

SOEN 390/F: SPRINT 3

REPORT (TEAM #14)

Software Engineering Team Design Project

Instructor: Dr. Yann-Gaël Guéhéneuc

Winter 2022

Samantha Guillemette (26609198)

Mohammad Ali Zahir (40077619)

Saleha Tariq (40006997)

Laila Alhalabi (40106558)

Hoda Nourbakhsh (40066450)

Tushar Raval (40124664)

Quang Tran (27740654)

Marwa Khalid (40155098)

Steven Markandu (23740137)

Submitted: March 16th, 2022

1.0 INTRODUCTION	4
1.1 Purpose	4
1.1.1 System	4
1.1.2 Document	4
1.2 Targeted Users	4
1.3 Targeted Readers	5
2.0 PROJECT DESCRIPTION	6
2.1 Sprint Schedule	6
3.0 REQUIREMENTS	7
3.1 Backlog	7
3.2 User Stories	8
4.0 RELEASE PLANNING	11
4.1 Sprint 3 Summary	11
4.2 Sprint 4 Planning	12
5.0 ARCHITECTURE	15
5.1 High-Level Overview of System	15
5.2 Use Case Diagrams	16
5.2.1 Use Cases related to the Admin App	16
5.2.1 Use Cases related to the Client App	23
5.3 Domain Model Diagram	27
5.4 Database Design	28
5.5 Physical Design of the Database	30
5.6 Sequence Diagrams	32
6.0 RISK MANAGEMENT	37
6.1 Purpose of the Risk Management Plan	37
6.2 Risk Management Procedure	37
6.2.1 Process	37
6.2.2 Risk Identification	37
6.3 Risk Analysis	39
6.3.1 Qualitative Risk Analysis	40

6.3.2 Quantitative Risk Analysis	40
6.4 Risk Response Planning	40
6.5 Risk Monitoring, Controlling and Reporting	41
7.0 USER INTERFACE DESIGN	42
7.1 Sprint 1 UI Mockups	42
7.1.1 Admin Portal Sprint 1	42
7.2 Sprint 2 UI Mockups	48
7.2.1 Admin Portal Sprint 2	48
7.2.2 Client Portal Sprint 2	50
7.3 Sprint 3 UI Mockups	57
7.3.1 Dark Theme Update	57
Figure 42:	63
7.3.1.2 Dark Theme Update Client Portal	64
7.3.2 Admin Portal Sprint 3	70
7.3.3 Client Portal Sprint 3	78
7.4 Sprint 4 UI Mockups	93
7.4.1 Admin Portal Sprint 4	93
7.4.2 Client Portal Sprint 4	95
8.0 TESTING PLAN AND REPORT	96
8.1 Unit Testing	96
8.2 Code Coverage	96
8.3 Acceptance Testing	97
8.4 System Tests	99
8.5 Testing Procedure	100
8.6 Test Results	100
8.6.1 Unit & integration test for Admin app	101
8.6.2 Unit & integration test for Client app	102
8.6.3 System test for Admin app (Cypress)	104
8.6.4 System test for Client app (Cypress)	105
9.0 SPRINT RETROSPECTIVES	106

9.1 What went wrong	106
9.2 What went well	106

1.0 INTRODUCTION

Team 14 is working towards developing an application called COVID-19 Tracking app which will allow medical doctors and government health officials to monitor patients infected with COVID-19. Currently, with the rise in cases, it is extremely difficult for doctors to prioritize infected patients and many of them are overloaded. The application will eliminate overloading and place a limitation on the number of patients assigned per doctor. Additionally, the COVID-19 Tracking app will ensure infected patients can be categorized as priority or not and emergency communication can be established between the doctor and patient. The application aims to facilitate the monitoring of COVID-19 patients and lowering the percentage of cases.

1.1 Purpose

The purpose of this document is to outline the content of the project's first iteration which consists of the primary user requirements, release planning for following iterations, architectural structure, risk management plan, user interface designs, and testing plan for the various software components. The project contains a total of five iterations involving the development of the main functionalities such as notifications through Bluetooth system to inform a patient to self-quarantine and more.

1.1.1 System

The purpose of this project is to find a solution to keep track of individuals infected with COVID-19. This would be done via a system which would give updates on the status of COVID-19 patients, as well as provide advice to them.

1.1.2 Document

The purpose of this document is to:

- Describe the Project
- Detail the requirements
- Summarize the release planning
- Provide an overview of the architecture of the system
- Summarize high risk issue and possible solutions to minimize their impact
- Detail the user interface of the system
- Provide a testing plan in order to ensure the correctness of the system in terms of functionality and performance.
- Provide any information on possible defects that may have occurred during Sprint 1
- Detail any quality measurements taken

1.2 Targeted Users

The targeted users would primarily be patients with COVID-19, doctors, immigration officers, health officials and the administrators of this system.

1.3 Targeted Readers

The targeted readers for this document are any possible stakeholders for this system. The intended audience of this report is Yann-Gaël Guéhéneuc and Wei Liu, the product owner for the development of COVID-19 Tracking app.

2.0 PROJECT DESCRIPTION

The system in development will allow medical doctors and government health officials to monitor patients infected with COVID-19.

Agile methodology was selected because it allows us to develop the system in increments and allows for continuous feedback from stakeholders during development and allows for continuous improvement of the product.

2.1 Sprint Schedule

The sprint schedule will be as follows:

Sprint	Date
1	11/01/2022 - 02/02/2022
2	03/02/2022 - 23/02/2022
3	24/02/2022 - 16/03/2022
4	17/03/2022 - 06/04/2022
5	07/04/2022 - 18/04/2022

Table 1: Sprint Schedule

3.0 REQUIREMENTS

The following requirements were elicited from the product owner and have been turned into user stories approved by the product owner. **4 user stories** have been elicited for a total of **17 user story points**.

In the following subsections, we first present the backlog as an overview, and then look at each user story in detail.

3.1 Backlog

Per our backlog, these were the user stories related to sprint 3::

ID	Name	USP	Priority
Issue-11	As a doctor or health official, I want to monitor the status & symptoms of confirmed and unconfirmed patients so that I can prescribe a suitable treatment for the patient.	2	3
Issue-16	As a doctor, I want to see which patient's updates I have reviewed and which ones I have not reviewed yet so that I dont review over the same file again	2	2
Issue-37	As a patient, I want to be able to edit my profile so that doctor has all the up to date information.	8	2
Issue-103	As a patient, I want to see my profile with basic details such as my status, temperature, weight, address, etc so that I can know if the information is correct or needs modification.	5	1
Total		17	

Table 2: Sprint 3 User Stories

3.2 User Stories

In the following subsection, we look at each user story in detail and provide additional information for each (if applicable)

<u>Issue-11</u>	As a doctor or health official, I want to monitor the status & symptoms of confirmed and unconfirmed patients so that I can prescribe a suitable treatment for the patient.
USP	2
Priority	3
Description	Acceptance criteria: Given: The user is logged in as a doctor or health official. When: The user is on the Profile of the Patient page Then: The user can select status from the dropdown.

Table 3: User Story 1

<u>Issue-16</u>	As a doctor, I want to see which patient's updates I have reviewed and which ones I have not reviewed yet so that I don't review over the same file again
USP	2
Priority	2
Description	<p>Acceptance criteria:</p> <p>Given: The user is logged in as a doctor and is on the patient page. When: The user updates and reviews a patient profile. Then: The application shows profiles that have not been reviewed yet.</p>

Table 4: User Story 2

<u>Issue-37</u>	As a patient, I want to be able to edit my profile so that the doctor has all the up to date information.
USP	8
Priority	2
Description	<p>Acceptance criteria:</p> <p>Given: Patient is logged in successfully When: Patient clicks on profile page Then: the basic fields should be editable</p>

Table 5: User Story 3

<u>Issue-103</u>	As a patient, I want to see my profile with basic details such as my status, temperature, weight, address, etc so that I can know if the information is correct or needs modification.
USP	3
Priority	1
Description	<p>Acceptance criteria:</p> <p>Given: The user is logged in as a patient on the client app.</p> <p>When: The user clicks on profile button on bottom of the navbar.</p> <p>Then: The user is redirected to their profile displaying a list of given details about the patient.</p>

Table 6: User Story 4

4.0 RELEASE PLANNING

4.1 Sprint 3 Summary

Sprint 3 focused on delivering a variety of features as can be seen in the table below. In this sprint, work was performed in the frontend and backend, although the majority of the work was completed in the frontend and some on the backend. The problems we faced in this sprint included merging changes from main to branches faster so there wouldn't be too many merge conflicts upon completion of branch code ready to merge to main. During this sprint, issues 10, 19 and 38 were brought forward to be completed in Sprint 4 due to time constraints.

Story ID	Story Title	USP	Status
<u>Issue-11</u>	Monitor status & symptoms of confirmed and unconfirmed patients	2	DONE
<u>Issue-16</u>	Display / show that a patients' status updates have been reviewed	2	DONE
<u>Issue-37</u>	Allow Patient to edit profile page	8	DONE
<u>Issue-103</u>	Backend for "Profile for patient"	3	DONE
<u>Issue-2</u>	Frontend Display for "Arrange Appointments"	5	DONE
<u>Issue-29</u>	Frontend Display for "Patient updates status everyday"	2	DONE
<u>Issue-39</u>	Frontend Display for "Allow patient to view the basic info of the assigned doctor"	2	DONE
<u>Issue-41</u>	Frontend Display for "Allow PATIENTS view the history of his/her updated diary"	2	DONE

<u>Issue-80</u>	Frontend Display for "Create a similar "List of symptoms" page for patients"	2	DONE
<u>Issue-4</u>	Frontend Display for "Trace & notify COVID-19 patients"	2	DONE
<u>Issue-10</u>	Notify doctor when status is re-updated on the same day	3	PUSHED TO SPRINT 4
<u>Issue-19</u>	Allow admin to manage client account (delete, re-assign role)	5	PUSHED TO SPRINT 4
<u>Issue-38</u>	Patient diary to update locations	5	PUSHED TO SPRINT 4
Total		43	30

Table 10: Sprint 3 Summary of Stories

An * would be a good way to show stories added during the sprint. And a strikethrough would be good to denote a deleted story.

4.2 Sprint 4 Planning

Sprint 4 will focus on delivering the next set of important features which can be seen in the table below. In this sprint, our team is planning on covering 17 user stories which sum up to a total of 64 user story points. As compared to the previous sprint, the workload can be seen to have remained relatively the same for Sprint 4. We might face the same problem as in the previous sprint since we might possibly have to push forward some user stories to later sprints due to time constraints. In this fourth sprint, we are also catching up on three user stories (Issue-10, Issue-19, and Issue-38) from the last sprint which have additionally added onto the overall sum of user story points to be completed.

Story ID	Story Title	USP	Status
<u>Issue-4</u>	Backend for "Trace & notify COVID-19 patients"	3	
<u>Issue-10</u>	Notify doctor when status is re-updated on the same day	3	
<u>Issue-2</u>	Backend for "Arrange Appointments"	8	
<u>Issue-12</u>	Notify Covid-19 patients & self-quarantine individuals to update status daily	3	
<u>Issue-14</u>	Different visibility for different users	3	
<u>Issue-80</u>	Backend for "Create a similar "List of symptoms" page for patients"	3	
<u>Issue-19</u>	Allow admin to manage client account	5	
<u>Issue-29</u>	Backend for "Patient updates status everyday"	5	
<u>Issue-38</u>	Patient diary to update locations	5	
<u>Issue-39</u>	Backend for "Allow patient to view the basic info of the assigned doctor"	3	
<u>Issue-41</u>	Backend for "Allow PATIENTS view the history of his/her updated diary"	3	
<u>Issue-43</u>	Allow DOCTORS to view the history of symptoms for each patient	2	

<u>Issue-238</u>	Admin List for superadmin to view/edit	5	
<u>Issue-42</u>	(optional) Allow patient to view the message from the assigned doctor via notification panel	3	
<u>Issue-44</u>	(optional) Allow doctor to view the list of assigned patients & past assigned patients	2	
<u>Issue-45</u>	(optional) Allow doctor to change status of a patient to a non-patient	3	
<u>Issue-46</u>	(optional) Allow health officer to view statistics like the number of patients for each district daily	5	
Total		64	

Table 11: Sprint 4 Story Planning

5.0 ARCHITECTURE

In this section, we present a high level overview of the system, illustrate all the use cases, provide a domain model diagram as well as detail our database design.

5.1 High-Level Overview of System

The diagram below illustrates how the application communicates with the backend via Redux store (the centralized state management component). In particular, for most of the events that need to manipulate / access some piece of data, the component will first dispatch an action to the local data storage to be processed, then the local data storage will communicate with the database to update information (if some local states are changed).

The frontend application is planned to build with Material-UI & Semantic-UI library in order to save time and boost developers' workflow. In addition, the usage of these libraries guarantees that the application will have a consistent look and feel.

The backend of the system is managed by Firebase (an abstraction layer of Google Cloud) which provides NoSQL database system, user authentication, and some advanced custom functions.

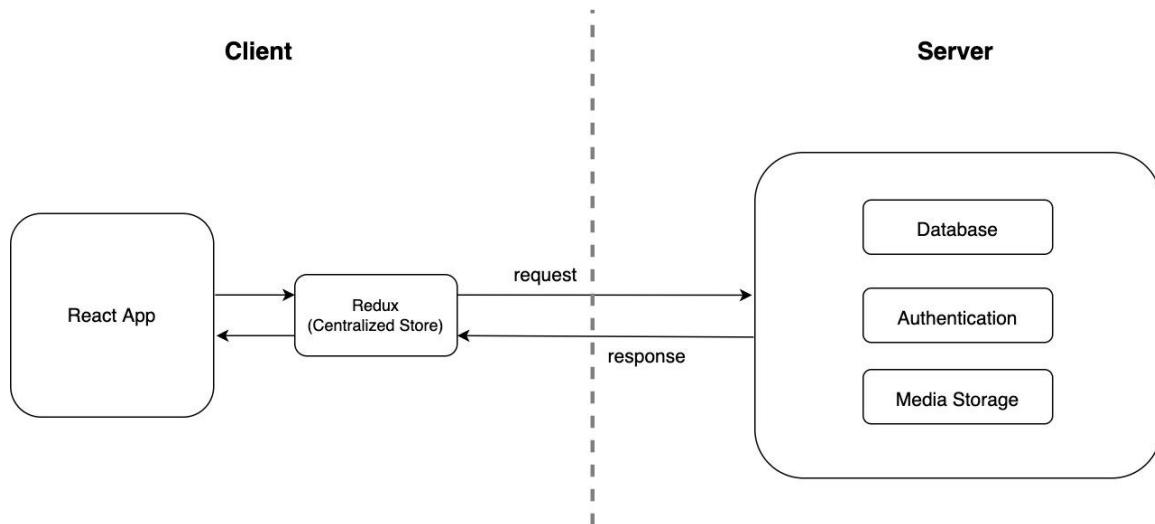


Figure 1: High-level overview system diagram

5.2 Use Case Diagrams

In our application, there are several important use cases to consider which illustrate the different interactions between the users and the system.

5.2.1 Use Cases related to the Admin App

For the Admin App, the “User” stakeholder contains doctors, admins, health officials and immigration officers.

- **Register:** A User of Covid Track must be able to register an account with email and password.
- **Login:** A registered user of Covid Track must be able to login with a registered email and password.

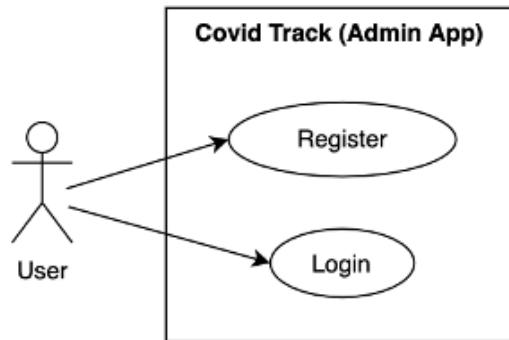


Figure 2: Use case diagram for authentication and registration on the Admin App.

- **View Dashboard:** A registered doctor of Covid Track can view their dashboard.

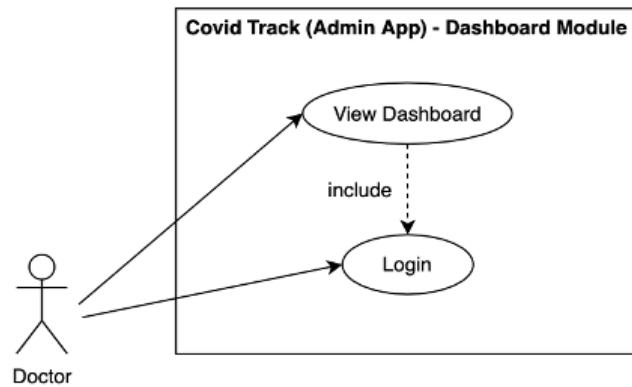


Figure 3: Use case diagram for dashboard view on the Admin App.

- **View Patient Profile:** Registered users of Covid Track can view a patient's profile.

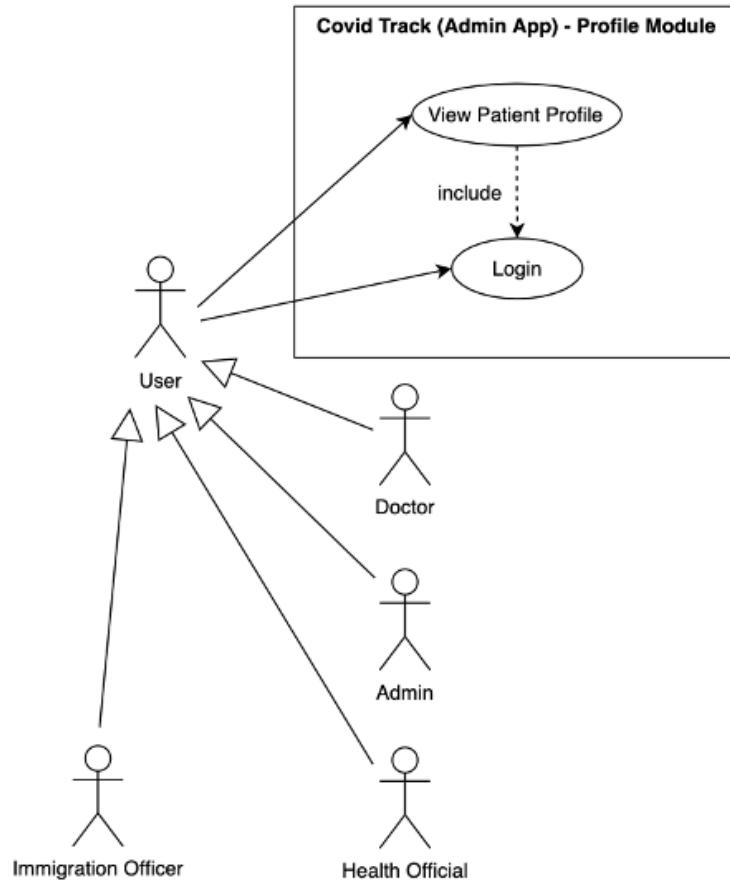


Figure 4: Use case diagram for patient profile view on the Admin App.

- **Flag Patient:** A registered doctor, health official or immigration officer of Covid Track can flag a patient in order to prioritize the patient's updates.

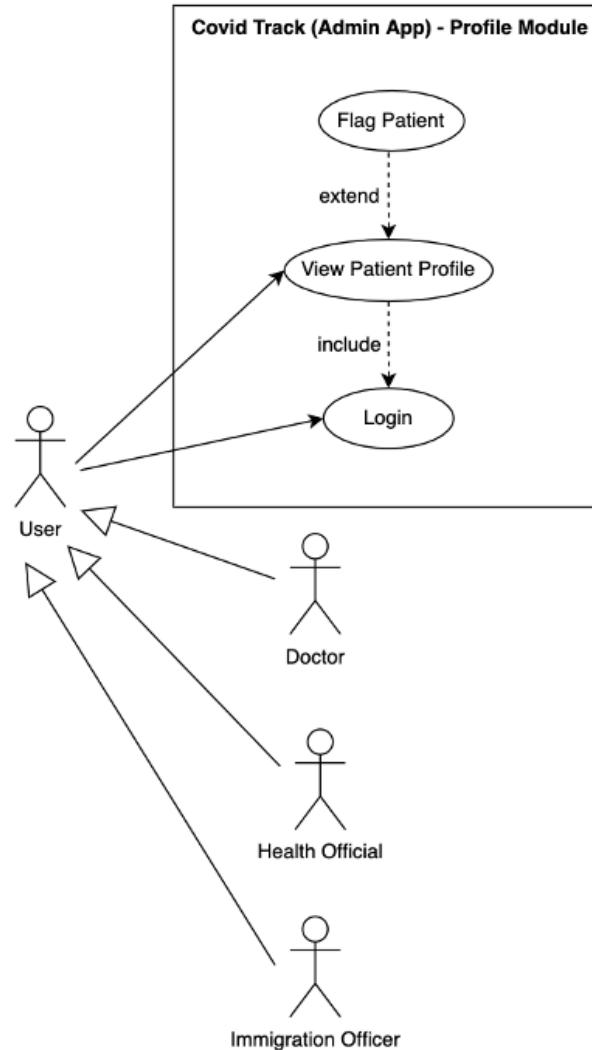


Figure 5: Use case diagram for prioritizing patient updates on the Admin App.

- **Patient Status Review:** A registered doctor of Covid Track can mark a profile as reviewed after reviewing the patient's status.

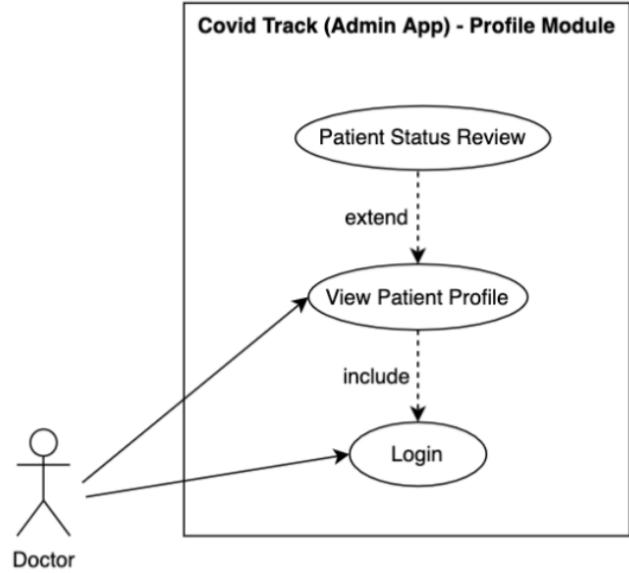


Figure 6: Use case diagram for patient status review on the Admin App.

- **Notify Covid Patient Contacts:** A registered health official of Covid Track can trace and notify the people with whom Covid-19 patients have been in contact.

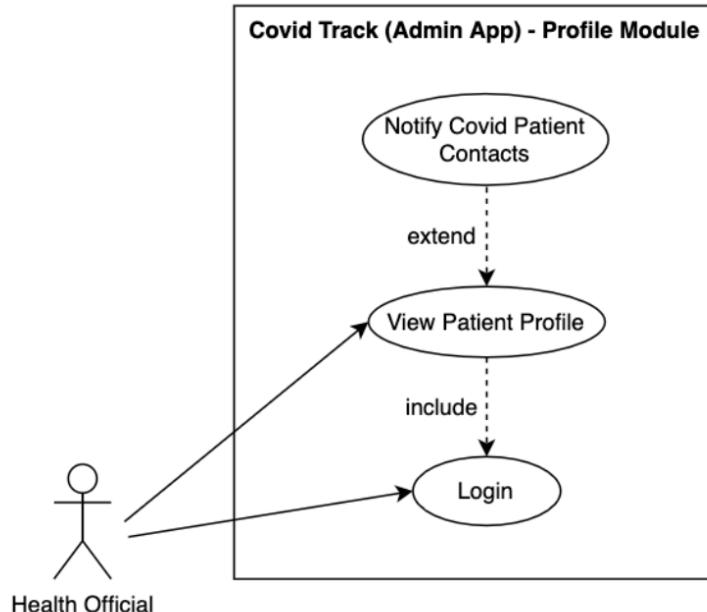


Figure 7: Use case diagram for notifying Covid-19 patient contacts on the Admin App.

- **Select Patient Status from Dropdown:** A registered doctor and health official of Covid Track can monitor and select the status of a patient from the dropdown located on the patient's profile.

