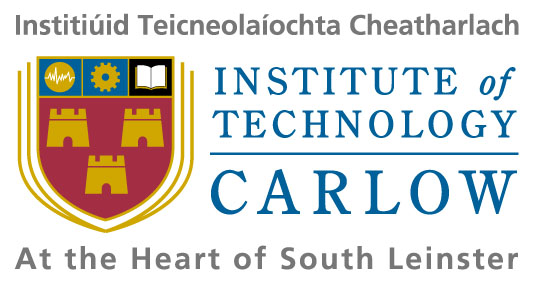
2020







Holly Gunning

C014010

Supervisor: DR. Hisain Elshaafi

# Abstract

The purpose of this project was to create a progressive, single page web application to aid small to medium size clinics and their patients with a more online based experience, so as to reduce the need for as many physical consultations, enabling a more direct approach to prognosis and thereby limiting the spread of contagions amongst patients.

The application allows patients to book appointments online, keep track of their personal medical record, request prescriptions digitally rather than in person as well as record any medication they are taking. Doctors can easily view a database of all registered patients and make amendments to their medical records as well as review prescription requests and either grant or deny them.

Table of Contents

[Abstract 1](#_Toc50109699)

[Table of Figures 5](#_Toc50109700)

[1. Introduction 7](#_Toc50109701)

[2. Additional Research Undertaken 8](#_Toc50109702)

[2.1. Backend Service Overlooked 8](#_Toc50109703)

[2.1.1. Firebase 8](#_Toc50109704)

[2.1.2. Authentication 8](#_Toc50109705)

[2.1.3. Hosting 9](#_Toc50109706)

[2.1.4. Cloud Functions 9](#_Toc50109707)

[2.1.5. Cloud Firestore 9](#_Toc50109708)

[2.1.6. Cloud Storage 9](#_Toc50109709)

[2.2. Introducing a Framework 9](#_Toc50109710)

[2.2.1. Vue JS 10](#_Toc50109711)

[2.2.2. Benefits of VueJS 10](#_Toc50109712)

[2.2.3. Vue Router 11](#_Toc50109713)

[2.2.4. Vuex 11](#_Toc50109714)

[2.2.5 Vuetify 11](#_Toc50109715)

[2.3. Additional Research Conclusion 12](#_Toc50109716)

[3. Project Description 13](#_Toc50109717)

[3.1. Application Screenshots & Explained 14](#_Toc50109718)

[3.1.1. Login Screen 14](#_Toc50109719)

[3.1.2. Registration Screen 15](#_Toc50109720)

[3.1.3. Dashboard Screen (Patient) 16](#_Toc50109721)

[3.1.4. Navigation Drawer Screen (Patient) 17](#_Toc50109722)

[3.1.5. Profile Screen (Patient) 18](#_Toc50109723)

[3.1.6. Medical Record Screen (Patient & Doctor) 21](#_Toc50109724)

[3.1.7. Appointments Screen (Patient) 26](#_Toc50109725)

[3.1.8. Chat Screen (Patient) 30](#_Toc50109726)

[3.1.9. Patient Database Screen (Doctor) 31](#_Toc50109727)

[3.1.10. Navigation Drawer Screen (Doctor) 32](#_Toc50109728)

[3.1.11. Appointments Screen (Doctor) 33](#_Toc50109729)

[3.1.12. Patient Chat Screen (Doctor) 34](#_Toc50109730)

[3.1.13. Admin Screen 35](#_Toc50109731)

[3.1.14. Track Medication Screen (Patient) 36](#_Toc50109732)

[3.1.15. Prescriptions Screen (Patient) 38](#_Toc50109733)

[3.1.16. Prescription Requests Screen (Doctor) 40](#_Toc50109734)

[Adherence to Specification 43](#_Toc50109735)

[General Specification 43](#_Toc50109736)

[Register, Login & Forgot Password 43](#_Toc50109737)

[Patient Specification 43](#_Toc50109738)

[User Profile 43](#_Toc50109739)

[Medical Record 43](#_Toc50109740)

[Medication Tracking 43](#_Toc50109741)

[Appointments 43](#_Toc50109742)

[Prescriptions 44](#_Toc50109743)

[Message Functionality 44](#_Toc50109744)

[Doctor Specification 44](#_Toc50109745)

[Appointment Schedule 44](#_Toc50109746)

[Prescriptions 44](#_Toc50109747)

[Message Functionality 44](#_Toc50109748)

[Deviation from Specification 45](#_Toc50109749)

[General Specification 45](#_Toc50109750)

[Register, Login & Forgot Password 45](#_Toc50109751)

[Patient & Doctor Specification 45](#_Toc50109752)

[Medical Record & Patient Database 45](#_Toc50109753)

[Message Functionality - Video Call Consultations 45](#_Toc50109754)

[Adherence to & Deviation from Design 46](#_Toc50109755)

[Issues Encountered 47](#_Toc50109756)

[Covid-19 47](#_Toc50109757)

[Internet Issues 47](#_Toc50109758)

[Insufficient Research 47](#_Toc50109759)

[Learning Outcomes 48](#_Toc50109760)

[Technical 48](#_Toc50109761)

[IDE – Visual Studio Code 48](#_Toc50109762)

[VueJS Framework & Vuetify 48](#_Toc50109763)

[Firebase 49](#_Toc50109764)

[Personal 49](#_Toc50109765)

[Time Management 49](#_Toc50109766)

[Coding Skills 49](#_Toc50109767)

[Reflections 50](#_Toc50109768)

[Time Management 50](#_Toc50109769)

[Research 50](#_Toc50109770)

[Coding 50](#_Toc50109771)

[Future Development 51](#_Toc50109772)

[Video Call Consultation 51](#_Toc50109773)

[Refactor the User Role System 51](#_Toc50109774)

[Expand Upon Medical Record Functionality 52](#_Toc50109775)

[Add in Notification Functionality 52](#_Toc50109776)

[Acknowledgments 53](#_Toc50109777)

[Plagiarism Declaration 54](#_Toc50109778)

[54](#_Toc50109779)

[Bibliography 55](#_Toc50109780)

[References 55](#_Toc50109781)

# Table of Figures

[Figure 1: Login screen on laptop 15](#_Toc50105158)

[Figure 2: Login screen on mobile 15](#_Toc50105159)

[Figure 3: Registration screen on laptop 16](#_Toc50105160)

[Figure 4: Dashboard Screen 17](#_Toc50105161)

[Figure 5: Contact Us Form 17](#_Toc50105162)

[Figure 6: Navigation Drawer Screen 18](#_Toc50105163)

[Figure 7: My Profile Screen 19](#_Toc50105164)

[Figure 8: Edit Profile Card Replacing Profile Card 19](#_Toc50105165)

[Figure 9: Appointment Dialog on Calendar 20](#_Toc50105166)

[Figure 10: Time Schedule Displayed on Calendar of User Screen 20](#_Toc50105167)

[Figure 11: Dark Mode Toggle 21](#_Toc50105168)

[Figure 12: My Profile on Mobile 21](#_Toc50105169)

[Figure 13: Medical Record Screen 22](#_Toc50105170)

[Figure 14: Allergies Tab on Medical Record Screen 23](#_Toc50105171)

[Figure 15: Adding an Allergy on Medical Record 23](#_Toc50105172)

[Figure 16: Allergy Type Drop Down Figure 17: Severity Drop Down 23](#_Toc50105173)

[Figure 18: Conditions Tab on Medical Record Screen 24](#_Toc50105174)

[Figure 19: Adding a Condition on Medical Record 24](#_Toc50105175)

[Figure 20: Search Condition Name Field Figure 21: Select Condition Date Date-Picker 25](#_Toc50105176)

[Figure 22: Immunisation Tab on Medical Record Screen 25](#_Toc50105177)

[Figure 23: Adding an Immunisation on Medical Record 26](#_Toc50105178)

[Figure 24: Search Immunisation Field Figure 25: Immunisation Reaction Field 26](#_Toc50105179)

[Figure 26: Appointments Screen (View Appointments Screen) 27](#_Toc50105180)

[Figure 27: Initial Book Appointment Form 27](#_Toc50105181)

[Figure 28: Book Appointment - Personal Details 28](#_Toc50105182)

[Figure 29: Select Appointment Type Drop-down 28](#_Toc50105183)

[Figure 30: Select Doctor Dropdown 28](#_Toc50105184)

[Figure 31: Available Appointment Times Populated to Booking Form 29](#_Toc50105185)

[Figure 32: View & Book Appointment on Mobile 30](#_Toc50105186)

[Figure 33: Chat Screen Pre-Appointment 31](#_Toc50105187)

[Figure 34: Patient & Doctor Conversing 31](#_Toc50105188)

[Figure 35: Patient Database Screen 32](#_Toc50105189)

[Figure 36: Data Table Showing Reduced Number of Patients 32](#_Toc50105190)

[Figure 37: Navigation Drawer Screen (Doctor) 33](#_Toc50105191)

[Figure 38: Navigation Drawer on Mobile 33](#_Toc50105192)

[Figure 39: Appointments Screen Doctor 34](#_Toc50105193)

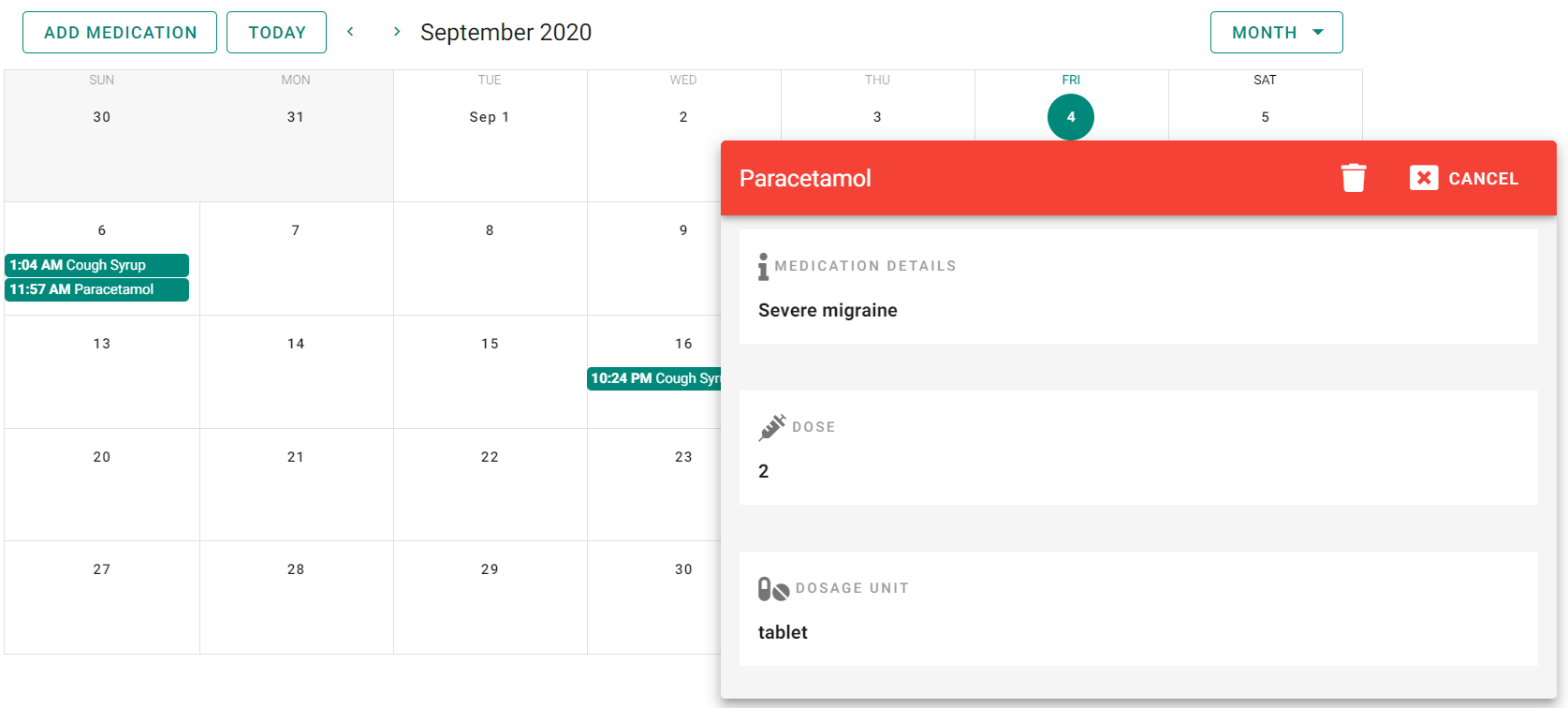
[Figure 40: Appointment Day Schedule on Mobile 34](#_Toc50105194)

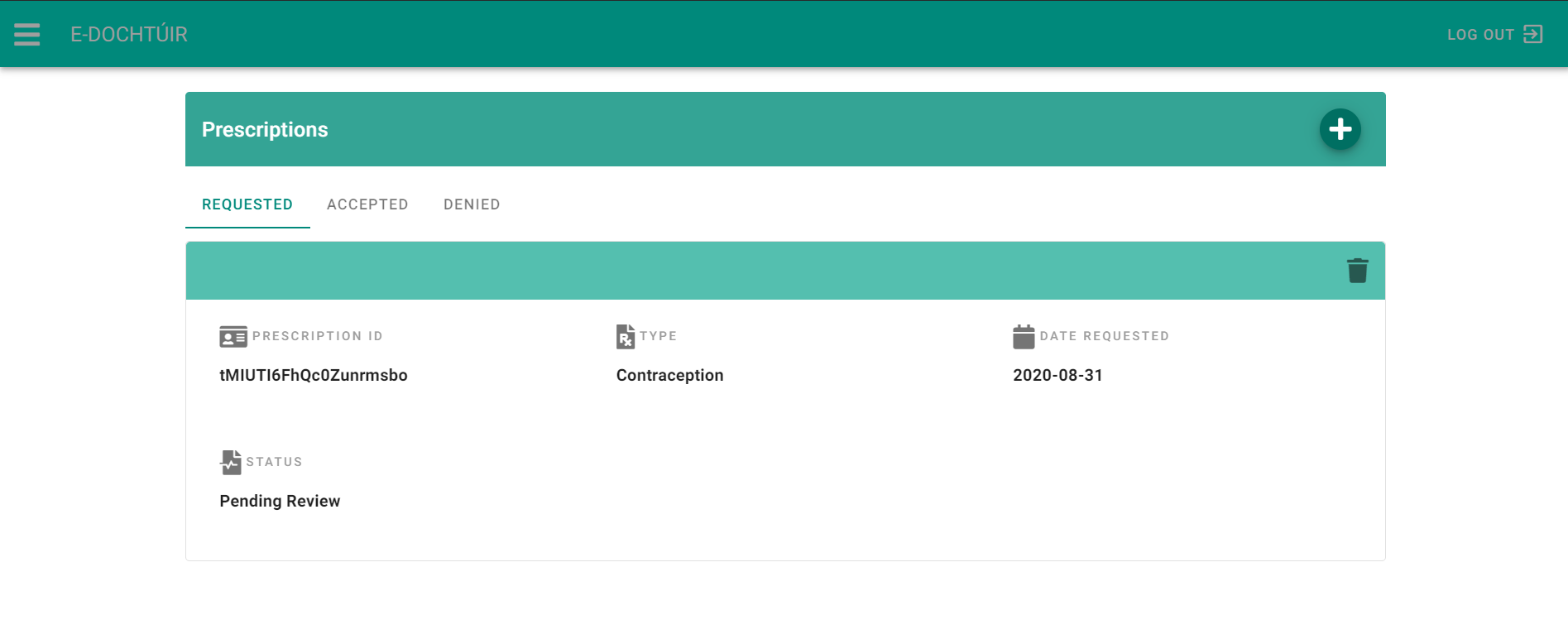
[Figure 41: Patient Chat Screen on Mobile Figure 42: Chat Created by Selecting Patient 35](#_Toc50105195)

[Figure 43: Admin Screen 36](#_Toc50105196)

[Figure 44: Track Medication Screen 37](#_Toc50105197)

[Figure 45: Add Medication Dialog 37](#_Toc50105198)

[Figure 46: Medication Information Menu 38](#_Toc50105199)

[Figure 47: Prescription Screen for Patient 39](#_Toc50105200)

[Figure 48: Prescription Request Form Pre-Treatment Selection 39](#_Toc50105201)

[Figure 49: Contraceptive Pill & Patch Request Form 40](#_Toc50105202)

[Figure 50: Prescription Screen for Doctor 41](#_Toc50105203)

[Figure 51: Example of Prescription Form 41](#_Toc50105204)

[Figure 52: Upload Prescription on Accepted 42](#_Toc50105205)

[Figure 53: Prescription Denied 42](#_Toc50105206)

# 1. Introduction

The following final report is intended to document the development process of the E-Dochtúir online healthcare application. E-Dochtúir is an application for small to medium sized clinics looking to both avail of and provide a more online experience for both their doctors and the clinics patients. From research undertaken in the first document it was discovered that while there are pre-existing online healthcare applications, there are fewer that are based in Ireland, and none that cater specifically to people with a medical card. This was the main inspiration behind the creation of this application.

This document describes the additional research that was undertaken to fully realise the scope of the applications development. It documents both the backend service and framework that were eventually utilised.

This document then describes the project as a whole and references every screen of the application with a description of the functionality behind it. It then states where the development adhered to the functional specification and design manual and where it deviated from it and why.

The overall learning outcomes of the project, both technical and personal are discussed, as well as a reflection of where the author felt they could have improved for the betterment of the applications development. Lastly, possible future developments that could be incorporated into the application to better the experience of both patients and doctors are explained.

# 2. Additional Research Undertaken

## 2.1. Backend Service Overlooked

Initially, when the research for this application was concluded, the original plan was to use a MySQL database along with C# for server-side when processing and returning information to the user as well as storing any necessary data. There are a number of reasons as to why this decision was abandoned and additional research undertaken.

Firstly, it was advised during the early stages of the project that MySQL should not be used as it is an older more well-known technology released in 1995 and would present less of a challenge during development. Furthermore, in choosing a more recent technology, it would allow for a more difficult learning curve. In doing so, this presented not only more of a challenge but ensured that by the end of the project a new technology had been adopted and learned, making for a much more worthy overall development experience.

Secondly, the initial research overlooked other requirements that should have been noted when deciding on backend technologies. For instance, there was no prior decision made as to how the application would be hosted, an oversight on the project owners’ part. Another oversight was how the authentication would be handled, as there is more than one user role or type.

The solution to all of these issues, was resolved with the discovery of Googles’ Firebase platform which allows access to a free spark plan with generous limits for new users. There is also a pay as you go blaze plan that calculates pricing for apps at scale, suitable for more established businesses or developers looking to use firebase to the full extent.

### 2.1.1. Firebase

Firebase was a platform invented in 2012 by an independent company and later acquired in 2014 by Google. It is now their flagship service for application development. It is a fully managed platform, providing developers with a variety of tools and services to enable them to create fully developed applications so that they can continue to expand their user base. Firebase can be used for developing native iOS and Android applications as well as web-based applications. Firebase comes with its’ own infrastructure and tools that simplify the overall development process. *“These toolkits are interconnected, scalable and integrable with third party software to overcome complex challenges with standard building blocks.*” [1]

### 2.1.2. Authentication

Firebase authentication provides the necessary backend services for authenticating users, in order to grant them access to the application and read/write privileges to the database. A number of authentication methods are available, including email and password, phone number, Google, Facebook, Twitter and more. These log in methods are made available through the user management system included with Firebase Auth. Authentication of a current users’ state is made easy thanks to the observer provided for authentication changes. As for password security, the service uses an internally modified version of scrypt to hash account passwords. *“Even when an account is uploaded with a password using a different algorithm, Firebase Auth will rehash the password the first time that account successfully logs in.”* [2]

### 2.1.3. Hosting

Firebase hosting provides fast and secure, production-grade web content hosting for developers. With regards to security, zero-configuration Secure Socket Layer (SSL) is built into the service to ensure that content is always delivered securely. On initial hosting, developers are provided with a firebase sub-domain, but are also able to serve their content on a custom domain if they so wish. Firebase hosting can seamlessly integrate with other services, such as Authentication and Cloud Firestore, straight out-of-the-box.

### 2.1.4. Cloud Functions

Cloud Functions provide developers with the ability to automatically run backend code in response to events triggered by features within the application and HTTPS requests. Servers are managed and scaled automatically, allowing developers to concentrate on the programming of the functions which are stored in Google’s cloud to be run in a managed environment. [3] To summarise, a developer need only write and deploy a function and the servers will manage the function immediately.

### 2.1.5. Cloud Firestore

Cloud Firestore provides developers with a flexible and scalable NoSQL document database. This database can be used not only to store data from the client-side into a hierarchical data stricture, but also to sync with the server side. Realtime listeners are used for responsive data synchronisation in order to update data on any connected device.

### 2.1.6. Cloud Storage

Cloud Storage provides the necessary means to store and serve user-generate content, such as files, photos and even videos. It is extremely powerful yet also a cost-effective object storage service. [4]

## 2.2. Introducing a Framework

When the initial research for this project was undertaken, the notion of implementing a framework for the basis of the application was undervalued and misunderstood. It was believed that the development of the application could rely solely on the Semantic UI library as well as HTML & JavaScript to create a fluid web-progressive single page application. As development progressed it became apparent that this was a crude and inexperienced approach to web application development in general. Whilst the application could create a navigational page flow without reload, trying to implement multi-user views that were clean was impossible. If a user had any basic knowledge of web design, they would be capable of displaying the hidden views through inspecting the page elements. In the end, not only did it feel unprofessional to not use a framework, but it was also creating a serious security risk to the user.

Though there was only a little over a month left to the deadline, a bold decision was made to completely redesign and refactor the project using the VueJS framework. Switching to this framework supported not only the progressive web app functionality which would allow the user to install the application to their device to use as if it was a real native android or iOS app, but also allowed for the implementation of navigational guarding for page routes and differentiation of users. Semantic UI was also swapped out in favour of Vuetify, VueJS’ user interface (UI) component and classes library.

## 2.2.1. Vue JS

Vue was initially released in February of 2014 by Evan You, and unlike other popular frameworks, it is entirely created and maintained by the author and its’ community. VueJS is open-source and follows the Model-View-Viewmodel software architectural pattern. It is marketed to be an approachable, versatile and performant JavaScript framework, and continues to rapidly grow in popularity each year. This in turn allows it to directly compete against other frameworks like Angular and React, due to being more learner friendly, less opinionated, and highly customizable. VueJS is incrementally adoptable, enabling developers to use it to create anything from small widgets on a webpage to full-fledged dynamic JavaScript driven web applications, making it an ideal choice whether it’s for integrating into existing projects or building sophisticated single-page applications (SPAS). Depending on the developers needs it can scale between a library and a full-featured framework. [5]

### 2.2.2. Benefits of VueJS

The following are just some of the many benefits to choosing VueJS over other frameworks:

**Flexible & Scalable** – As mentioned above, VueJS features an incrementally adoptable ecosystem, making it accessible for large scale web applications, as well as for smaller integration with other technologies, making it a more well-rounded technology. Simply put, the framework enforces simplicity without the loss of flexibility.

**Developer Friendly** – VueJS is made by developers for developers. It features a command line interface called vue-cli, which enables developers to easily configure and run a new project in a matter of minutes. For those who prefer a more graphical user interface, Vue also provides vue-ui, which provides the user with easy access to serve and build their projects, view and add plugins and dependencies as well as view their project configurations. This helpful tool eliminates the need to continually type commands into their terminal, therefore reducing the time spent on meaningless tasks.

**Efficiency** – According to official benchmarks, due to being significantly smaller than others in size, applications developed in Vue load much swifter and use less bandwidth overall. Pair this with the frameworks scalability, upgrading smaller scale applications to larger more advanced systems is a relatively simple process with little to no performance decrease. [6]

**Reusability** – Perhaps one of the features that makes VueJS so accessible are its’ components. These components essentially allow developers to create custom elements which can then be imported into a view page, dynamically loading the content from the component into the main template view at a desired section. While this is beneficial in that it helps to declutter a page of redundant code, it also eliminates repetition of code, for instance, the navigation bar of an application can be loaded using a component, rather than coding it individually for each page.

