
3. The doctor enters a username and password for their account (GUI 1.3)
4. The app checks with the database that the username isn’t taken
5. The app checks with the database that the chosen password meets the password requirements of the app
6. The app enters the users details to the database thus registering them on the app
7. The use case continues at position 2 of the A1 flow

* Exceptional flow

E1: Patient log in fail

1. The use case continues at position 2 of the main flow
2. The app displays an error message for the patient as the details provided are invalid
3. The user is unable to log in
4. The use case fails

E2: Doctor log in fails

1. The use case continues at position 2 of the main flow
2. The app displays an error message for the doctor as the details provided are invalid
3. The user is unable to log in
4. The use case fails

E3: Patient is unable to register

1. The use case continues at position 2 of the A2 flow
2. The user aborts the registration process
3. The use case fails

E4: Doctor is unable to register

1. The use case continues at position 3 of the A3 flow
2. The user aborts the registration process
3. The use case fails

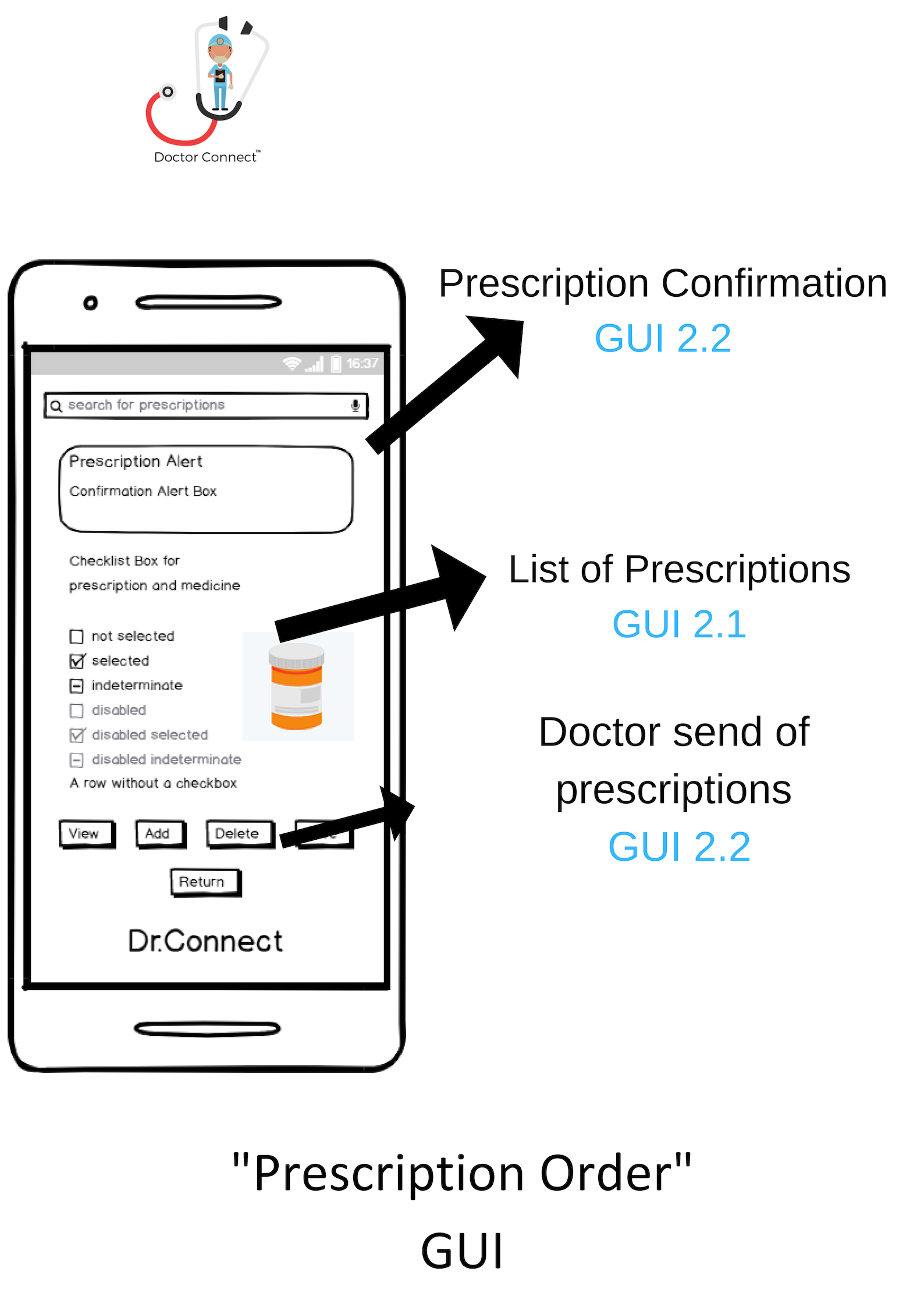
* Termination

The app displays the home screen of the app with the prescriptions, appointments, notifications and doctor/patients’ areas displayed.