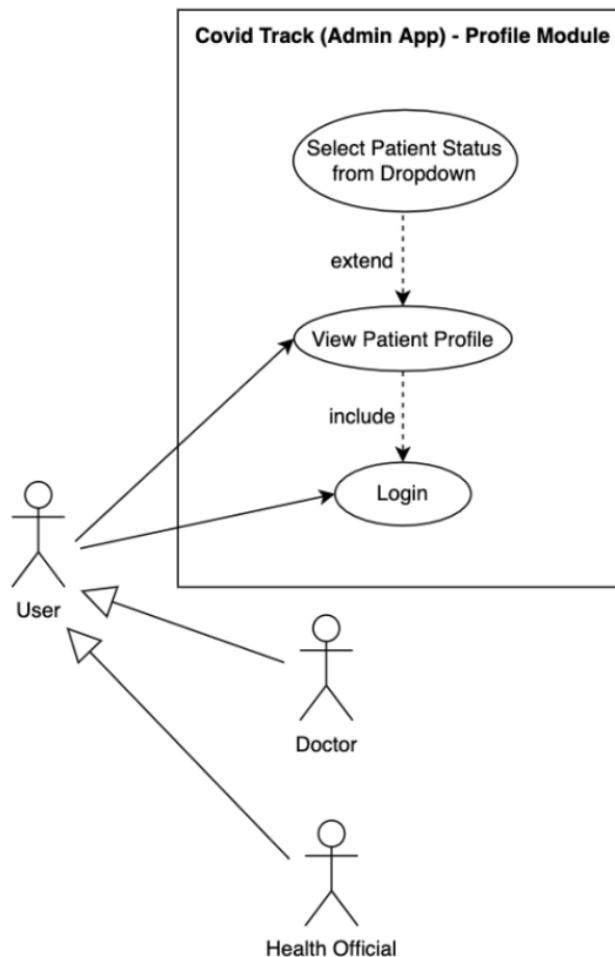


Figure 8: Use case diagram for monitoring status of confirmed and unconfirmed patients on the Admin App.

- **View Patient List:** A registered doctor and health official of Covid Track can view the patient list.

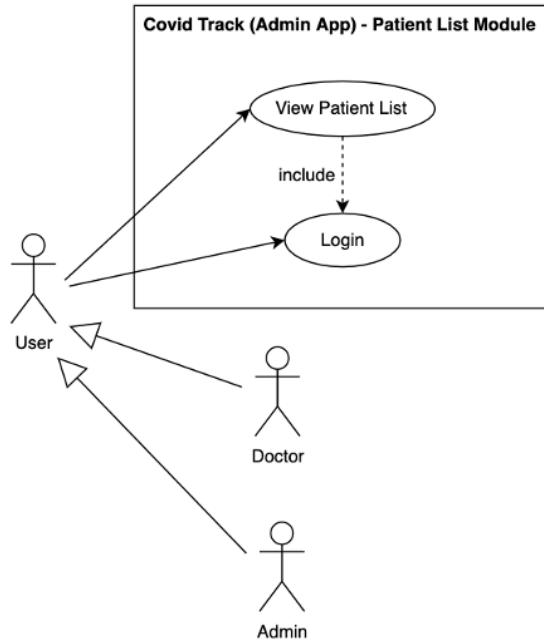


Figure 9: Use case diagram for patient list view on the Admin App.

- **Assign Doctor to Patient:** A registered admin can assign doctors to incoming patients.
- **Re-assign Patient to Another Doctor:** A registered admin can re-assign a patient to another doctor.

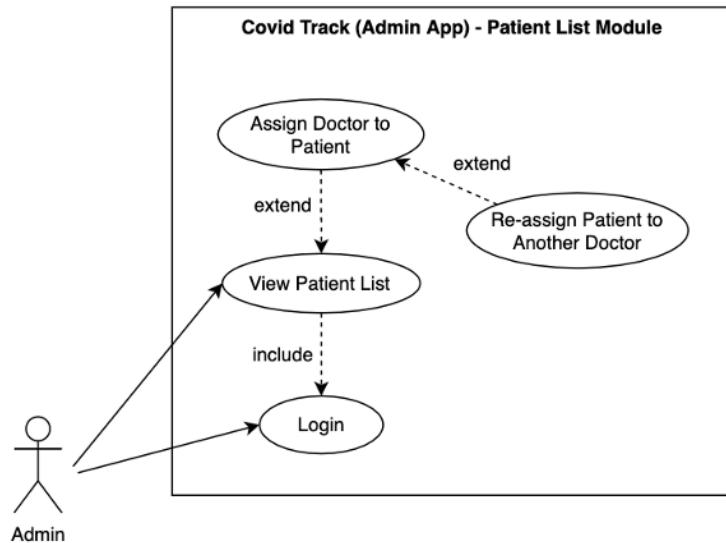


Figure 10: Use case diagram for assigning doctors to patients and re-assigning a patient to another doctor on the Admin App.

- **View Inbox:** A registered doctor can view their inbox.
- **Contact Patient:** A registered doctor can contact a patient.

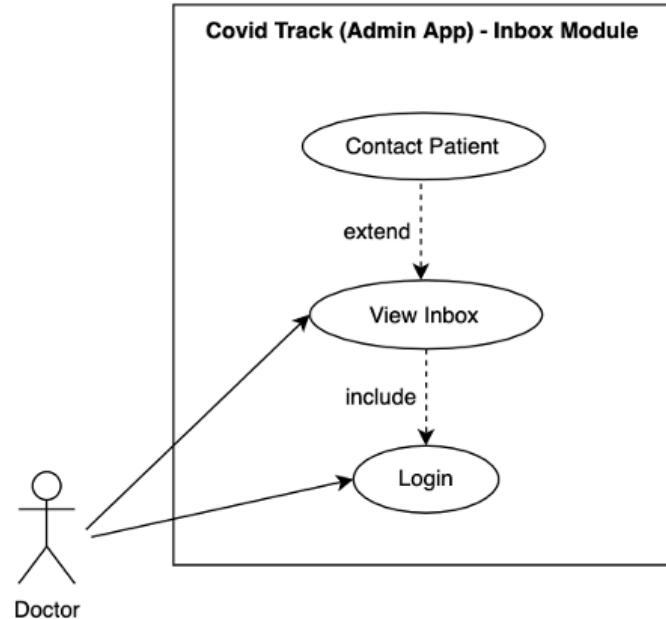


Figure 11: Use case diagram for contacting a patient on the Admin App.

- **View Appointments:** A registered doctor can view their appointments.

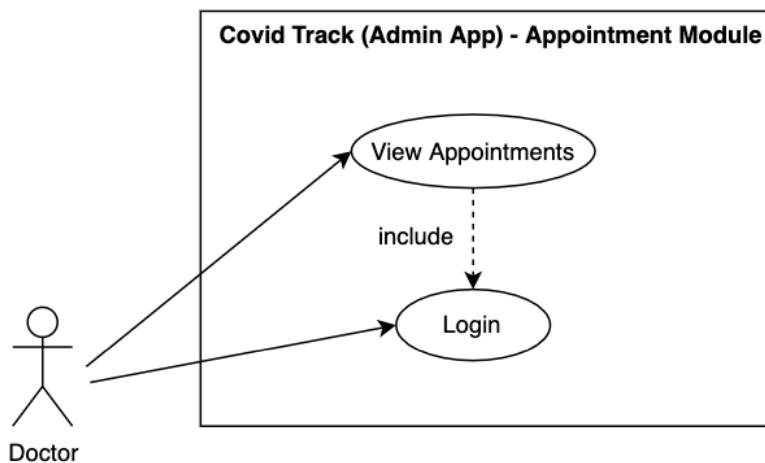


Figure 12: Use case diagram for appointments view on the Admin App.

5.2.1 Use Cases related to the Client App

For the Client App, the “User” stakeholder only contains Patients.

- **Register:** A User of Covid Track must be able to register an account with email and password.
- **Login:** A registered user of Covid Track must be able to login with a registered email and password.

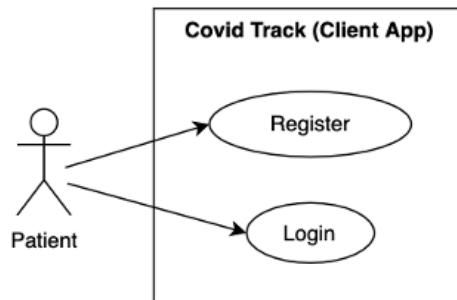


Figure 13: Use case diagram for authentication and registration on the Client App.

- **View Profile:** A registered patient can view their profile.
- **Edit Profile:** A registered patient can edit their profile.

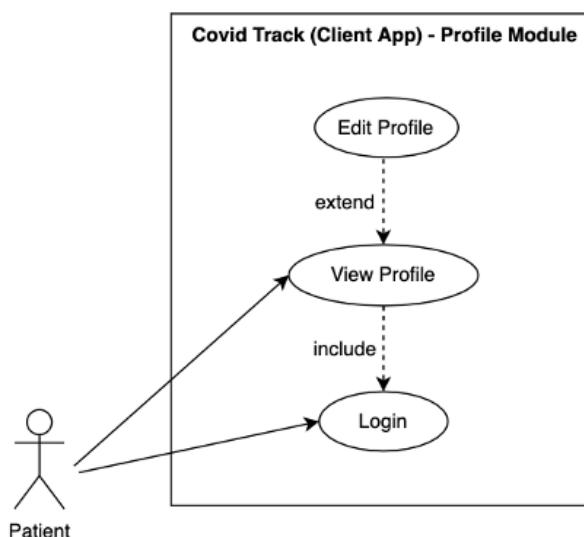


Figure 14: Use case diagram for profile view and edit on the Client App.

- **View Send Message Page:** A registered patient can view the page to send messages.
- **Contact Doctor:** A registered patient can contact a doctor.

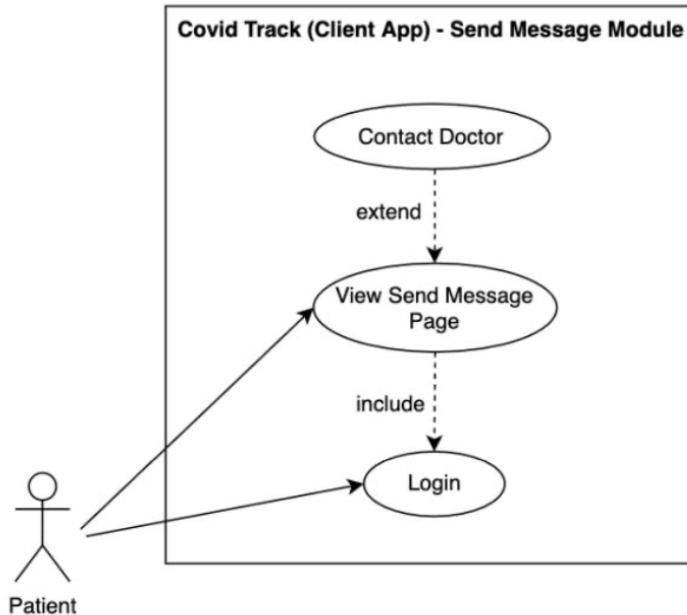


Figure 15: Use case diagram for contacting a doctor on the Client App.

- **View My Doctor:** A registered patient can view the basic information of their assigned doctor.

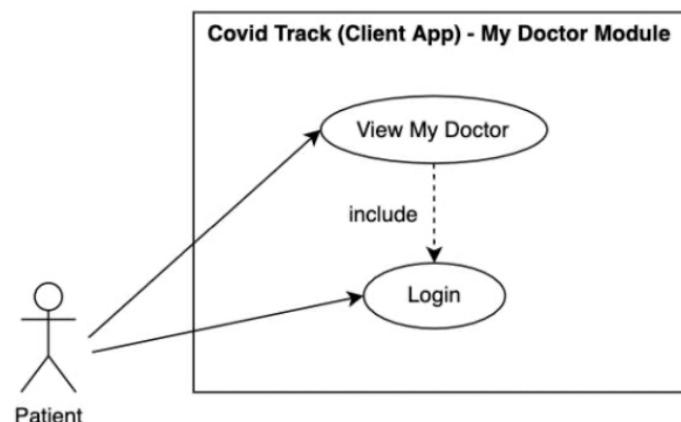


Figure 16: Use case diagram for viewing the basic information of the assigned doctor on the Client App.

- **View Diary:** A registered patient can view their diary.

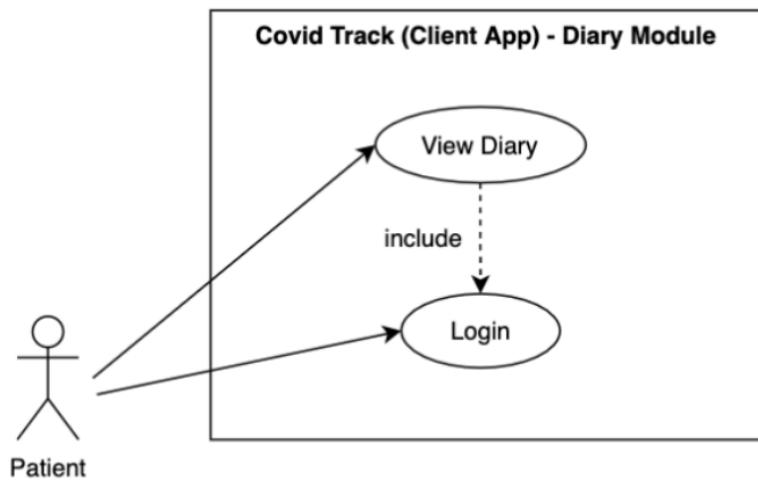


Figure 17: Use case diagram for diary view on the Client App.

- **View Update Status Page:** A registered patient can view their updated status page.
- **Add Status:** A registered patient can add a status.

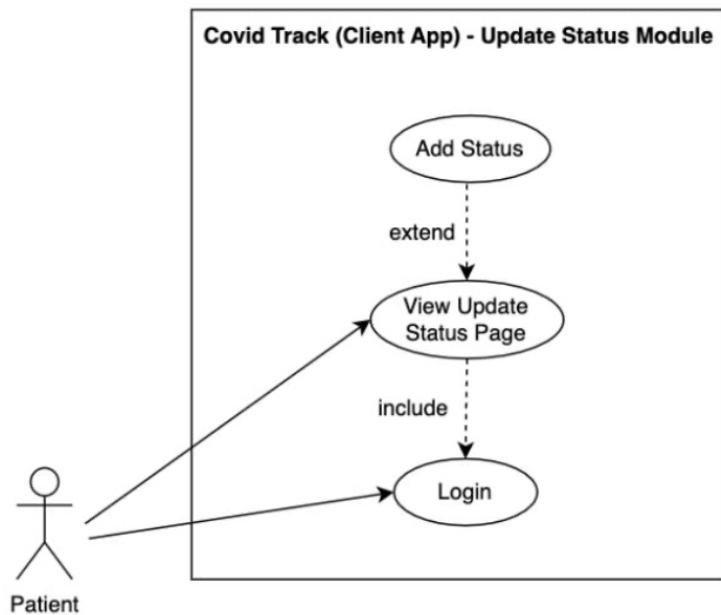


Figure 18: Use case diagram for viewing and adding status on the Client App.

- **View Appointment:** A registered patient can view their appointment.

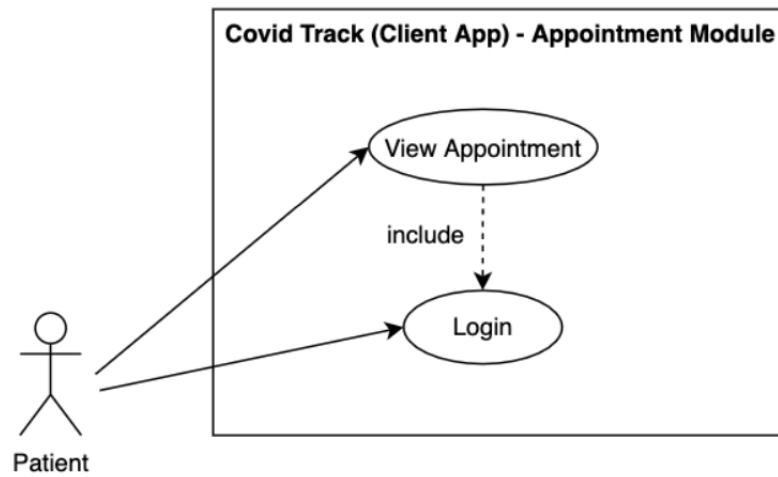


Figure 19: Use case diagram for appointment view on the Client App.

5.3 Domain Model Diagram

The domain model diagram describes the relationship between different entities in our application. In particular, each patient could have only one *ProfileDetails* and each *ProfileDetails* should only belong to one patient. Besides the basic patient details such as date of birth, name, address; every *ProfileDetails* is additionally composed of a *ListOfSymtoms*. Each patient registered in our platform will also own a *QRCode* and a *Diary*.

Both *Doctor* and *Patient* entities can only have one *Inbox* which contains one or more messages. The difference here is each *Inbox* belongs to one patient but it could be assigned to different doctors so that the new doctor can join and continue the conversation from the previously assigned doctor.

For the *Appointment*, each patient is allowed to book only one (concurrent) appointment at a time to ensure that they do not overload the system. However, for doctors, they can have multiple (concurrent) appointments with their patients. In addition, patients are able to see the history of all of their previous appointments.

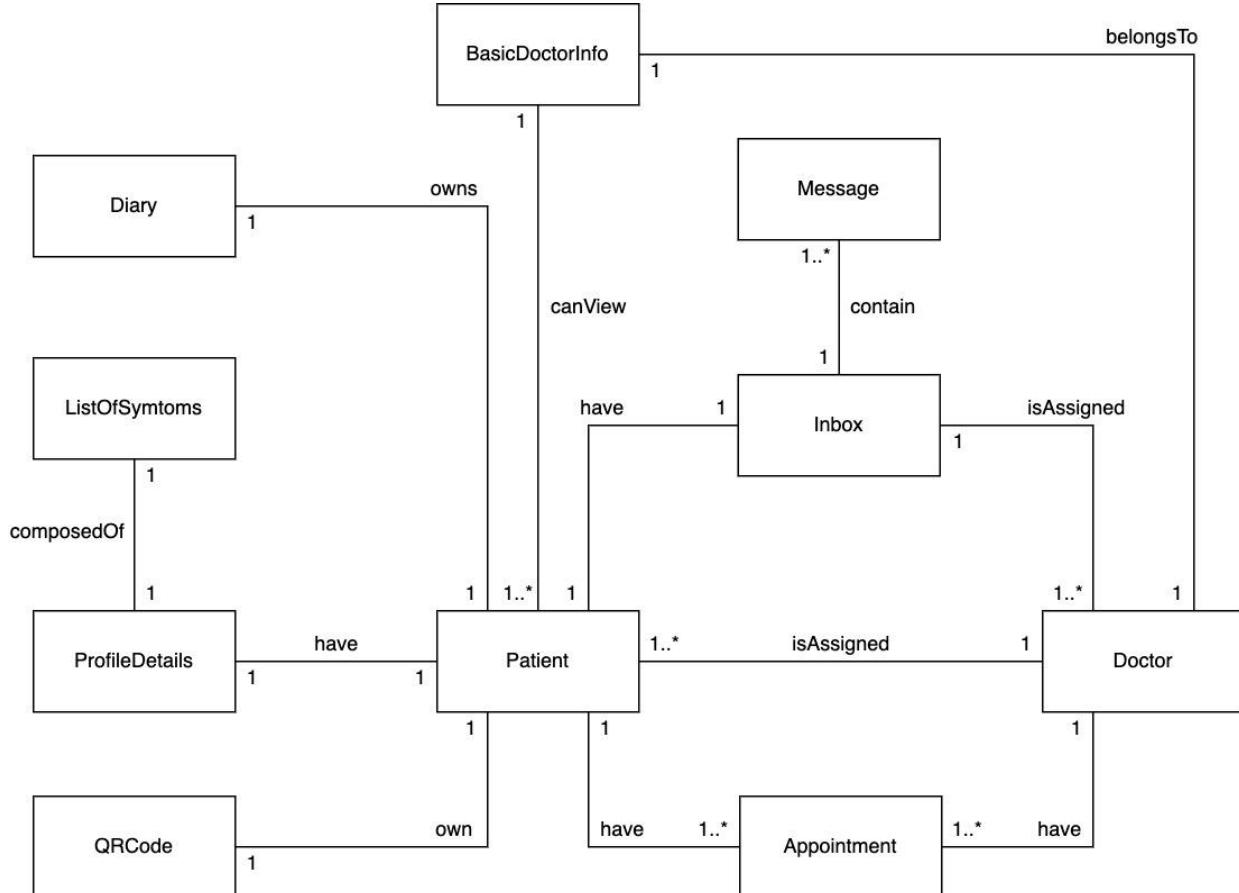


Figure 12: Domain model diagram

5.4 Database Design

The firebase backend comes with a NoSQL database due to its capability of being highly scalable. Even though it may not be as straightforward as a SQL database when modeling relationships, NoSQL “collections” and “documents” structures are simple and fast to create. Moreover, it can be modeled to mimic the table relationships from SQL using JSON-like syntax.

In this schema design, a doctor can have many patients but every patient can only be assigned to one doctor at a time. Both doctors and patients can have an inbox, and each inbox can contain a list of messages (conversations). Every patient in our platform will have a *ProfileDetails*, a *ListOfSymptoms*, a *Diary*, and a unique *QRCode* that belongs to his/her profile. For the appointment, it makes sense to allow a doctor to have many concurrent *Appointments*, however, each patient should be allowed to make only one concurrent appointment at a time to avoid overwhelming the system. The reason that the relationship between a patient to an appointment is a one-to-many relationship is that the patient needs to have access to (many) past appointments that they have.

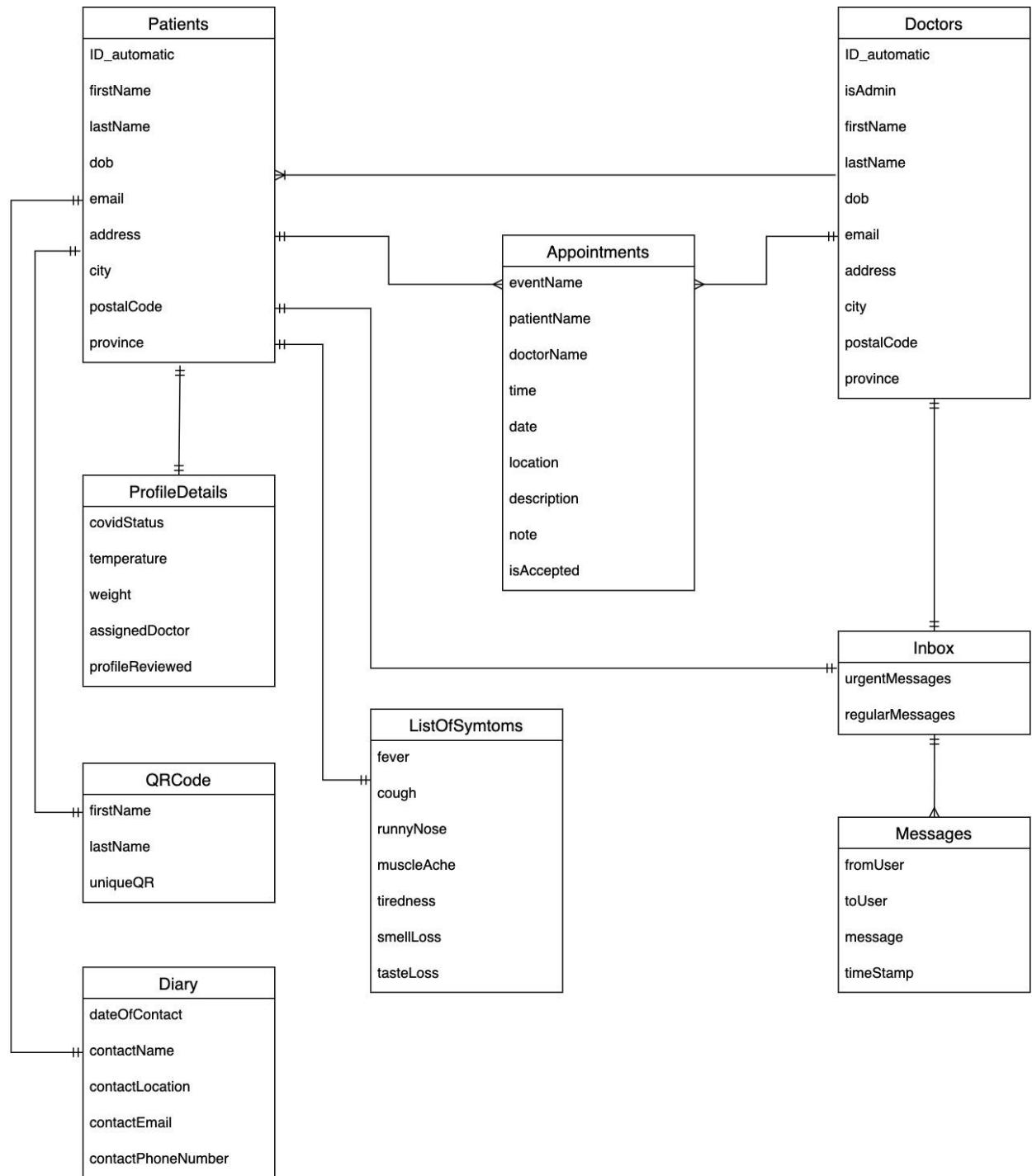


Figure 13: Database design

5.5 Physical Design of the Database

Firebase is a part of Google Cloud Service. It offers Cloud Firestore which is a highly scalable NoSQL database that can be used through various business applications regardless of the size. Using real-time listeners, Firebase stores and keeps the data in sync across the application. Cloud storage provides robust operations with highly secured data transactions. In addition, it serves user-generated content from the database in real-time. Moreover, it offers data analytics so that developers could draw meaningful insights from the application they build. Figure 14 below illustrates the architecture of the system. On a regular workflow, everytime a request arrives to the backend, it could be passed through different flows of connected cloud services.