### 2.2.3. Vue Router

The Vue Router plugin is the official router for VueJS, integrating with the frameworks core to provide developers with the ability to make secure and fluid single page applications. User routes can be set up by creating a new route instance and programmatically telling the router each route within the application. Each route can contain different properties, such as the path that can be used in the address bar to access a specific route, the name property which is used to reference the route to run and the component property which defines the component or view to load when a user visits the route or path. A developer can set up the default page on initial load of the application, ensuring the user is always loaded into this default page first. If a user attempts to load a page that does not exist the router can simply redirect them to the default page. If a user wants to quickly get to a specific page, they can type in the address path in the browser to access this page. When using a vue router, whilst the user might be going to a different page they are not actually making any additional requests to the server as this is all handled by VueJS, making the flow of the app feel quick and efficient to the user. The overall transition from page to page is smooth thanks to view transition effects powered by VueJs’ transition system.

Within the router file, developers can also implement route params, queries and wildcards. For instance, if the application requires that the user be registered and logged in, the developer can ensure this by using navigation guards and route meta fields, which essentially redirect or cancel a user’s flow or determine if the user is authorized. By implementing security checks to determine if the user is authorized or is a specific user type i.e. employee and employer, the vue router can keep each page pertaining to each user secure from one another.

### 2.2.4. Vuex

*“Vuex is a state management pattern and library for VueJS that acts as a centralized store for components, with rules ensuring that the state can only be mutated in a predictable fashion.”* [7] Incorporating Vuex into an application supports developers when dealing with shared state management at the cost of more concepts and boilerplate, making it a trade-off between short-term and long-term productivity.

### 2.2.5 Vuetify

Vuetify is a UI component and classes library, based on Googles’ Material Design philosophy, but designed specifically for VueJS. This design philosophy aims to provide the user with a good user experience and a clean UI. Vuetify gives developers access to different UI components such as buttons, toolbars, popups and a grid system used for flexible and dynamic device design in order to adhere to this philosophy.

## 2.3. Additional Research Conclusion

From the additional research undertaken, it was decided that Firebase would be used as the backend of the application. Firebase Authentication was to be used with Cloud Functions to successfully register the users and assign them a base role of patient. Client-side information will be stored in the Cloud Firestore to avail of real time updates of the data, for instance, as the doctor updates a patient’s medical record, so that the patient can see these updates straight away. As Cloud Firestore was chosen for the database this meant that MySQL was no longer necessary. The entire application will be hosted with Firebase Hosting to ensure that it is made publicly available, just as a real software product would. Firebase Storage will be used to allow doctors to upload documents i.e. prescriptions.

As for the applications framework, while the research document was re-read and the previously researched frameworks were reconsidered, it was decided that Vue was the most appropriate framework for this application, due to the limited time left to complete the project, the relatively low learning curve and the developer friendly nature of the framework. It was also the most suitable for enabling the progressive web application nature of the project as well as for developing it in a fluid single page application.

# 3. Project Description

The E-Dochtúir application has been developed as a single page web application with progressive web application capabilities, achieved through the utilization of the VueJS framework and its’ official Vue Router plugin. The Vue Router enables navigation guards that protects the applications many views from being accessed by unauthorised users, as well as prevents authorised users from accessing pages that are not associated with their user role within the application. The applications user interface has been designed using VueJS’ corresponding UI component and classes library, Vuetify, in order to comply with Googles’ design philosophy of providing a good user experience and a clean and smooth UI. Much of the applications front-end is developed by using components and adding props which give said components specific functionality or class styles. VueJS’ framework and documentation continues to receive regular updates and while this has occasionally led to some unpredictable bug issues, it has a thriving online community of developers willing to share their knowledge and expertise, making these issues manageable.

The back-end of the application was created using Googles’ Firebase platform. Application users are registered using firebase authentication with email and password and their passwords stored securely. On submission of registration data, Cloud Functions assign a role to the user upon successful registration. Cloud Firestore is used to store client-side data inside a NoSQL database system and update any data in real-time for a fast and responsive experience where the user can see updates happening before their eyes.

With regards to the code, there are individual vue pages for each piece of functionality, with separate components being pulled in from a components folder to reduce the amount of repetitive code, such as the navigation bar which is featured on each page. The layout for each vue is comprised of two sections; a template section that comprises of the HTML and Vuetify components that make up the UI and a script section where the functionality of the application is stored. The script is broken down into different properties to be exported, such as created, computed, data, validations, methods etc. All functionality is written in JavaScript, including any querying or firebase functions.

The chosen integrated development environment (IDE) to work on during the course of the project is Visual Studio Code, as it has a number of supporting extensions for Vue such as vue syntax highlight and Vetur for auto-completion of tags, linting/error-checking and component data for hover-information.

## 3.1. Application Screenshots & Explained

### 3.1.1. Login Screen

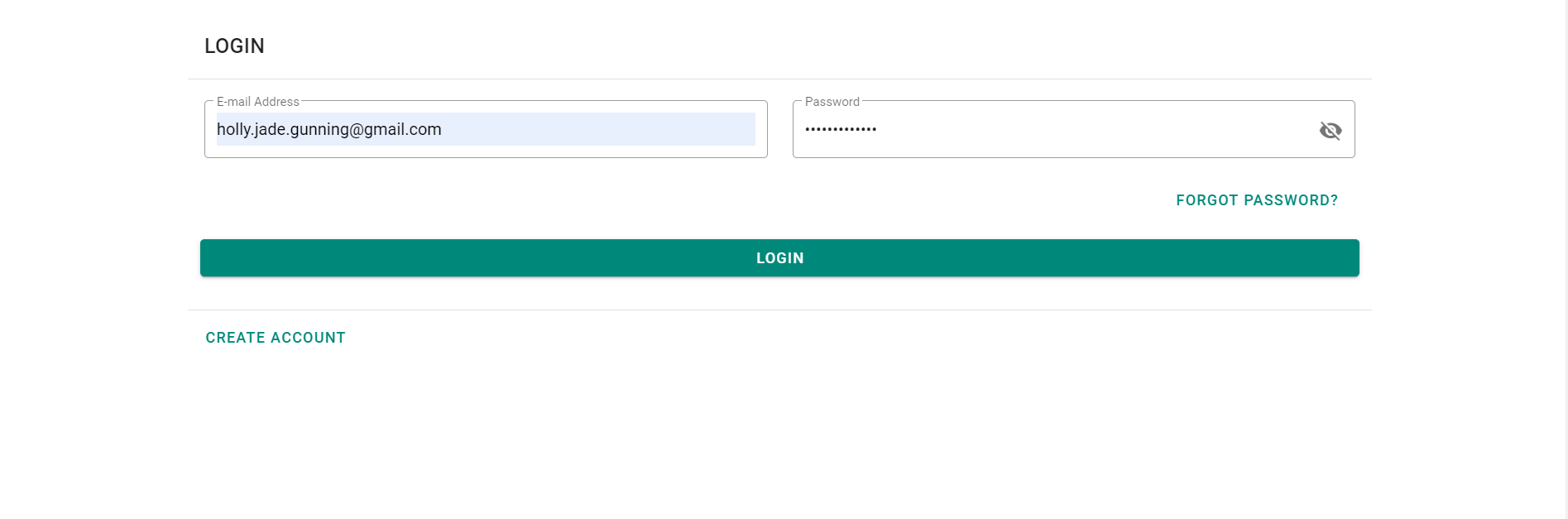


Figure 1: Login screen on laptop

The applications login screen allows a user to log in with their user credentials. They can select the eye icon on the password field if they want to see the true value of their password. Upon successful login the user is redirected to the dashboard screen of the application if they are a patient, the patient database page if they are a doctor, or the single user management page if they are an admin.

If the user is not already registered then they can select the ‘Create Account’ text button which will change their view to the Sign-Up screen. If the user has forgotten their password then they can select the ‘Forgot Password?’ text button which will pop out a dialog box.

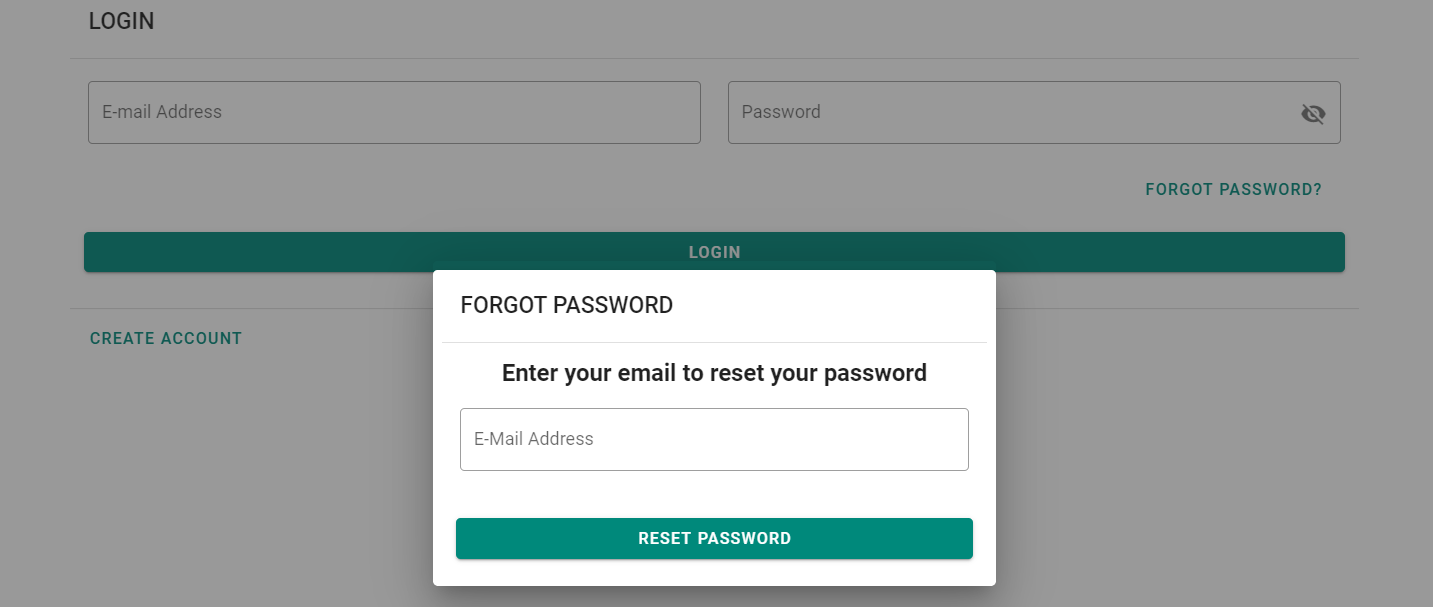


Figure 2: Login screen on mobile

All fields are validated, and appropriate error messages will appear on failed log in or invalid data.

### 3.1.2. Registration Screen



Figure 3: Registration screen on laptop

The applications register screen allows the user to register a number of user profile fields and their credentials. All fields are required and if they are not filled in then the form will display the appropriate validation error messages. If the user selects the Terms of Service highlighted text then a dialog box will open up that will allow them to read the applications terms. As with the login screen if the user wishes to see the input for their password they can select the eye icon to reveal it on their screen. If the user is already registered then a firebase error message will populate the error message box telling them that these credentials are already in use. If the user is already registered they can select the ‘Already Registered’ text button which will take them back to the login screen.

Upon successful registration, the user’s authentication credentials are created, storing their unique ID as well as their email and password. Firebase Authentication hashes the password using an internally modified version of scrypt.

At the same time, a cloud function is fired off which sets the role of the registered user to a type patient. A record is created in the “roles” collection in Cloud Firestore, which stores the users roll and email under a document ID that uses the same ID as the users’ unique ID from authentication to connect them to the user. Finally, a record of the users form details, excluding password, is created in the “users” collection in Cloud Firestore, again under a document ID using the same unique ID to connect them to the user.

The user is then redirected to the applications dashboard.

### 3.1.3. Dashboard Screen (Patient)

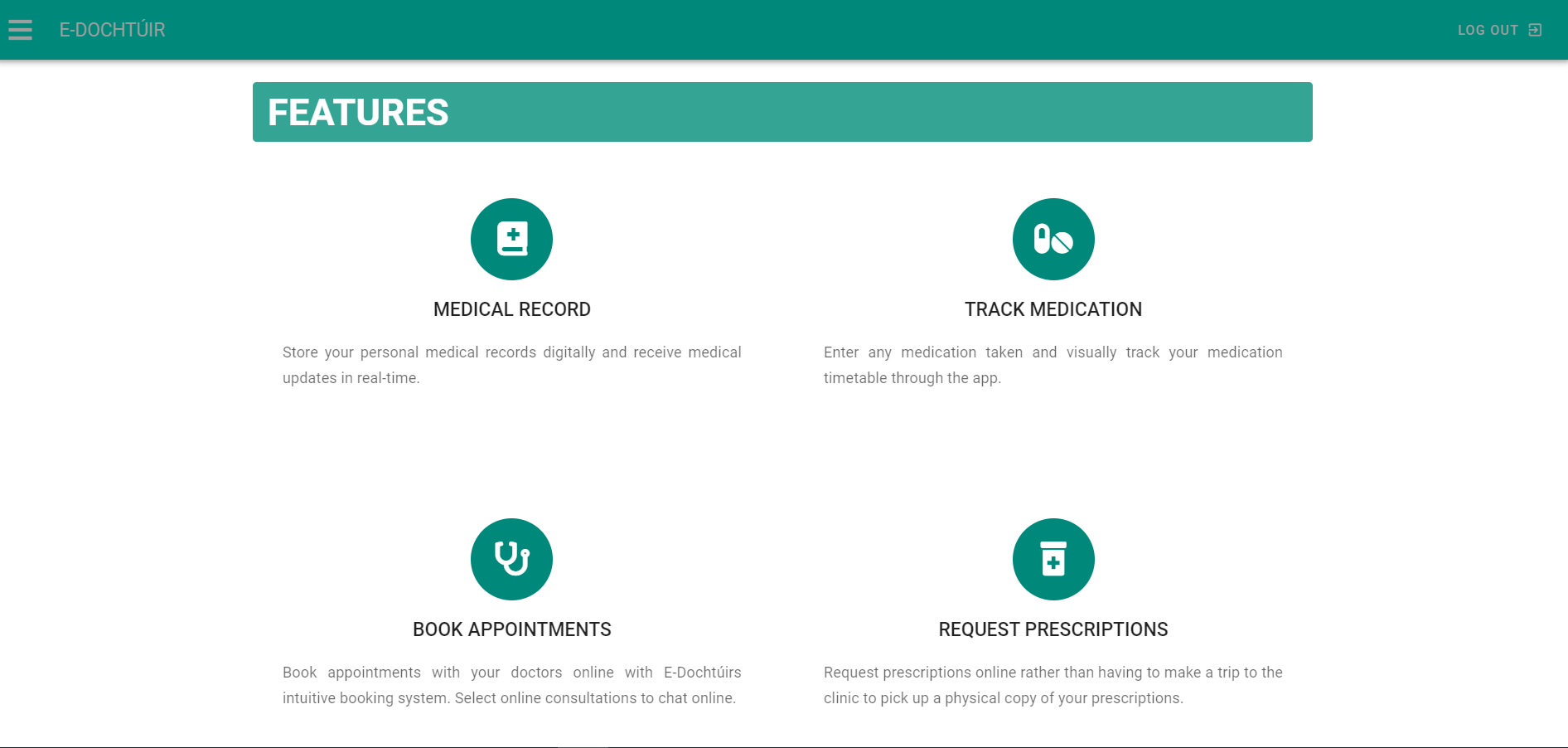


Figure 4: Dashboard Screen

The dashboard screen is the initial page that a patient is loaded to upon logging into the application. It is simplistic in design as its only real function is to explain to the patient some of the features available to them. At the bottom of the page there is a contact us form, that allows the patient to create a ticket that will be stored in the “tickets” collection. The name and email address are auto populated through the patients’ record in the “users” collection. This record will store the Patients ID, name, email and message. This information is solely useful to the developers of the application and perhaps the clinic, who may take any queries or grievances on board to better the services being supplied. The message field has been properly validated to ensure no blank messages are stored in the database.

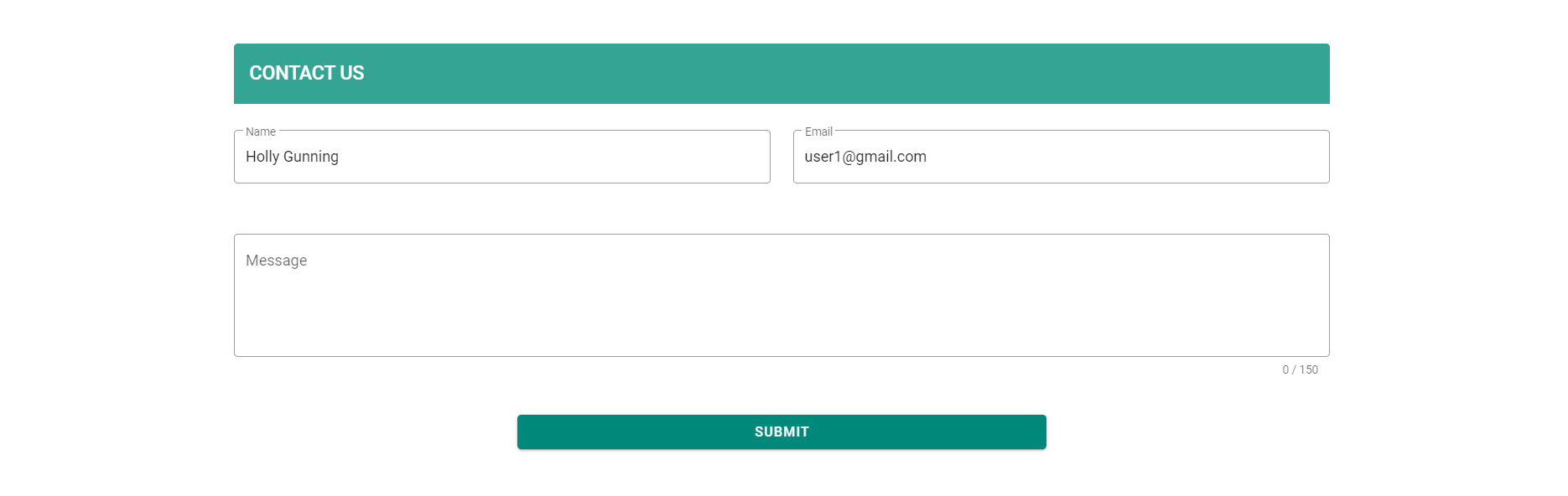


Figure 5: Contact Us Form

### 3.1.4. Navigation Drawer Screen (Patient)

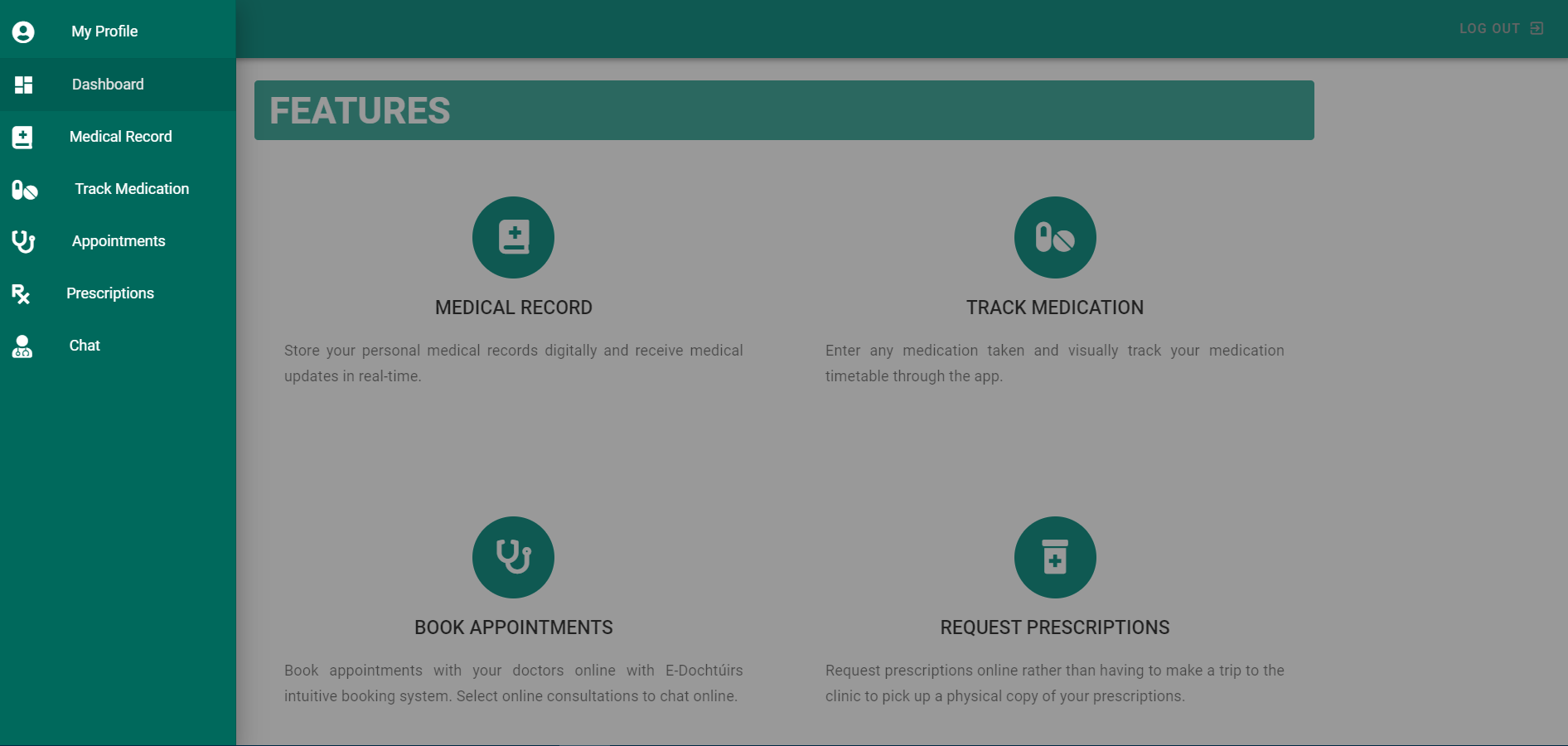


Figure 6: Navigation Drawer Screen

The navigation drawer screen shows the navigational bar for the patient on the left-hand side of the screen. As can be seen from the dashboard screen, the navigation drawer is typically not displayed by default. Instead, the patient must click the three-line icon on the top left of the page toolbar. This in turn makes the drawer active. The drawer will close if anywhere on the page is selected or the patient navigates to a different screen. Depending on which page the patient is on, the button for that page will be slightly darker on the navigation drawer. The decision to hide the initial navigation inside of a drawer was to give the application a cleaner more fluid appearance and a smoother user experience, given the fact that whilst this application is web based, it also needs to scale to iOS and Android devices to reach a broader consumer market.