* Post condition

The user is allowed access to the other areas of the apps: prescriptions, appointments, notifications and doctor/patients.

4.1.2. Requirement 2 <Prescription Order>



* Prescription Order Aim

For our prescription ordering requirement, we have infused our NFC tags with being able to store prescriptions for each individual patient and being able to be created by the doctor while being read by a pharmacy.

This is a lot easier for pharmacists and doctors as they both don't have to work off of handwritten notes which are prone to misreading the prescription and also not allowing a patient to keep track of what past prescriptions, he/she has had.

* How It Works

First a patient meets up with their doctor and receives a prescription. The doctor simply does this on the app or the website which he can then link to the patients account. Only a registered doctor can set a prescription which removes an issue with current doctors notes in which some patients forge their own allowing to receive medicine needed with a prescription. The patient then has their prescription stored on their app which allows them to go to a pharmacy and receive their prescription. The pharmacist then scans their NFC tag which shows the prescription and allows them to give the prescription to the patient while letting the server know they received it.

* Use case

Prescriptions

* Scope

The scope of this use case is for a pharmacy to view a patient’s prescriptions. (GUI 2.1)

* Description

This use case is how the pharmacy can read and state that this prescription has been used.

* Flow Description
* Precondition

The pharmacy is logged in.

* Activation

This use case starts when a pharmacy scans the NFC tag.

* Main flow

1. The pharmacy scans the NFC device.
2. The pharmacy reads the prescription. <See A1.>
3. The pharmacy gives the patient its his/her prescription.  (GUI 2.1)
4. The pharmacy states that the prescription has been given.

* Alternate flow

(A1 No prescription)