The Cloud Logging service: is a fully managed, real-time log management system that comes with search, analysis and alerting features.

The Analytics service: enables developers and business owners to apply Machine Learning models, or other analytics assets to increase the ROI.

The Dataflow service: allows batch data processing with automated provisioning and horizontal scaling to maximize resource utilization.

The Data Storage service: allows reliable, high performance and secured data transfer.

The App Engine service: enables developers to build scalable applications on a fully managed serverless platform.

The Compute Engine service: helps creating and running virtual machines on Google's infrastructure.

The Big Query service: a multi cloud data warehouse designed for business agility.

The Dataproc service: a managed and a scalable service for running different open source tools and frameworks.

The Datalab service: used to explore, visualize, analyze, and transform data easily.

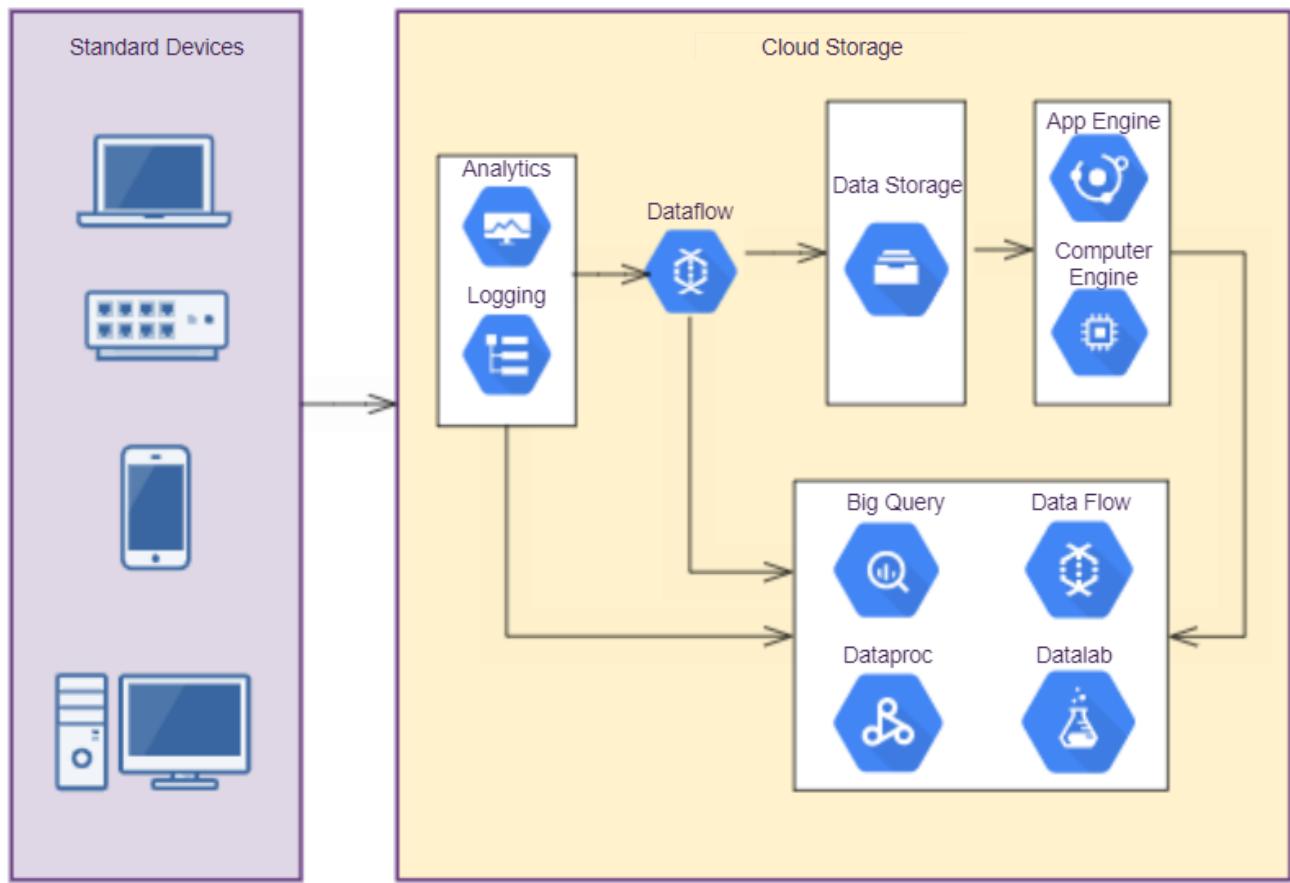


Figure 14: Physical Design of the Database

5.6 Sequence Diagrams

All processes in the COVID-19 tracking application system can be presented by a sequence diagram. The first process is the login/sign in process that is illustrated in figure 5. After the patients register, they will be directed to the dashboard page and they will be assigned to a doctor if they are new registered patients.

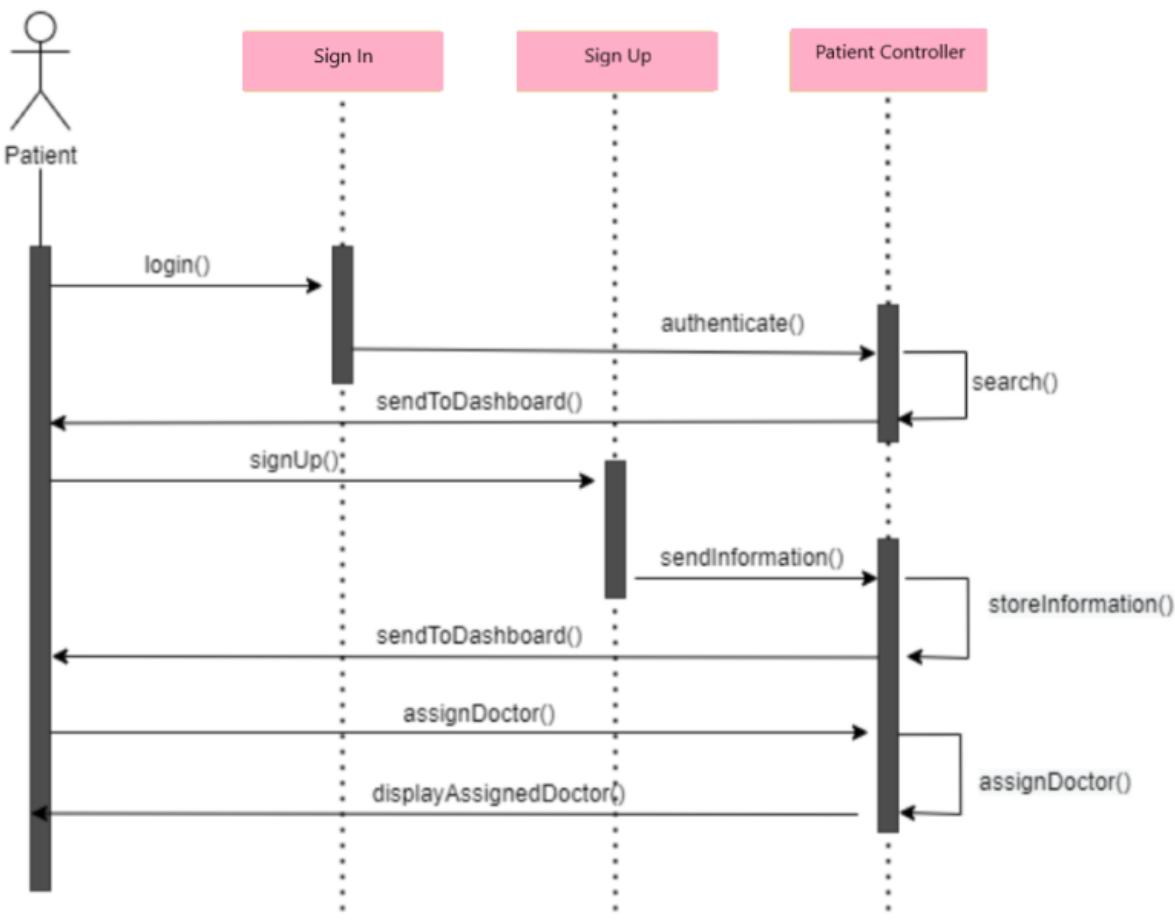


Figure 15: Sign In/ Sign Up for Patient Sequence Diagram

After signing in, the patients will be able to update their profiles. If a patient was tested positive, she/he can update their status to notify the doctor with the updates. The patient will get a notification therefore when the doctor reviews their status as represented in figure 6.

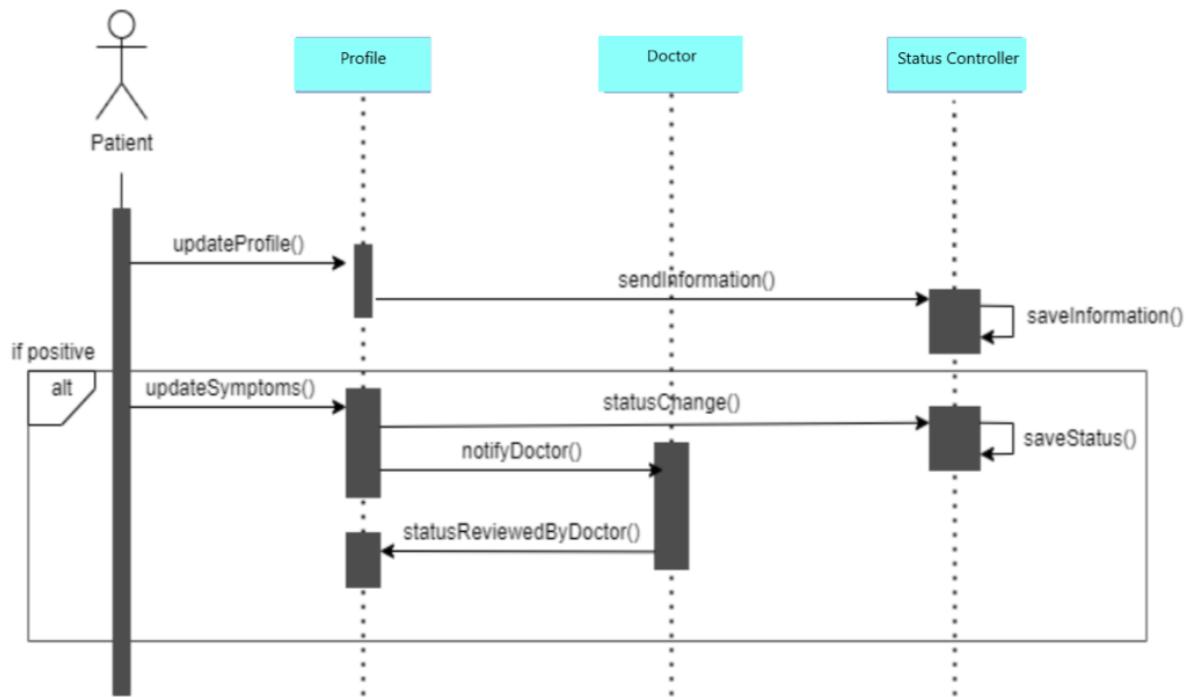


Figure 16: Status Update for Patients Sequence Diagram

If a patient has a question, they can always message their assigned doctor. The patients will be prompted a question to determine the urgency of their situation. If urgent, they will be given the priority to be answered by the doctor.

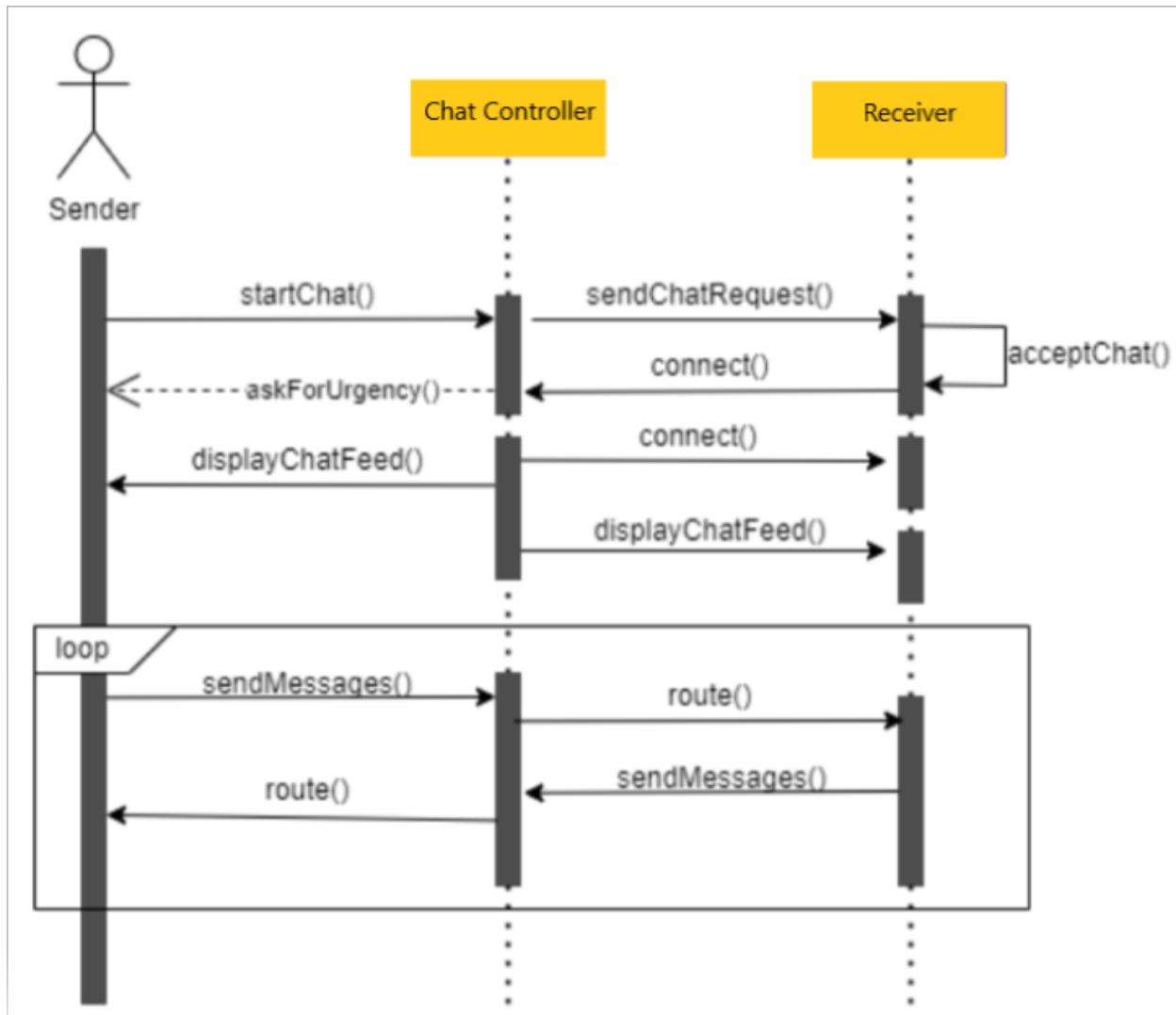


Figure 17: Chatting System Sequence Diagram

All patients will have a unique QR Code, that they can view on their account. The admins will be able to scan a QR code to get a patient's record. If the record doesn't exist, it will return an error message. Otherwise, it will return the profile of the patient as displayed in Figure 8.

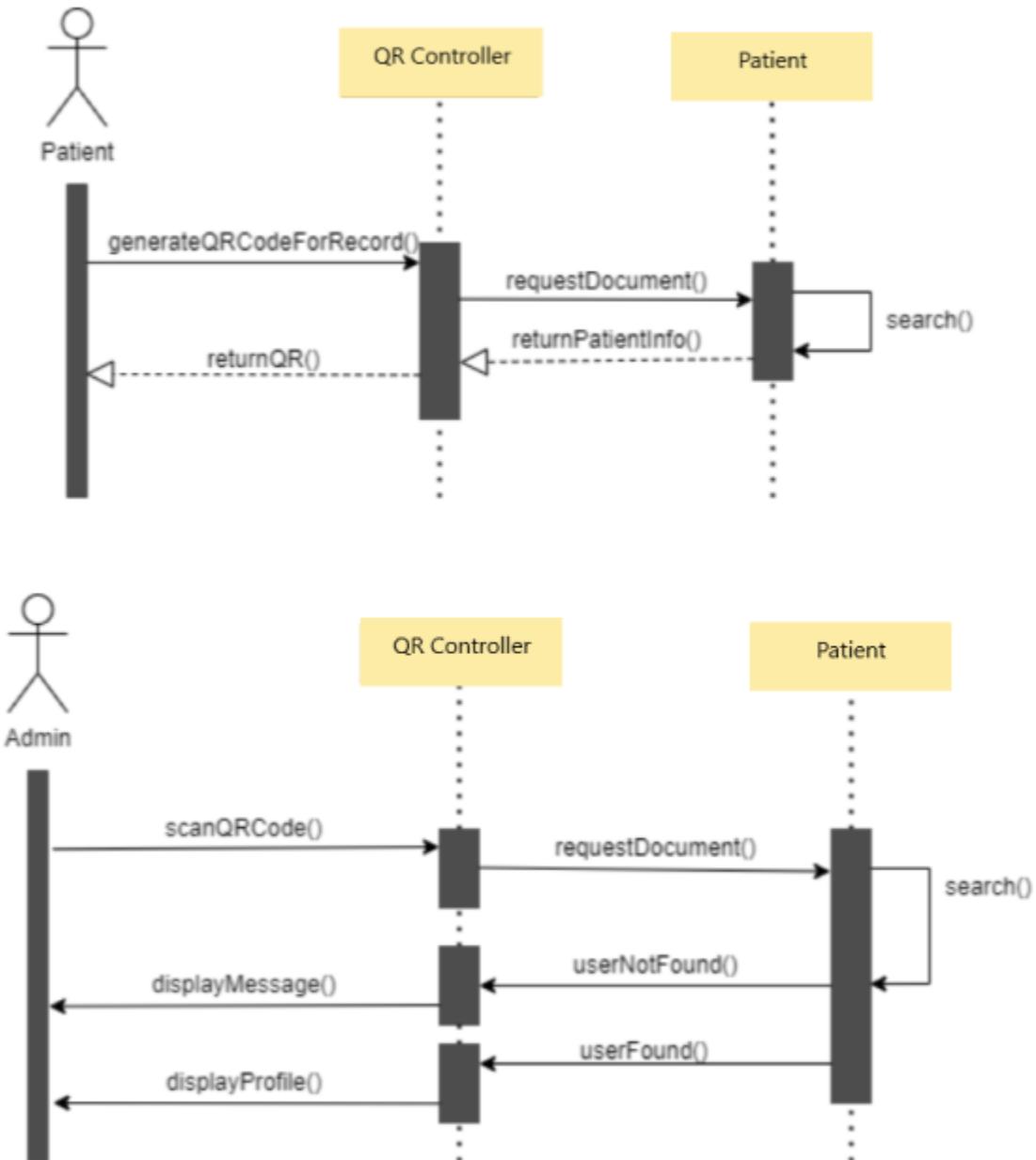


Figure 18: QR Feature Sequence Diagram for Patients and Admins

Every patient will be assigned to an available doctor when registering in the covid tracking application as shown in figure 19.

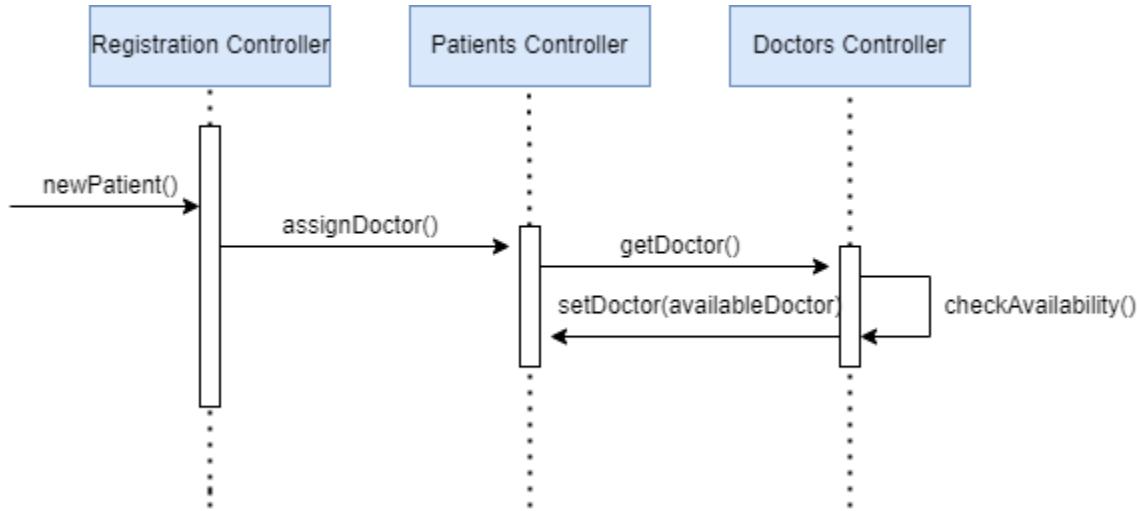


Figure 19: Assigning Patients to Doctors Feature Sequence Diagram

Every patient will be required to update the symptoms everyday. However, the doctor will be notified if the patient updates her/his status twice a day.

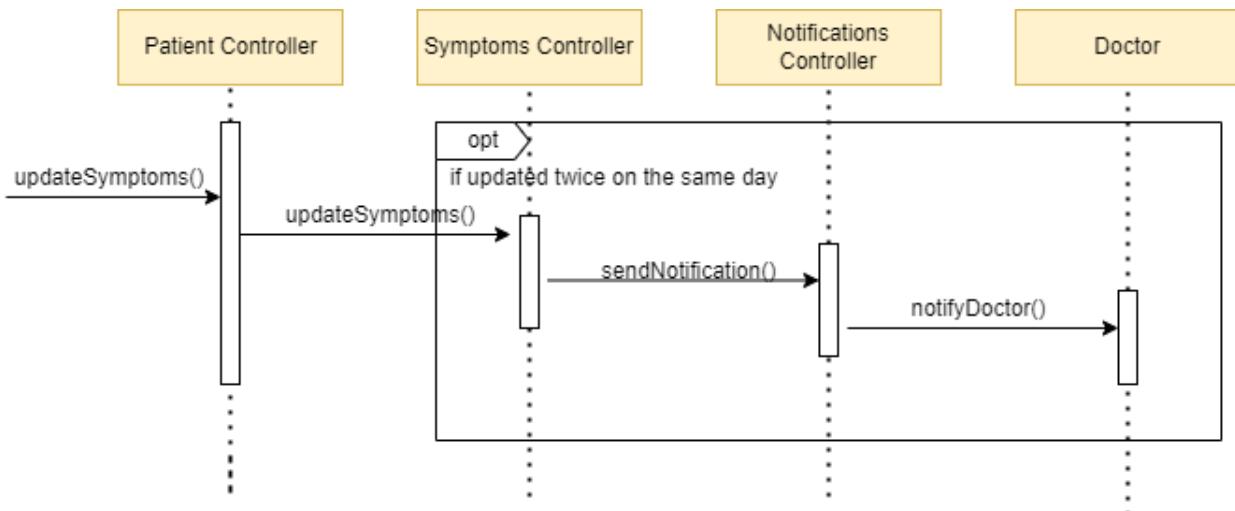


Figure 20: Updating Symptoms Feature Sequence Diagram

6.0 RISK MANAGEMENT

In this section, we present the purpose of the risk management plan, its procedure, an analysis, a response, and methods for monitoring, controlling and reporting risks.