### 3.1.5. Profile Screen (Patient)

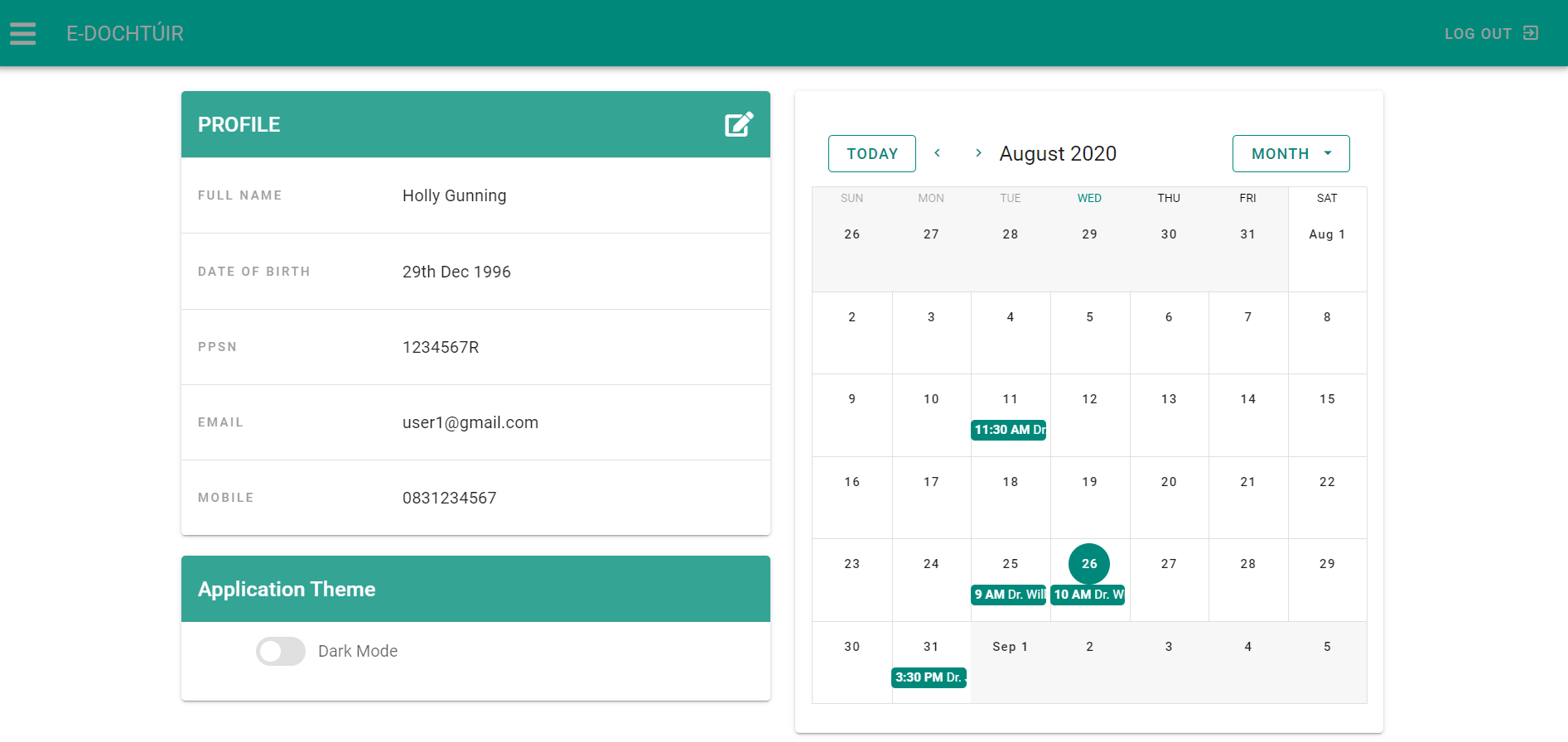


Figure 7: My Profile Screen

The applications profile screen allows the patient to view their user information, edit this information. View a scaled down calendar of their appointments and change the theme of the application to dark mode.

The patients’ information was stored to the “users” collection in the database and connected to the patient through their unique credential ID that was given to them during registration. They can edit this information by selecting the edit icon in the top right of the profile card. When the button is selected the profile card will change to the edit profile card seamlessly.

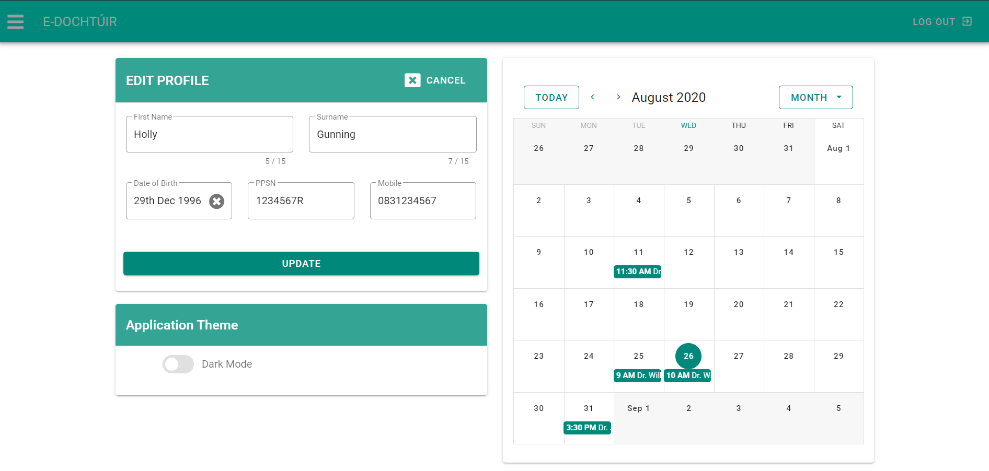


Figure 8: Edit Profile Card Replacing Profile Card

The user can edit any of the fields shown to them on this card. The date of birth field is validated to ensure that the user cannot select an age below 16 years of age. Once changes are saved the card reverts to the profile card view and the updated values are immediately seen on the card.

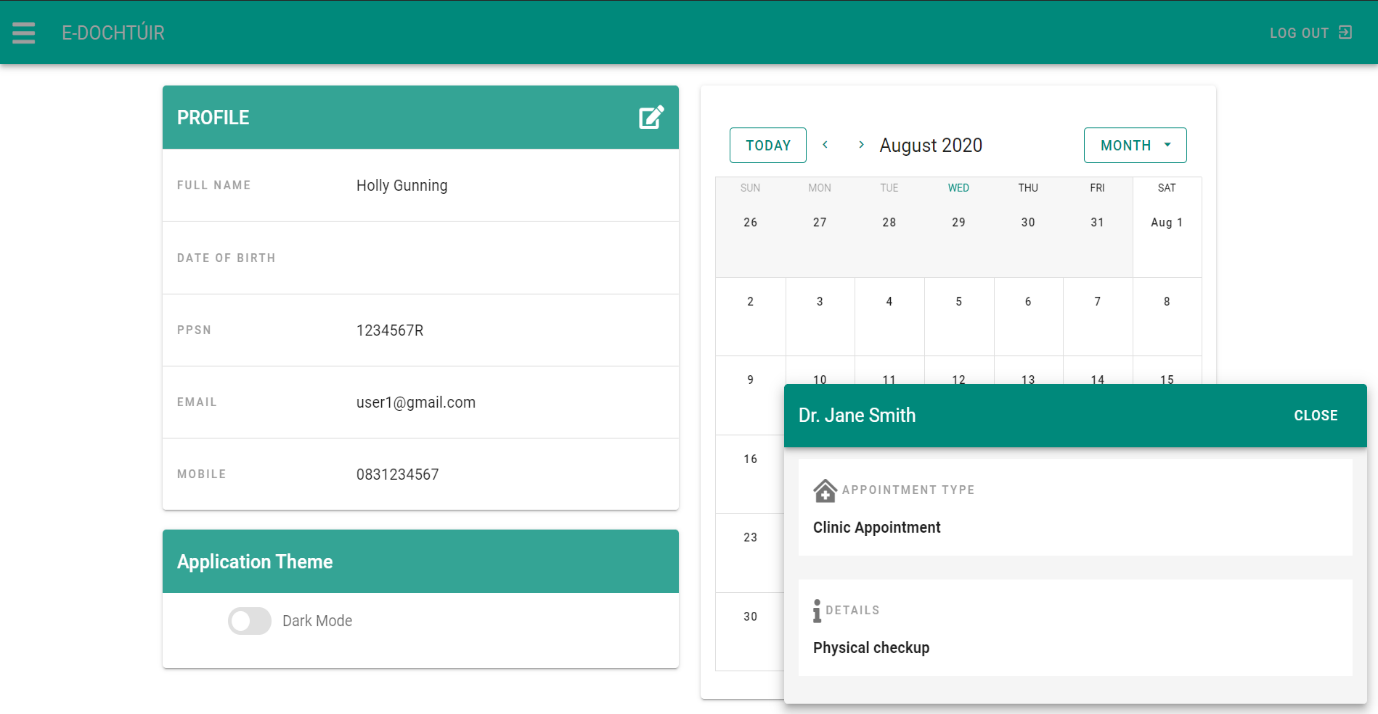
On the right-hand side of the screen a small calendar provides patients with a calendar view of their appointments. If they select the coloured information on the calendar, they can view the appointment details and type of appointment as well as see the name of the doctor they have booked with. This dialog can be closed by either selecting the close button or simply selecting the page outside of the boxes perimeter.

Figure 9: Appointment Dialog on Calendar

If the patient selects a date on the calendar the view expands to display a timed schedule of that specific day. From here they can see their appointment on the specific time slot corresponding to the time that their appointment was booked.

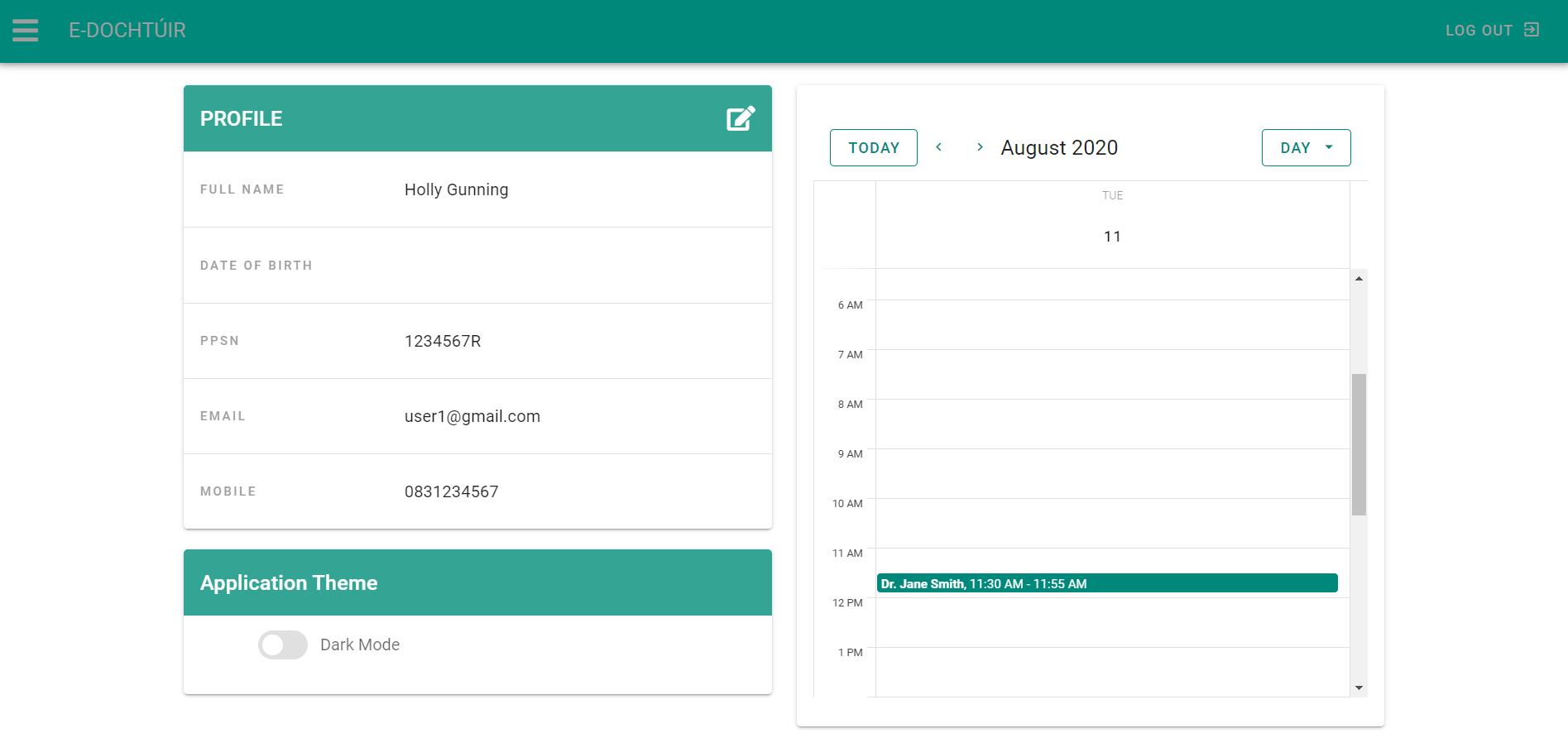


Figure 10: Time Schedule Displayed on Calendar of User Screen

Lastly, under the profile/edit profile card, there is a toggle button. When toggled this allows the user to change the theme of the application from light mode or default to dark mode. Once toggled, this changes the theme of the application entirely. It can be reverted back to light mode at any time.

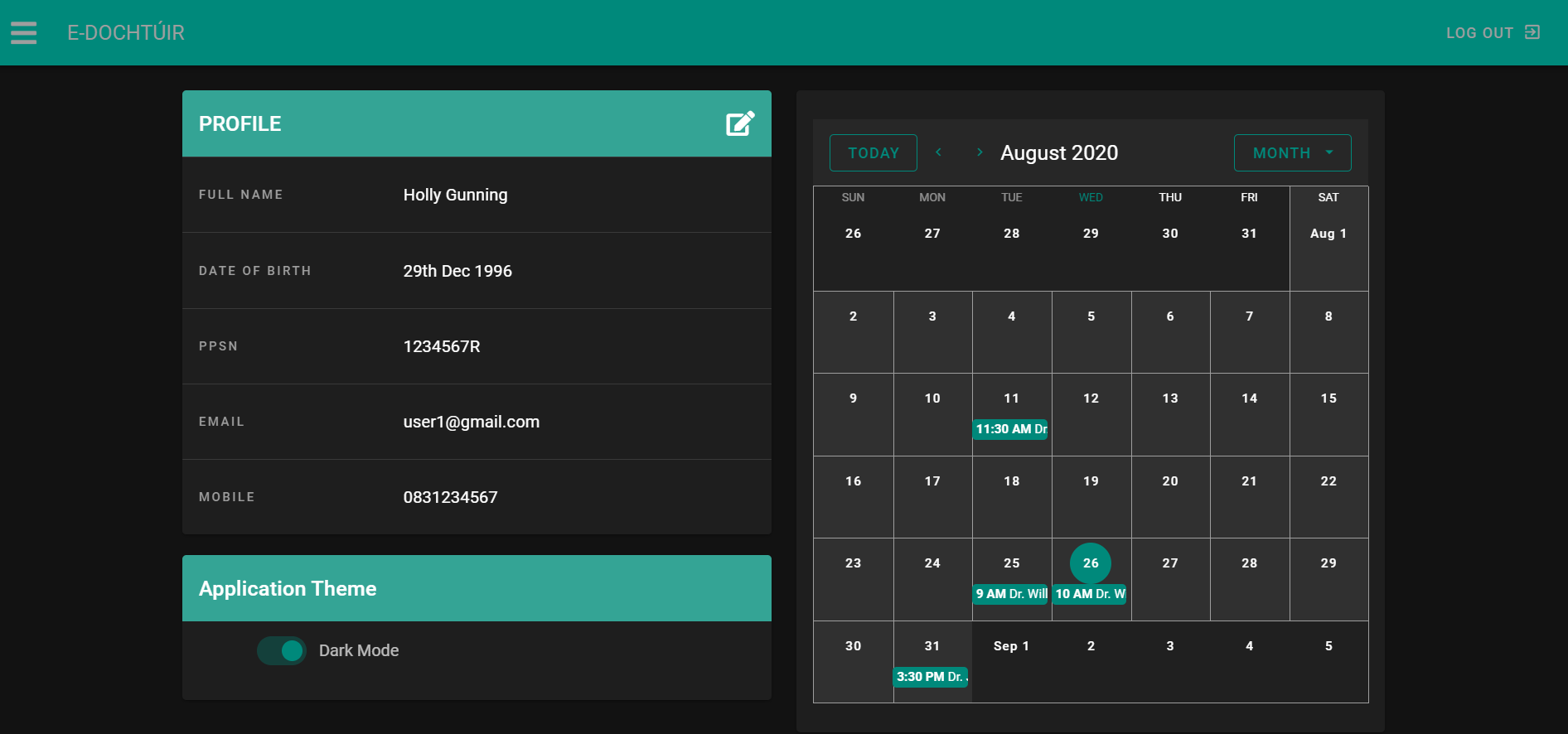
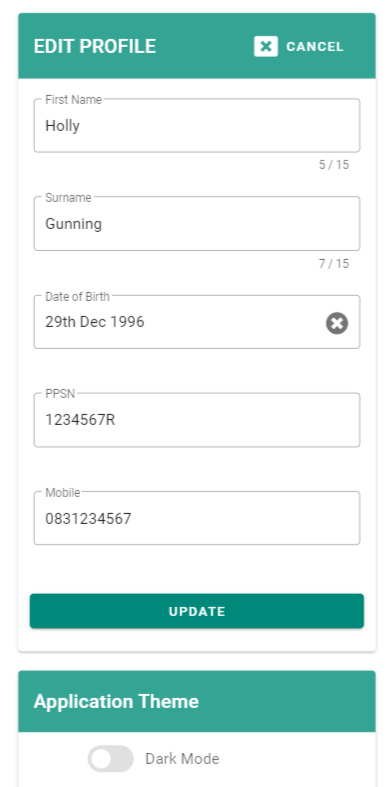


Figure 11: Dark Mode Toggle

Figure : My Profile on Mobile

### 3.1.6. Medical Record Screen (Patient & Doctor)

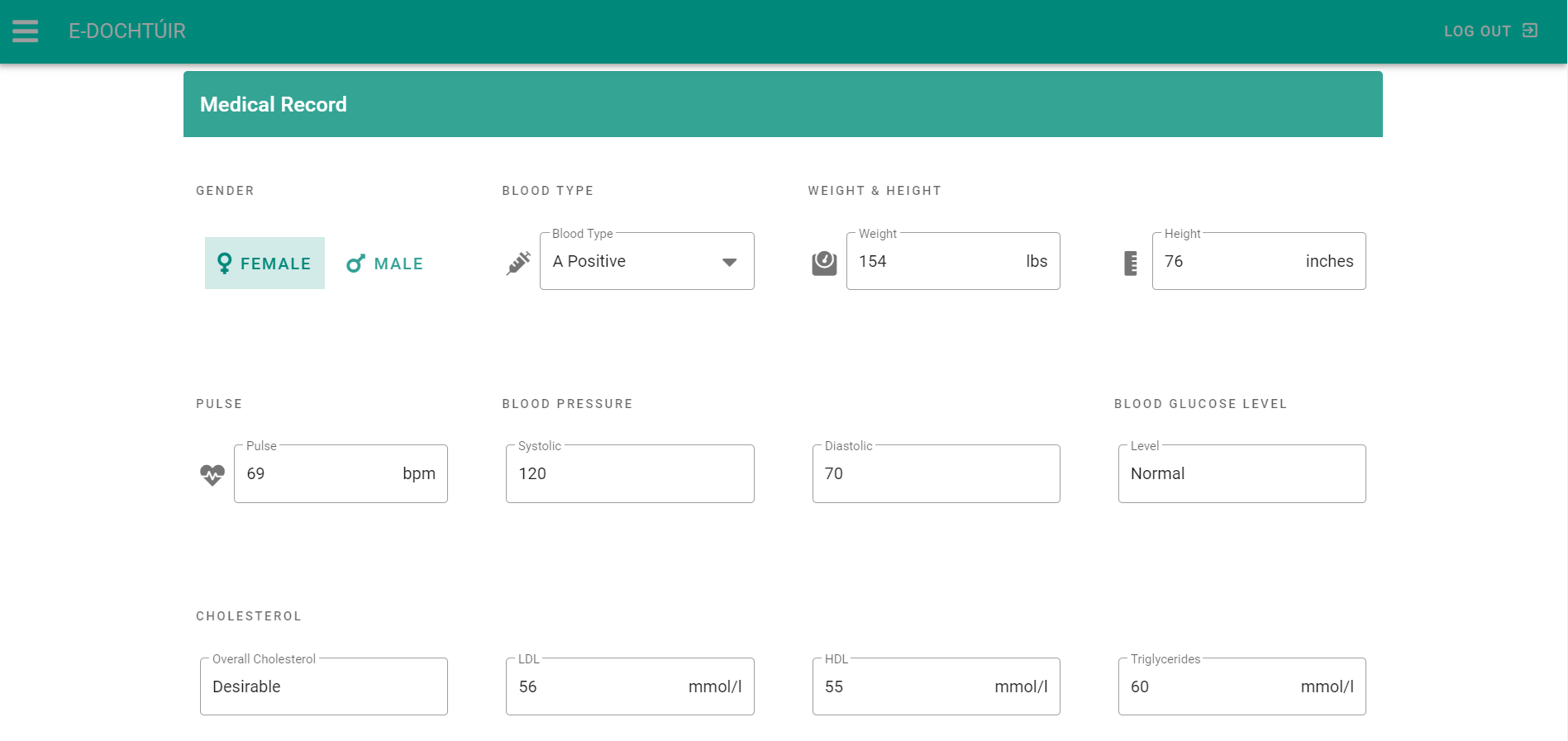


Figure 13: Medical Record Screen

The medical record screen allows patients and doctors to view and update information associated with their medical history. For the patient, the gender field is a grouped button selection where they can switch between male and female. The blood type field is a drop-down menu. Weight and height allow for numerical input for pounds and inches measurements. The pulse, blood pressure, blood glucose and cholesterol fields are not editable by the patient, only viewable, as the doctor is in charge of updating these fields. Below these fields is a tab bar that allows the user to switch between allergies, conditions and immunisations.

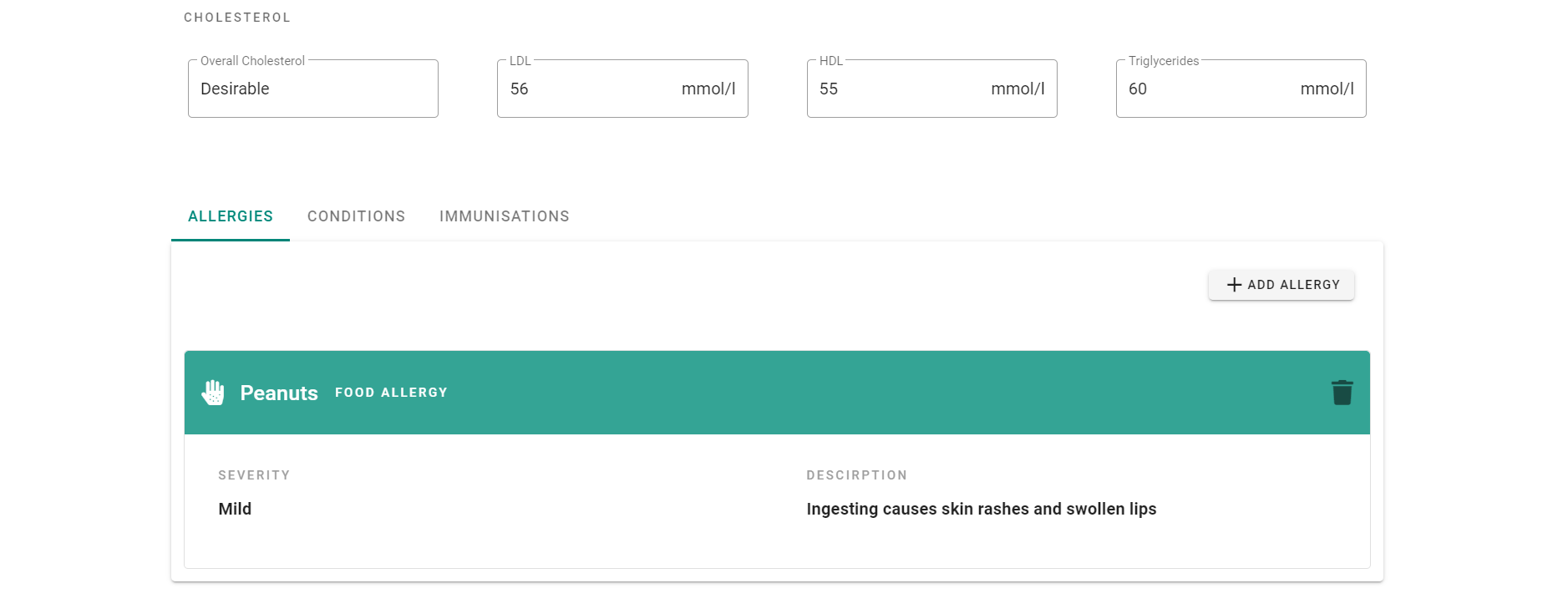


Figure 14: Allergies Tab on Medical Record Screen

In the allergies tab, the user can view, add and delete all allergies pertaining to the patient When the user selects on the ‘Add Allergy’ button, a dialog box opens presenting a small form.