1. Error message appears on app screen.

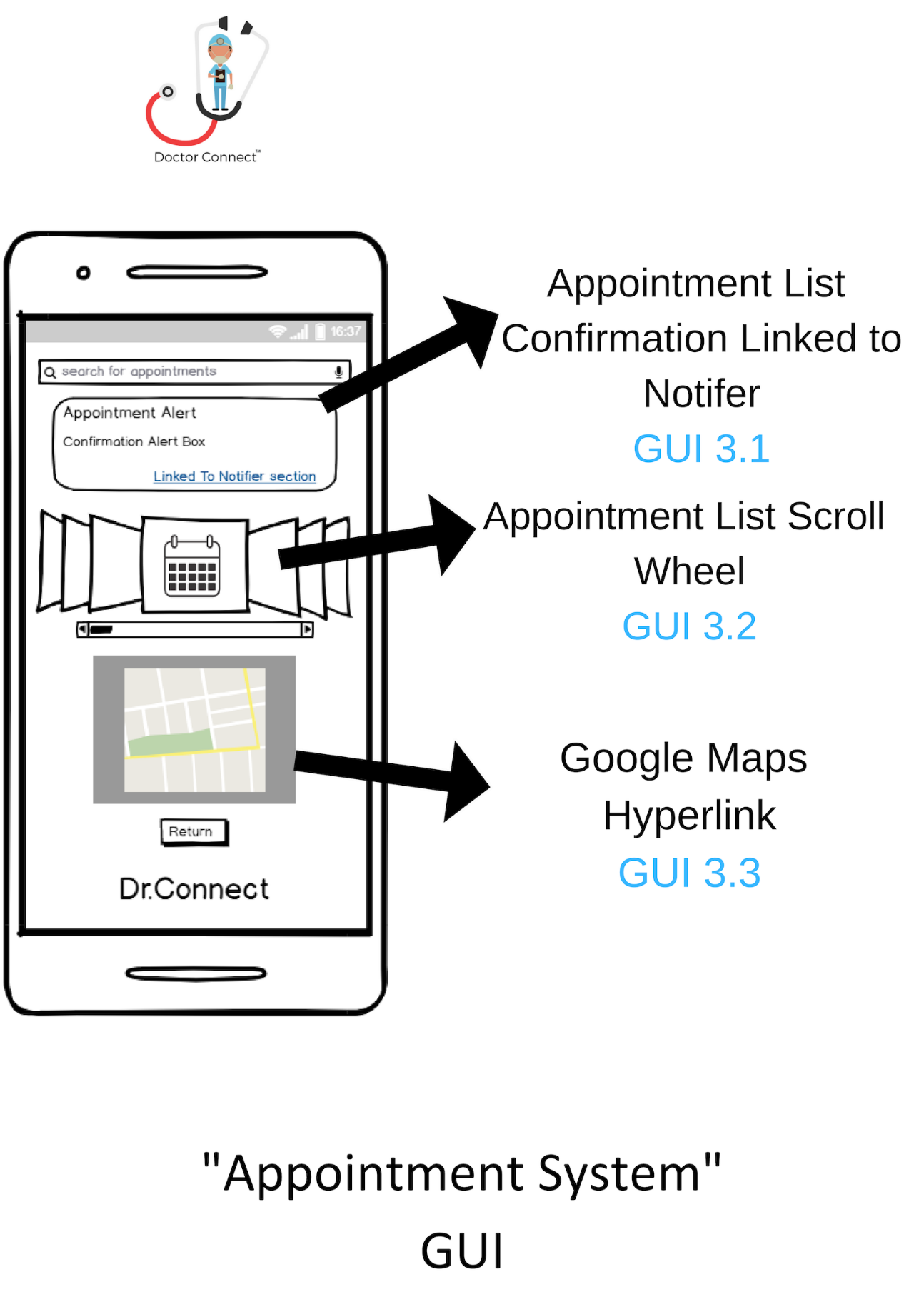
* Exceptional flow
* Termination

The patient receives his/her prescription. (GUI 2.2)

* Post condition

The system goes to a wait state.

4.1.3. Requirement 3 <Appointment System>



# Use case

Appointment System

# Scope

The scope of this use case is the implementation of an Appointment System section built into the Dr.Connect Application

# Actors

User / Server / Dr.Connect Application

# Description

This use case breaks down the Appointment system process and how communication can be made between doctor and patient via the application.

# Flow Description

## Precondition

App is installed on phone / Notifications are enabled / User has provided us with permissions

## Activation

This use case starts when a new the user or doctor enters a date / time / patient / doctor inside the appointments section based within the Dr.Connect Application.

Main flow

1. When patient clicks into his doctor information page / when doctor clicks into his patients list.
2. Patient clicks scroll wheel bookings based in Appointments / Doctor click scroll wheel bookings based in Appointments. (GUI 3.2)
3. App registers Users response and stores it in database with date and time of appointment when information entered on BOTH ends of sever (notifier confirmation).
4. App then confirms to user of appointment on screen with date and time plus Google Map hyperlink for location of appointment. (GUI 3.3)
5. App goes into a wait state until notifier is ready to also update with appointment info. (GUI 3.1)

## Alternate flow

1. User clicks ‘cancel’
   1. App registers Users response and stores it in database and notifier initiates showing confirmation of cancellation on BOTH sides of sever patient and doctor.
   2. App goes into a wait state, ignoring all potential notifications until appointment can be modified, deleted, or created.

## Exceptional flow

E1 User switches off phone/app while appointment booking is being made

1. App resets, assumes User has not made an appointment sent to the database.
2. User must go through the process all over again to make an appointment booking.