VERSION HISTORY

Versio n #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Laila Alhalabi, Marwa Khalid	01/17/22	Quang Tran	02/02/22	Initial Risk Management Plan draft

Table 12: Risk Management Plan Version History

6.1 Purpose of the Risk Management Plan

Risk management is one of the most important aspects of any business as it has a direct effect on a project's objectives. The RMP is a process designed to identify and evaluate all possible risks and hazards that can affect the COVID-19 Tracking application. The process involves performing activities such as: identifying and analyzing risks, assessing the impact of risks as well as monitoring and controlling risks throughout the project lifecycle. The Risk Management Plan (RMP) will be updated at each stage of the project life-cycle.

6.2 Risk Management Procedure

In this subsection, we detail the risk management process and identifying risks.

6.2.1 Process

The project manager will serve as the Risk Manager within this project, with responsibilities such as developing and updating the project scope and schedule, communicating with stakeholders, and maintaining the project in a manner that ensures a high-quality product in a timely manner. Nonetheless, it is important to identify all the risks that could possibly happen in the project so that they can be avoided. The following sections will identify the steps for accomplishing risk management.

6.2.2 Risk Identification

The identification of risks is an important part of the project management process. It helps in identifying all the possible risks that could occur during the project. The risk identification will include all the risks of the project.

Risk ID	Description	Probability	Impact
R-1	Patient can alter another patient's data	Moderate	High
R-2	Application crashes unexpectedly	Moderate	High
R-3	Patient data accessible by anyone.	Moderate	High
R-4	Login Security is Weak	Moderate	High
R-5	Inability to complete requirements/features by the deadline	High	High
R-6	User unable to update their status	Moderate	Moderate
R-7	Doctor isn't notified when patient changes status on the same day	Moderate	Moderate
R-8	Not enough test coverage to identify bugs/issues	Moderate	Moderate
R-9	User is unable to log in	Low	Low
R-10	User Interface isn't intuitive	Low	Low

Risk ID	Description	Resolved in Sprint	Strategy and Effectiveness
R-1	Patient can alter another patient's data		Nothing has been decided yet.
R-2	Application crashes unexpectedly		Nothing has been decided yet.
R-3	Patient data accessible by anyone.		Nothing has been decided yet.
R-4	Login Security is Weak		Nothing has been decided yet.
R-5	Inability to complete requirements/features by the deadline		Nothing has been decided yet.
R-6	User unable to update their status		Nothing has been decided yet.
R-7	Doctor isn't notified when patient changes status on the same day		Nothing has been decided yet.
R-8	Not enough test coverage to identify bugs/issues		Nothing has been decided yet.
R-9	User is unable to log in		Nothing has been decided yet.
R-10	User Interface isn't intuitive		Nothing has been decided yet.

Table 13: Risk Identification Table

6.3 Risk Analysis

In this section, we will identify the risks that are likely to affect the project and assess the possible outcomes. Risks that may affect this project include work-breakdown structure (WBS), deliverables and cost and effort assessments of the project.

6.3.1 Qualitative Risk Analysis

The project manager will analyze the likelihood and impact of occurrence for each identified risk. Risks Probability:

- High Risk: If there is a greater than 70% chance that a risk can happen, it is considered high.
- Medium Risk: If there is between 30%-70% chance that a risk can happen, it is considered medium.
- Low Risk: If there is less than 30% chance that a risk can happen, it is considered low.

PROBABILITY	THREATS				
	Very low	Low	Moderate	High	Very High
High				R5	
Moderate		R6, R7, R8		R1, R2, R3, R4	
Low	R9, R10				
	IMPACT				

Table 14: Risk Analysis Table

6.3.2 Quantitative Risk Analysis

The qualitative risk analysis technique is used to examine risk occurrences that have been prioritized.

6.4 Risk Response Planning

Each high risk will be allocated to a project team member for monitoring purposes, and they will be responsible for monitoring the risks that they are assigned. One of the following approaches will be used to handle each main risk:

- Avoid: Avoiding the risk by avoiding the cause of this risk.
- Mitigate: Mitigating risk by minimizing its impacts on the system.
- Accept: Accepting the risk - nothing to be done.
- Transfer: Transferring the responsibility of monitoring and controlling a risk to a third party, such as buying insurance.

6.5 Risk Monitoring, Controlling and Reporting

Throughout the project lifespan, the amount of risk will be measured, managed, and reported. The project team will keep a "Top 10 Risk List" that will be disclosed as part of the project status reporting procedure for this project.

7.0 USER INTERFACE DESIGN

To construct the user design, Figma was used as well as the following community templates;
<https://www.figma.com/community/file/987982429991608576/Semantic-UI---Figma>,
[https://www.figma.com/community/file/912837788133317724/MUI-for-Figma-v5.4.0-\(Community\)](https://www.figma.com/community/file/912837788133317724/MUI-for-Figma-v5.4.0-(Community))

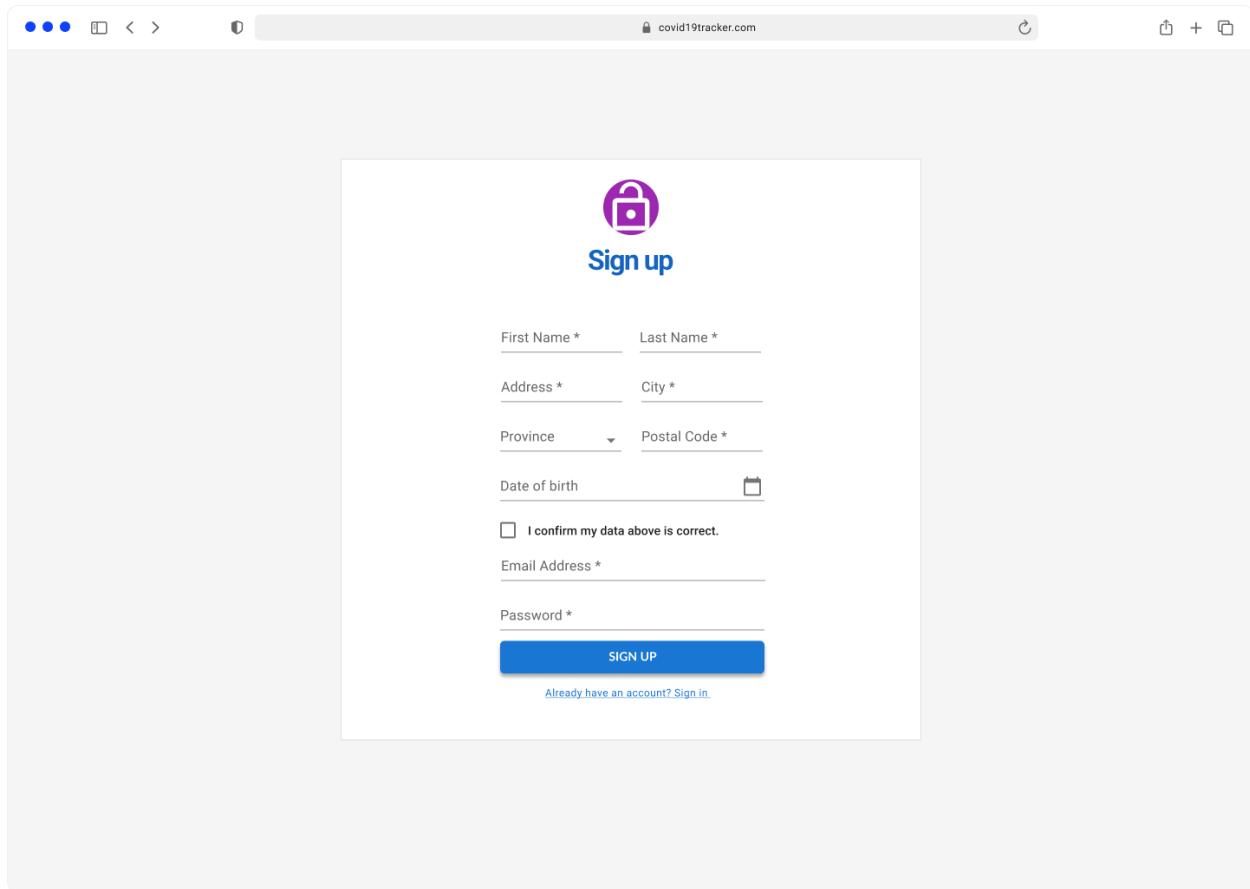
In this section, the **UI mockup** for each feature (user story) completed in Sprint 1 and the ones we intend to complete in Sprint 2.

7.1 Sprint 1 UI Mockups

Here are the UI mockups of user stories completed in Sprint 1.

7.1.1 Admin Portal Sprint 1

The following figures are the UI mockups for the Admin portal.



A screenshot of a web browser displaying a sign-up form for the Admin Portal. The URL in the address bar is covid19tracker.com. The form is titled "Sign up" and features a purple padlock icon above it. It includes fields for First Name * and Last Name *. Below these are fields for Address * and City *. A dropdown menu for Province and a field for Postal Code * are also present. A Date of birth field with a calendar icon follows. A checkbox labeled "I confirm my data above is correct." is shown next to a "SIGN UP" button. At the bottom of the form, there is a link "Already have an account? Sign in."

Figure 21: UI Admin Sign-up for User Story [Issue-7](#)

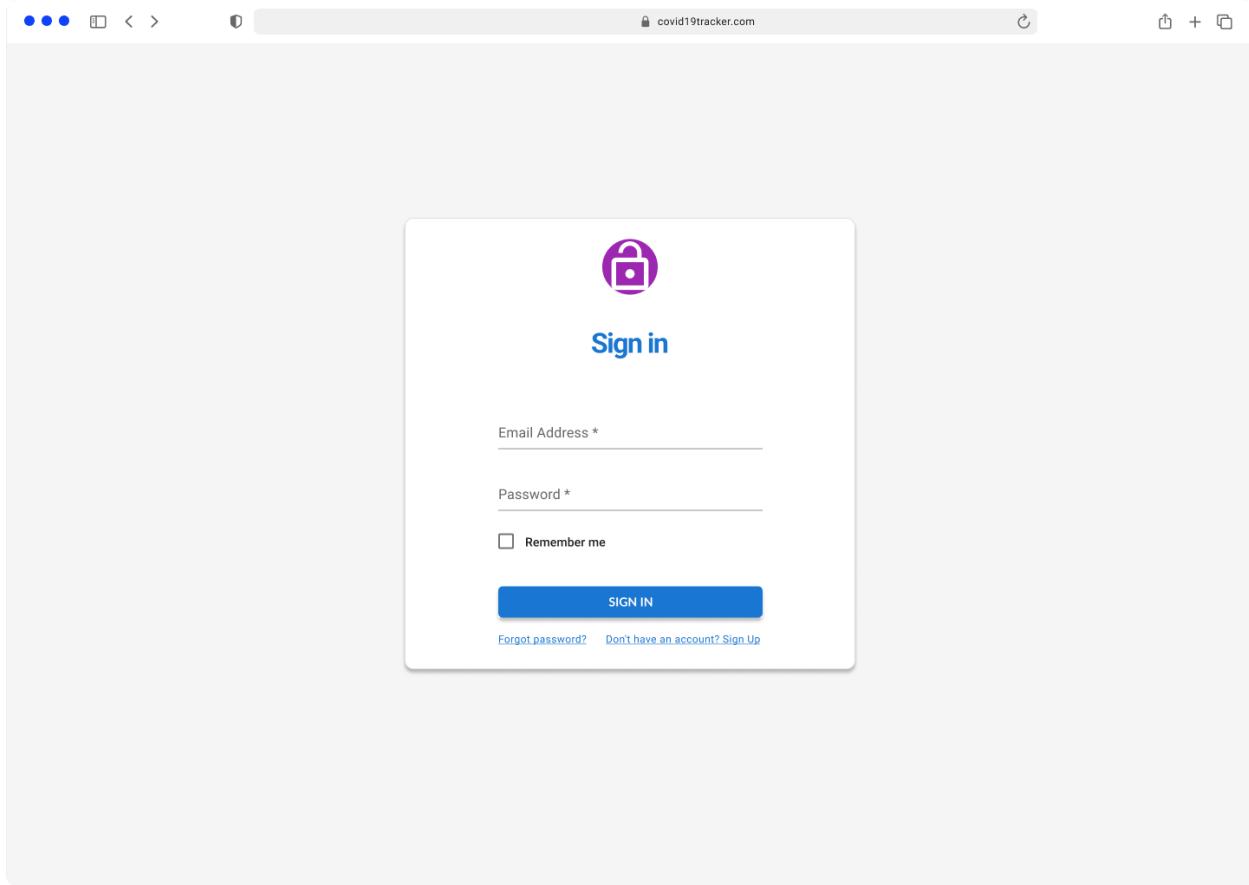


Figure 22: UI Admin Sign-in for User Story [Issue-7](#)

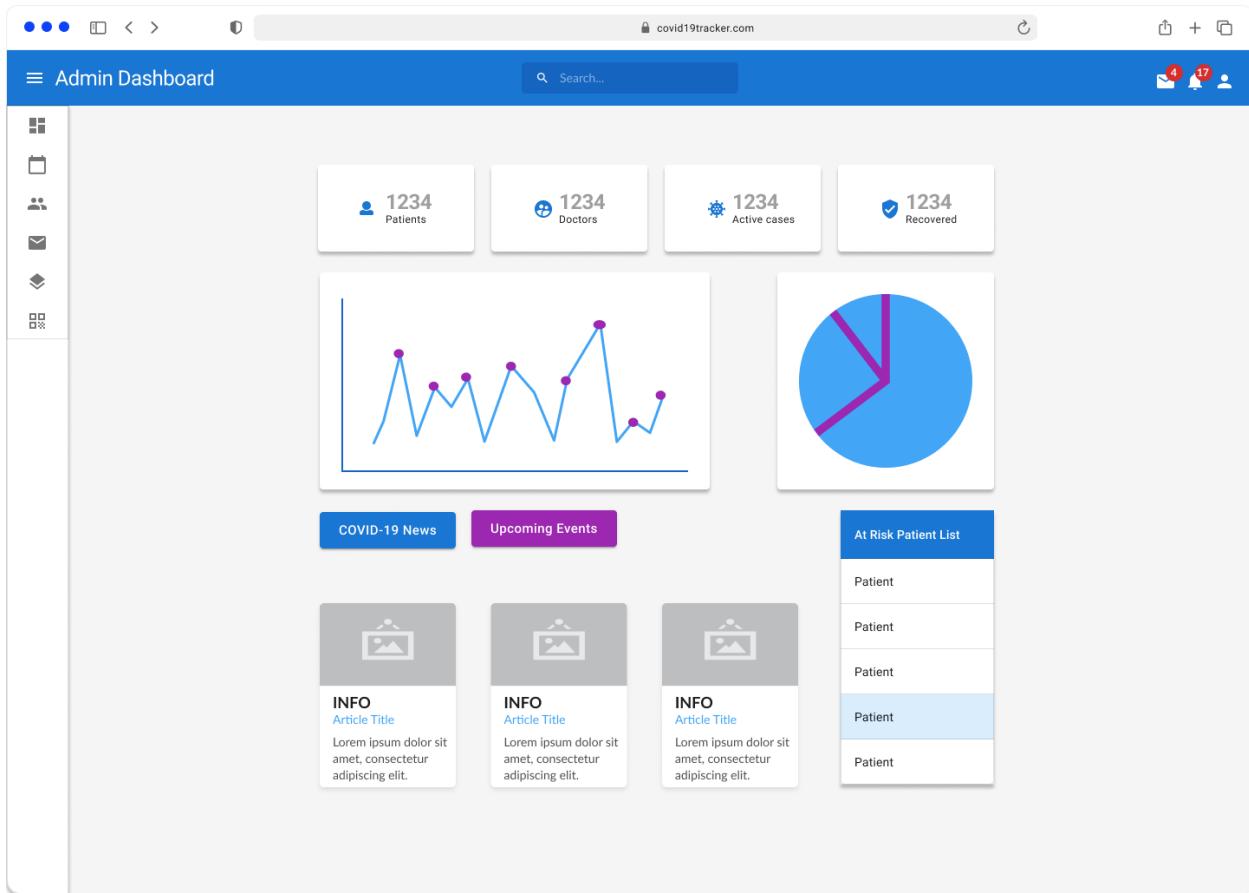


Figure 23: UI Admin Dashboard for User Story [Issue-5](#)

The screenshot shows the Admin Dashboard for covid19tracker.com. The main title is "Appointments" and the date is "January 2022". The calendar grid shows several events: "event 1" on Jan 2, "event 4" from Jan 4 to Jan 6, "event 5" from Jan 13 to Jan 14, "event 3" from Jan 18 to Jan 19, and "event 2" from Jan 20 to Jan 21. The sidebar on the left includes icons for Home, Calendar, People, Mail, and Reports.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	31	1
2 event 1	3	4	5 event 4	6	7	8
9	10	11	12	13 event 5	14	15
16	17	18	19 event 3	20	21 event 2	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Figure 24: UI Admin Appointments for User Story [Issue-5](#)

Patient List

PATIENT NAME	ID	STATUS	UPCOMING APPOINTMENT	ASSIGNED DOCTOR	FLAGGED PRIORITY
Cell	Cell	POSITIVE	Cell	Cell	<input checked="" type="checkbox"/>
Cell	Cell	NEGATIVE	Cell	Cell	<input type="checkbox"/>

Symptoms

Temperature (°C)	Weight (lbs)	Height (feet)
Cell	Cell	Cell

PATIENT NAME	ID	STATUS	UPCOMING APPOINTMENT	ASSIGNED DOCTOR	FLAGGED PRIORITY
Cell	Cell	POSITIVE	Cell	Cell	<input type="checkbox"/>
Cell	Cell	POSITIVE	Cell	Cell	<input type="checkbox"/>

Rows per page: 10 1-5 of 13 < >

Figure 26: UI Admin Patient List for User Story [Issue-89](#)

The screenshot shows the Admin Dashboard for covid19tracker.com. The top navigation bar includes a search bar and notification icons for 4 messages and 17 pending actions. The left sidebar has a navigation menu with icons for Home, Patients, Doctors, Reports, and Settings.

Patient Details:

- Profile:** A circular icon of a woman with glasses and a pink background.
- Name:** JANE DOE
- Age:** 50
- Birthday:** 1 July 1957
- Address:** 101 Brooke, Montréal, L5L 9T9

Status:

- Confirmation: Unconfirmed (dropdown menu)
- POSITIVE (button)
- Temperature: 39°C
- Weight: 154 lbs

Assigned Doctor: Dr. Micheal Scott

Status Review: Review Complete:

Symptom Details:

Date	Fever	Cough	Runny Nose	Muscle Ache	Tiredness	Smell Loss	Taste Loss
Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell	Cell

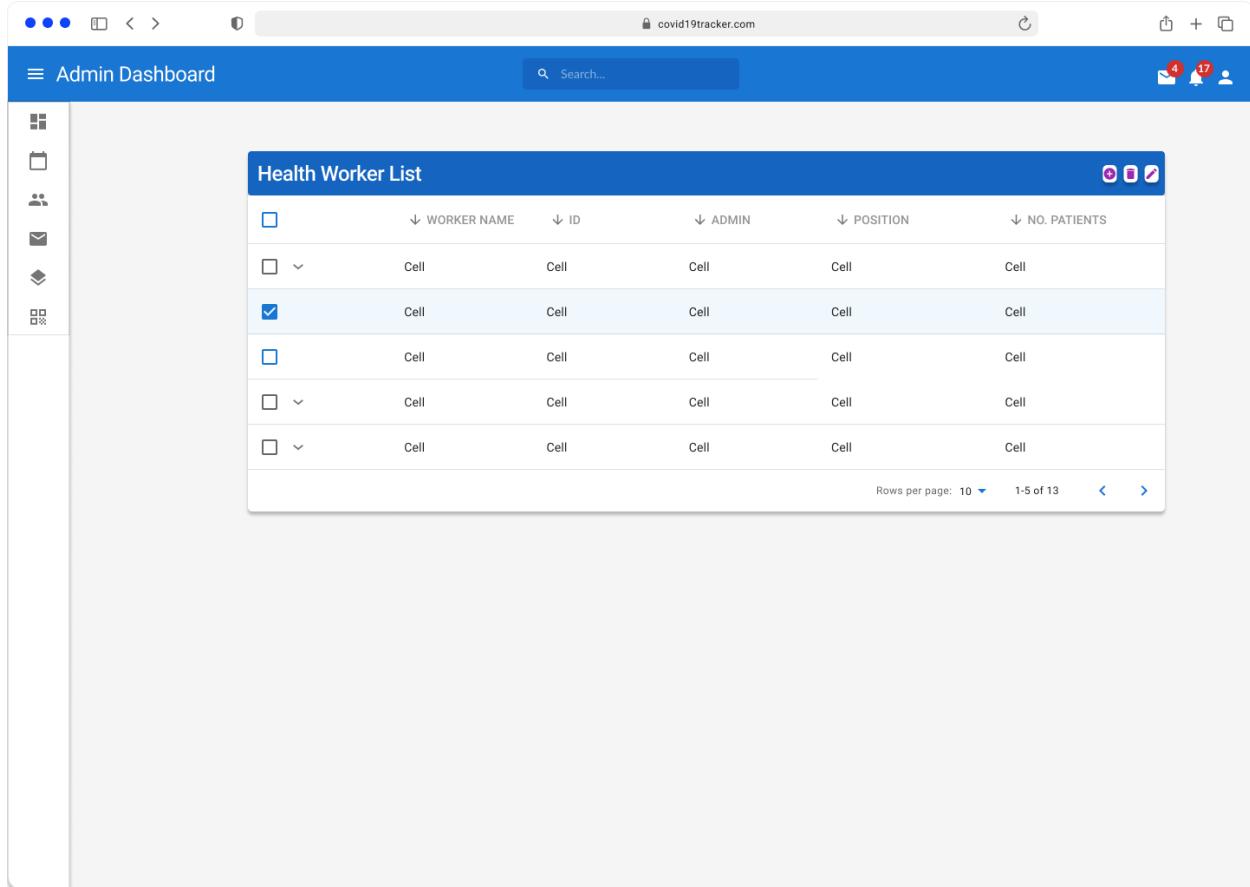
Figure 26: UI Admin Patient Details for User Story [Issue-9](#)

7.2 Sprint 2 UI Mockups

Here are the UI mockups of user stories completed in Sprint 2.

7.2.1 Admin Portal Sprint 2

The following figures are the UI mockups for the Admin portal.



A screenshot of a web browser displaying the Admin Dashboard for covid19tracker.com. The dashboard features a blue header with the title "Admin Dashboard" and a search bar. On the left, there is a sidebar with various icons. The main content area is titled "Health Worker List" and contains a table with six columns: WORKER NAME, ID, ADMIN, POSITION, and NO. PATIENTS. The table has six rows, with the second row containing a checked checkbox. At the bottom of the table, there are pagination controls showing "Rows per page: 10" and "1-5 of 13".