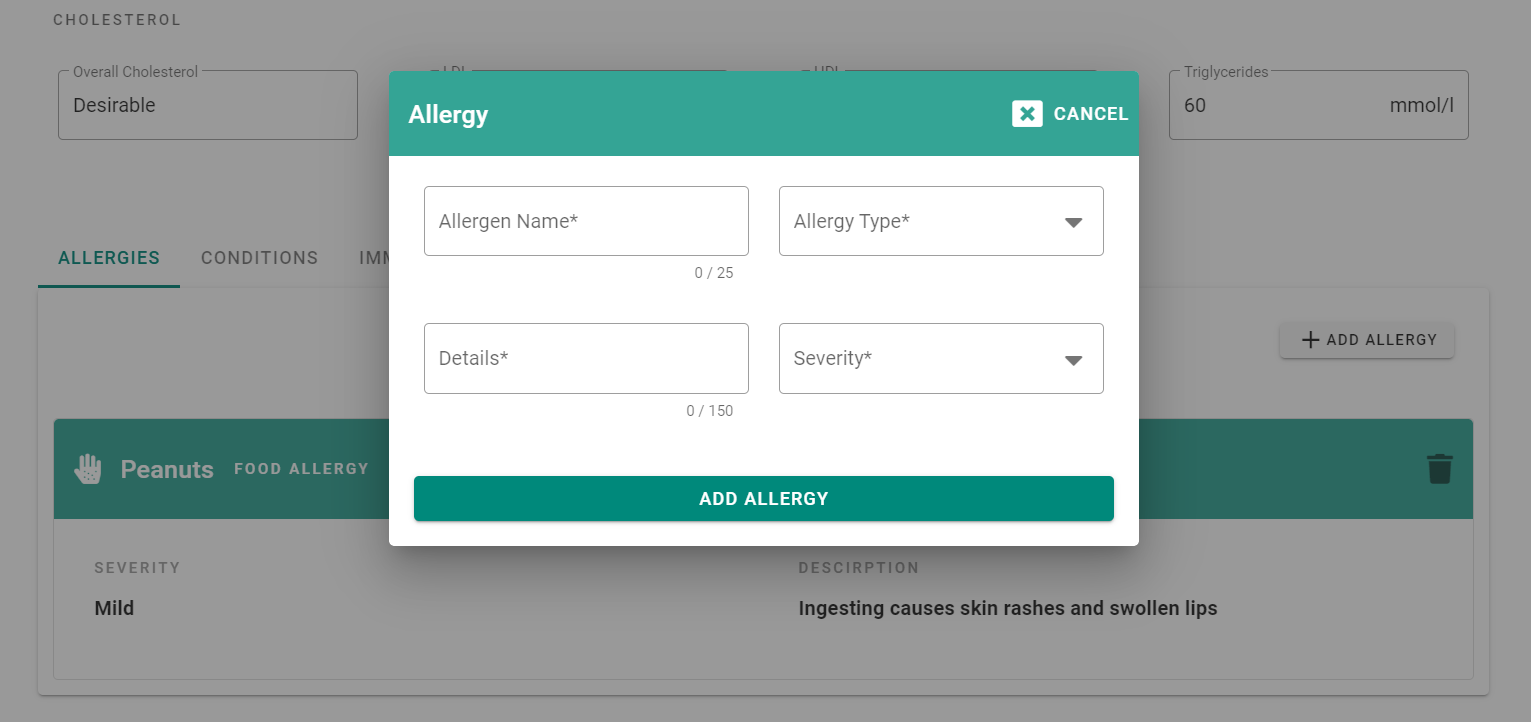


Figure 15: Adding an Allergy on Medical Record

They can give the allergy a name, select a type such as food or pollen from the allergy type dropdown, select a severity from mild to life threatening in the severity drop down and include details on the allergy.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 16: Allergy Type Drop Down Figure 17: Severity Drop Down

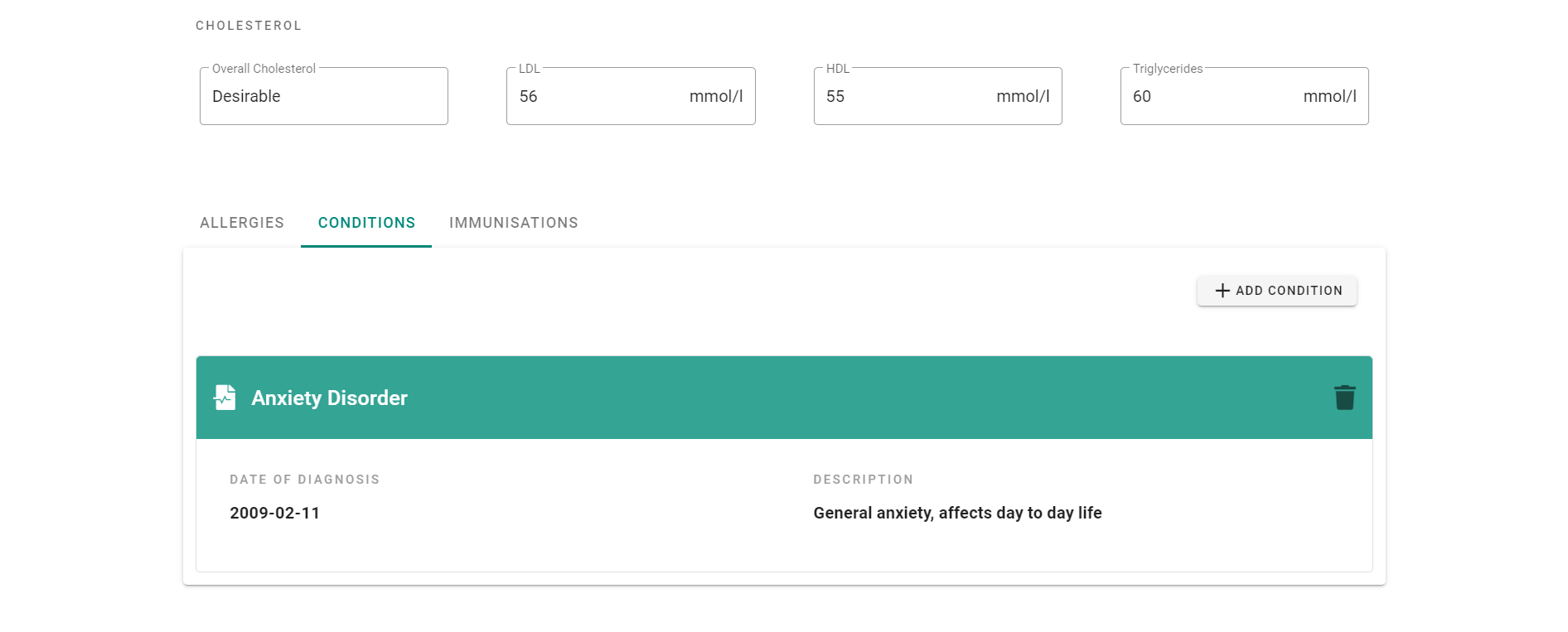


Figure 18: Conditions Tab on Medical Record Screen

In the conditions tab, the user can view, add and delete all conditions pertaining to the patient When the user selects on the ‘Add Condition’ button, a dialog box opens presenting a small form.

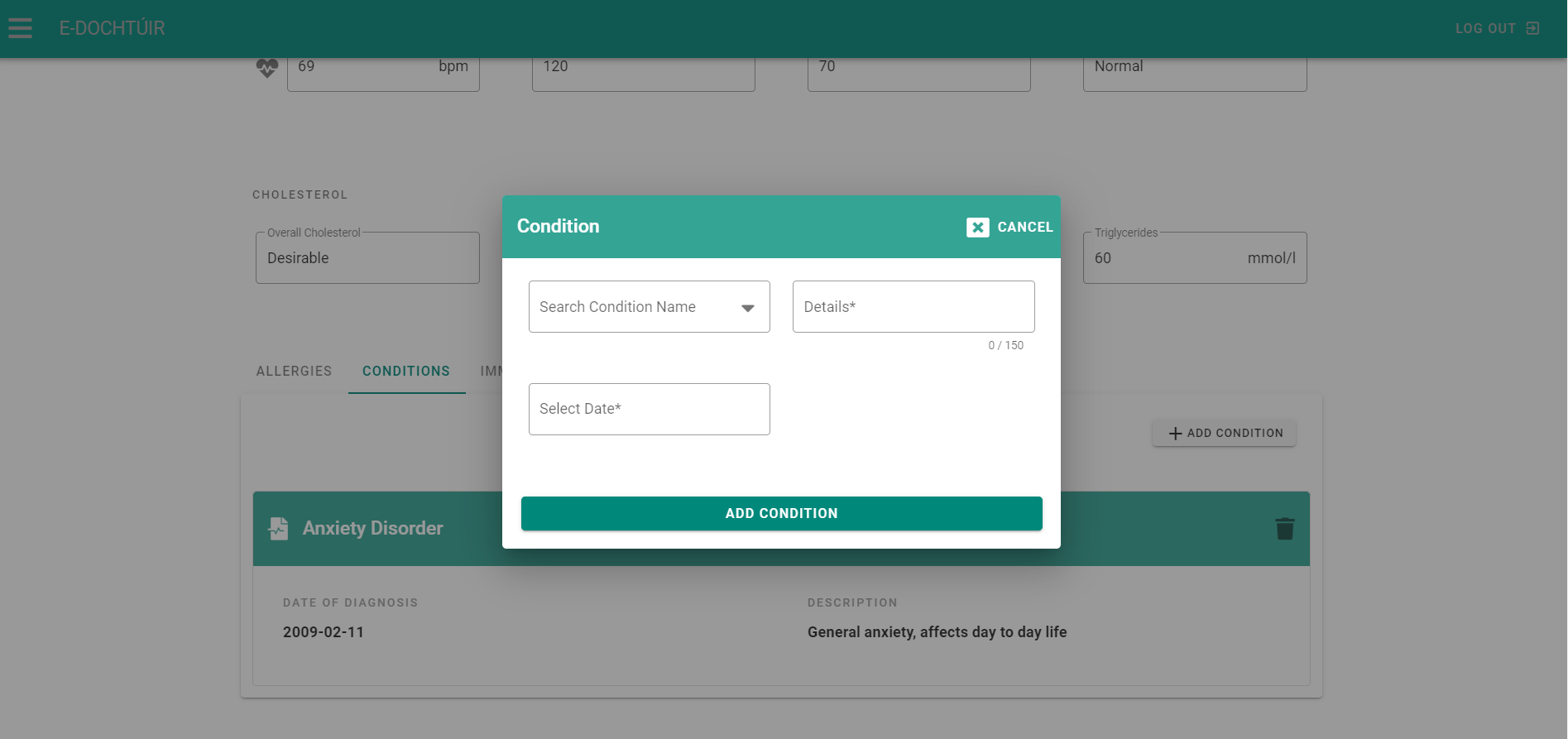


Figure 19: Adding a Condition on Medical Record

They can search for a condition name by typing it in the field, provide details on the condition and select a date for condition diagnosis. The selected date cannot predate the patients date of birth and cannot exceed the current date.

A screenshot of a computer screen

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 20: Search Condition Name Field Figure 21: Select Condition Date Date-Picker

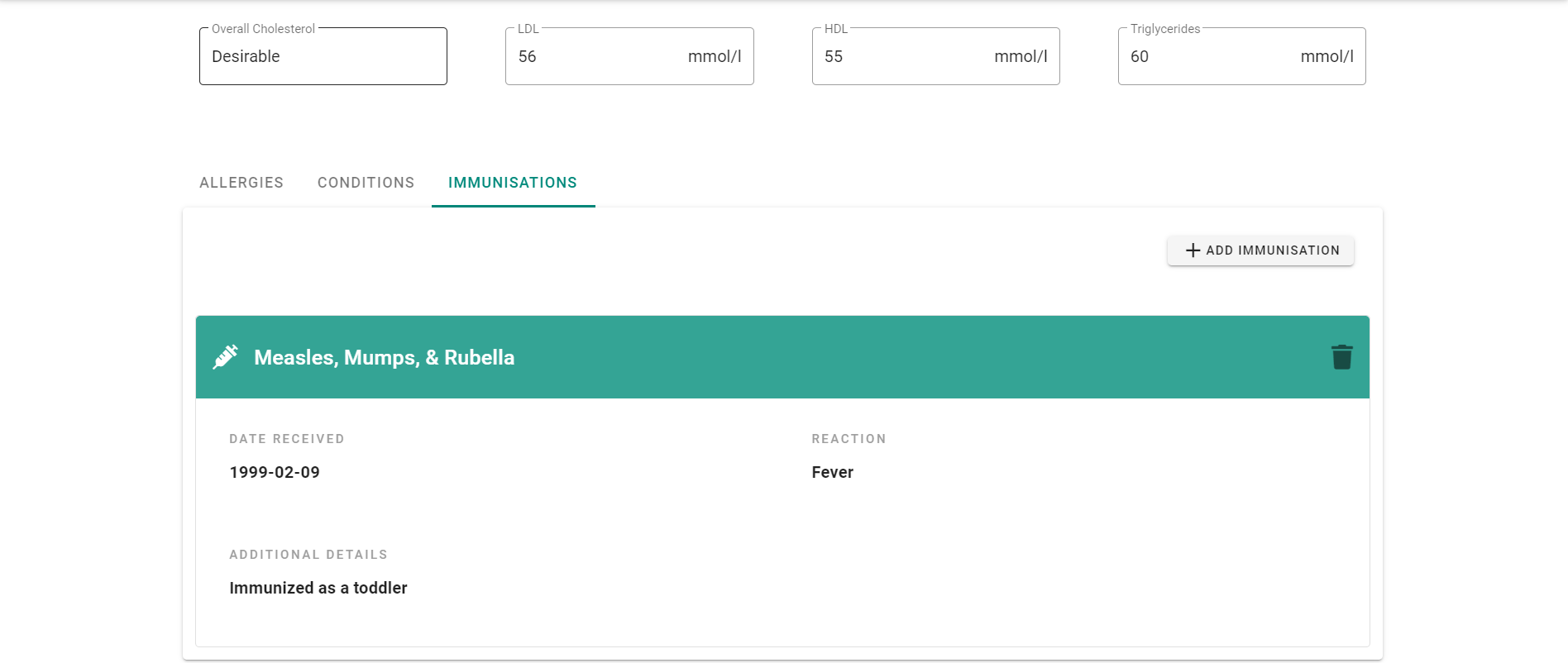


Figure 22: Immunisation Tab on Medical Record Screen

This is the immunisations tab, where the user can view, add and delete all immunisations pertaining to the patient When the user selects on the ‘Add Immunisation button, a dialog box opens presenting a small form.

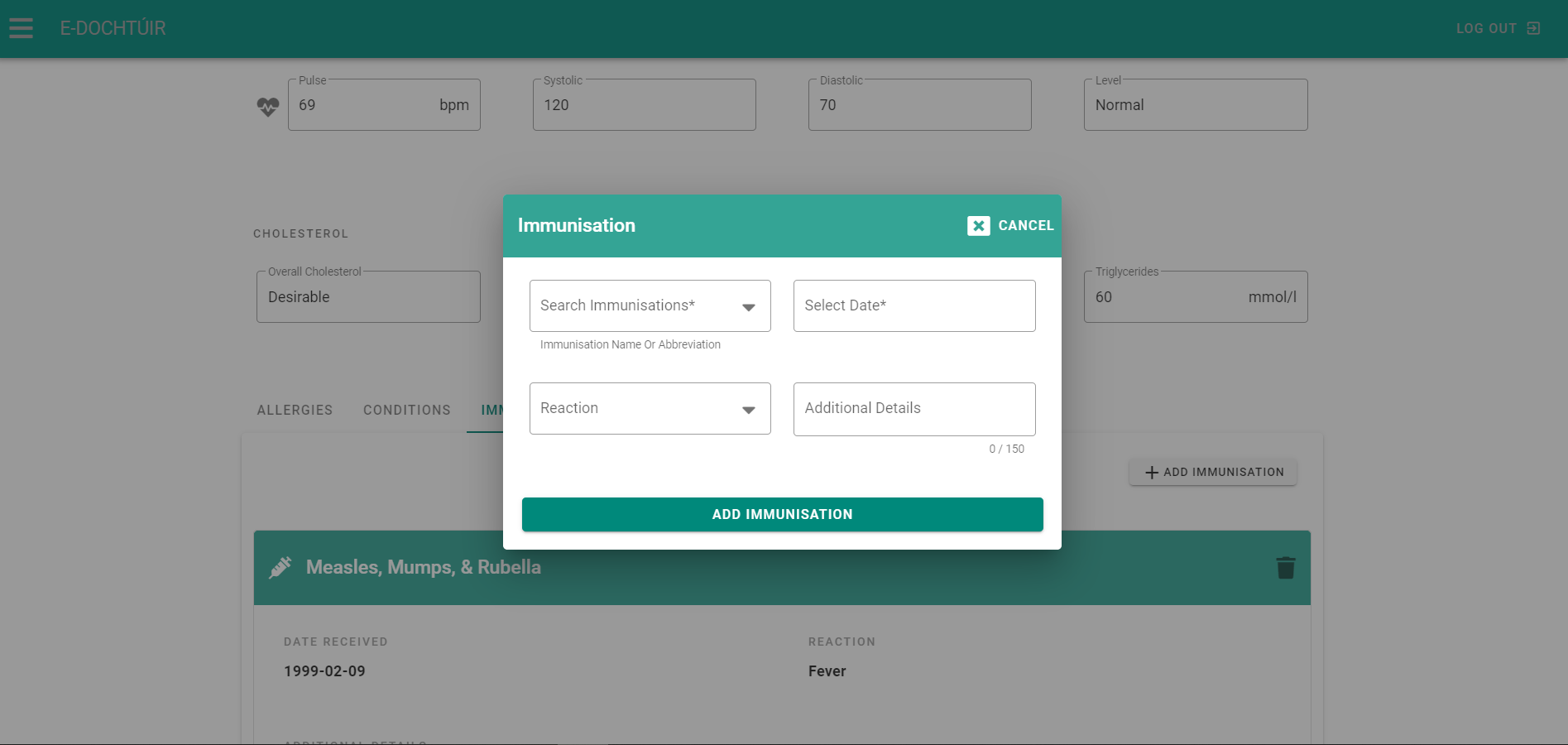


Figure 23: Adding an Immunisation on Medical Record

The user can search for the immunisation name and optionally search with its’ abbreviation if they know it. They can then select the date the patient received the immunisation, the reaction they had and provide additional details. The selected date cannot predate the patients date of birth and cannot exceed the current date.

A screenshot of a computer screen

Description automatically generated A screenshot of a computer

Description automatically generated

Figure 24: Search Immunisation Field Figure 25: Immunisation Reaction Field

All fields on the medical record screen have validation checks to ensure correct information. All fields are validated, and appropriate error messages will appear on failed submission or invalid data.

### 3.1.7. Appointments Screen (Patient)

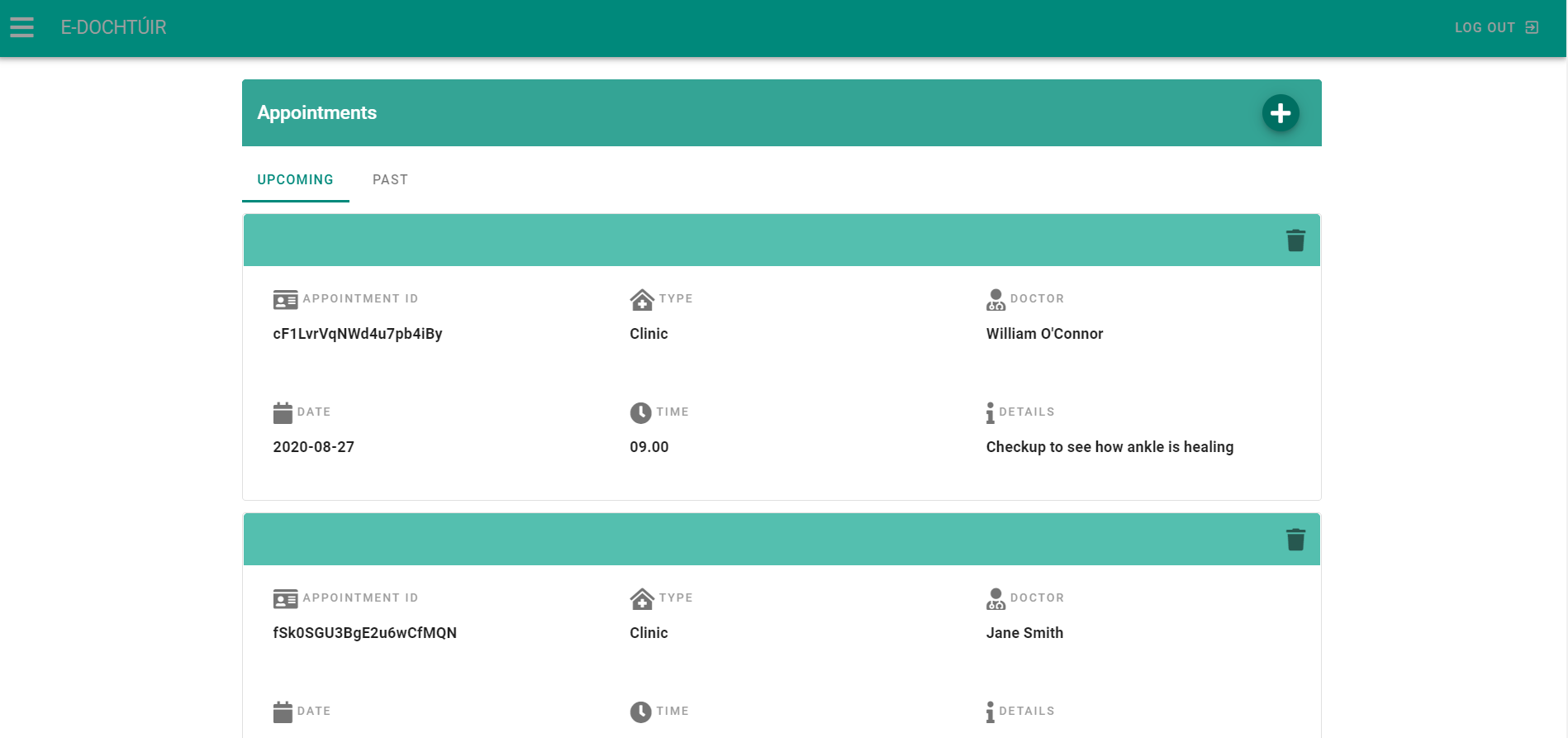


Figure 26: Appointments Screen (View Appointments Screen)

The appointments screen allows a patient to view and delete both upcoming and past appointments, as well as book a new appointment. On the main page, a tab bar separates upcoming and past appointments, which filters appointments to each tab based on the appointment date and whether the date predates or exceeds the current calendar date. Each record is then ordered by the date of the appointment and then by the appointment time, in ascending order. To book a new appointment, the patient simply selects the “+” button on the title bar which opens up a full screen dialog box with the book appointment form.