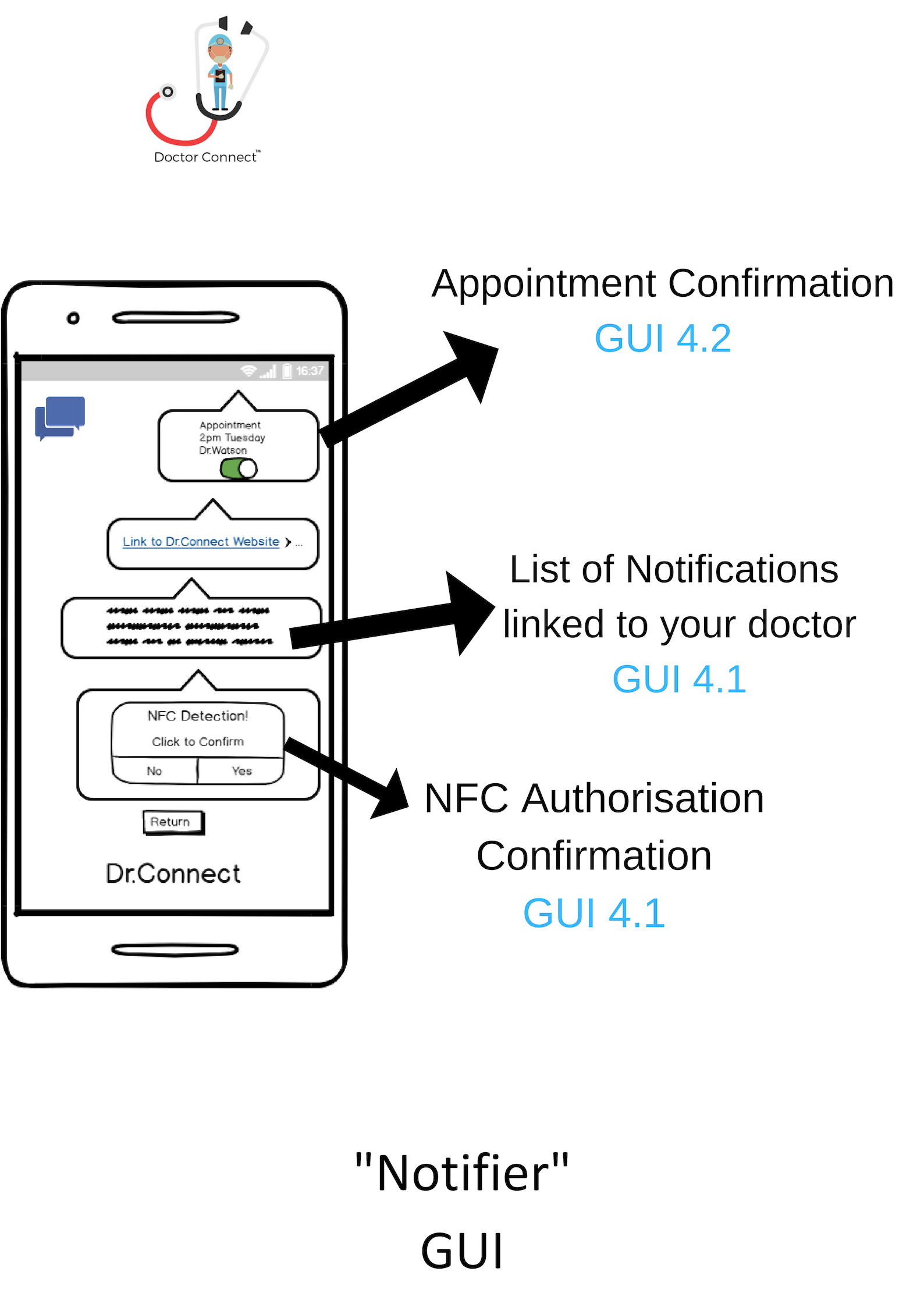
## Termination

Returning back to the main feature page of Dr.Connect Application and its features.

## Post condition

Appointment Section goes into Wait State until accessed again.

# 4.1.4. Requirement 4 <Notifier>



# Use case

Push Notifications

# Scope

The scope of this use case is the implementation of Notifications (GUI 4.2)

# Actors

User

NFC equipped Phone

Server

Dr.Connect Application

# Description

This use case describes the ways Notifications are pushed to a User’s device

# Flow Description

## Precondition

App is installed on phone

Phone is capable of reading NFC (GUI 4.1)

Notifications are enabled

User has provided us with permissions

## Activation

This use case starts when a new user is asked to give our app permission to push notifications

## Main flow

1. Upon account creation, App requests permission to send notifications.
2. User clicks ‘accept’. (See A1/E1)
3. App registers Users response and stores it in database.
4. App goes into a wait state until potential notification is ready.

## Alternate flow

1. User clicks ‘reject’
   1. App registers Users response and stores it in database.
   2. App goes into a wait state, ignoring all potential notifications.