	WORKER NAME	ID	ADMIN	POSITION	NO. PATIENTS
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell
<input checked="" type="checkbox"/>	Cell	Cell	Cell	Cell	Cell
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell

Figure 27: UI Admin Health Worker List for User Story [Issue-17](#) and [Issue-19](#)

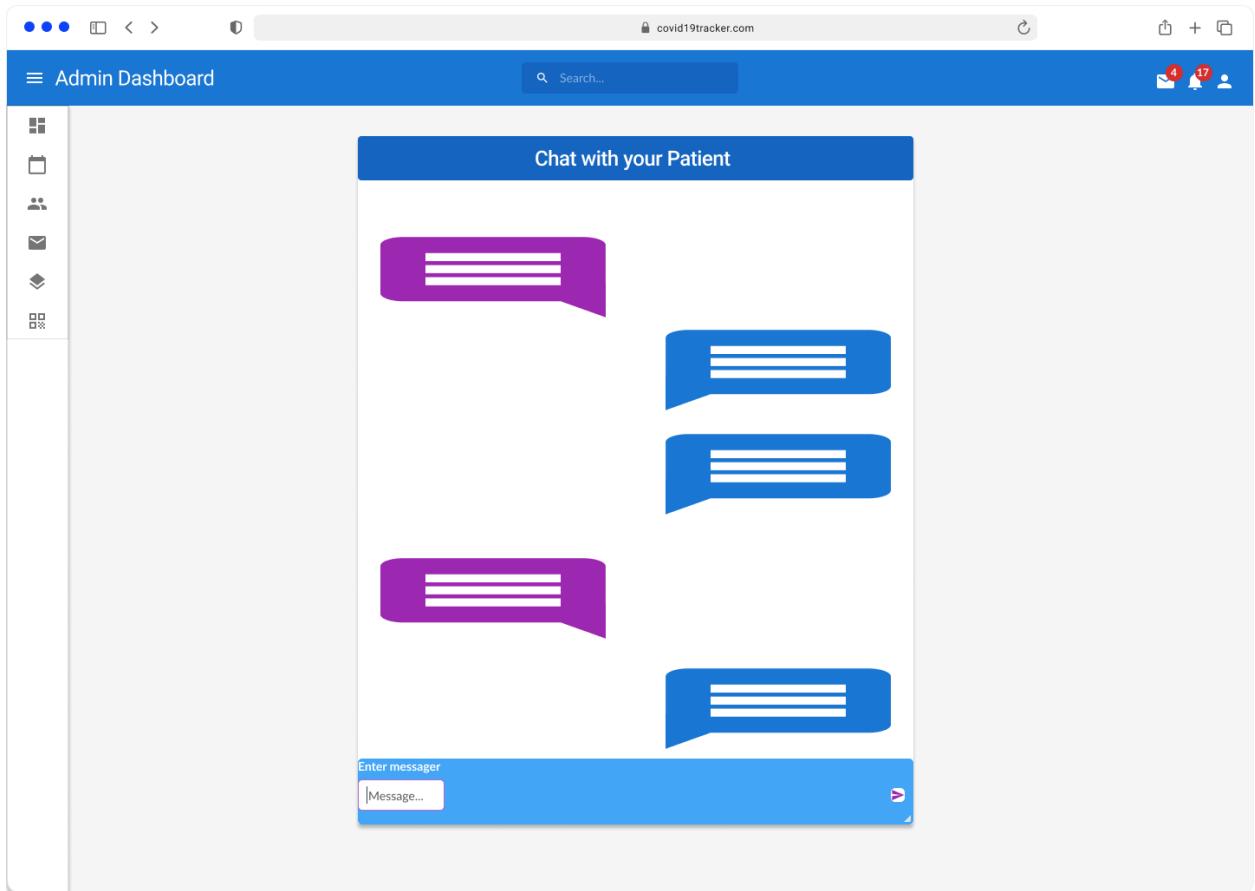


Figure 28: UI Admin Chat with patient for User Story [Issue-13](#)

The screenshot shows the Admin Dashboard interface for covid19tracker.com. On the left is a vertical sidebar with icons for Home, Calendar, Patients, Messages, and Reports. The main area has a blue header bar with the title 'Admin Dashboard' and a search bar. In the center, there's a circular profile picture of a woman with glasses and pink hair, labeled 'JANE DOE'. Below the profile are her details: Age: 50, Birthday: 1 July 1957, Address: 101 Brooke, Montréal, L5L 9T9. To the right of the profile is a 'Flagged' status indicator. Below it is a card with 'STATUS' (Unconfirmed) and 'Confirmation' (Positive). It also shows 'Temperature: 39°C' and 'Weight: 154 lbs'. Further down are sections for 'ASSIGNED DOCTOR' (Dr. Julie Joy) and 'STATUS REVIEW' (Review Complete:). At the bottom is a table titled 'SYMPTOM DETAILS' with columns for Date, Fever, Cough, Runny Nose, Muscle Ache, Tiredness, Smell Loss, and Taste Loss. The table contains several rows of data, with the third row having a checked checkbox in the Date column.

Date	Fever	Cough	Runny Nose	Muscle Ache	Tiredness	Smell Loss	Taste Loss
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell	Cell	Cell
<input checked="" type="checkbox"/>	Cell	Cell	Cell	Cell	Cell	Cell	Cell
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell	Cell	Cell
<input type="checkbox"/>	Cell	Cell	Cell	Cell	Cell	Cell	Cell

Figure 29: UI Patient Profile where you can edit symptoms, re-assign doctor, view if patient is flagged, review status, and edit patient info, for User Stories [Issue-3](#), [Issue-16](#), [Issue-18](#), [Issue-20](#).

7.2.2 Client Portal Sprint 2

The following figures are the UI mockups for the Client portal.

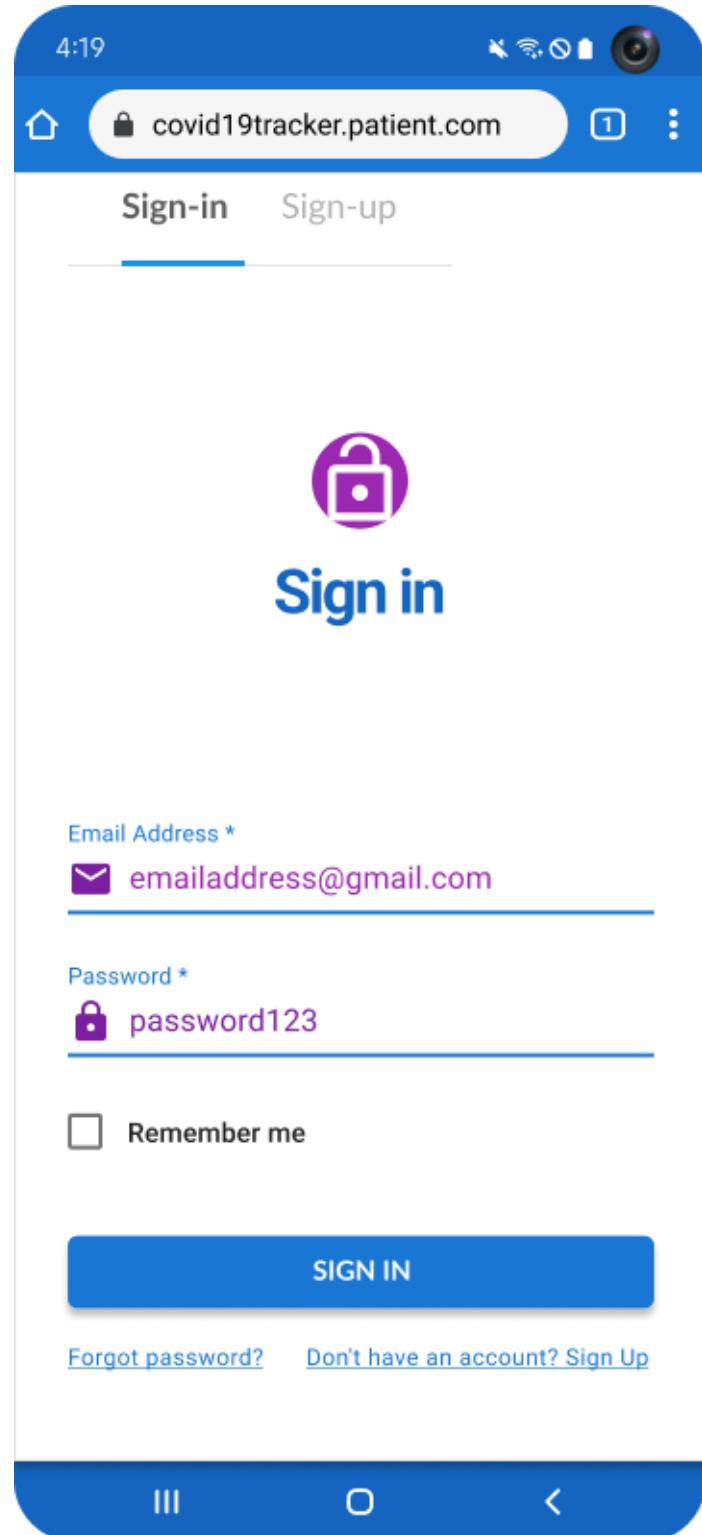


Figure 30: UI Client Sign-in for User Story [Issue-7](#)

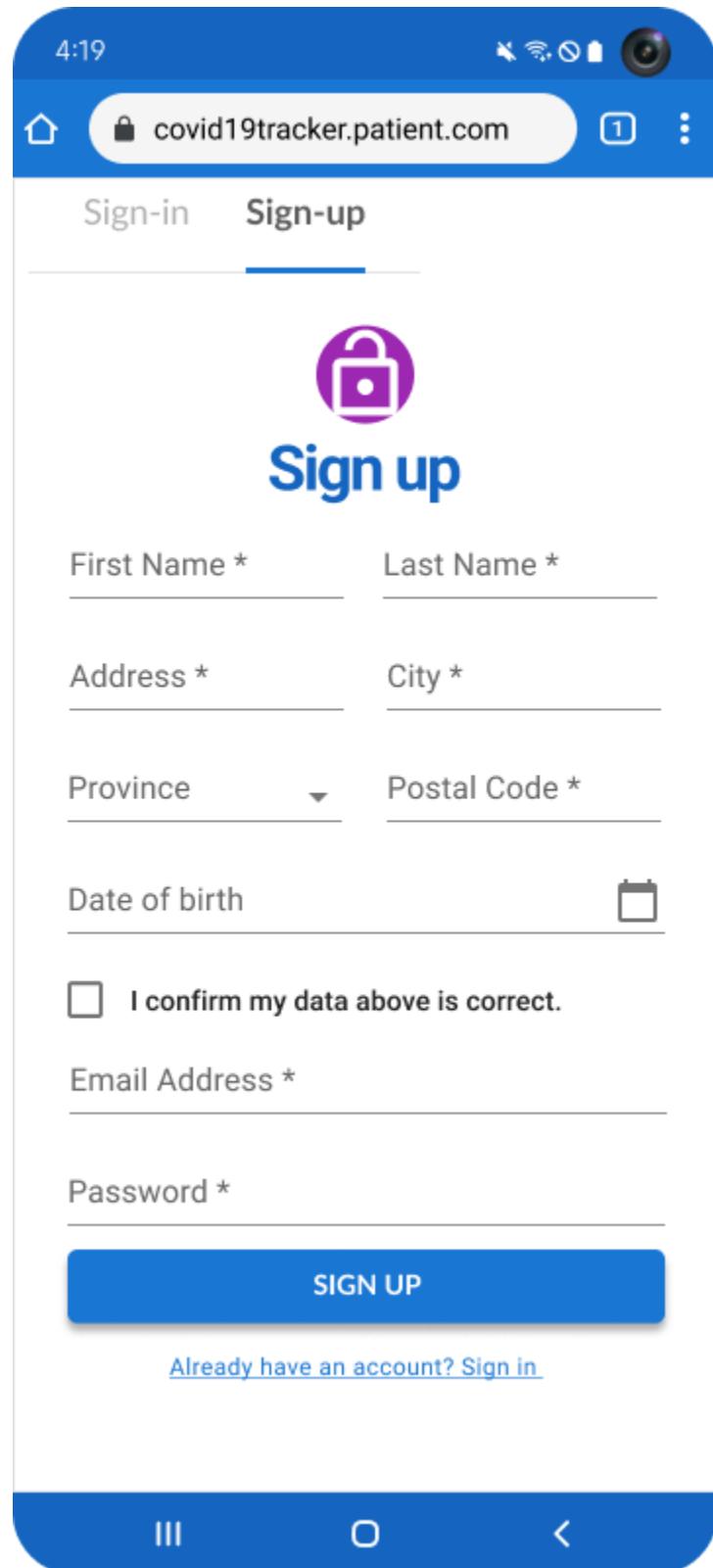


Figure 31: UI Client Sign-up for User Story [Issue-7](#)

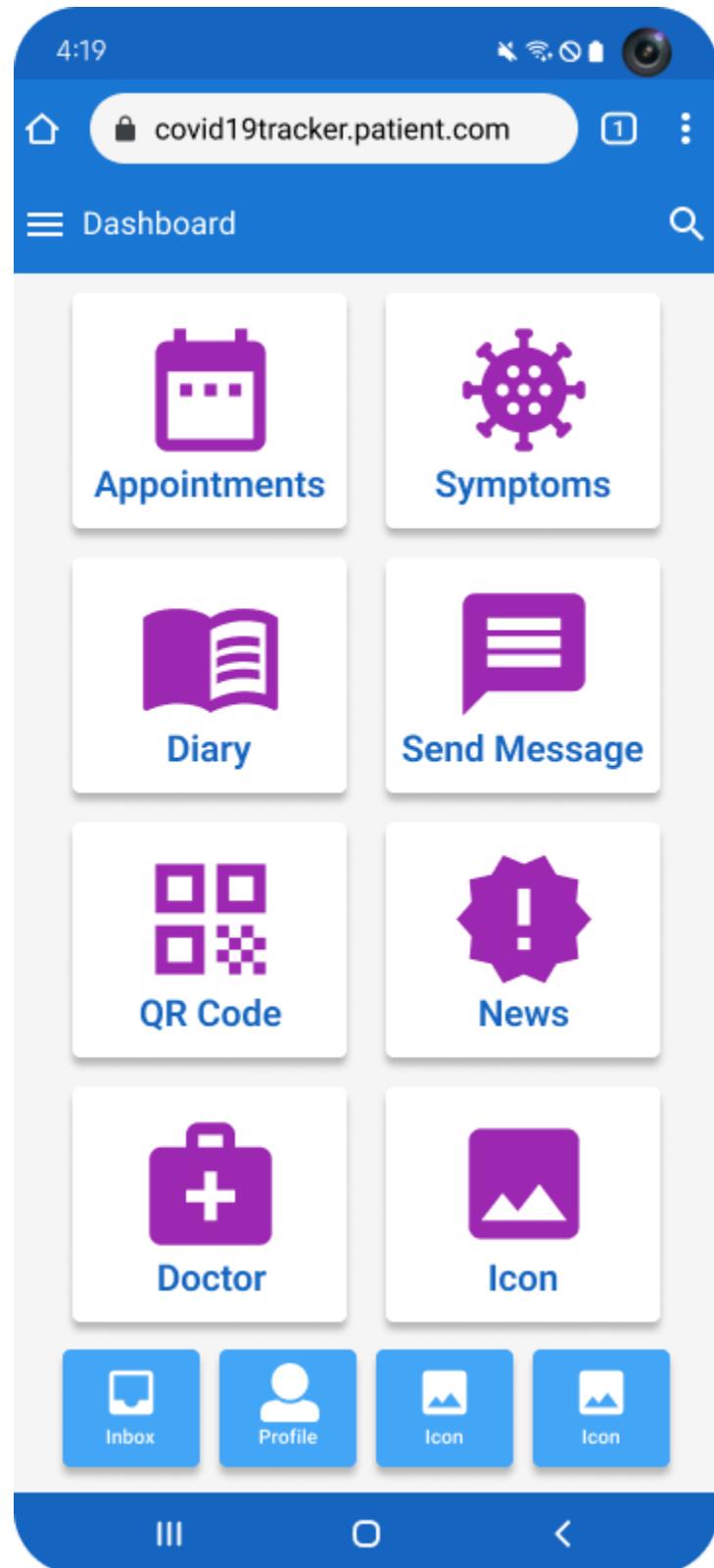


Figure 32: UI Client Dashboard

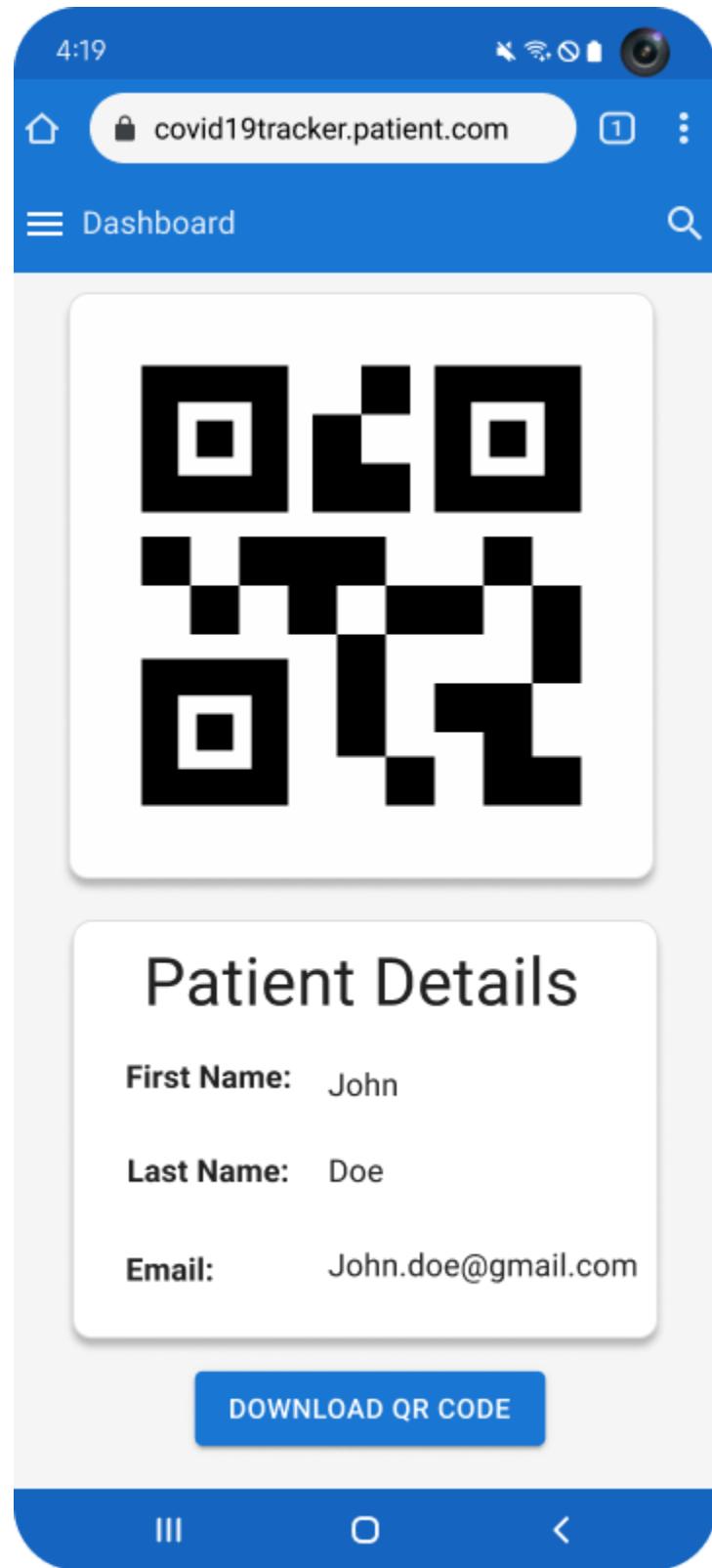


Figure 33: UI Client QR code for User Story [Issue-22](#)

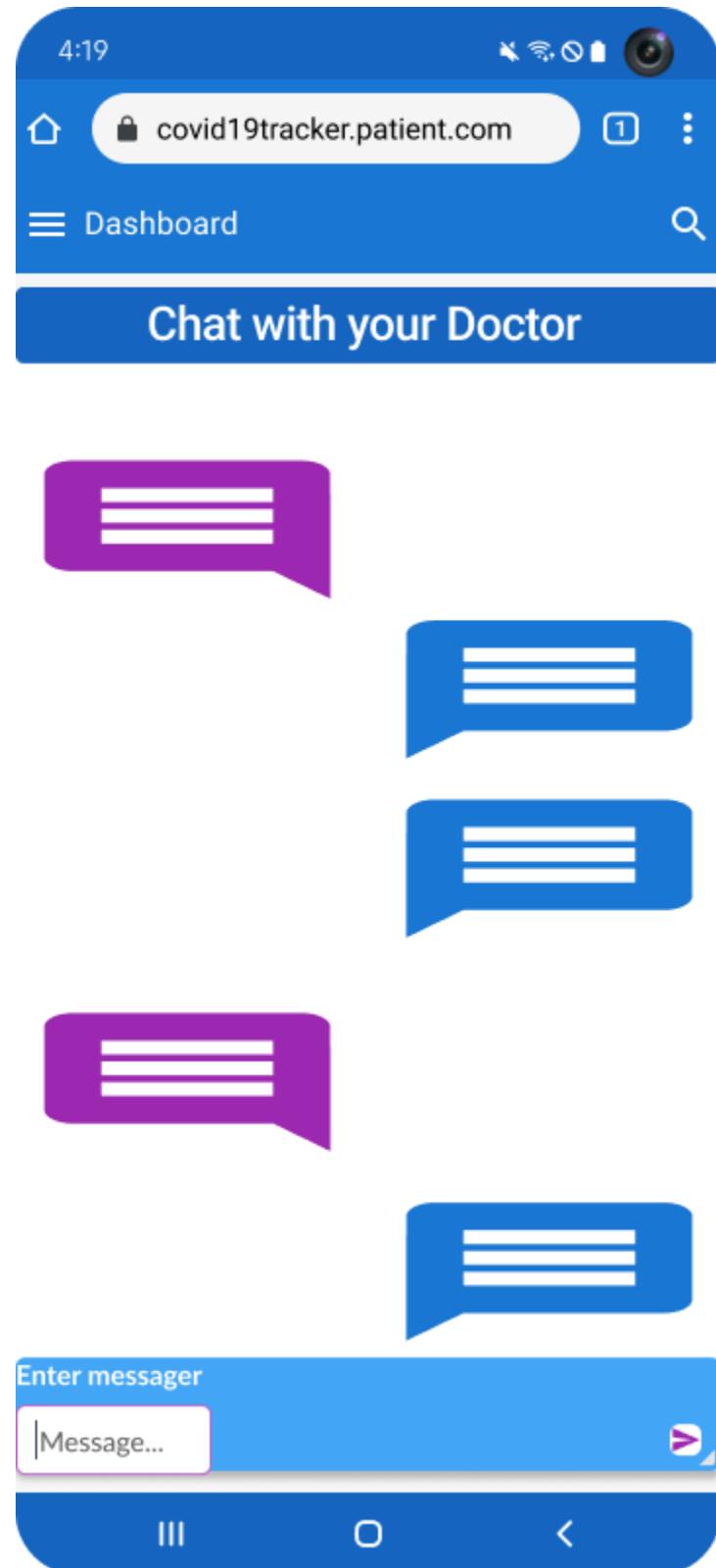


Figure 34: UI Client Chat with patient for User Story [Issue-13](#)

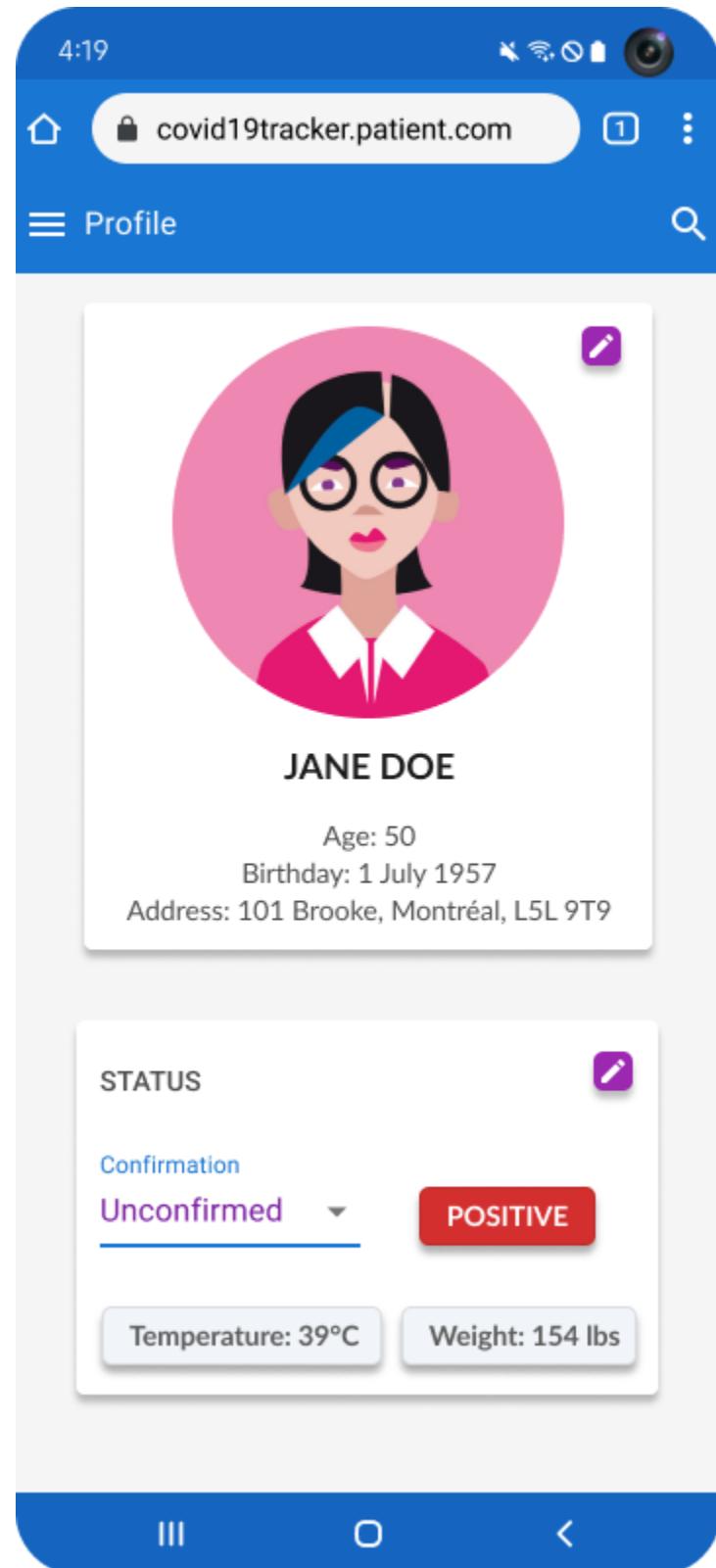


Figure 35: UI Client Patient for User Story [Issue-37](#)

7.3 Sprint 3 UI Mockups

Here are the UI mockups of user stories completed in Sprint 3.