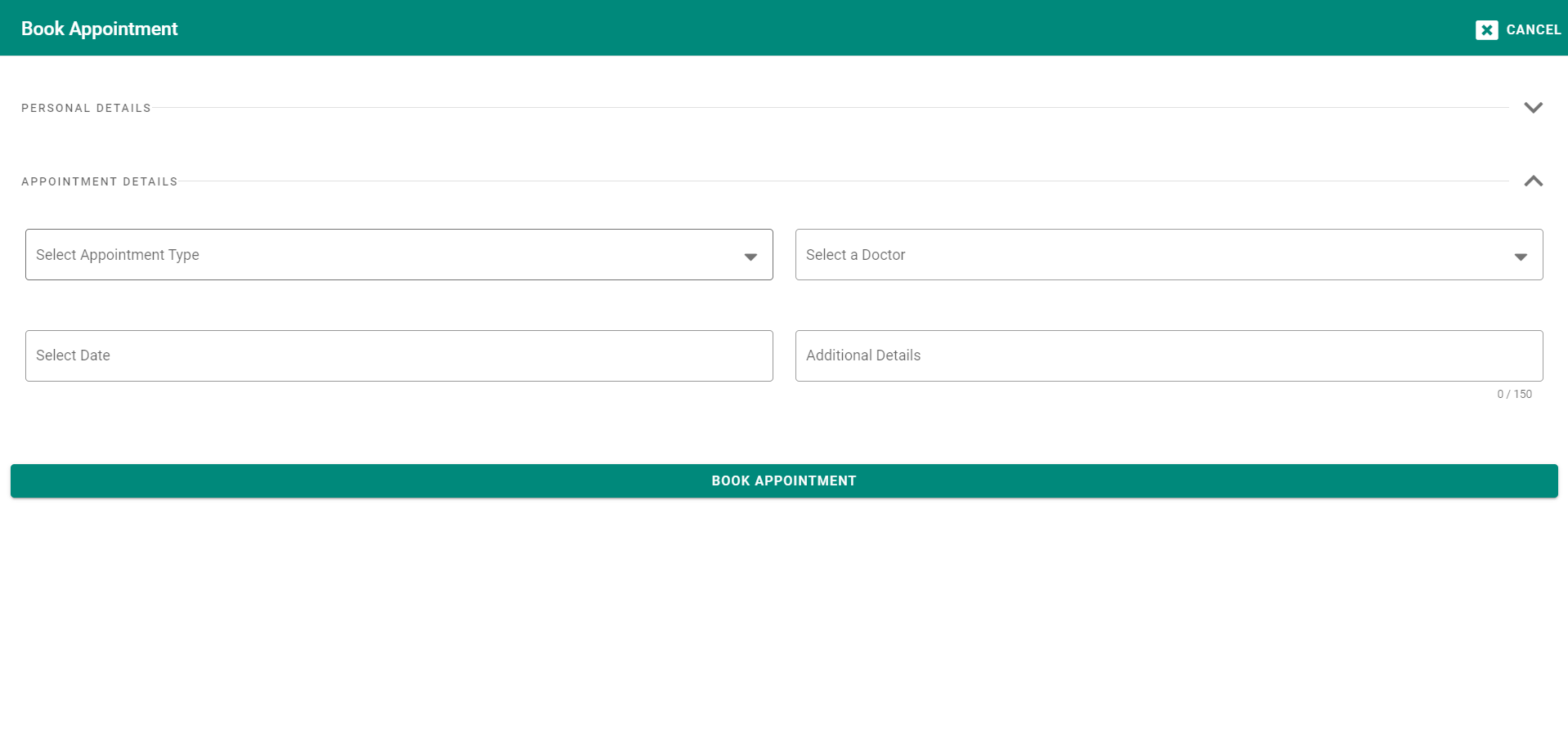


Figure 27: Initial Book Appointment Form

The book appointment screen has a cancel button to return to the view appointments screen as well as a form with two sections: personal details and appointment details. Personal details consist of the patients first name, surname, date of birth, email, PPSN and mobile number. These fields are in a closed accordion, as these details are pre-populated from the patients record in the “users” collection of the database and are not editable. They are simply there to convey to the user that this information will be submitted when they book an appointment.

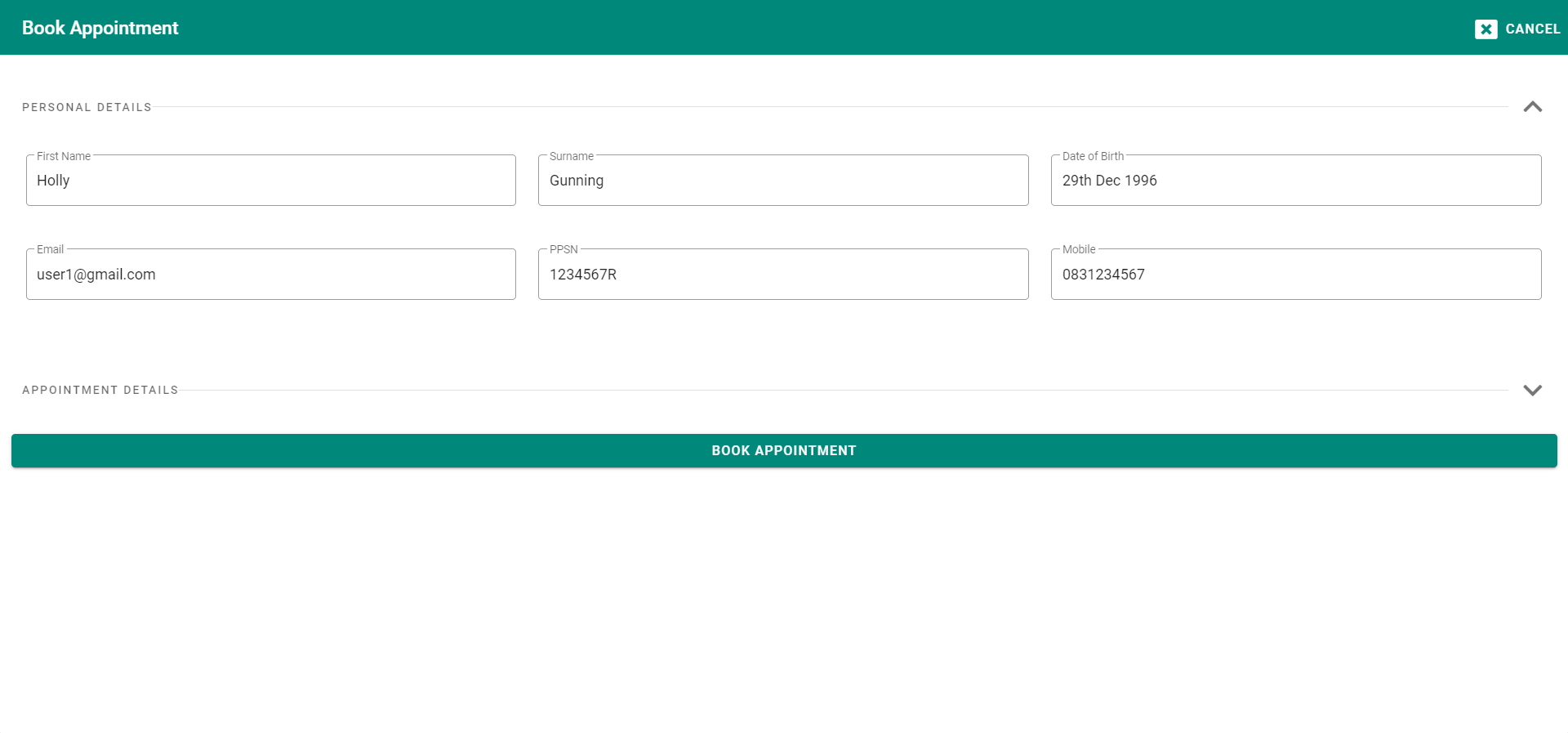


Figure 28: Book Appointment - Personal Details

As for the appointment details section, a patient can select whether the type of appointment is an online consultation or a clinic consultation. This is important for the chat functionality of the application. They can then select a doctor for their appointment. Doctors are populated into this selection menu based on a query which determines which users of the application have a role of doctor and which do not. Any with the doctor role are passed into a “doctors” array which populates the drop-down selection menu.

A screenshot of a cell phone

Description automatically generated

Figure 29: Select Appointment Type Drop-down

A screenshot of a cell phone

Description automatically generated

Figure 30: Select Doctor Dropdown

The appointment date-picker will only allow appointments to be booked from tomorrow onwards. Once a patient has selected both a doctor and a date for their appointment, a method is fired off in the background which looks at all of the appointments booked for the chosen doctor on the chosen date, and returns only the times which are not already booked to the form so that the patient may select a time for their appointment. This prevents double booking of appointments with any one doctor.

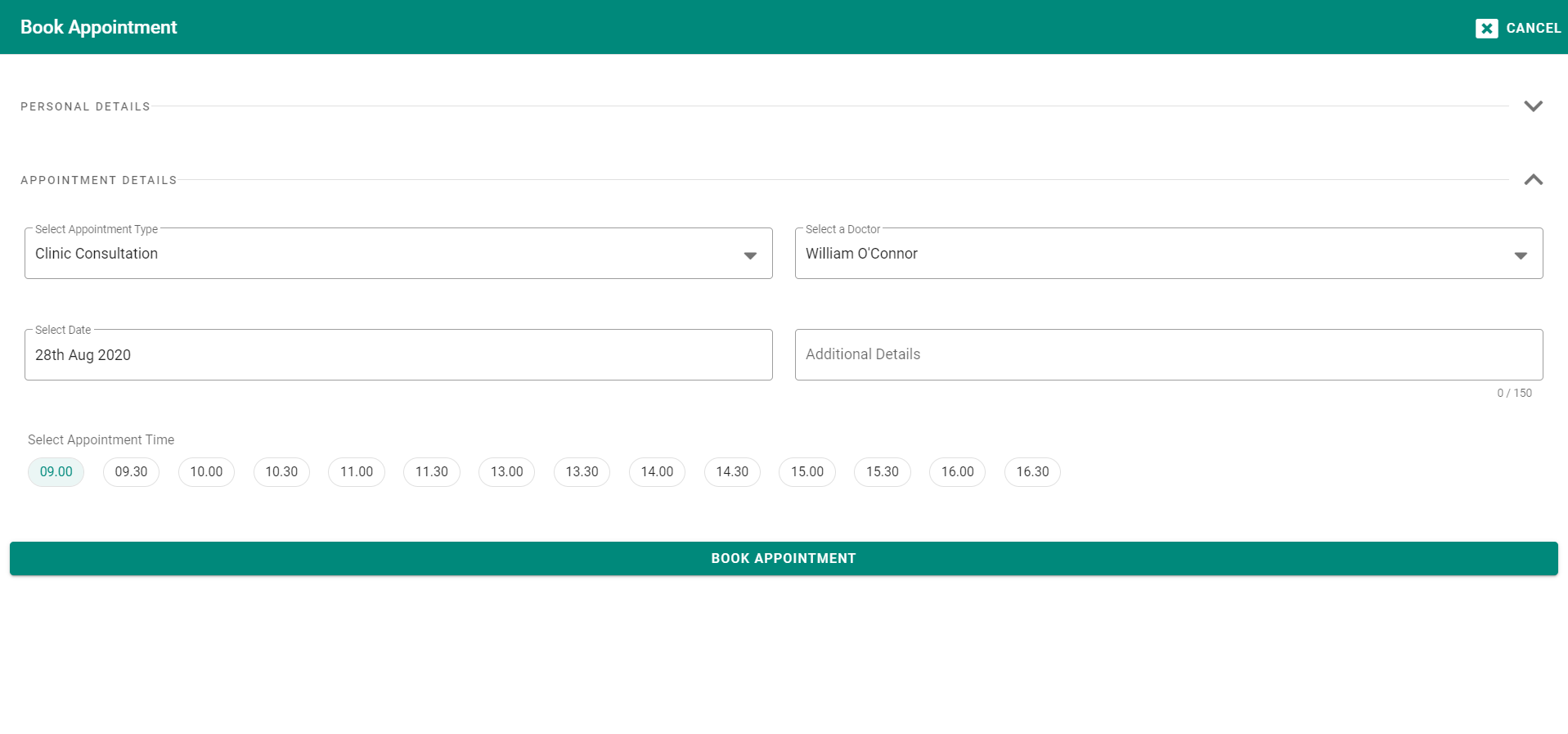


Figure 31: Available Appointment Times Populated to Booking Form

As well as checking for available appointment times, there is a secondary check that queries the “appointments” collection in the database to identify any appointments belonging to the patient. If any of those appointments have the same appointment date as the date of the appointment the patient is attempting to book, an error message will alert them that an appointment has already been booked on this date and the times will not display. This prevents the patient from having an excessive amount of appointments.

Upon submission of the form, the patient is alerted of their successful booking, the book appointment dialog will close, and they will see their new appointment populated into their upcoming appointment tab in real-time, in the correct order. The appointments details are stored in the “appointments” collection in the database and connected to the patient by storing a “patientID” field with the users’ unique ID inside of the record.

All fields on the appointments screen, in particular the appointment booking form, have validation checks to ensure correct information. All fields are validated, and appropriate error messages will appear on failed submission or invalid data.

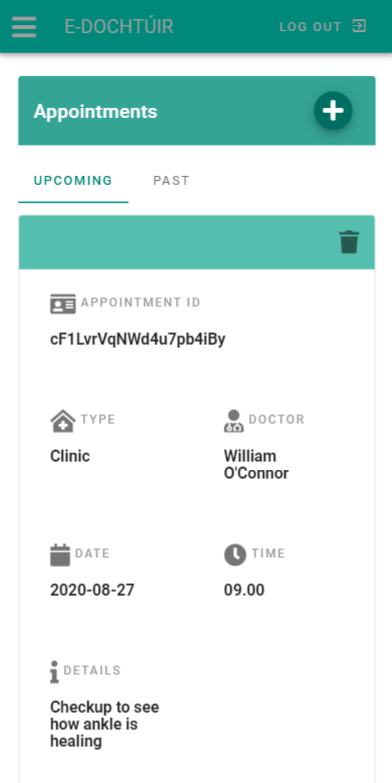
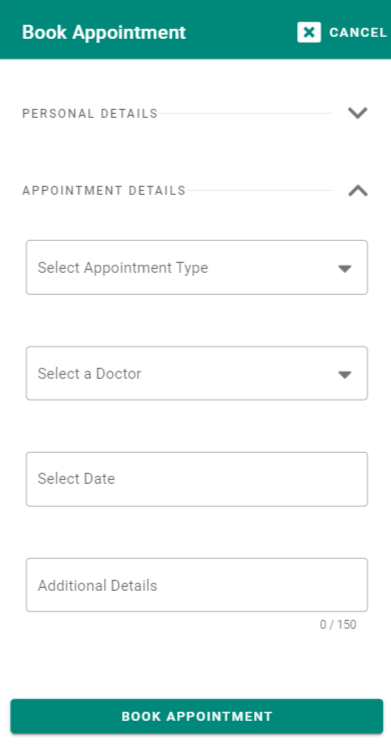
 

Figure 32: View & Book Appointment on Mobile

### 3.1.8. Chat Screen (Patient)

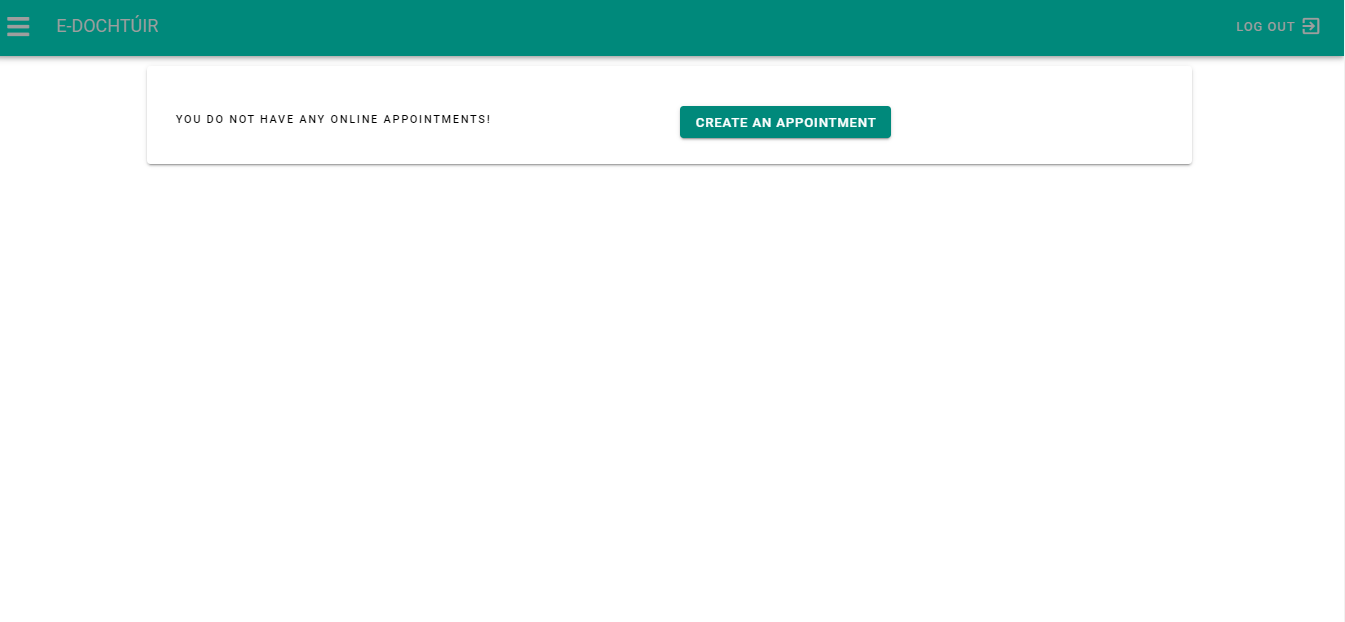


Figure 33: Chat Screen Pre-Appointment

The chat screen is where the patient converse with a doctor for any appointments that have booked with a type of online consultation. Prior to a doctor creating a room, the patient is presented with a simple message stating they have no appointments and an optional button to redirect quickly to the appointment screen of the application. Once a doctor has created a room on their end, a message window will appear, replacing the above-mentioned message and button. Messages are stored into an array and passed to the specific room record in the collection with the users roleID, name, text and a timestamp.

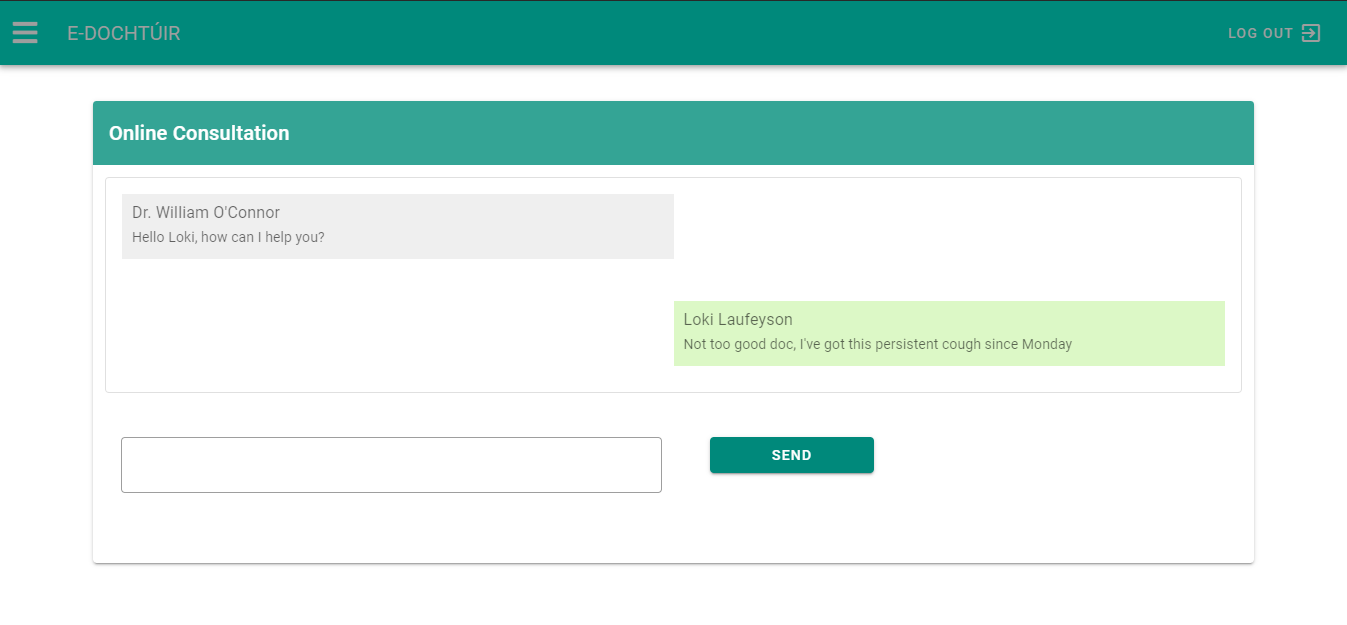


Figure 34: Patient & Doctor Conversing

### 3.1.9. Patient Database Screen (Doctor)

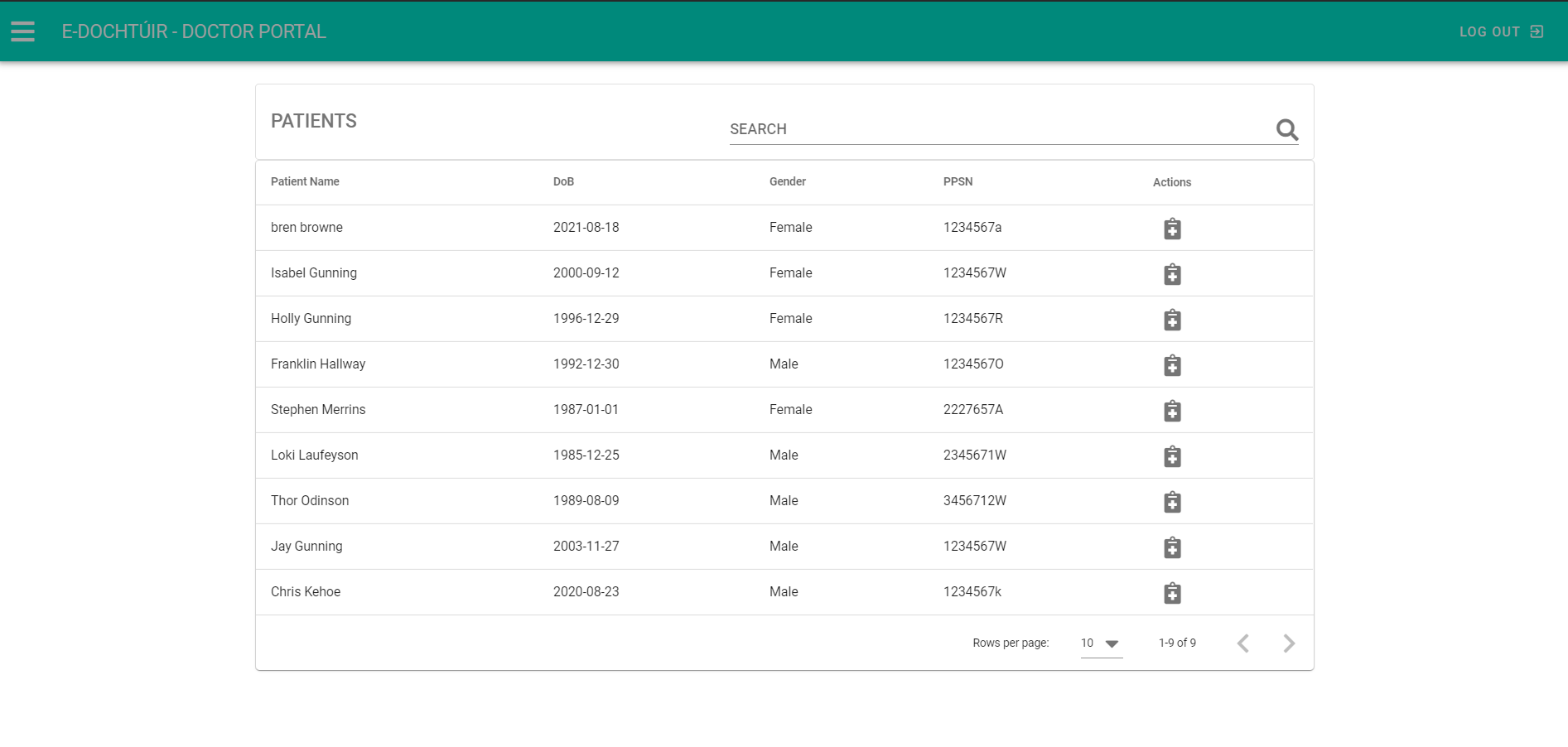


Figure 35: Patient Database Screen

The patient database screen allows a doctor to see a data table of all the patient users of the application. They can use the search bar to finite search for a particular patient, or they can sort the table by patient name, date of birth, gender or PPSN. Each record has an action field. Selecting the clipboard icon in this field will open a full-page dialog of the patients’ medical record, as previously mentioned in the medical record screen section of this document. At the bottom of the data table, the doctor can limit the number of patients to 5, 10 and 15 per page or load all of the patients to the screen. The arrows allow the doctor to navigate through the patients if they are limited in number on the page.