## Exceptional flow

E1 User switches off phone/app while request is being sent

1. App resets, assumes User has not yet been asked.

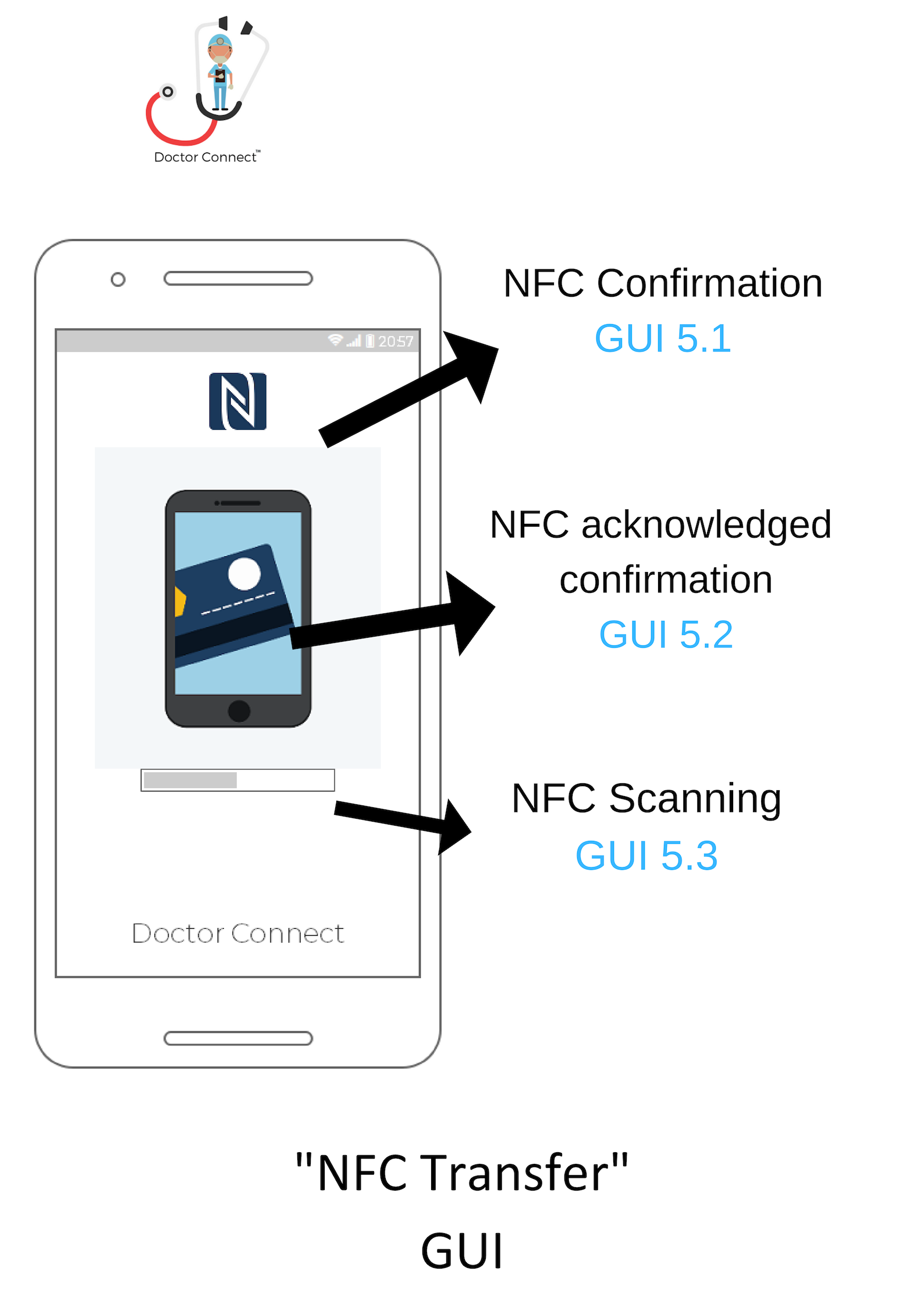
## Termination

Notification setting has been chosen and applied.

## Post condition

App goes into wait state.

4.1.6. Requirement 5 <NFC Transfer>



# Use Case

NFC Technology

# Scope

The scope of this use case is the Applications use of NFC

# Actors

Patient

Doctor

NFC scanning device (phone, etc.)

# Description

This use case describes the ways in which NFC is implemented in the App.

# Flow Description

## Precondition

The App is installed on user’s NFC capable phone

Pharmacy is capable of reading NFC equipped ID card (GUI 5.1)

## Activation

This use case starts any time NFC is used to authorise access to private information/the app

## Main flow

1. Patient/Emergency Services open Doctor Connect App on phone/device (See A1)
2. ‘Dr.Connect’ ID Card is held next to phone until App acknowledges connection (GUI 5.2)
3. App authorises ID of card and requests server to search for that account
4. Server finds ID number in database and authorises access to App (See E1)
5. App opens account specified by server on main profile page

## Alternate flow

**A1** Pharmacy scans ID card

1. Pharmacist scans ID Card
2. Scanning Device requests ‘Dr.Connect’ server to search for record of ID number (GUI 5.3)
3. Server locates ID number and displays prescription code(s) associated with it to Device (See E1)

**A2** Doctor/GP uses Dr.Connect App

1. Dr.Connect doctor ID Card is held next to device until App acknowledges connection
2. App authorises ID of card and requests server to search for that account
3. Server finds ID number in database and authorises access to App (See E1)
4. App opens account specified by server on main profile page

## Exceptional flow

**E1** No ID number match found in database

1. Server informs App that no record was found
2. Notification pushed to user: “Card was not recognised”

## Termination

1. The use case terminates when App is no longer using NFC

## Post condition

The App goes into a wait mode.