7.3.1 Dark Theme Update

During the course of sprint 2, a styling decision was made to opt for a dark mode palette. The following figure illustrates the palette selection.

Colors



Figure 36: Theme Palette

Here are the previous sprints' mockups supporting the new dark theme.

7.3.1.1 Dark Theme Update Admin Portal

The screenshot shows a dark-themed user interface for an admin portal. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the URL "covid19tracker.com". On the right side of the header are notification icons for messages, alerts, and users. Below the header, the main content area has a title "APPOINTMENTS" and a search bar. A sidebar on the left contains icons for dashboard, calendar, users, messages, and other system functions. The central part of the screen is a "Appointments" section with a title, a date range from "Today" to "January 2022", and a "Month", "Week", or "Day" view selector. The main feature is a monthly calendar grid for January 2022. The days of the week are labeled "Sun" through "Sat". Specific dates are highlighted with blue boxes: "event 1" on Monday, "event 4" spanning Tuesday through Friday, "event 5" spanning Saturday through Monday, "event 3" on Wednesday, and "event 2" on Saturday.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Figure 37: UI Admin Appointments for User Story [Issue-5](#)

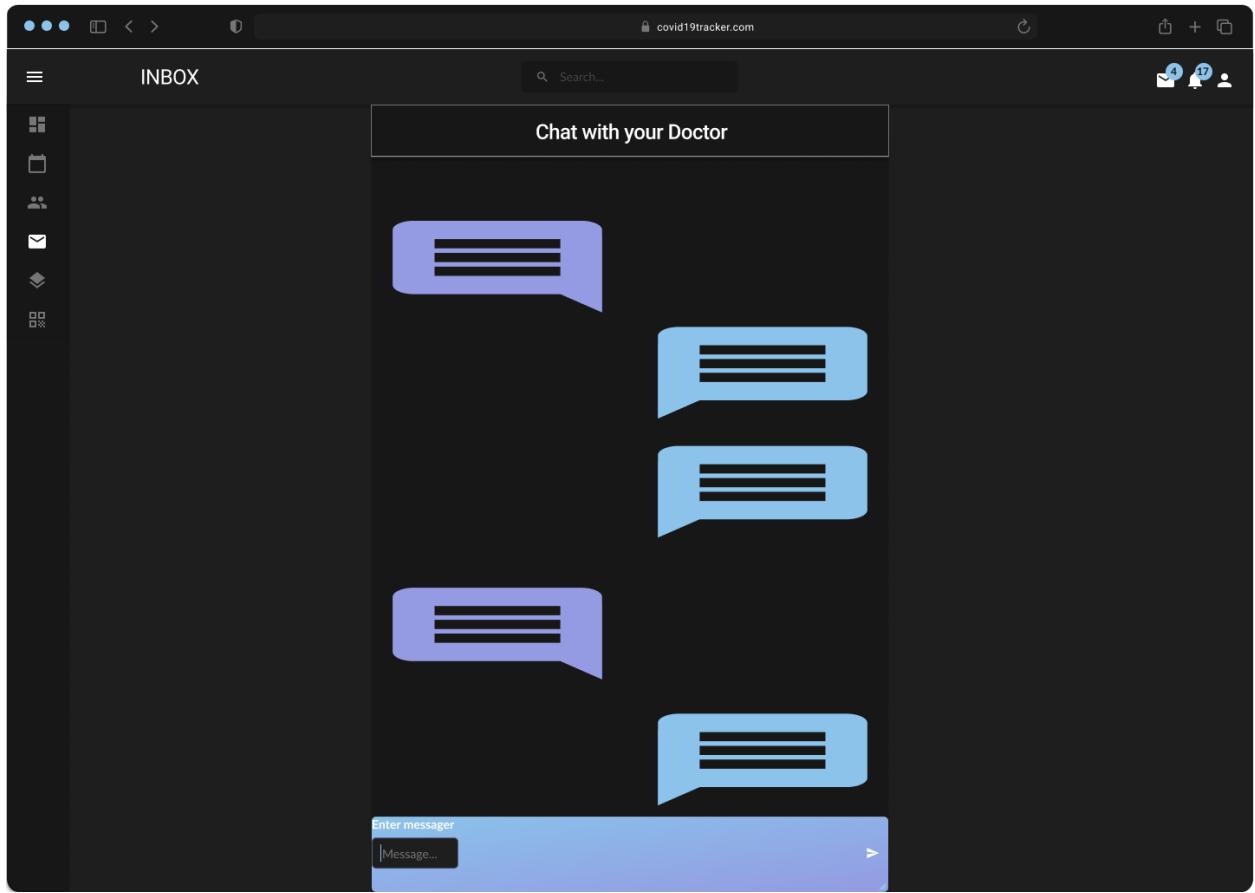


Figure 38: UI Admin Chat with patient for User Story [Issue-13](#)

The screenshot shows the 'PATIENTS' section of the COVID-19 Tracker application. At the top, there's a navigation bar with icons for home, search, and user profile, along with a notification badge showing 17 notifications.

The main area is titled 'Patient List'. It features a table with columns: PATIENT NAME, ID, STATUS, UPCOMING APPOINTMENT, ASSIGNED DOCTOR, and FLAGGED PRIORITY. The 'STATUS' column has two buttons: 'POSITIVE' (highlighted in blue) and 'NEGATIVE'. The 'FLAGGED PRIORITY' column contains two flag icons.

Below the table, there's a section titled 'Symptoms' with three columns: Temperature (°C), Weight (lbs), and Height (feet). Each column has a 'Cell' placeholder.

At the bottom, there's a 'Doctors' section with a table:

Doctors	Patient No.
Dr.	4/10
Dr.	3/15
Dr.	10/15

Navigation controls at the bottom right include 'Rows per page: 10', '1-5 of 13', and arrows for page navigation.

Figure 39: UI Admin Patient List for User Story [Issue-89](#)

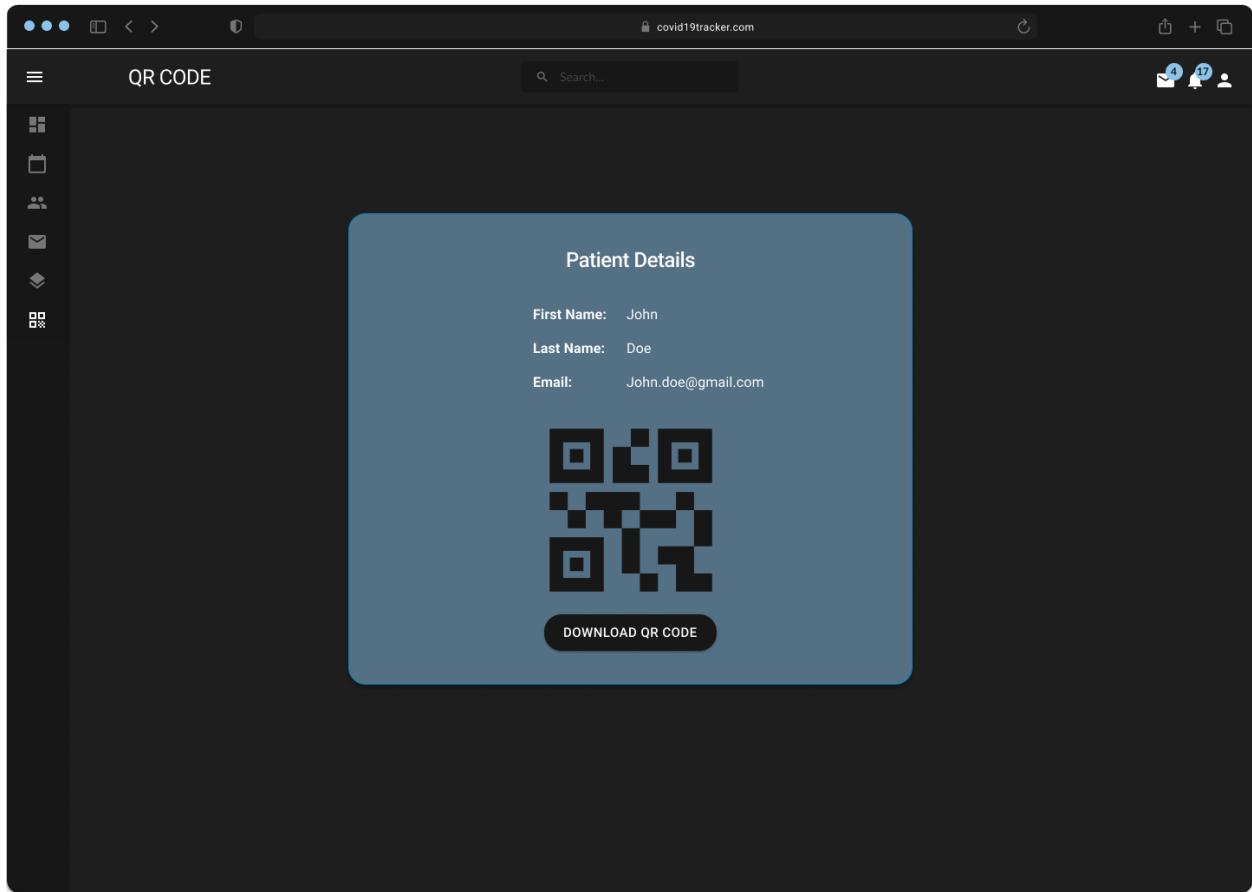
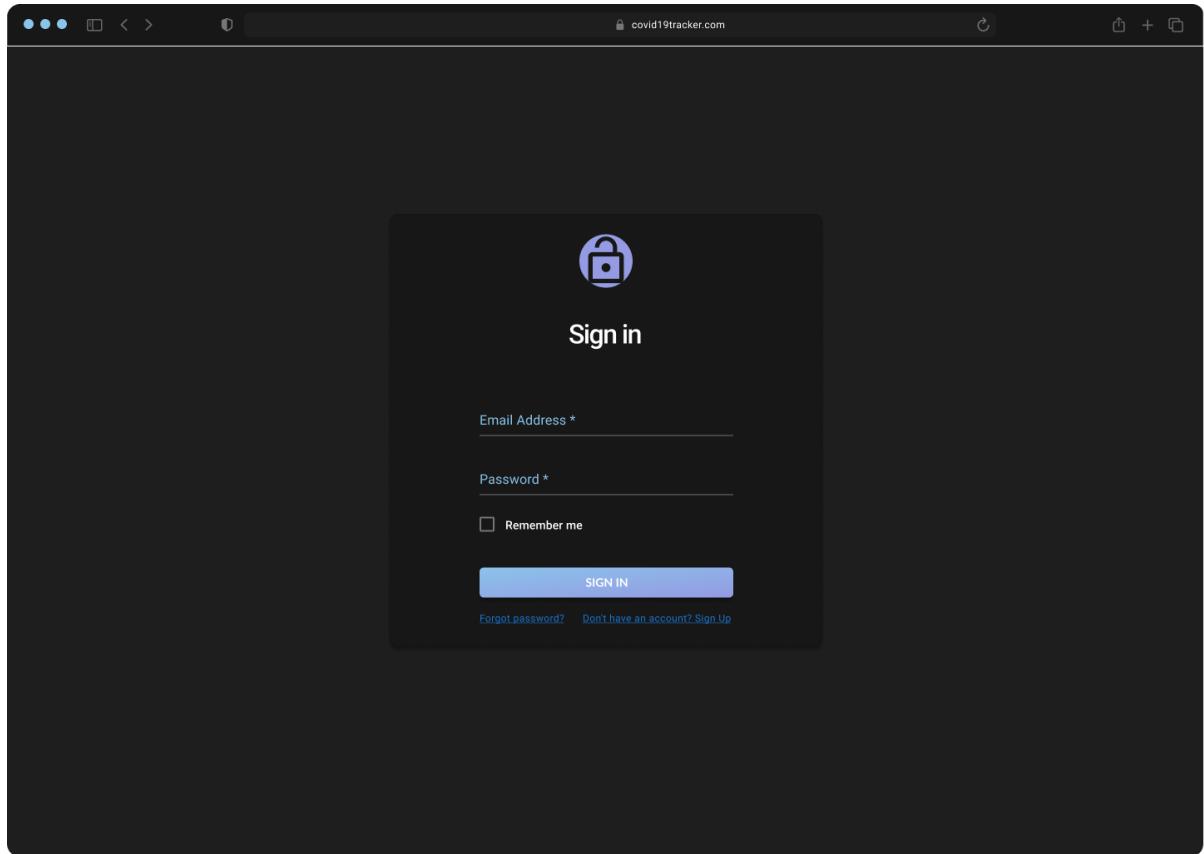


Figure 40: UI Client QR code for User Story [Issue-22](#)



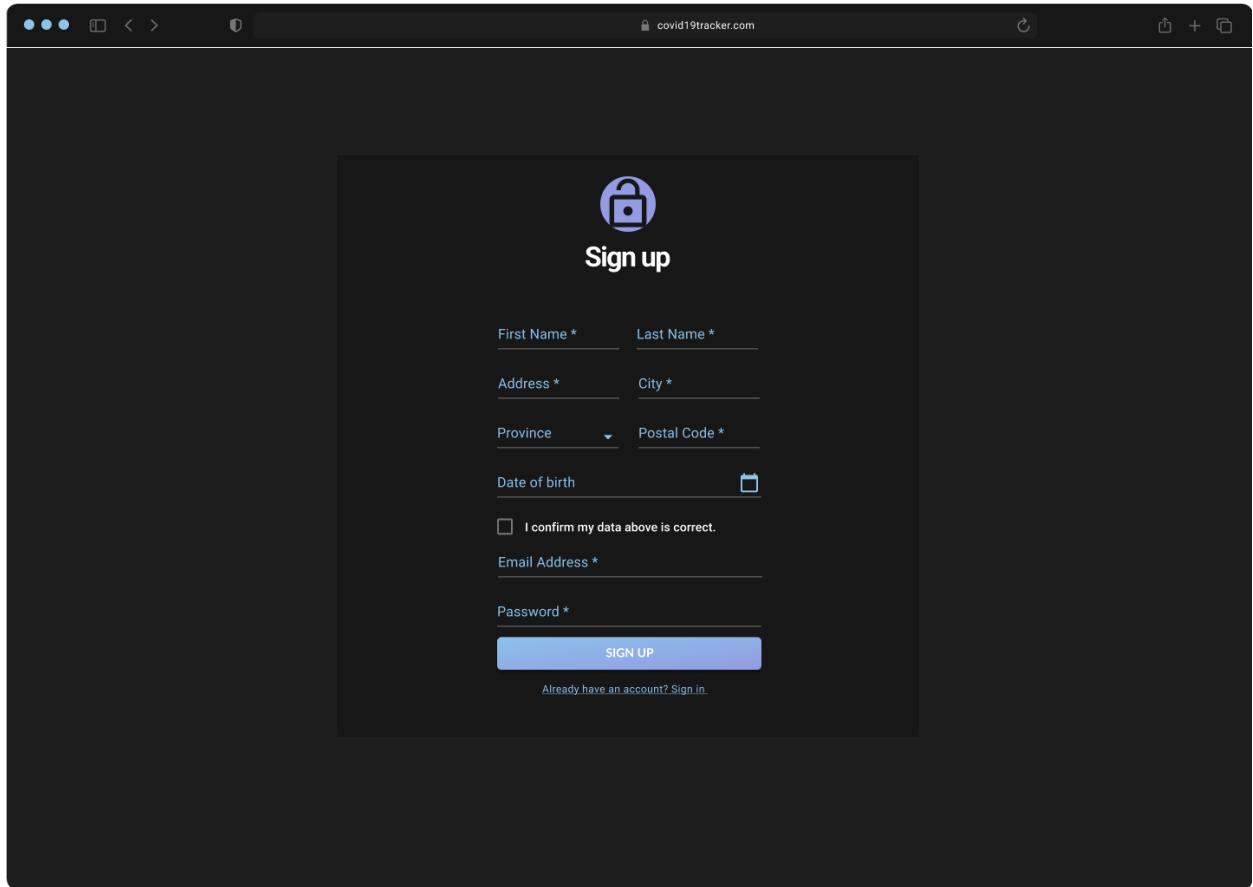


Figure 42:

7.3.1.2 Dark Theme Update Client Portal

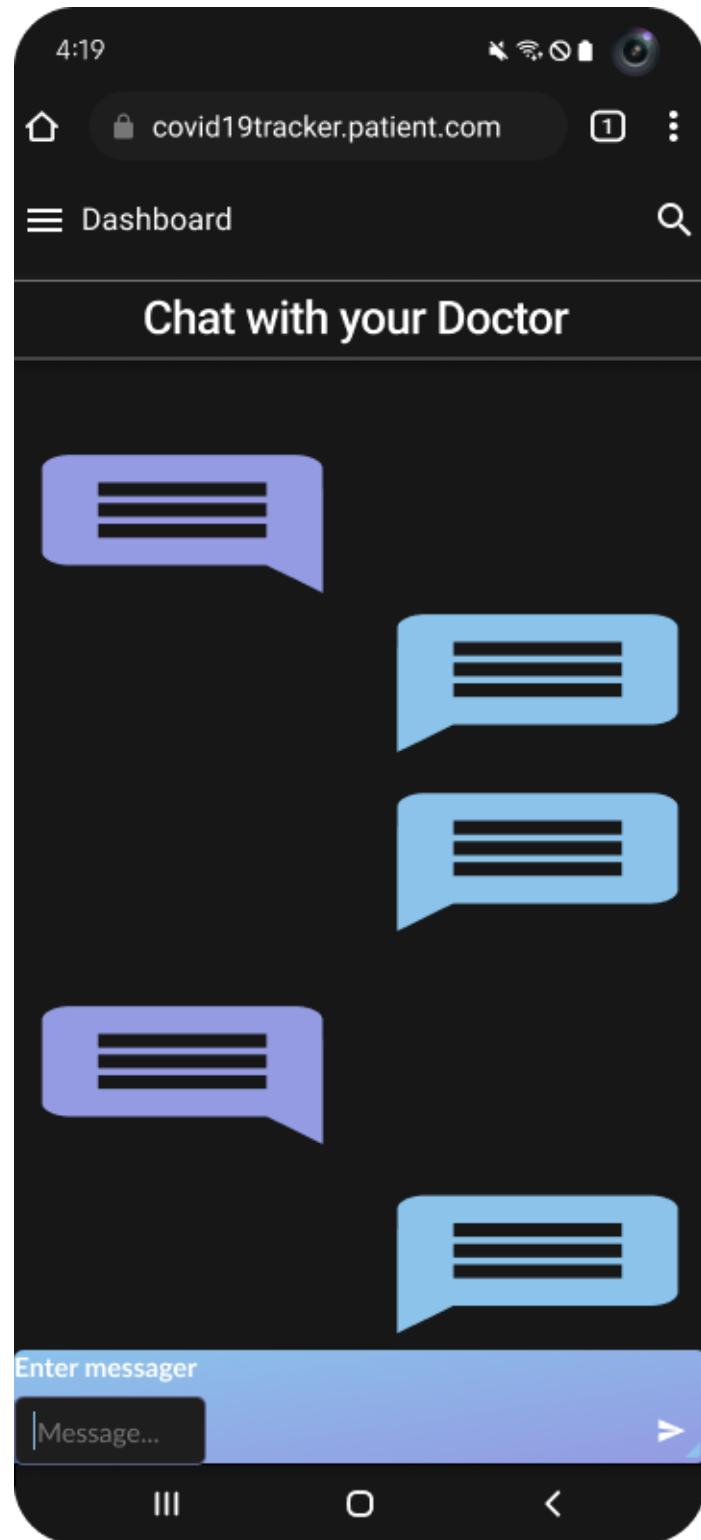


Figure 43: UI Patient Chat

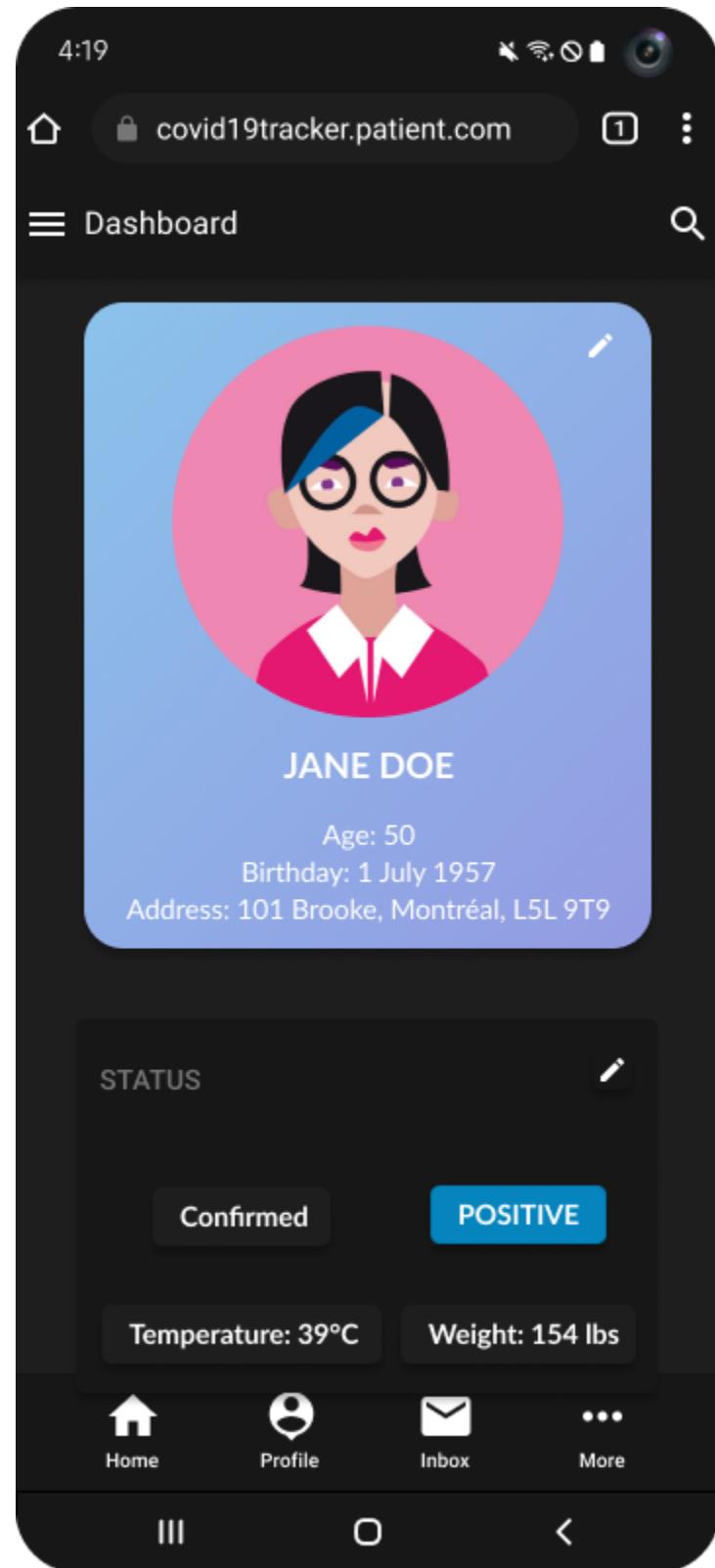


Figure 44: UI Patient Profile

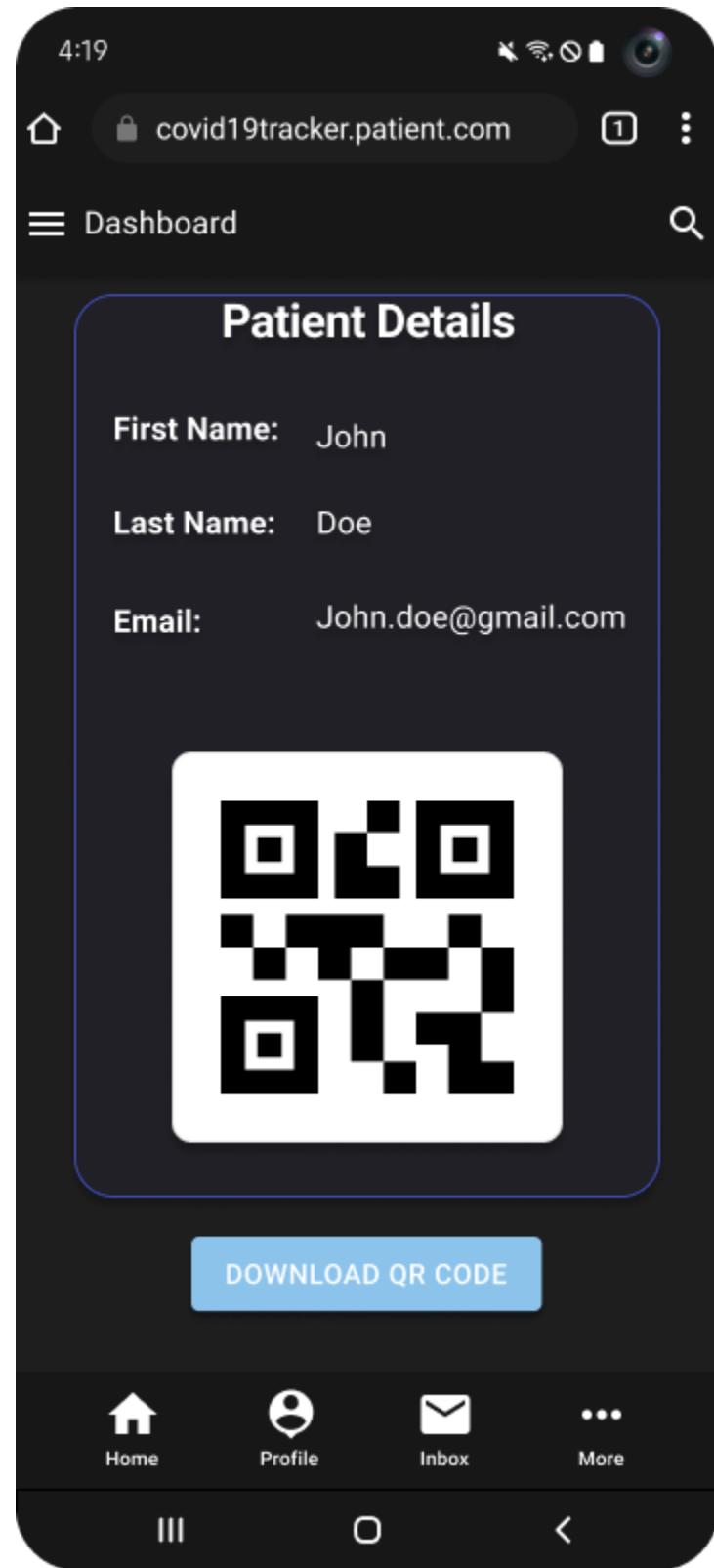


Figure 45: UI Patient QR Code

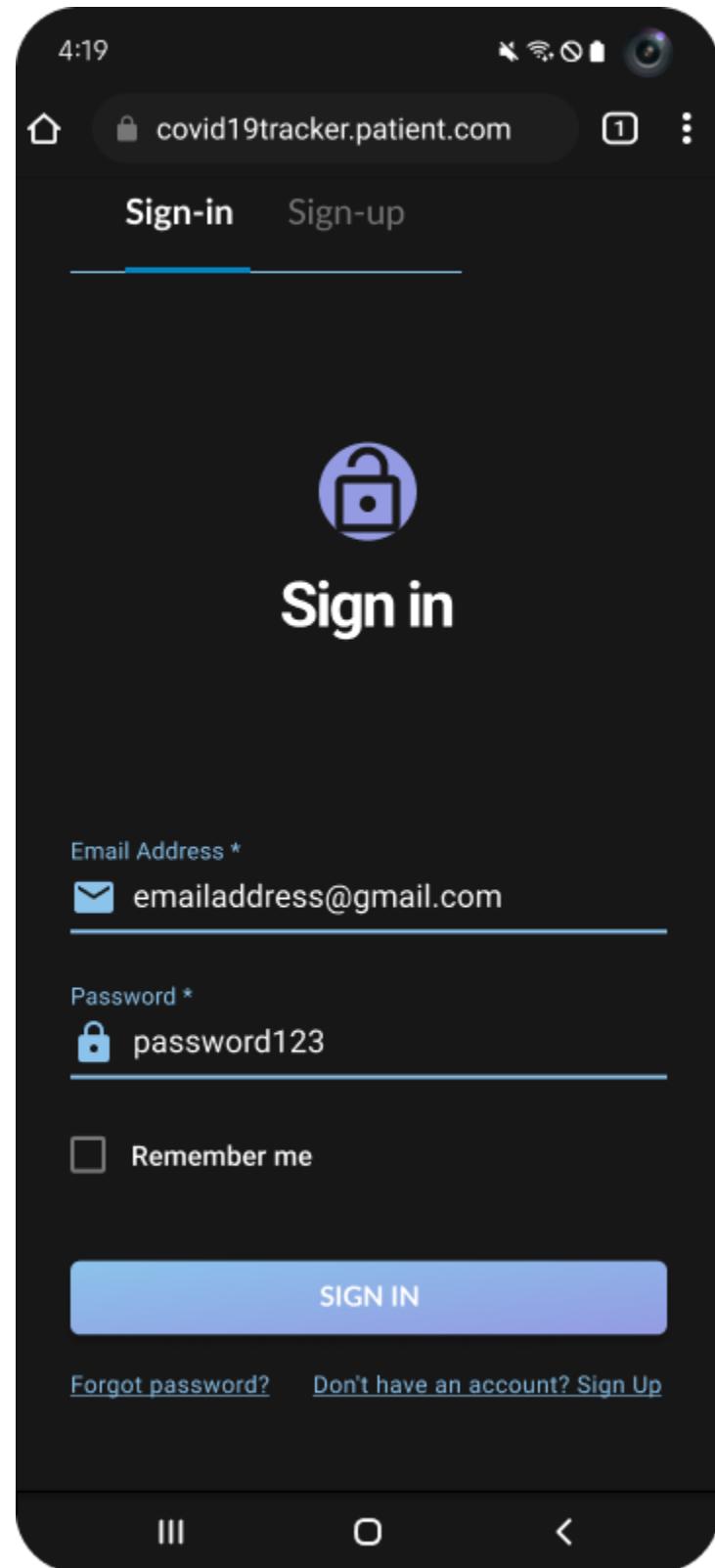


Figure 46: UI Client App Sign In

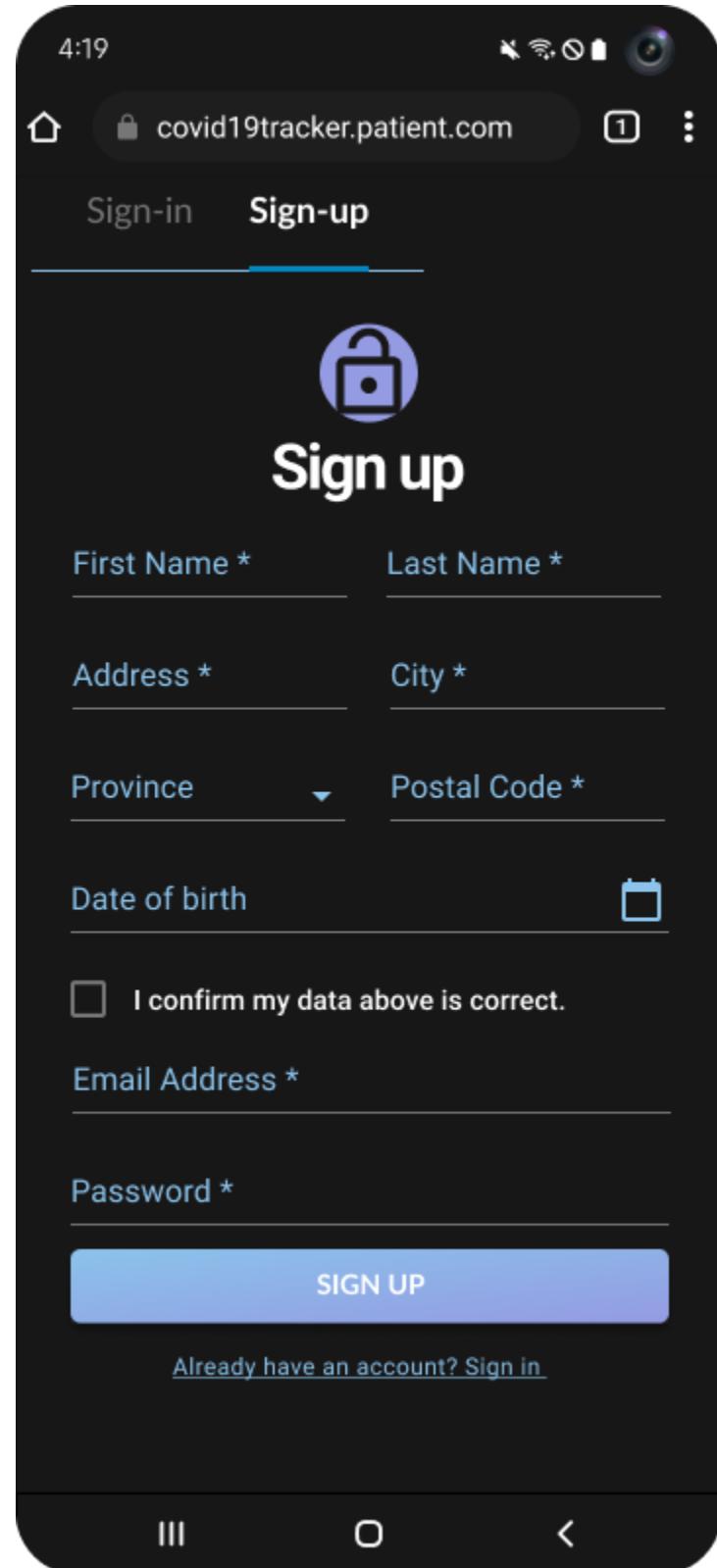


Figure 47: UI Client App Sign In

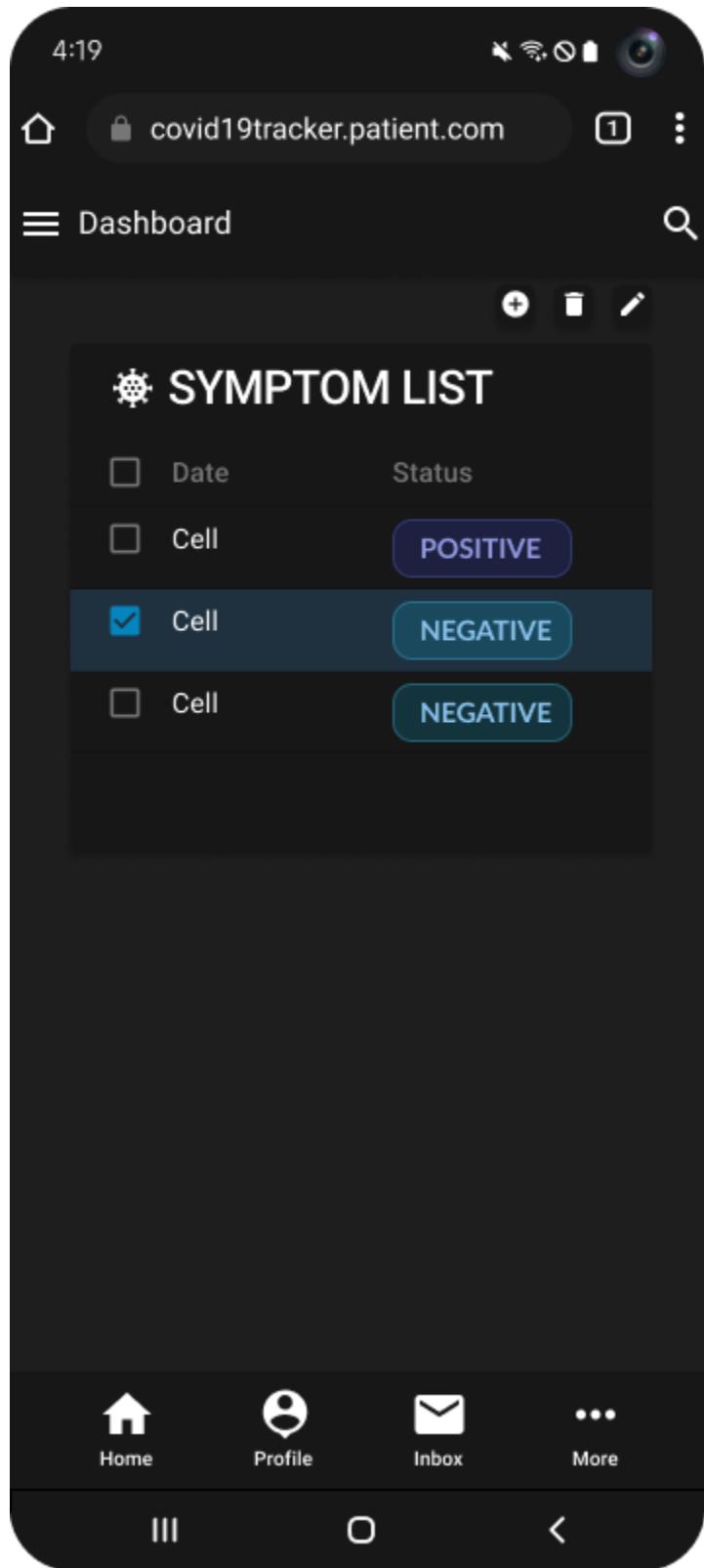
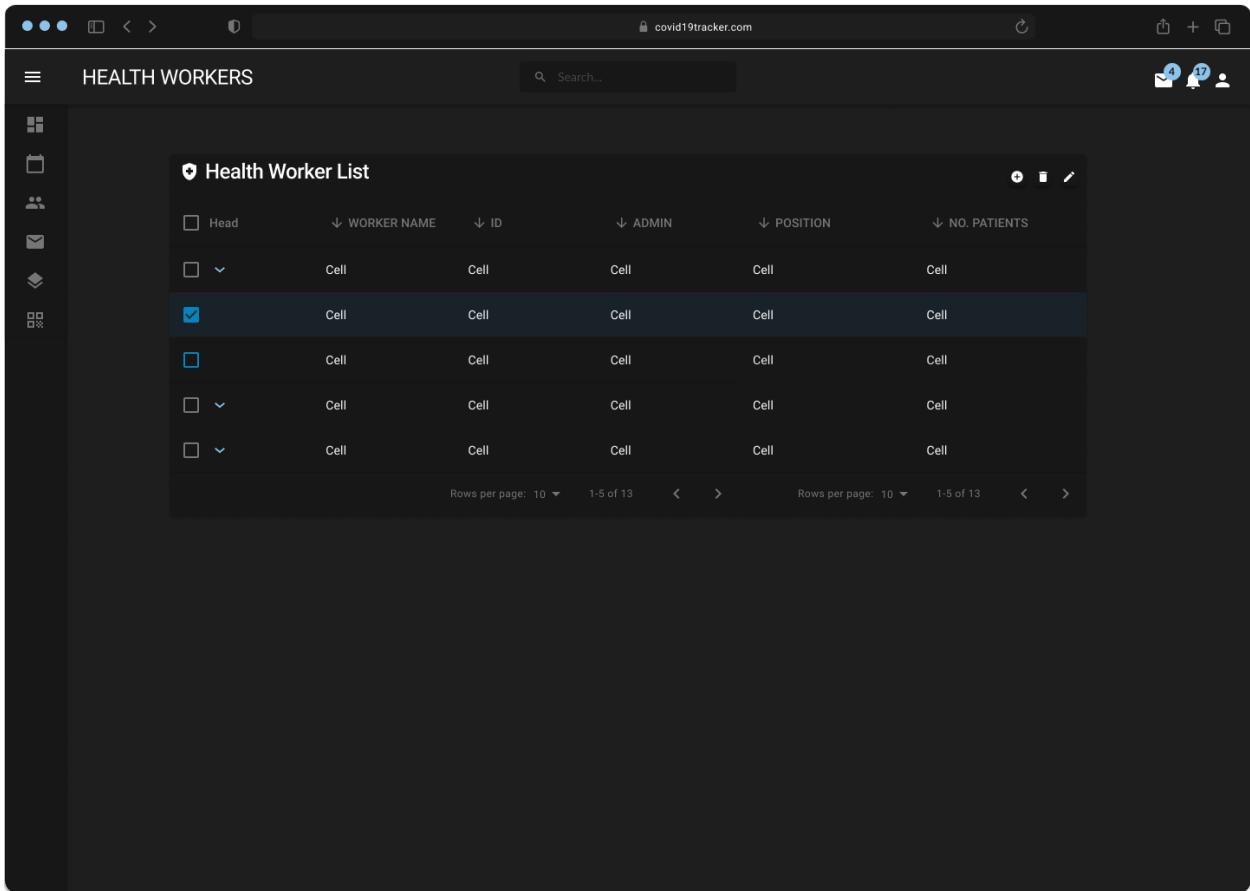


Figure 48: UI Client Symptoms List

7.3.2 Admin Portal Sprint 3

The following figures are the UI mockups for the Admin portal.



A screenshot of a web browser displaying the 'Health Worker List' page from the 'covid19tracker.com' website. The page has a dark theme. On the left, there is a vertical sidebar with icons for Home, Dashboard, Reports, and Settings. The main header says 'HEALTH WORKERS'. Below it is a search bar with placeholder text 'Search...'. The main content area is titled 'Health Worker List' with a shield icon. It features a table with the following columns: Head, WORKER NAME, ID, ADMIN, POSITION, and NO. PATIENTS. The first row is labeled 'Head' and includes a checkbox column. The second row has a checked checkbox in the first column. The third row has an unchecked checkbox in the first column. The fourth and fifth rows have unchecked checkboxes in the first column. At the bottom of the table, there are pagination controls: 'Rows per page: 10', '1-5 of 13', and arrows for navigating between pages. There are also buttons for 'Add', 'Edit', and 'Delete' at the top right of the table.

Figure 49: UI Admin Health Worker List for User Story [Issue-19](#)

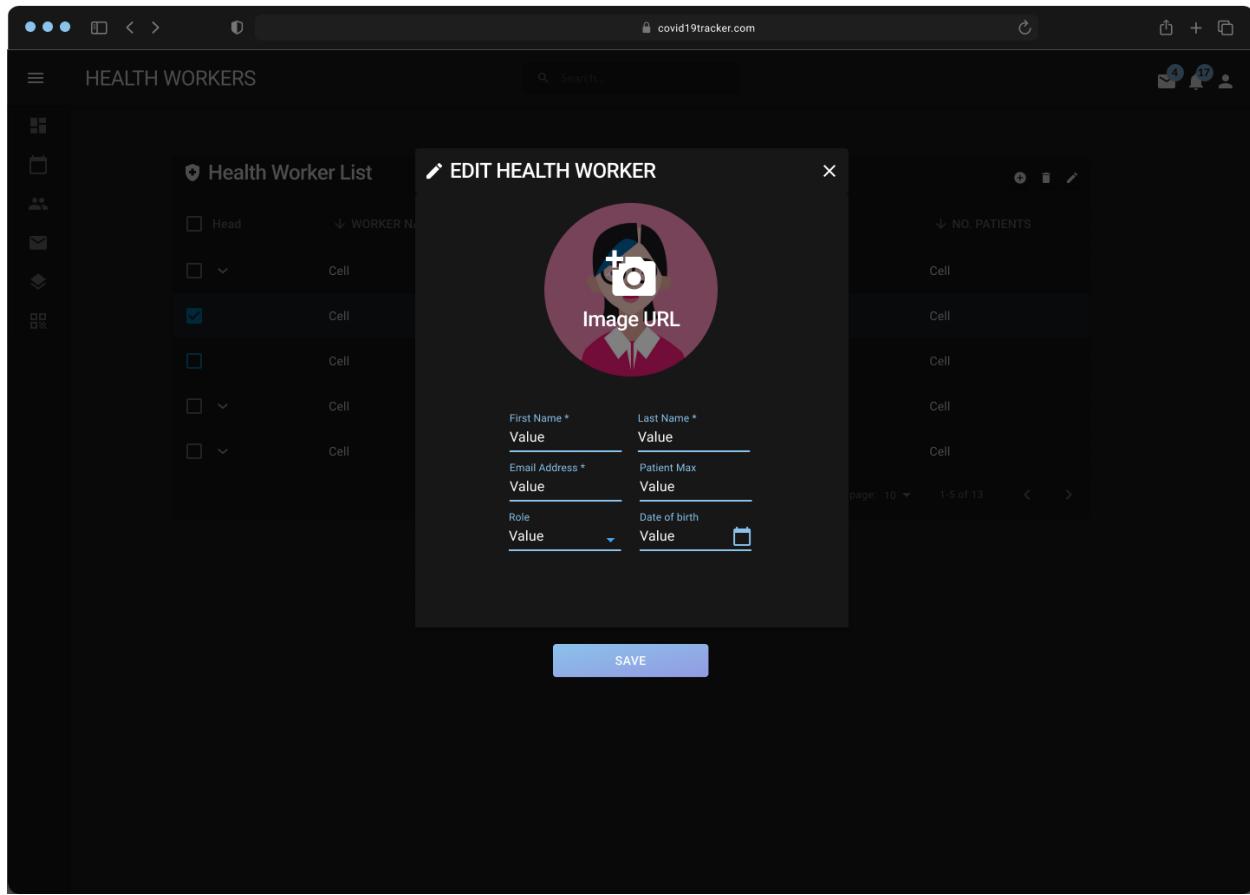


Figure 50: UI Admin Health Worker Edit for User Story [Issue-19](#)

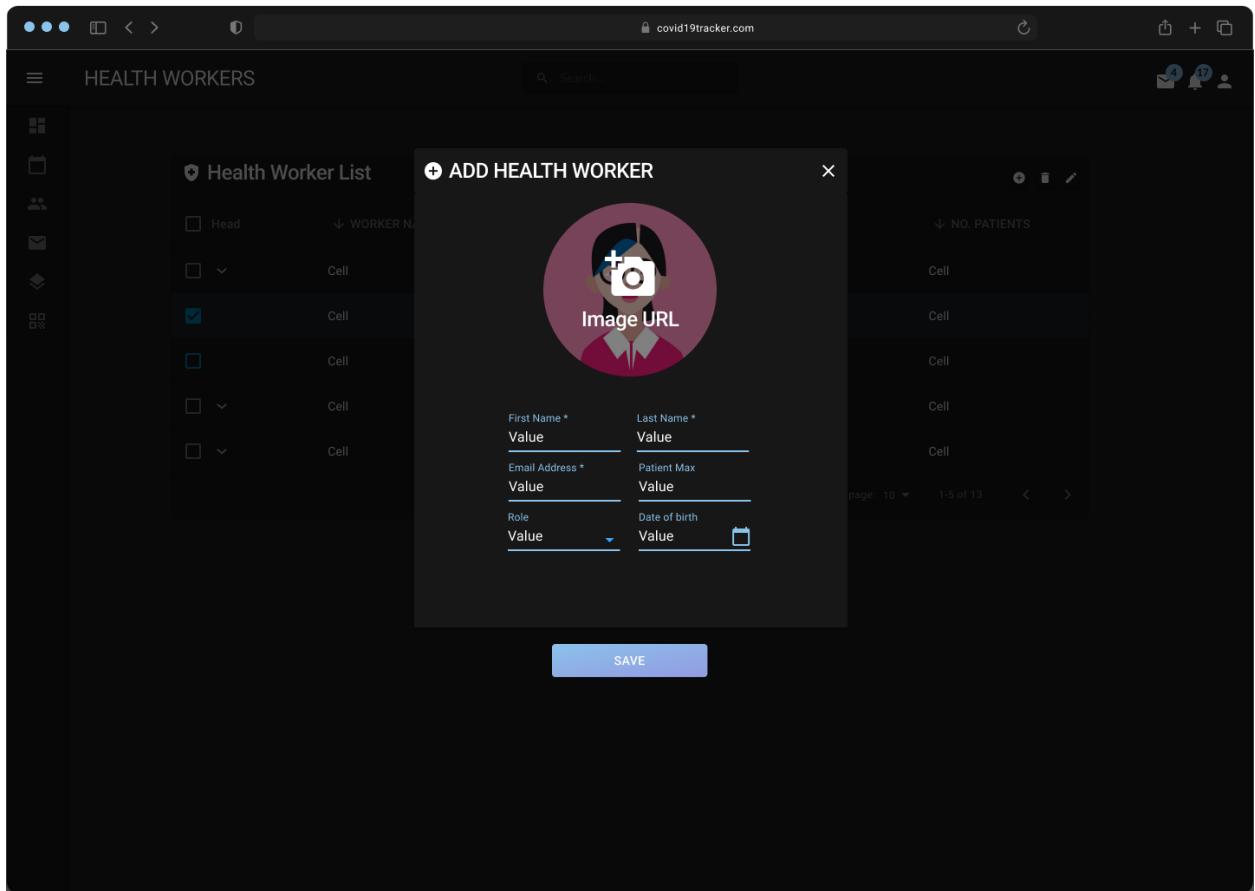


Figure 51: UI Admin Health Worker Add for User Story [Issue-19](#)