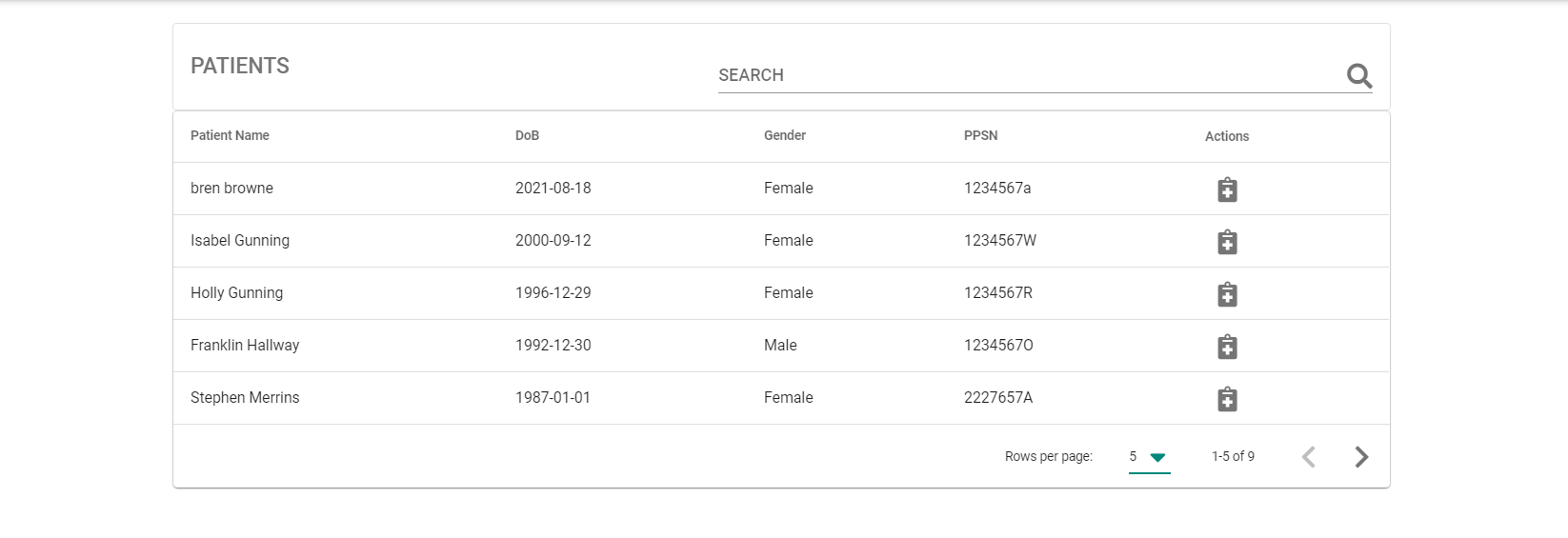


Figure 36: Data Table Showing Reduced Number of Patients

### 3.1.10. Navigation Drawer Screen (Doctor)

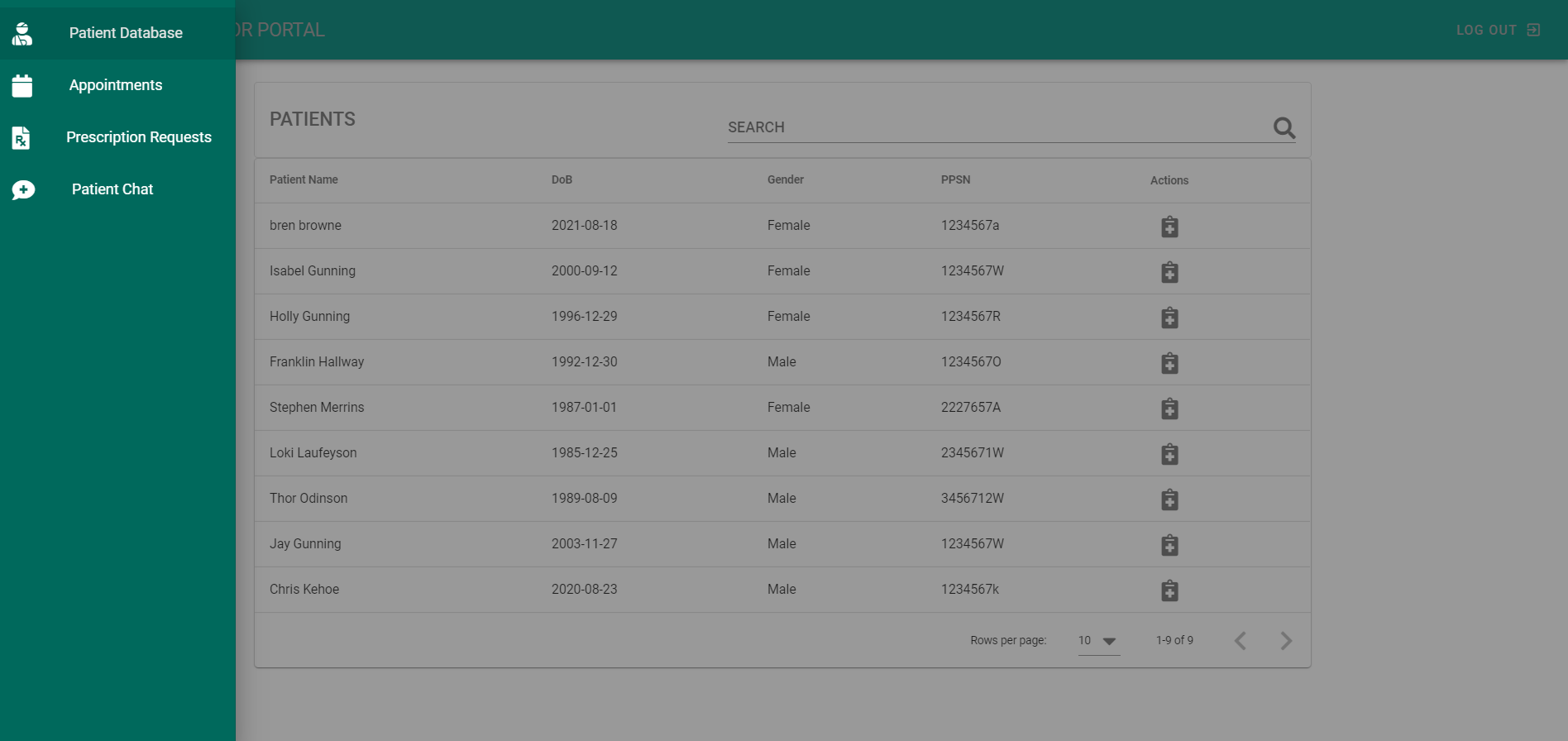


Figure 37: Navigation Drawer Screen (Doctor)

The navigation drawer screen shows the navigational bar for the doctor on the left-hand side of the screen. As is the case with the patient navigation drawer, this drawer is hidden until the three-line button is pressed and disappears when the doctor selects the page or navigates to a new page.

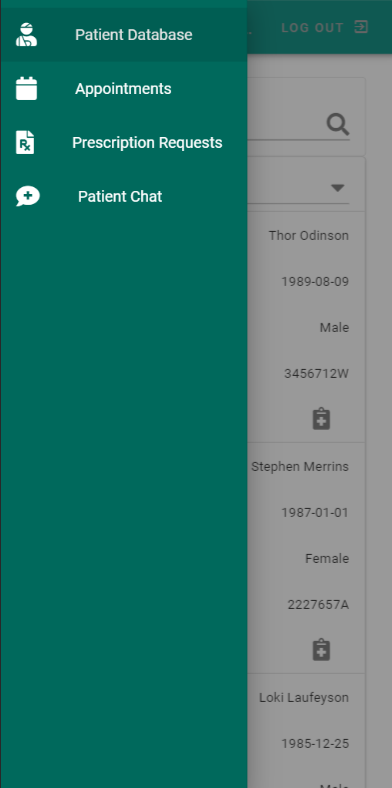


Figure : Navigation Drawer on Mobile

### 3.1.11. Appointments Screen (Doctor)

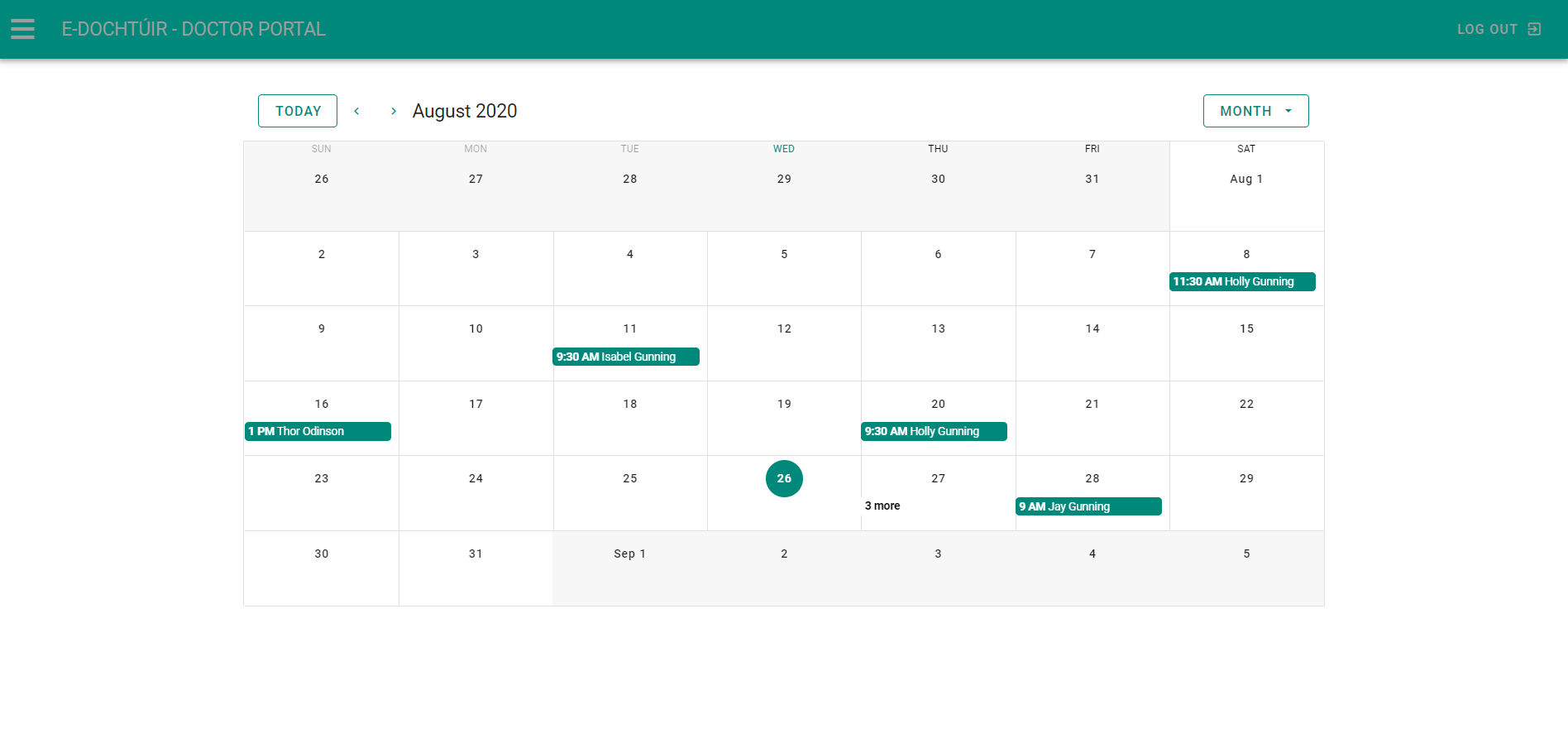


Figure 39: Appointments Screen Doctor

The appointment screen on the doctor side of the application allows the doctor to view all appointments of which a patient has booked with them. Much like in the calendar in “My Profile”, selecting the appointment shows more details about the appointment, such as the patients name, the type of appointment and the description. Selecting the date of the appointment will display the daily schedule for the doctor and show them the time slots for each appointment.

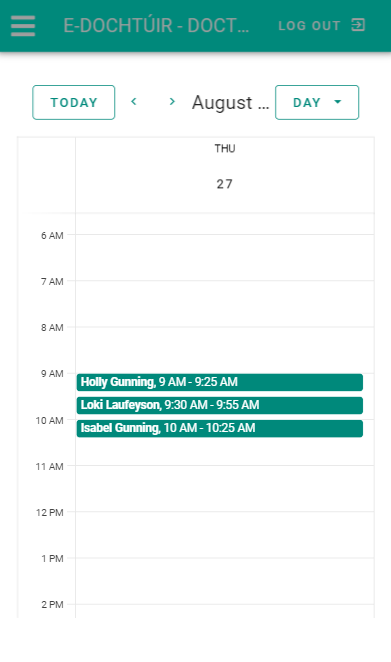
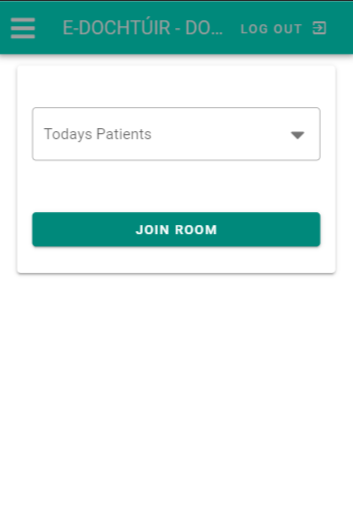
The appointments for this calendar are directly populated by querying the “appointments” collection and pulling back any records whose “doctorID” corresponds to the currently logged in doctor.

Figure 40: Appointment Day Schedule on Mobile

### 3.1.12. Patient Chat Screen (Doctor)

 A screenshot of a computer screen

Description automatically generated

Figure 41: Patient Chat Screen on Mobile Figure 42: Chat Created by Selecting Patient

The patient chat screen allows the doctor to create a “room” with a patient so that the doctor and patient can communicate online. Any patients who have booked an appointment online and whose appointment is for the current date will be filtered into the drop-down menu. When the doctor selects the “Join Room” button, a room is created by taking the doctor and patients’ IDs and storing them in a new document in the “rooms” collection.

By creating a room with just these IDs it allows the room to stay private from other doctors and patients. Messages are passed into a messages array and are stored in the room record in the “rooms” collection. When the doctor has concluded their chat with the patient, pressing the “Close Room” button will delete the room record by finding it’s corresponding record ID in the “rooms” collection, thereby deleting the instance and all messages along with it. Once the room has been deleted, the close room card will revert back to the join room card on the screen.

### 3.1.13. Admin Screen

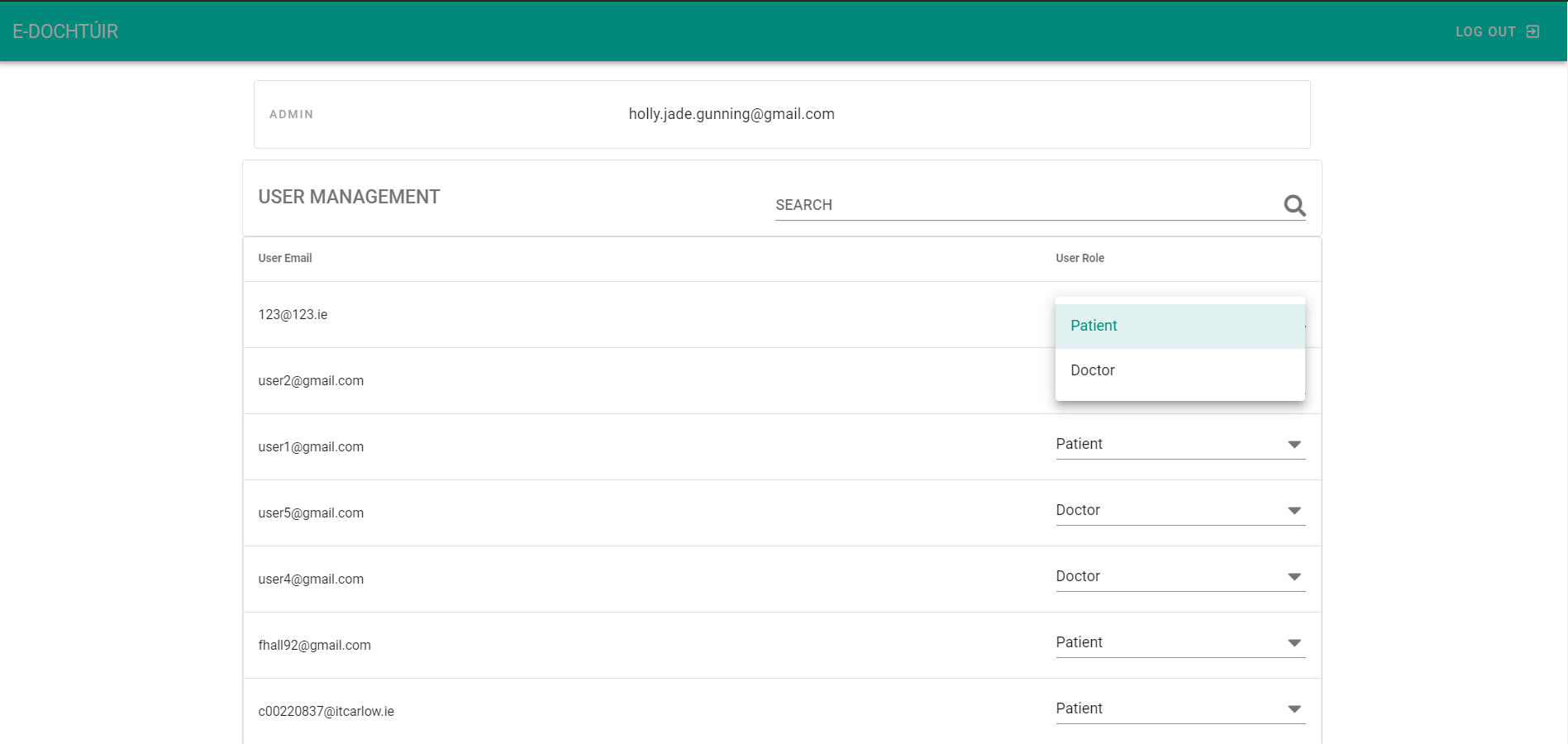


Figure 43: Admin Screen

The admin screen is a single page dedicated to an admin role; whose sole role is to change the predefined role of patient to doctor where applicable. The screen shows the user their admin credentials and a data-table which populates all non-admin users along with their current role in the application. This screen was designed solely so that once a clinic has vetted the medical professionals employed by them they can have them register on the application and then re-define their role to that of a doctor on the application.

On the data-table, the admin can search for a specific email using the search bar. To change a user from patient to doctor, they simply select the drop-down menu and select doctor. Doing so fires off a cloud function call to update the role of that user in the “roles” collection in the database.

### 3.1.14. Track Medication Screen (Patient)

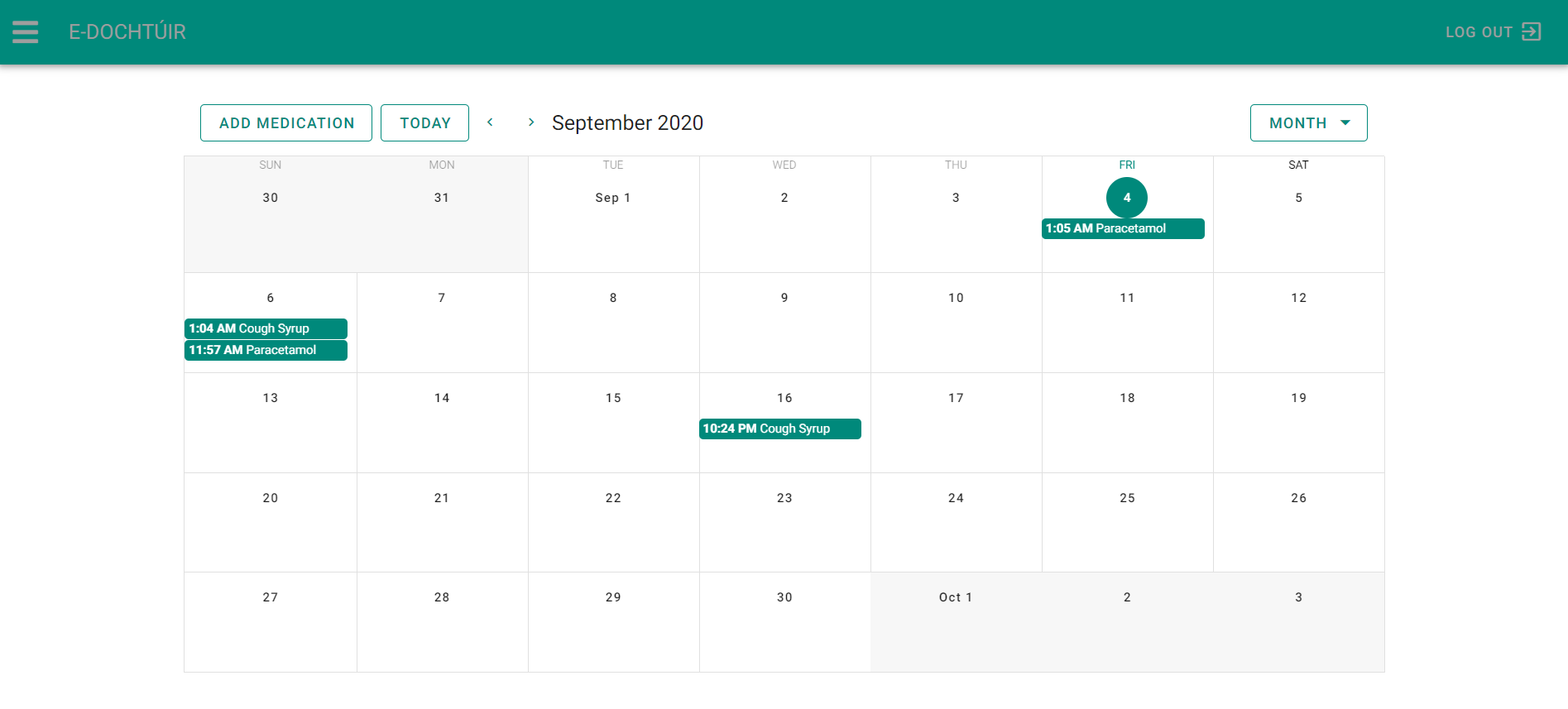


Figure 44: Track Medication Screen

The track medication screen allows patients to keep track of any medication they take in a more visual format through the aid of a calendar. They can access the specific date to see the time that they took the medication. If the patient presses the “Add Medication” button on the top left of the screen it opens a dialog box with the medication form to add to the calendar.