1. Non-Functional Requirements

### Performance / Response time requirement

On concentration of the topic of performance and time management the first thing that comes to mind is that the team want the application to be “fast”. This requirement to an extent is true but we also wish to give the user a more defined balance with reliability. More importantly we do not want to have the application crash due to processes not keeping up with the user running in the background.

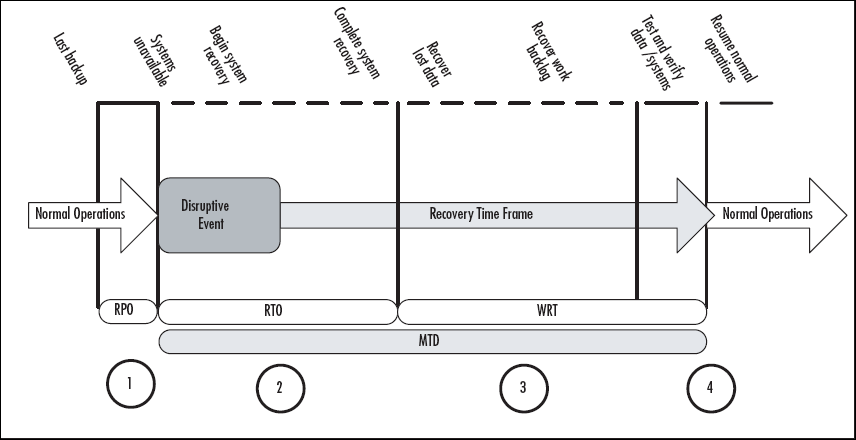
### Availability Requirement

This section of Non-Functional Requirement is quite important. We also must come to the fact that is several factors that will give the desired availability performance.

This will include what mobile devices and what Android firmware we aim to demographic the application that gives a bigger catch, and what hardware of mobile technology can sit comfortably run without running into issues. The logical choice would to pick an Android firmware that is the most dominantly compatible with most users while also can use NFC function as older Android firmware could not provide this function. During the development stage and along with the power of Android Studio, we can choose to upgrade more to other forms of Android firmware on the market while not sacrificing older models. This makes way for more people catch across the board who wish to download the application.

### Recover Requirement

On the topic of Recover Requirements, the first thing that comes to our mind is that if our application or the mobile device in use manages to die or run out of power or for some other reason, can the data that was accessible in our servers be safe and secure and not have third party access. Or that it won’t be destroyed and kept safe in our database. Below is an image that connects Recover Requirements that breaks down the **Recovery Point**, **Recovery Time** and **Work Recovery** to get systems back up and running to normal operations. There is an ability to include security measures for this in Android Studio.



### Robustness Requirement

With Robustness Requirements the pure definition of this is that if the users or whoever uses the application is unfamiliar with the layout or structure will happen to put a number of mistakes or accidental errors for data or information. We can respond to this by making sure we have precautions or messages in our application for instruction for the user or a “how to guide!”. Along with making sure the user can not cause the application to break the app by putting in unfamiliar information.

This can be tackled with the coding element of the project, along giving an informative pop up screen to show which features is which in the application allowing the user a great introduction in the world of DR. Connect and what it can bring to the table.

5.1.5. Security Requirement

Security Requirements is of course the element of security and GDPR. Safety on either end whether it be user or doctor, needs to be safe and secure and that it doesn’t facilitate interruption from other third-party elements or through fault of ourselves.

Thankfully with the java development end of Android Studio and the Firebase server, we will introduce we bring security elements and cryptography architecture into the equation, that allows a system of unique security providing a safety blanket to anyone who wishes to divulge their private data.

This is of massive priority to the team especially when we will be accessing the vast features of NFC technology, and how that information can be transferred safely while also not violating GDPR law.

* + 1. Reliability Requirement

Having the application work. Well. This then also leads elements that give the user full flow and freedom to go in and out of the application without any slowdown or issues on feedback.

It’s finding that balance of reliability and speed and its connection with the backend information connected, that will provide a technical challenge for the team. We look forward to seeing how can implement these tasks. This of course must be primarily balanced with their first requirement, going hand in and hand with performance and response.