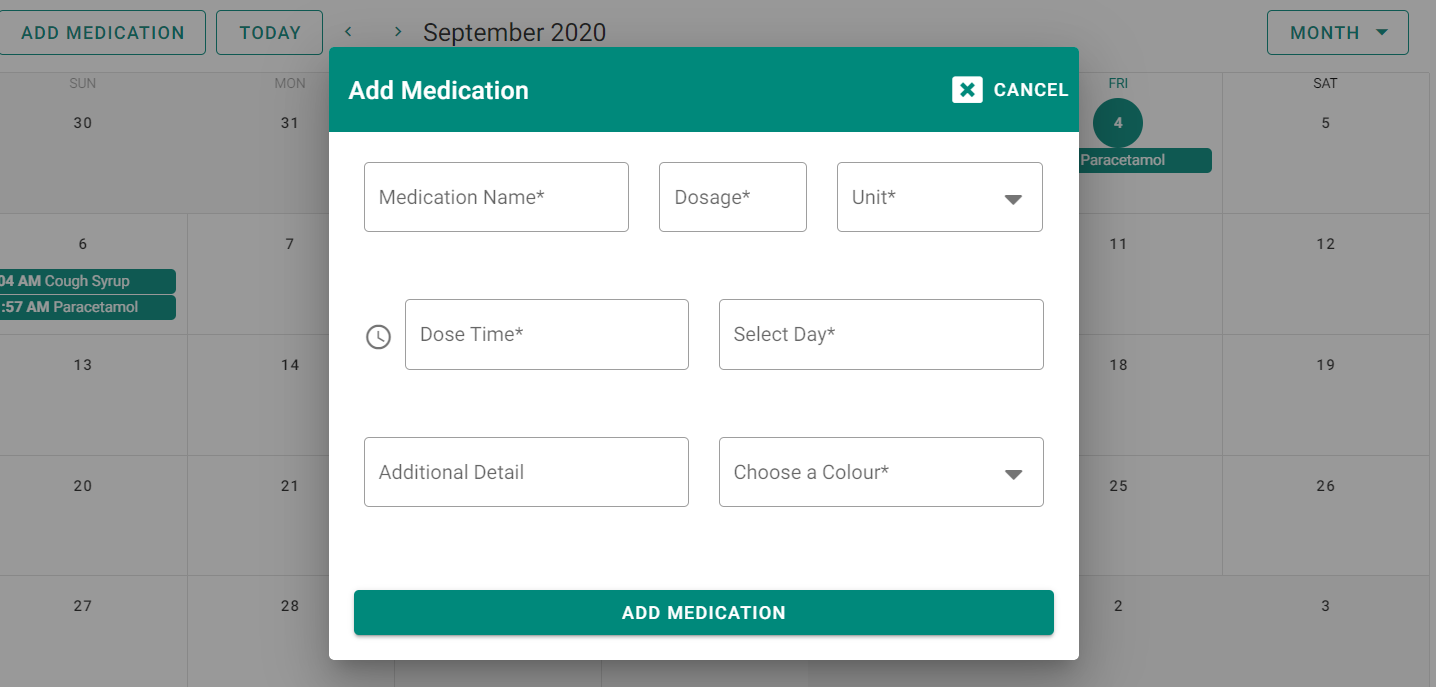
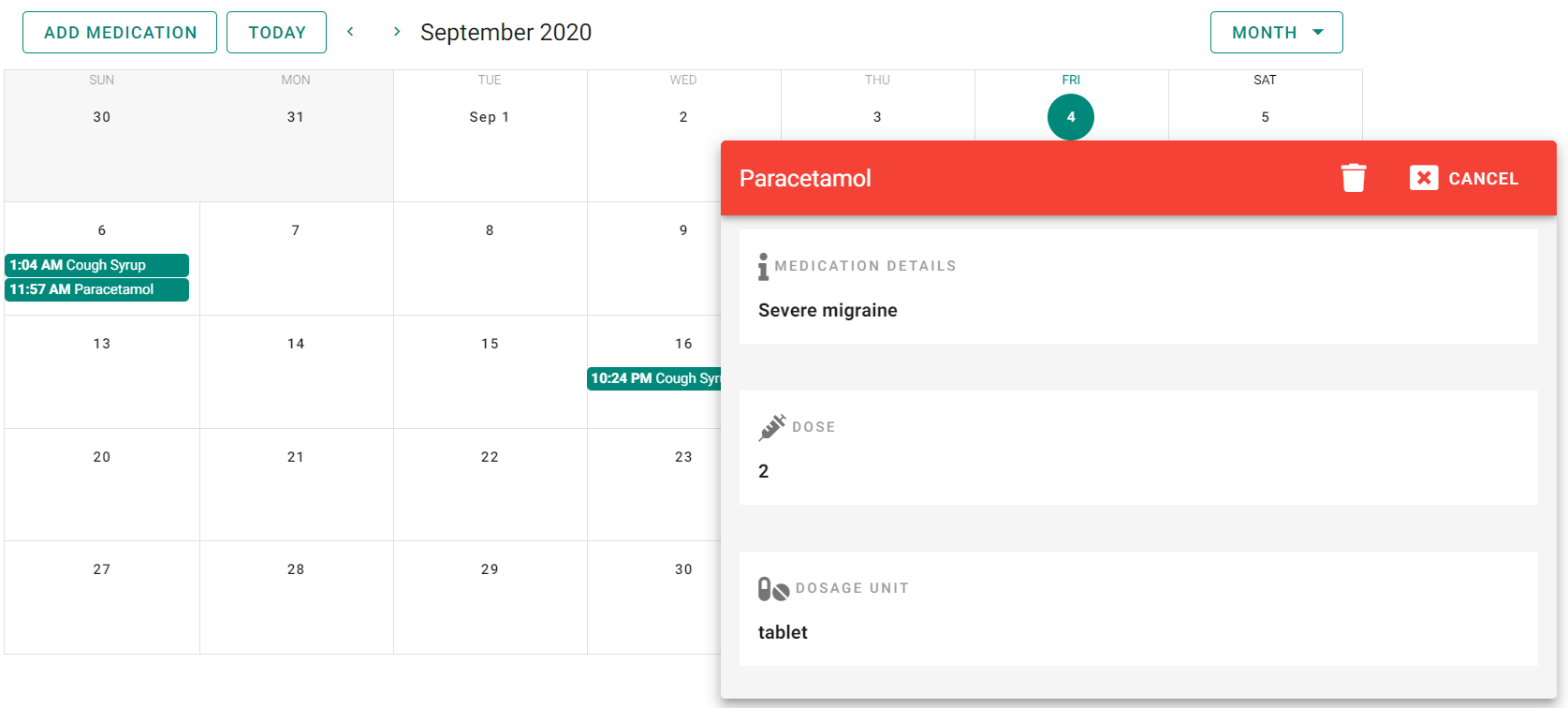


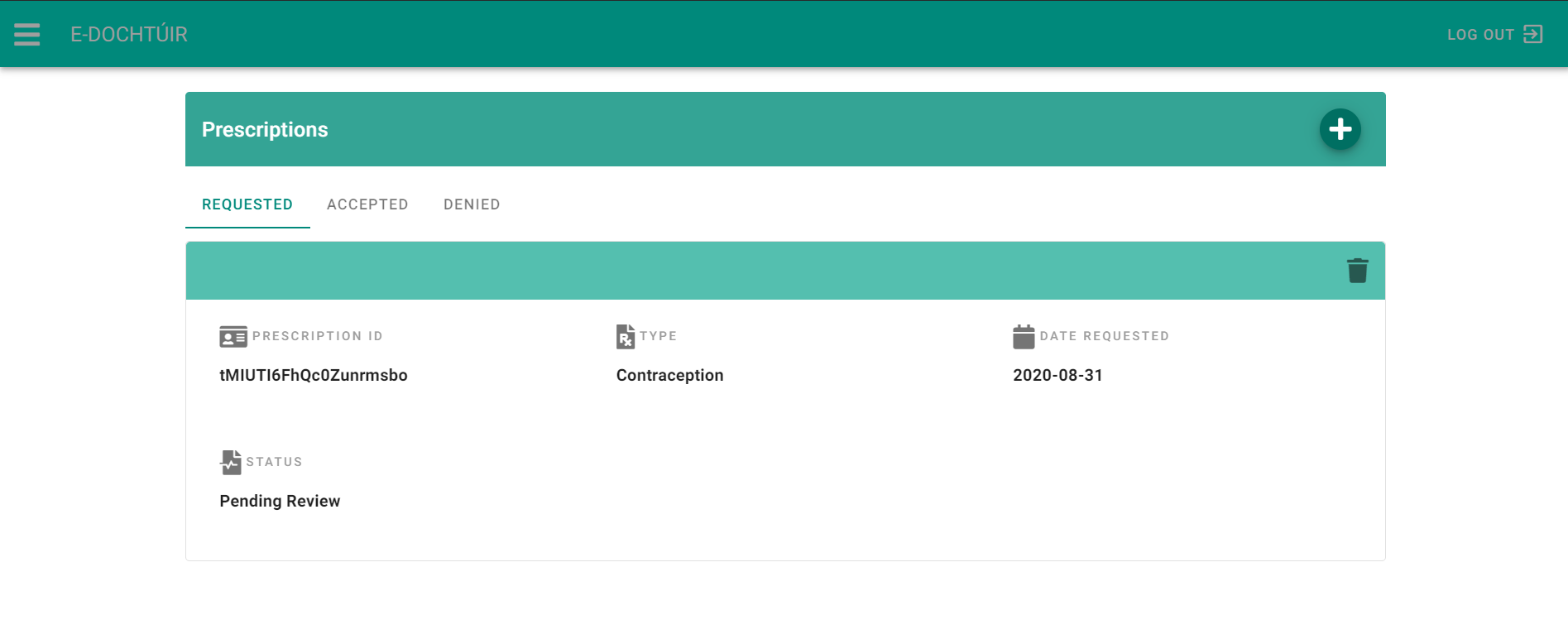
Figure 45: Add Medication Dialog

The medication form includes information such as medication name, dosage, unit, dose time, select day, additional details and a colour before submitting the medication entry. The entry is then saved in the “users” collection within a medication array. Each separate medication record is then fed into the events array so be displayed on the calendar.

Figure 46: Medication Information Menu

The patient can select the entry on the calendar, and it will enlarge a menu box with the information. The patient has the option to delete the entry, which will clear it from the calendar and the array in the collection.

### 3.1.15. Prescriptions Screen (Patient)

Figure 47: Prescription Screen for Patient

The prescription screen allows the patient to view any appointments they have requested as well as view any that have been accepted or denied, in ascending order. If the patient wishes to make a new prescription request, pressing the “+” button on the Prescriptions banner will open up a page dialog that will allow them to pick from a choice of prescriptions.

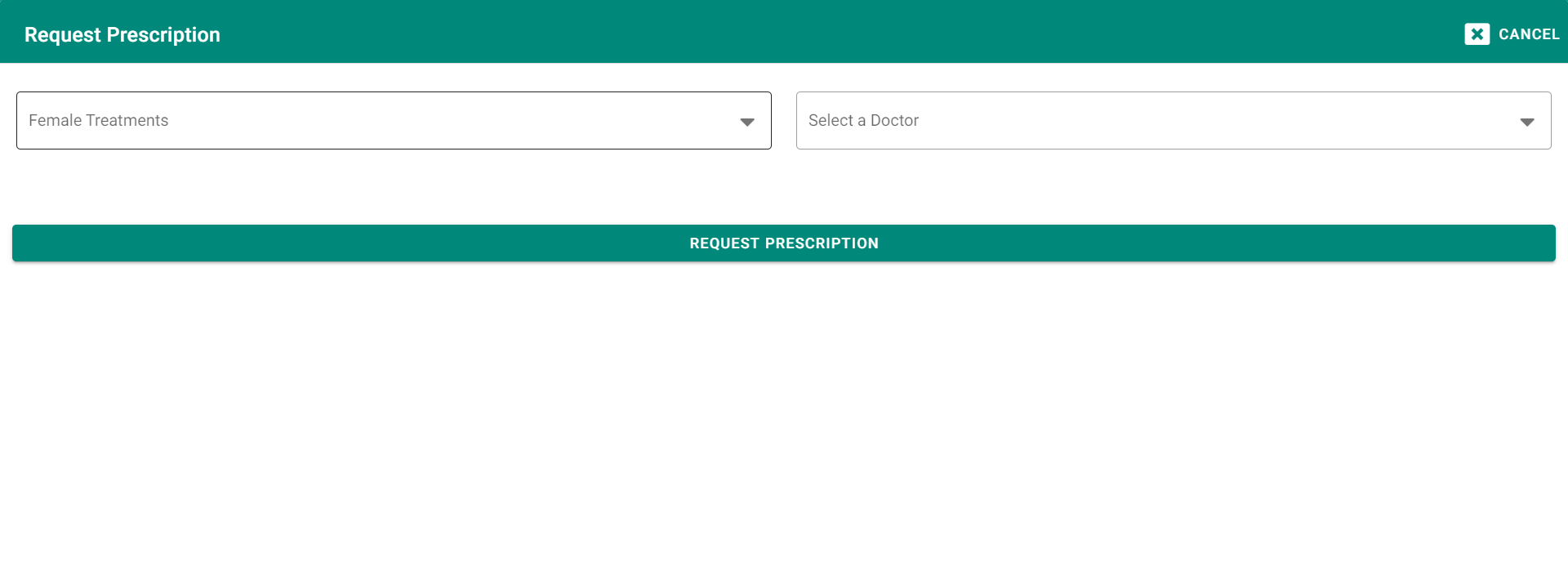


Figure 48: Prescription Request Form Pre-Treatment Selection

The patient then has access to a treatment selection box, that is determined by the gender that the patient selected in their Medical Record screen. They can select from a list of treatments and select a doctor to request the prescription from. Depending on the treatment selected, a form corresponding to that treatment will be shown to the patient once they have also chosen a doctor. For instance, if the patient selects Contraceptive Pill & Patch, the form for that request will be shown.

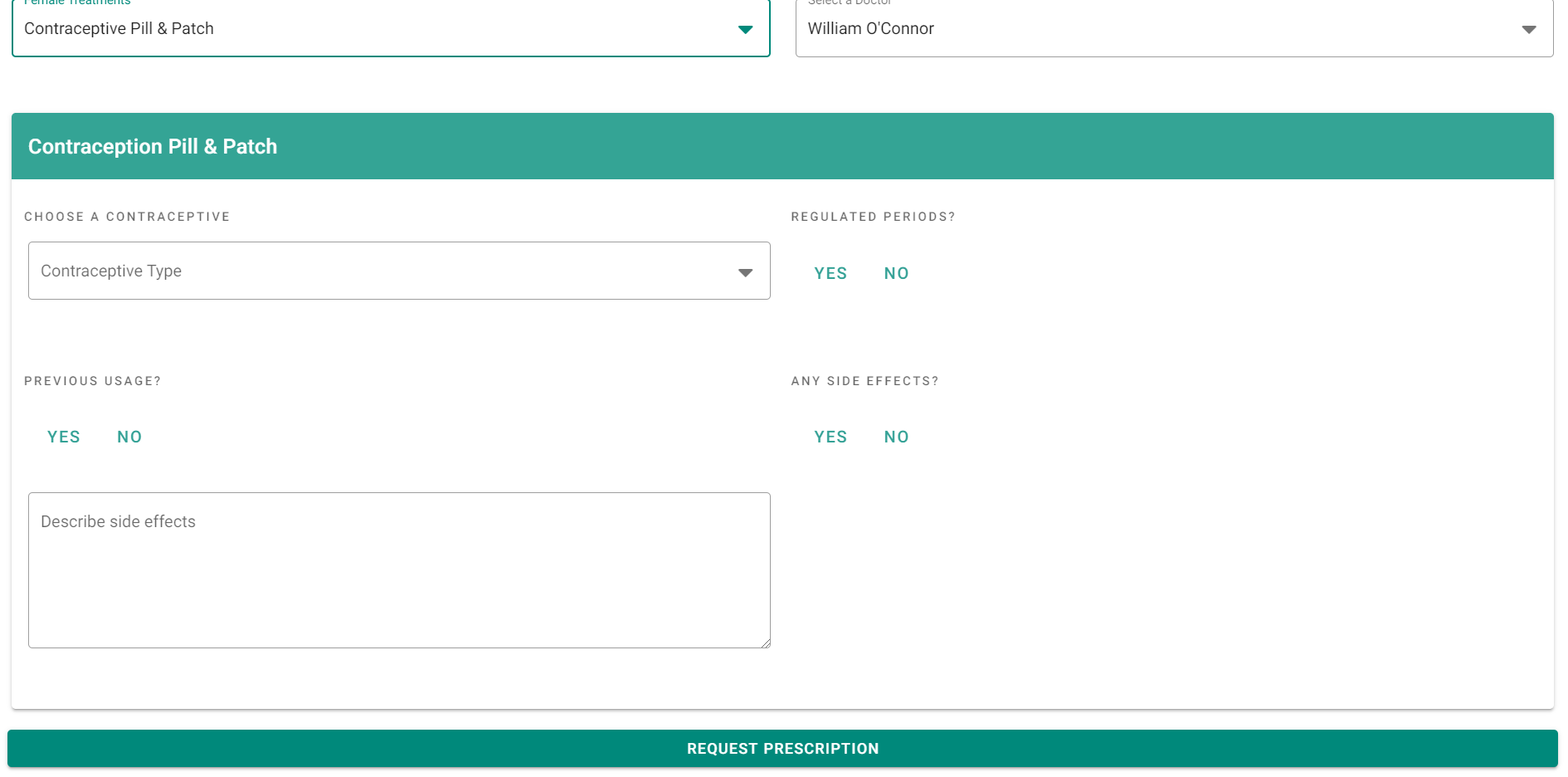


Figure 49: Contraceptive Pill & Patch Request Form

Once the patient has filled in all the appropriate information and submitted a request, the dialog will be closed, and they will be returned to the view prescription page. The prescription request is sent to the prescriptions collection in the database. The patients’ ID, as well as the date of request and a status of “Pending Review” are also additionally added to the record, so that the doctor can see all pending requests in order of the date requested and the patient. Patients may not make additional prescription request whilst there is one of that prescription types is pending.

### 3.1.16. Prescription Requests Screen (Doctor)

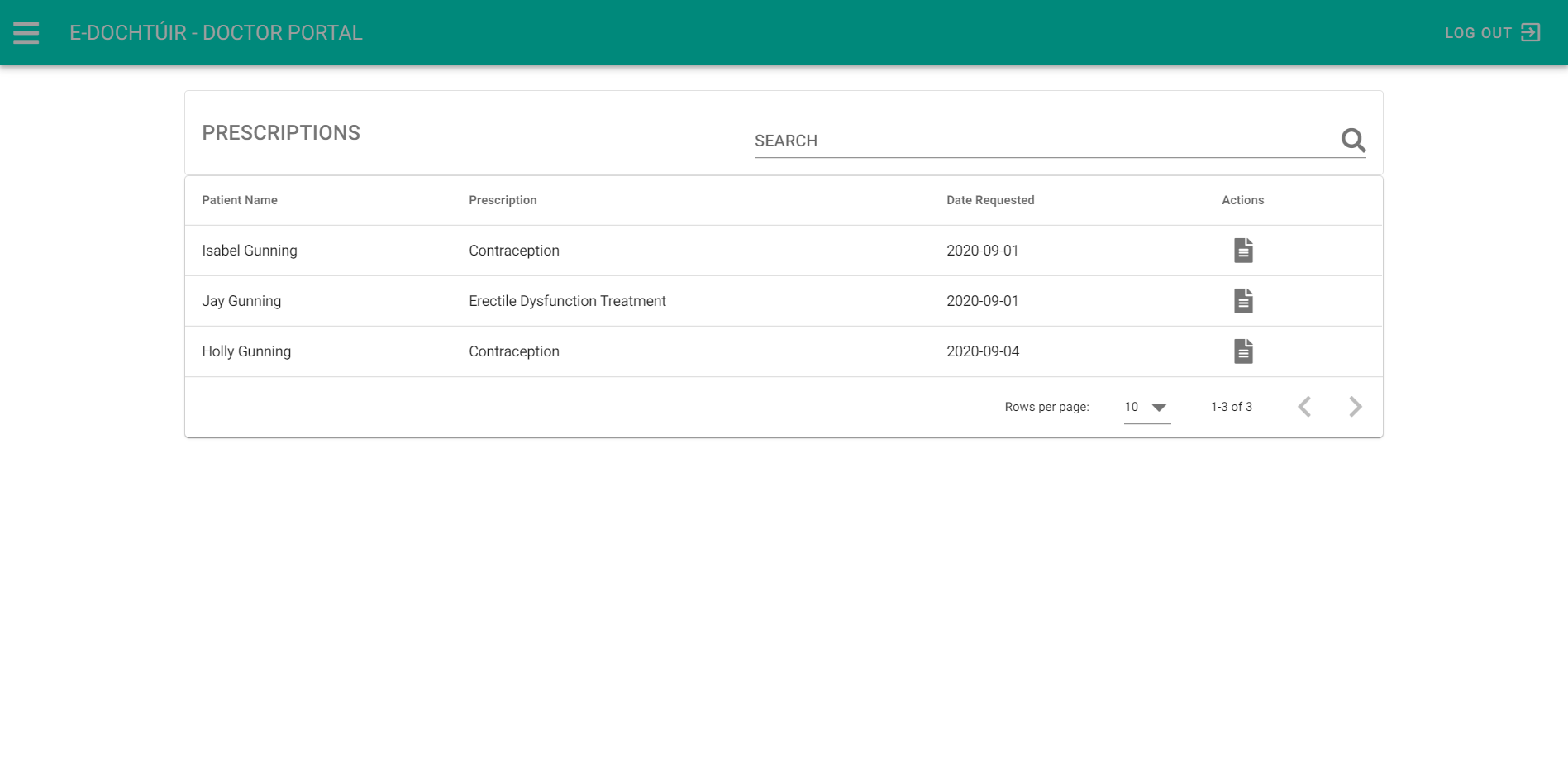


Figure 50: Prescription Screen for Doctor

The prescription request screen allows a doctor to see any prescriptions that have been requested of them by a patient in order of date requested and the name of the patient. Depending on how many requests there are, the doctor can search for specific information, or filter the prescriptions by name, date or type of prescription requested. When the doctor selects the action button next to a record, it will open the specific request form, based on the chosen type of prescription, and populate it with the necessary information.

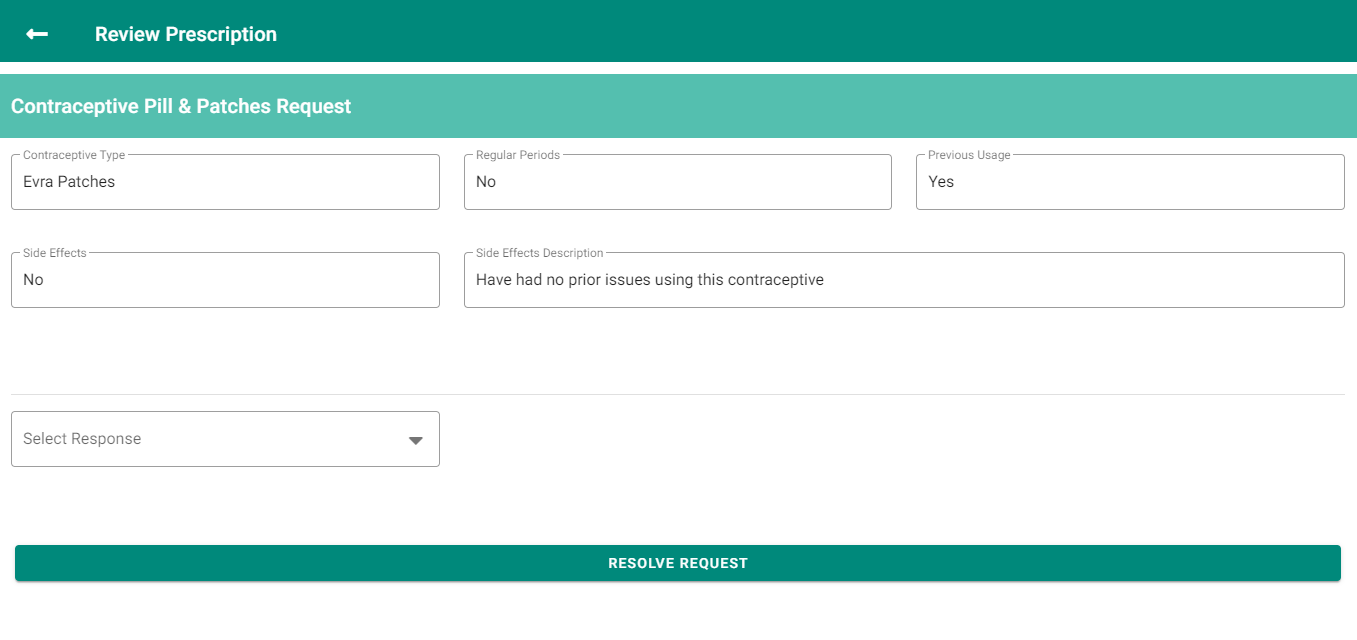


Figure 51: Example of Prescription Form

From this form, a doctor is able to see any information they might need to make a decision to accept or deny the prescription. If the doctor chooses to accept the prescription they are then shown a prescription upload box. When pressed this will give them the option to select a prescription from their computer and upload it with the record IDs name.

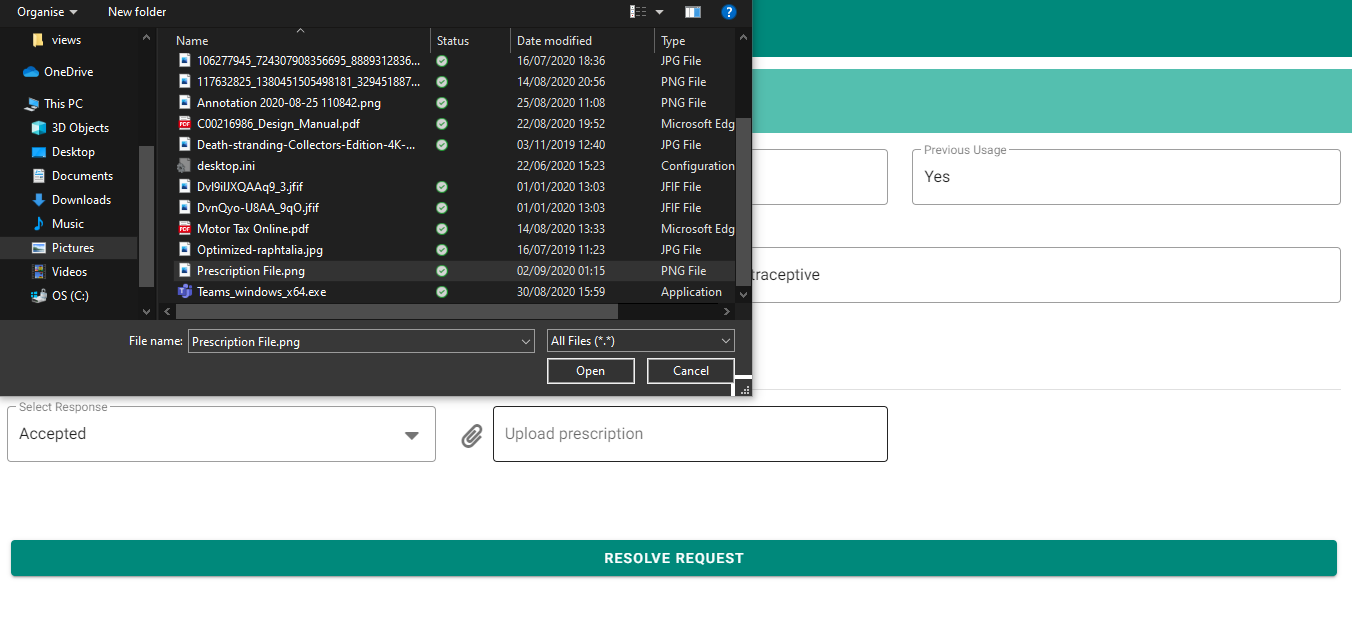


Figure 52: Upload Prescription on Accepted

If the doctor feels that the form answers convey that the prescription is unnecessary or the patient needs to first get checked up in person, the doctor can deny the request with a reason.