* + 1. Maintainability Requirement

Upon release of the software and when we come to the early testing stages of deployment, we will be bound to come across major issues or problems that we will inevitably have to fix.

With maintainability it’s fixing these problems and keeping consistent updates along with a balance of scale. Scale is how big or vast the project can be. We aim to keep a big development in the works but one we can fundamentally control and keep on top of which ties into maintainability and the general upkeep of the application. As a team we have the numbers to keep on top of a live system, and to give the user experience we intended people to receive.

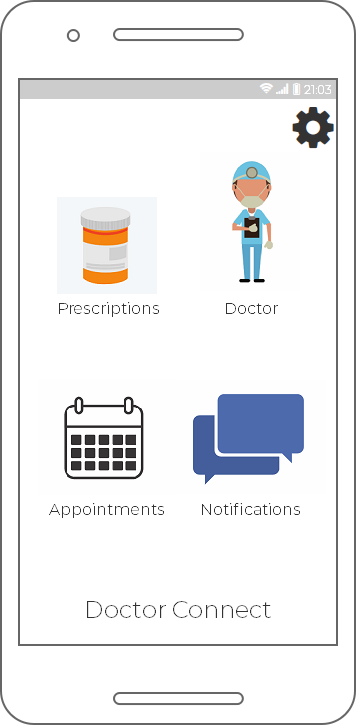
* + 1. Portability and Extendibility Requirement

With the development of Android Studios in the works, the idea of portability is already firmly hand in hand with mobile device technology. That of course can change with a various amount of screen sizes, Android firmware, and other factors that could affect performance, reliability and response times. The application will also need its own website to provide detailed information on its creation and features. This will not include a desktop version of our application. We wish to concentrate first and perfect one version of the program we are happy with before we branch out to other options.

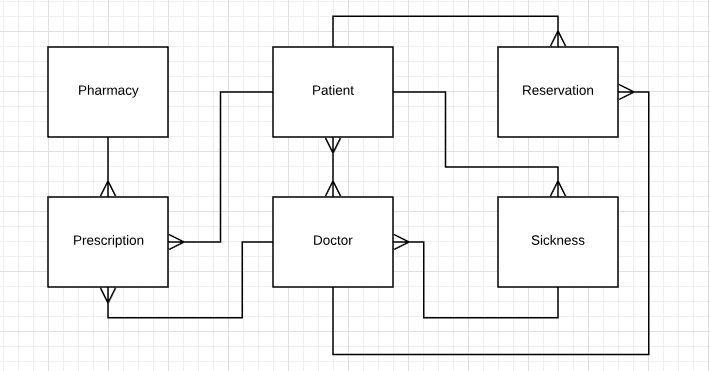
With Android Studios we are of course limited to the world of APK’s and certain O.S software, so in the future as they claim a massive dominance in the market, we wish to include IOS compatible into the frame. An idea or element of this can be introduced with the NFC development kit we have which supports IOS tagging. More information on this will be additionally noted in the “System Evolution” section closer to the end of this project.