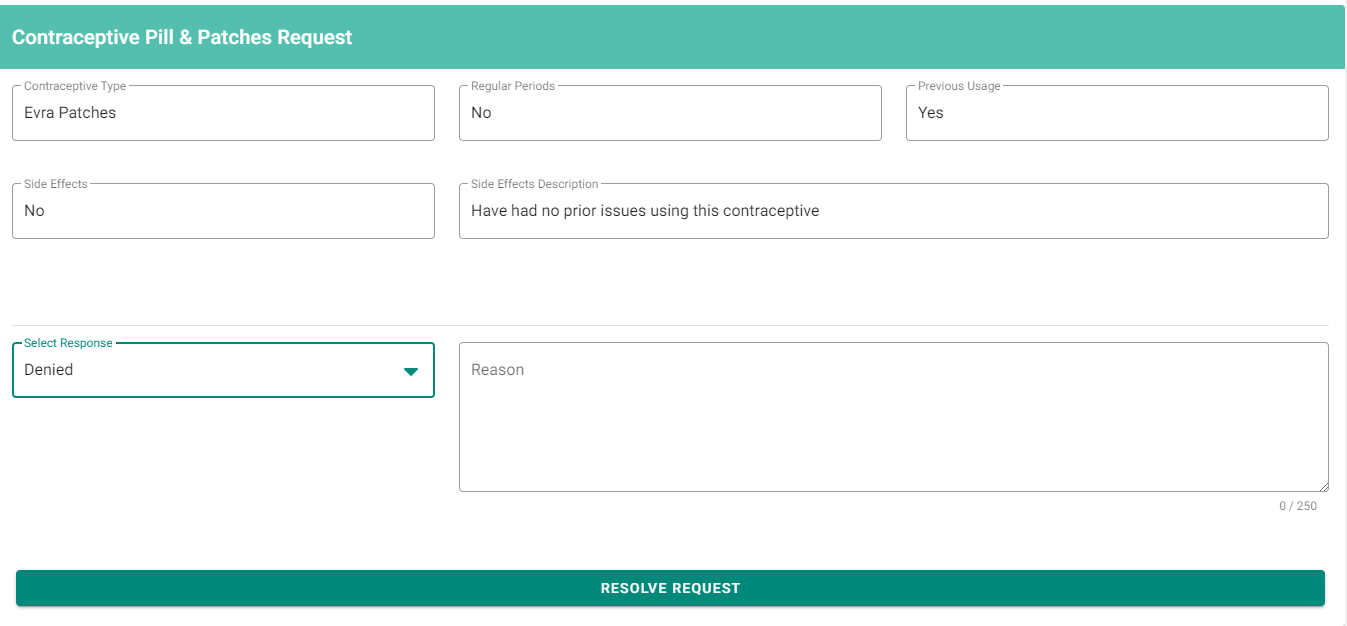


Figure 53: Prescription Denied

Once the doctor has responded to the prescription, the prescription dialog closes and the doctor is back on the prescription request data table, where the request responded to is cleared from the data table and the patients’ prescription is moved to either the accepted or denied tab in prescriptions view.

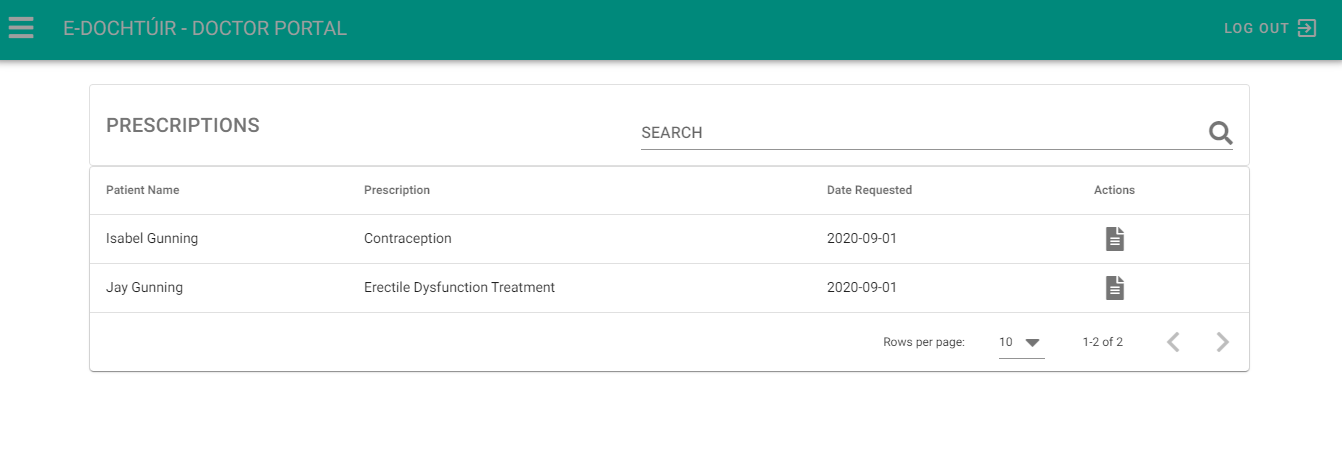


Figure 54: Request handled no longer in data-table

# Adherence to Specification

## General Specification

### Register, Login & Forgot Password

As previously outlined in the functional specification, any user of the application can log in using their email and password and will be redirected to the application dashboard. If the user does not have an account then they can select the ‘Create Account’ option at the bottom left of the login page. This will alternate their view to the register page where the user can enter a number of different details pertaining to themselves, including an email and password which will become their application login credentials. Meanwhile, if the user has forgotten their password, they can select the ‘Forgot Password’ option to the right of the login screen under the password field. This will open a dialog box which will allow the user to enter their email address. Provided the email address exists, an email will be sent to it with a link for the user to change their password.

## Patient Specification

### User Profile

Patients of the application can go to their user profile, view the information they provided when creating the application and make edits to the information, excluding the patients email as it is their authentication credential. They can view all of their upcoming appointments from here on a small-scale calendar. There is also a dark mode button to control the theme of the application if the patient finds the light theme too hard on their eyes.

### Medical Record

Patients are able to store their medical information on the application in the medical record section, allowing them to keep a portable record for easy accessibility. The patient can add and make changes to their gender, blood type, weight and height and add records about their allergies, conditions and any immunisations they have had.

### Medication Tracking

Patients can track any medication they ingest through this feature. They are provided with a calendar. On the top left is a button which will allow them to add the details for a medication as well as the date and time it was taken. This record will then be populated to the calendar in the appropriate date and time slot, as seen when the patient goes to the specific day on the calendar. If they wish, they can delete the entry also.

### Appointments

Patients have access to an appointments section within the application. In this section they can view upcoming and past appointments. On the Appointments header bar, a plus icon acts as the ‘Book Appointment’ button. When selected, it expands a book appointment dialog form. The form consists of two drop down accordions. The first accordion contains the personal details of the patient making the appointment while the second which is the default open accordion contains the form fields for booking the appointment. The patient can select the appointment type, a doctor a date for their appointment, details and an appointment time. When the patient successfully books an appointment, it will populate in their upcoming appointment tab in order of date.

### Prescriptions

Patients are able to view their prescriptions, pending, accepted or denied, and can request a new prescription. When requesting a prescription, they may only request one of every available type whilst prescription is pending. The treatments available are populated depending on whether the patients’ gender is female or male. If no gender is recorded then the patient will be prompted to redirect to their medical record to supply this information. Depending on the treatment selected, a form specific to that treatment will be displayed to the patient, but only after the patient has selected both a treatment and a doctor to request the form from. Once the form has been submitted, the patient will be able to view it pending in their prescription view. On acceptance or denial, this record will be moved to the respective tab.

### Message Functionality

Patients who have booked an appointment and have selected to have an online consultation can avail of the online chat functionality within the application. On the date of the appointment, the doctor will instigate the creation of a chat room, where both the doctor and the patient can talk privately with one another through text message. This form of communication is suitable for queries that do not necessarily require face to face interactions.

## Doctor Specification

### Appointment Schedule

As previously outlined in the functional specification, doctors have access to an appointments section within the application. Within this section, they have a calendar that acts as a real-time scheduler, populating appointments of patients pertaining to them. These appointments are updated in real-time as patients book their appointments. The doctor can also see information about the appointment and click on the specific date to open up a time breakdown where they can see the time slot for the appointment.

### Prescriptions

Doctors who have received a prescription request from a patient can see this request populated in their prescription request data-table. The table provides them with the name of the patient, the prescription type and the date it was requested. The doctor can then review every individual request and either deny the request with a reason or accept it and send a prescription.

### Message Functionality

Doctors who have received a booking from a patient with the appointment type “online consultation” have the ability to chat to a patient on the booked day. A drop-down menu is available that populates with all patients that have appointments on the current day. Once a patient is selected the window with the chat box opens and the doctor can exchange messages with the patient.

# Deviation from Specification

## General Specification

### Register, Login & Forgot Password

Originally it was outlined that the registration of a medical user would differ from that of a patient. Due to a lack of information on the registration process of medical professionals in Ireland, there was a deviation from this specification. Rather than have an individual registration for user types, all users currently register through the same form. In order to sort user roles, once a doctor has registered, an admin can identify the validity of the user’s medical credentials through whatever means medical professionals are usually identified. They can then log in to the Admin Portal, where they have access to a database of all registered users of the application. From there they identify the user and update their role from patient to doctor within the application.

## Patient & Doctor Specification

### Medical Record & Patient Database

Every aspect of functionality stated originally was completed to specification. However, during development, it became clear that expanding this functionality to be accessible by the doctor would be greatly beneficial to both parties. Pulse, blood pressure, blood glucose levels, and cholesterol fields have been added to the patients’ side but are read-only fields. On the doctors’ side, a patient database page has been added where the doctor can view every patient associated with the clinic. On each record an action field containing a clipboard icon is used to access the patients’ medical record. All fields of the medical record for that specific patient are made visible, except for gender which is read-only. In implementing this, a doctor can make real time updates to a patient’s medical record during an appointment. Another example would be if the patient has been waiting for their cholesterol results. Upon arrival of the results the doctor can input them into the patients’ medical record and these fields will immediately update on the patients’ side.

Upgrading the functionality of the medical record feature was a logical step forward in making the application more well-rounded for both the patient and doctor and reducing the need for extra contact if the patient does not desire to discuss the results.

### Message Functionality - Video Call Consultations

Originally, it was intended that there would be multiple forms of communication between the patient and doctor. Whilst a text chat was implemented, a number of problems with the open-source technology chosen for video call implementation, WebRTC, made the development of this feature problematic. The implementation was at a local state and could pick up sound and video. There were problems with the sound where if headphones were not in use, a feedback loop was created which in turn produced a loud pitched echo which was painful on the ears and made it impossible to hear anything coherent. The problems with the technology, as well as the lack of a stable internet connection when working from home during the last few months, determined the difficult decision to shelf this feature indefinitely.

# Adherence to & Deviation from Design

Regarding the overall design of the application, most of the design manual was adhered to, with the exception of minor tweaks in field widths, dictated by the grid system of Vuetify. In places, such as in the medical record and appointment view cards, icons were added to the pages. It was felt that these small additions added a cleaner look and feel to the pages. Any additional features or pages kept in line with the original design in order to make the application seamless, rather than disjointed. Additional screens were added, such as the patient database screen, which adhered to the overall design themes.

# Issues Encountered

Below are a number of issues encountered during the project.

## Covid-19

During the course of the college year, the country went into lockdown due to the Covid-19 pandemic. This meant that the college was no longer operating in a physical capacity. In comparison to the work like atmosphere found on campus, working from home was much more difficult. The household consists of six members, three of which are a toddler and two young teenagers. It was difficult to find the space to work from that was quiet and this made it difficult to concentrate in general.

## Internet Issues

Another huge disadvantage to working from home besides the lack of space is the lack of facilities. Due to being situated in the rural Midlands, there is a distinct lack of internet services. No broadband companies will supply a wired internet connection and due to the house being positioned in a dip in the land, there is not a high enough vantage point to avail of a satellite broadband. The only form of internet connectivity available is in the form of a mobile sim, which delivers 2 mbps at its’ best and frequently loses service most of the time.

Considering the online nature of this application, it made it very difficult to progress in a number of ways. Firstly, it made accessing any online documentation difficult, which was required to understand how to proceed with technologies and frameworks due to not having any prior experience with these. Secondly, the application works of an authentication system and also requires the Cloud Firestore database to pass and receive data for the entire application. Without the internet this was virtually impossible. The application would often read false negatives because it was not computing properly, or the database was not responding fast enough. Deployment was also an issue with the application often failing half-way through or not uploading all of the necessary files.

These slow speeds coupled with the fact that the two teenagers in the house were also required to try and be online to receive their school work and stay in contact with their teachers made it difficult to maintain any kind of feasible working connection for the majority of this project.

## Insufficient Research

One of the biggest problems with the project was some of the initial oversights with aspects of the research document. These oversights wasted a lot more time than was necessary going down the wrong routes with regards to development. This is expanded on more in the reflections segment of the document.

# Learning Outcomes

From conceptualisation to development, the final year project provided a great learning experience, both with regard to what is required to develop a full functioning application, as well as the learning curve and upscale that derived of working with tools and technologies not formerly known.

## Technical

### IDE – Visual Studio Code

During the development of the application, visual studio code was explored and used as the IDE of choice. This IDE had not been previously used until 4th year, as prior experience to IDEs consisted of JCreator, Eclipse and, briefly, Visual Studio. VS Code was an ideal choice for this project. It was straightforward and easy to use, setting up a link to a GitHub repository was simple and it also features many great extensions which can help with improving the visual quality of code as well as providing syntax and error support.

### VueJS Framework & Vuetify

As mentioned in the additional research of this application at the start of the document, using a framework was a completely new experience and challenge. During the course of the college, none of the modules touched on the use of frameworks, the benefits to their use or any tutorials on how to use them. All of this knowledge was procured within the final month of development after realising the full extent of their capabilities. The framework has support for progressive web application design, including the creation and maintenance of a service worker. It also allows for navigation security throughout the application, both by incorporating firebase authentication to stop non-user access as well as to keep the pages specific to patients and doctors private from one another. VueJS also comes with a useful GUI tool for handling localhost serving as well as application builds, accessed by simply calling Vue ui in the terminal. Using this framework allowed for the application to be developed as a single page application, rather than having to page load or refresh on pages after each navigational change.

Not only does the framework provide a multitude of development options, it also comes with the option to use Vuetify, a UI component and classes library, based on Googles’ Material Design philosophy. Learning to utilise this library made it possible to create an application that scaled smoothly between large device screens like desktops to smaller ones like mobiles and tablets. Whilst there is documentation on how to use these libraries, the documentation provided concentrates more on the different props that a component can use to change the functionality of say a date-picker, from single date to range date, rather than full scale implementation instructions. This made it difficult and slightly time-consuming to learn the nuances of each component with regards to its limits and use.

### Firebase

As was the case with embracing the framework, the firebase platform was another technology that had never been used before the project. A number of the tools offered on this platform were utilised. Firebase authentication made it possible to register a user to the application and provide that user with a way to securely log in, as well as change their password if they had forgotten it. The auth state observer allowed for any authentication changes with the user to be tracked and the necessary actions taken depending on their state. Implementing these features was relatively simple as was the creation and handling of data through firebase Firestore. This real-time database allowed for the application to pass data to and from patient to doctor dynamically with little to no delay. Patients could make changes to their medication record and see those changes instantly as a result. The query capabilities of firebase made it possible to implement a vast majority of the application. The online documentation was extensive, all be it a little confusing at times but thanks to the ever-growing online community, any issues were soon resolved. Cloud storage made it possible to handle the uploading and receiving of documentation between doctor and patient.

## Personal

### Time Management

Time management is a skill that was greatly lacking at the start of this applications development. Due to a number of contributing factors, mentioned in later reflections, this was a skill that required the most work to develop. By the end of the project, there was a noticeable improvement in productivity. For instance, at the start of the project the research manual took a month to write up. At the time of writing this final document, which is larger is both page and word count, this took less than a week to write, reducing the time of documenting work by three quarters the original time.

### Coding Skills

In part, due to time spent on travel and assignments as well as other commitments, no individual projects had ever been taken on outside of those given by the college. In general, this factored greatly to the difficulty in which was faced with developing a full application as one person. This lack of experience greatly shows in the earlier stages of development. One of the biggest learning outcomes and lessons of this project was perhaps not the development of a working application, but rather the experience gained from making a lot of mistakes and learning how to rectify those mistakes. This experience has proven that the best way to learn to code and develop projects, is not through the theory gained in college, but rather the amount of time and effort that is put in outside of college to apply this theory as well as learn and experience different technologies and tools.

# Reflections

Having completed this project, were the chance to start again presented, there are a number of elements that should have been undertaken in a different manner.

## Time Management

Throughout the year, there were difficulties keeping up with the project on top of all the other course modules. On average, roughly four hours a day were spent travelling to and from the college, which greatly reduced the amount of time that could be spent on assignments and project work. Looking back, whilst there were some assignments that took longer due to difficulties grasping the concepts involved, there were others that more time was given to then necessary. If one piece of advice could be given, it would be to set up a schedule for each assignment and strictly adhere to it, only working on each module within an allotted time.

## Research

Many of the problems encountered during this project stemmed from overlooked concepts within the research document. One of the major mistakes made was that a considerably large portion of time was spent trying to get the application to work progressively (PWA) so that it could be downloaded just as a native app would download on iOS or Android. This was also the case with regards to getting the application to work fluidly as a single page application (SPA). Had more extensive research into frameworks been undertaken, it would have been apparent that a framework such as VueJS would have solved both of these issues at the same time. The amount of time wasted on these issues alone could have been time spent refining and expanding the concepts within the application.

## Coding

Perhaps the most important piece of advice that could have been given would be to start coding as soon as physically possible. During the course of the project, the research document, functional specification and design manual were all completed before any coding even began. Whilst research must be undertaken before the scope of the project becomes realised, picking possible technologies and using them immediately should have been a top priority. For instance, knowing that the project would have more than one user, if more in depth research into frameworks had been carried out, then the logical next step would be to attempt a number of tutorials to better understand those frameworks. In doing so, it would become apparent whether the technology would be appropriate for the development of the application. The biggest regret in this project is that too much time was spent on conceptualisation and not enough on actual physical development.

# Future Development

There are many features that could be integrated into the existing E-Dochtúir application in future development iterations, including both new features as well as refactorization of existing features to enhance the current application.

## Video Call Consultation

As mentioned above in the changes to design and specification, a number of difficulties were encountered, largely in part to the COVID-19 pandemic outbreak. As a direct result of this, some of the originally planned features were never successfully implemented. A feature that would have made this application that bit more accessible for both patients and doctors would have been the addition of video call consultation functionality.

In the future, with a more stable working environment, and either continued perseverance with the open-source WebRTC communication technology or further research into alternative forms of video communication, this feature should definitely be implemented in order to grow the applications user base.

## Refactor the User Role System

Due to the limited resources available it was difficult to determine how a real-life application would go about determining who is a doctor and who is a patient upon registration of a user. Though attempts were made to source this information from medical professionals, with the growing rate of the pandemic, it was decided that their time was better spent on real world matters then on communications for a final year project. Therefore, the solution for this application was to have every user register in the same manner and predefine their role as patient. A single admin user would have control over a database of all the users and could change the role of each user from patient to doctor where applicable.

In the future, this aspect of the application could be greatly improved by refactoring the current process. Further research would have to be done to identify the correct method of registering a medical professional. One possible method might go as follows. On the registration page a user could select between a patient or doctor registration type, which would then populate the registration form with the appropriate fields. The patient registration process would be identical to the current one; they fill in the form, their credentials are stored, and the Cloud Function assigns them a role of patient. As for the doctor, their form would consist of fields for name, date of birth, email, password and confirm password as well as an additional field for their medical ID or whatever identification confirms their role as a doctor. They will then be given a message and told that their credentials are being evaluated. An admin will see their credentials and perform the necessary checks to determine if they are in fact a registered medical professional.

## Expand Upon Medical Record Functionality

Whilst the Medical Record section of the application was perfectly designed to specification, future iterations might expand upon it to incorporate more in-depth features. For blood pressure, blood glucose and cholesterol all the patient really needs to know is there most up-to-date values. However, on the doctors’ side of the application, it might be valuable to store their previous values so that, say the patient has a new doctor, that doctor can view the patients’ medical history and catch up to speed on their requirements.

Information such as the position the patient was in or the measurement site that their blood pressure was taken at might also be of value to the doctor to ensure a more accurate reading or to ensure that a new reading is taken in the same manner as the previous one.

## Add in Notification Functionality

Another piece of functionality that would make the application more appealing to mobile users would be the addition of notifications. Specifically, if the patient or doctor were due to have an appointment with one another, if the appointment type were clinic, the application might send out a notification a day or so before the appointment and if the appointment type were online, the application might send a reminder an hour to thirty minutes before the appointment start time. This would make the application more accessible for those who do not regularly check their appointment calendar on the application.

# Acknowledgments

I would like to thank my project supervisor Dr Hisain Elshaafi for his support and guidance during the duration of my project.

I would also like to thank my father who made it possible for me to attend college and whos’ own medical history became a driving force for my inspiration for this project.

# Plagiarism Declaration

# Machine generated alternative text: Work submitted for assessment which does not include this declaration will not be assessed. DECLARATION *I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged. *I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside. *I have provided a complete bibliography of all works and sources used in the preparation of this submission. *I understand that failure to comply with the Institute’s regulations governing plagiarism constitutes a serious offence. Student Name: (Printed) ____________________________________________ Student Number(s): ____________________________________________ Signature(s): ____________________________________________ Date: ____________________________________________ ------------------------------------------------------------------------------------------------------- Please note: a) * Individual declaration is required by each student for joint projects. b) Where projects are submitted electronically, students are required to type their name under signature. c) The Institute regulations on plagarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook. INSTITUTE of TECHNOLOGY CARLOW



30/04/2020

C00214010

Holly Gunning

# Bibliography

## References

[1] blog, B., Busschere, P., REFERENCE&nbsp;, T. and AMERICA, T., 2020. All You Need To Know About Firebase. [online] Site name. Available at: <https://www.the- reference.com/en/blog/pieter-de-busschere/firebase.> [Accessed 3 September 2020].

[2] Firebase Open Source. 2020. Firebase Authentication Password Hashing. [online] Available at: <https://firebaseopensource.com/projects/firebase/scrypt/> [Accessed 3 September 2020].

[3] Firebase. 2020. Cloud Functions For Firebase. [online] Available at: <https://firebase.google.com/docs/functions> [Accessed 3 September 2020].

[4] Firebase. 2020. Cloud Storage | Firebase. [online] Available at: <https://firebase.google.com/docs/storage> [Accessed 3 September 2020].

[5] Vuejs.org. 2020. Vue.Js. [online] Available at: <https://vuejs.org/> [Accessed 3 September 2020].

[6] Netguru.com. 2020. 8 Reasons Why Vue.Js Is Worth Considering For Your Next Project. [online] Available at: <https://www.netguru.com/blog/8-reasons-why-to-use-vuejs> [Accessed 3 September 2020].

[7] Vuex.vuejs.org. 2020. What Is Vuex? | Vuex. [online] Available at: <https://vuex.vuejs.org/> [Accessed 4 September 2020].