5. G.U.I





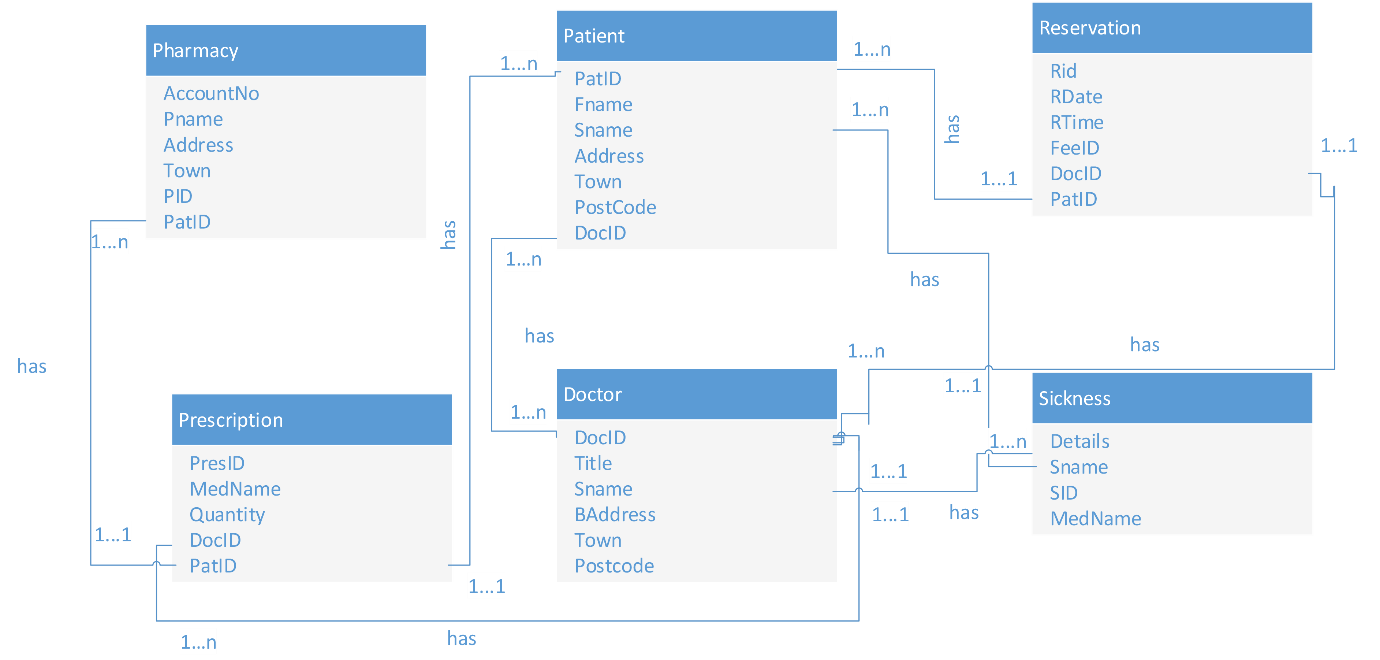
1. System Architecture



We chose this system architecture as we wanted something that has a clear, detailed view of our system as a whole.

With the patient class we included all details that would encompass everything the patient’s doctor and pharmacy would need to care for them. It also allows the doctor and pharmacy to ensure the right address and by extension medication is assigned to the correct patients.

Regarding the reservation class, all the attributes found within will aid us in implementing the appointments feature in Dr.Connect. It will allow both the patients themselves and doctors to grab the date, time, doctor and fee associated with a patient’s appointment. The prescription class, however, allows for a clear record of prescriptions that can be accessed by the pharmacy, or doctor whenever necessary.

Overall, we wanted a system that was clearly defined but allows for scalability in the future.

7. System Evolution

As high as our aspirations are for the initial version of our App, our plans and ideas don’t stop there. We believe the technology we are producing has potential beyond providing simple conveniences.

Expanded Feature Set

We had a shopping list of features and technologies that we wanted to implement in our plans for V1 of Dr.Connect, but the realities of time, resources and the limitations that those two place on us got in our way. We had plans for a messaging service, linking patients and doctors together even closer. Patients could ask their doctor specific questions related to their treatment, for example. Building on from that, we were interested in a phone call system: A patient could click a phone symbol under their doctors’ name/profile and be redirected to that Doctors clinic/secretary. We also had ideas centred around a ‘panic button’ of sorts, allowing the patient or emergency services to alert a doctor to their patients potentially dangerous condition.

Third Party Support



Medical Cards, GP Visit Cards, or even documents not linked directly to medical needs. We believe our App could incorporate these forms of identification, further increasing our convenience/ease of use, and potentially making our application more useful to more people. We are currently limited by the reach of NFC and its range of compatibility. Our usefulness is directly related to our potential Users current choice of phone, for example. iOS doesn’t support the NFC necessary for our App to function, which cuts out a huge portion of our target market. This is something we would need to address down the line, whether that’s via proprietary hardware or in this case collaboration with third parties.

Spin off Title

Our Application is dedicated to medical services, but we have many different ideas for how else we could use our infrastructure. The ability to use a card to scan and the unlock/authenticate/ identify information or data held on a secure server from anywhere in the world has huge potential if used in a useful or exciting new way. Is there a Mr and Mrs. Connect version of this App somewhere in our future? Or a way to store passwords or even unlock devices without the need to remember the hundreds of different variations of passwords tied to each of our online accounts.

8. Summary.

Overall this is the continued direction that is on course for Dr.Connect application.

Whether in the future the dynamics or requirements needed may change is something that the team will have considered. This will be an important learning curve for the project.

We aim to deliver as many of these requirements as we can to the application, giving users who interact a great experience. The architecture and understanding what are working with and what is feasible with team discussions, is key to what people can view the application in twelve week’s time.

So now the main key with all the specifications we have wrote down in this document is to fundamentally deliver these aims and technically get the work done. This is massive task with a big workload, so the challenge is there for everyone in the team. Wish us luck!

­­­­­­­­­­­­­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_