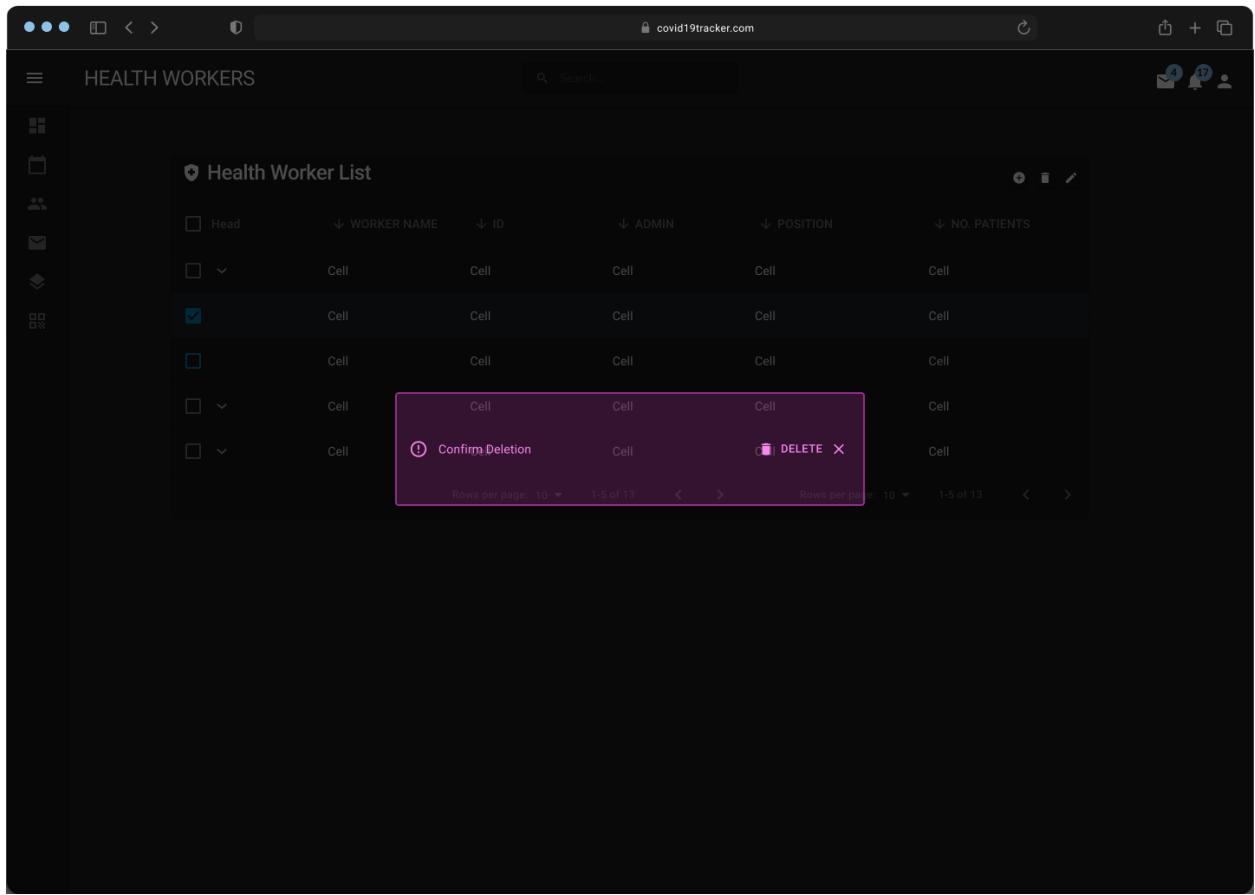


Figure 52: UI Admin Health Worker Delete for User Story [Issue-19](#)

The screenshot shows a web browser window for 'covid19tracker.com' with a dark-themed user interface. The main title is 'HEALTH WORKERS'. On the left, there's a sidebar with various icons. The main content area is titled 'Health Worker List' and contains a table with columns: Head, WORKER NAME, ID, ADMIN, POSITION, and NO. PATIENTS. The first row has a checked checkbox in the 'Head' column. A modal dialog box is overlaid on the table, indicating a 'Success' status with a checkmark icon and the message 'Fields saved'. The modal also includes a close button ('ACTION X') and pagination controls showing '1-5 of 13'.

Figure 53: UI Admin Health Worker Success for User Story [Issue-19](#)

The screenshot shows the 'PATIENT PAGE' section of the COVID-19 Tracker website. At the top, there's a navigation bar with icons for home, search, and notifications (17). Below it is a sidebar with icons for dashboard, calendar, users, messages, and settings.

The main area displays a patient profile for 'JANE DOE' (Age: 50, Birthday: 1 July 1957, Address: 101 Brooke, Montréal, L5L 9T9) with a pink circular placeholder image. To the right, a 'Flagged' status box indicates 'Unconfirmed' status (with a blue 'POSITIVE' button), a temperature of 39°C, and a weight of 154 lbs.

Below the profile, sections for 'ASSIGNED DOCTOR' (Dr. Julie Joy) and 'STATUS REVIEW' (Review Complete:) are shown. A 'SYMPTOM DETAILS' table lists symptoms like Fever, Cough, Runny Nose, Muscle Ache, Tiredness, Smell Loss, and Taste Loss, with the 'Cell' row for Fever checked.

The 'DIARY ENTRIES' section shows a table with columns for Date, Name, Phone Number, Email, and Address of Contact. The 'Cell' row for the first entry is checked. A 'Notify' button with an arrow icon is located next to the diary entries table.

At the bottom, pagination controls show 'Rows per page: 10' and '1-5 of 13'.

Figure 57: UI Patient Page for [Issue-4](#), [Issue-19](#)

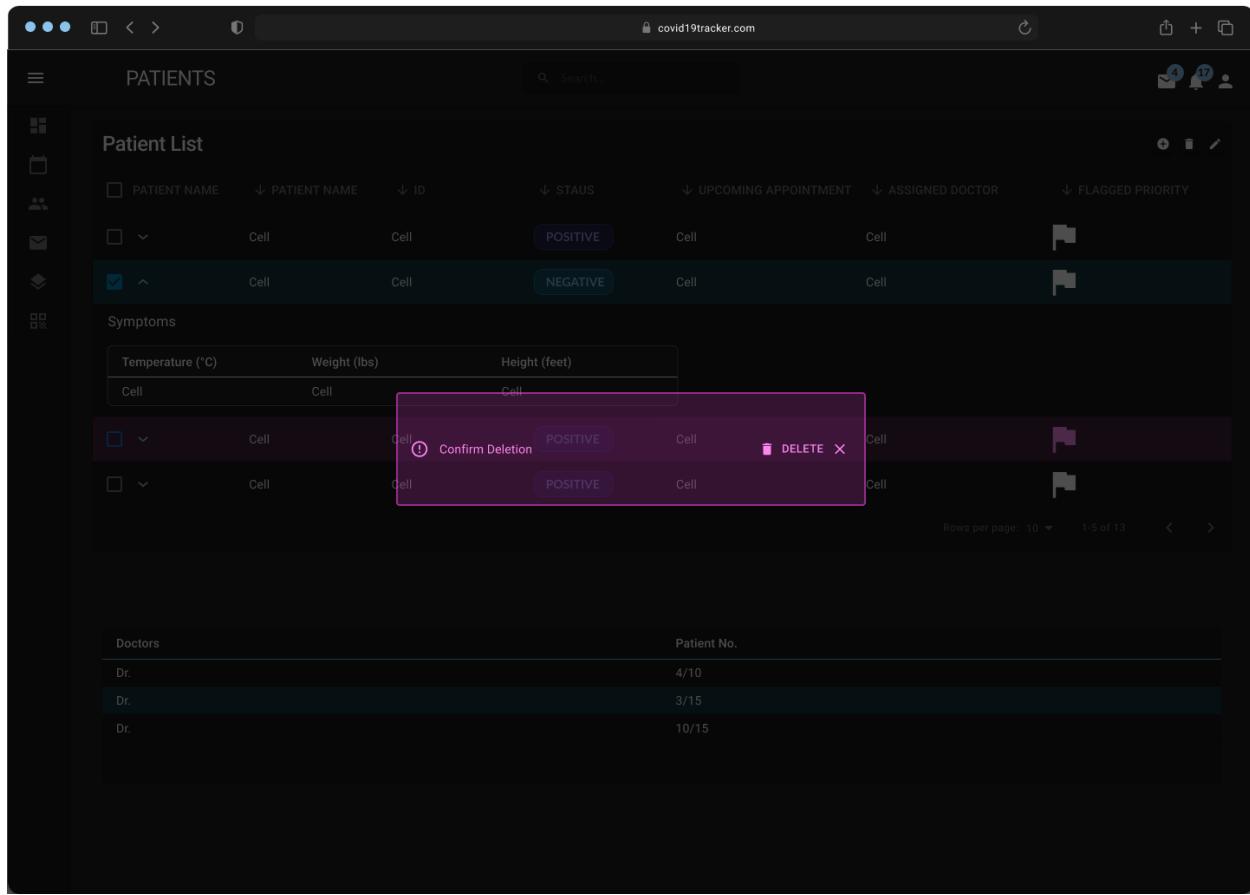


Figure 58: UI Patient Delete Modal

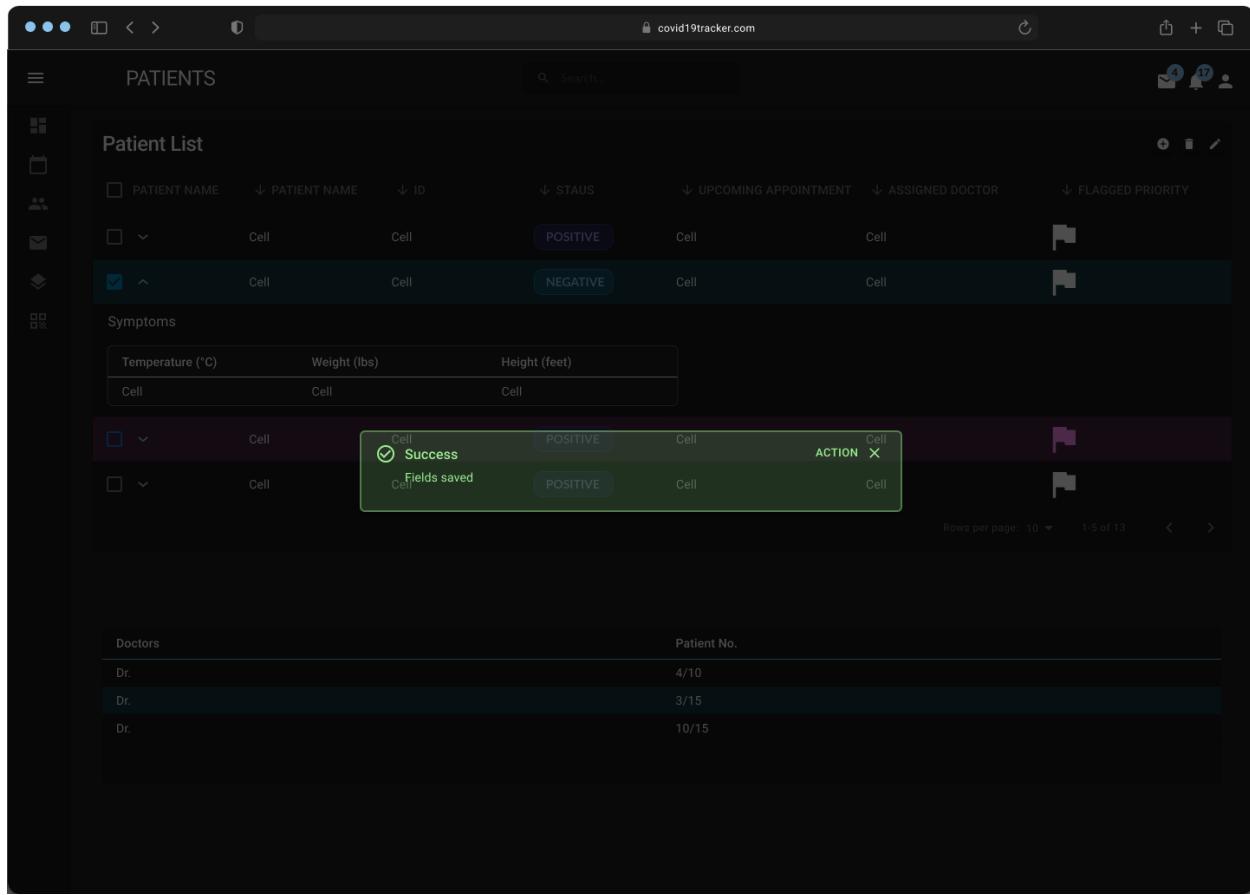


Figure 59: UI Patient Success Modal

7.3.3 Client Portal Sprint 3

The following figures are the UI mockups for the Client portal.

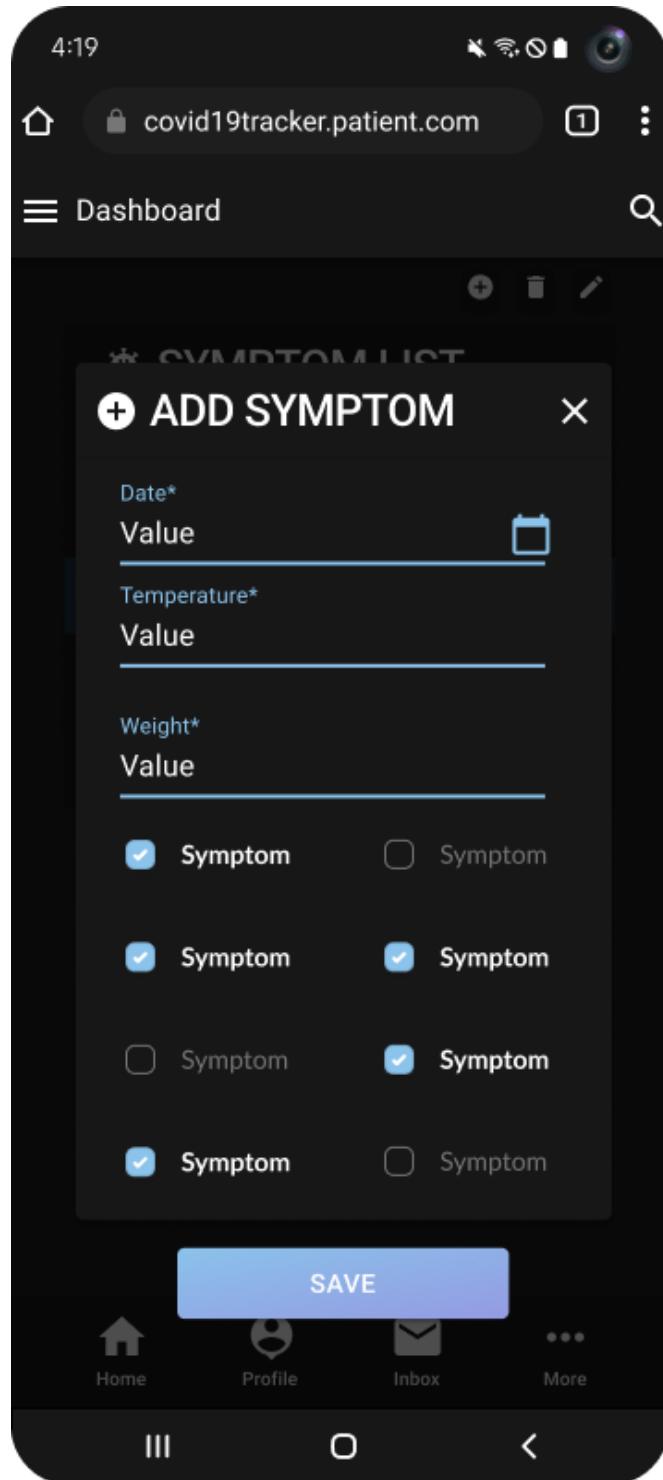


Figure 60: UI Symptoms Entry Add for User Story [Issue-41](#)

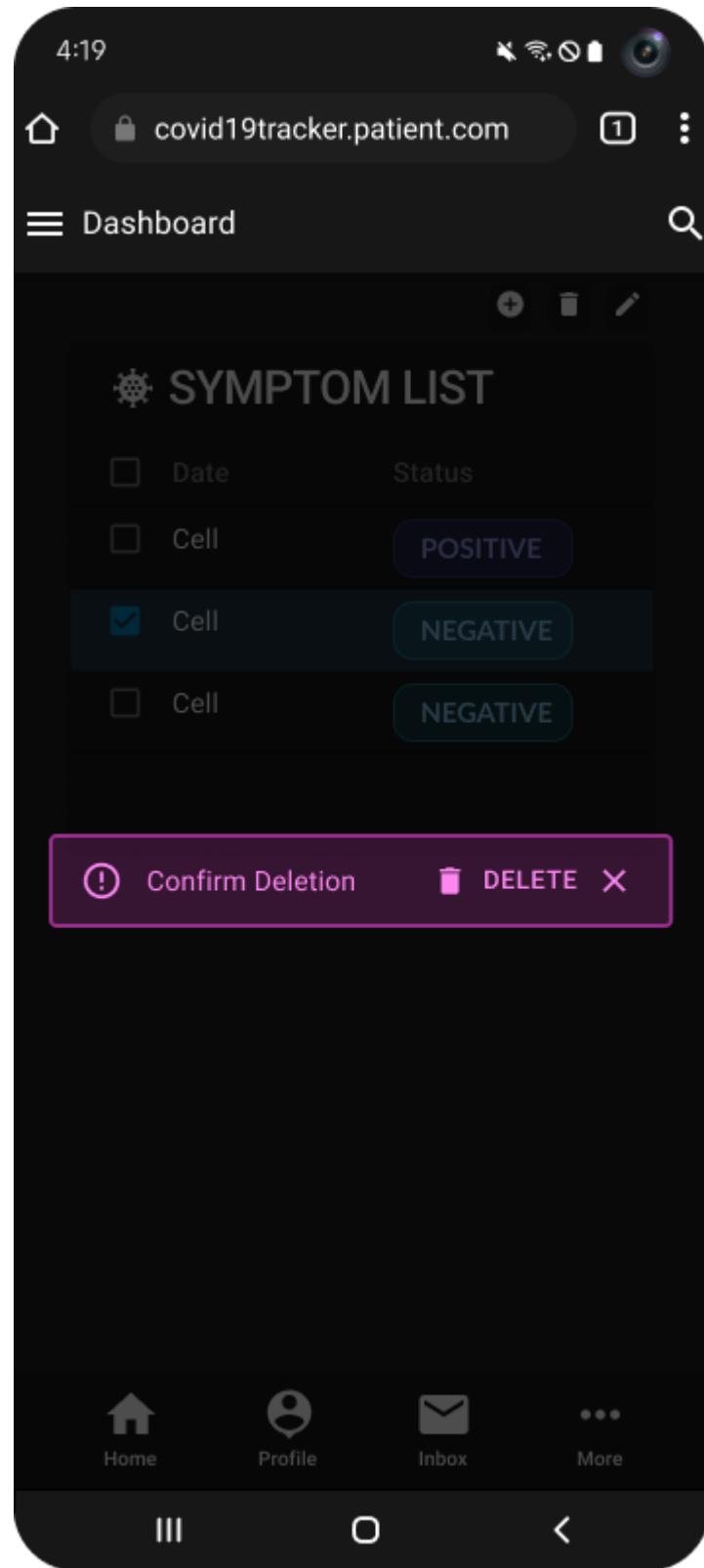


Figure 61: UI Symptoms Entry Delete for User Story [Issue-41](#)

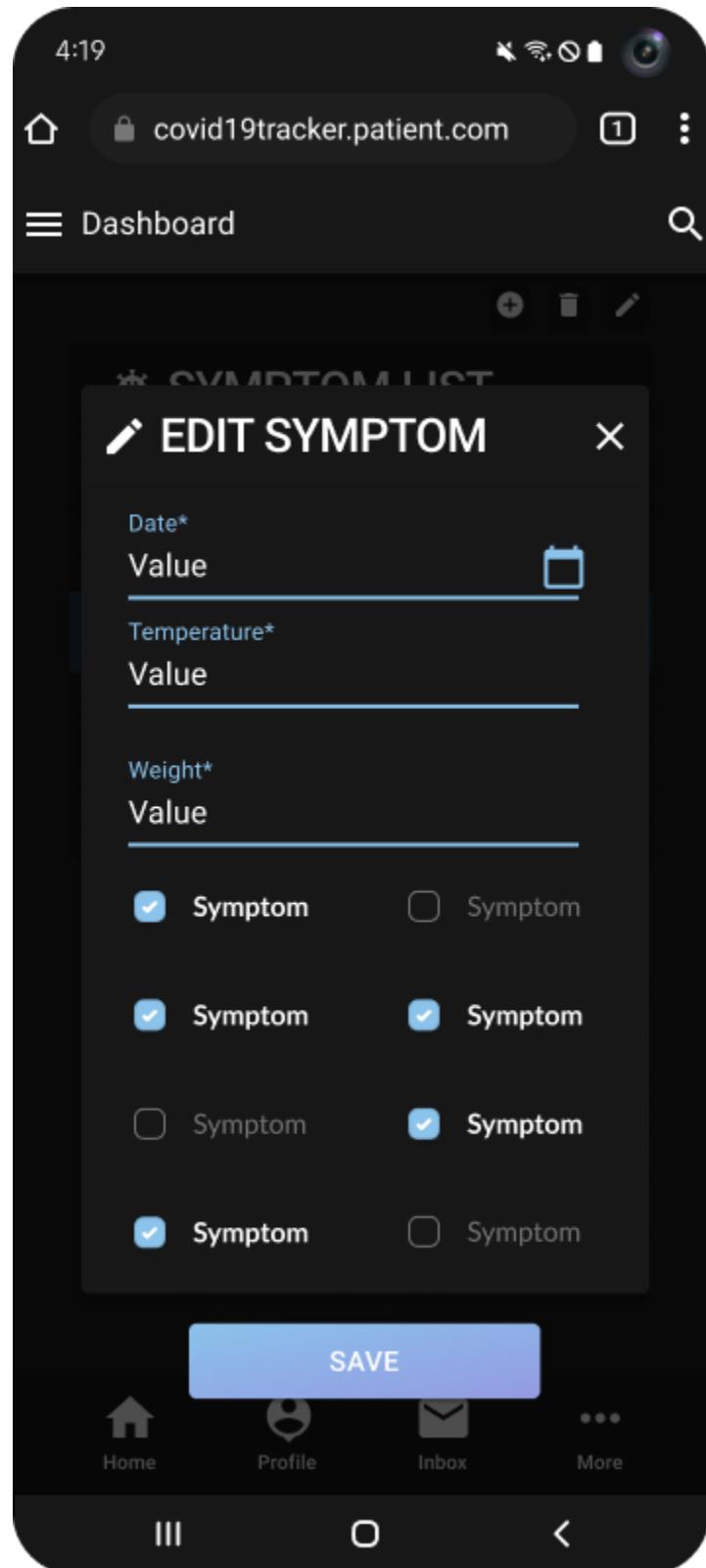


Figure 62: UI Symptoms Entry Edit for User Story [Issue-41](#)

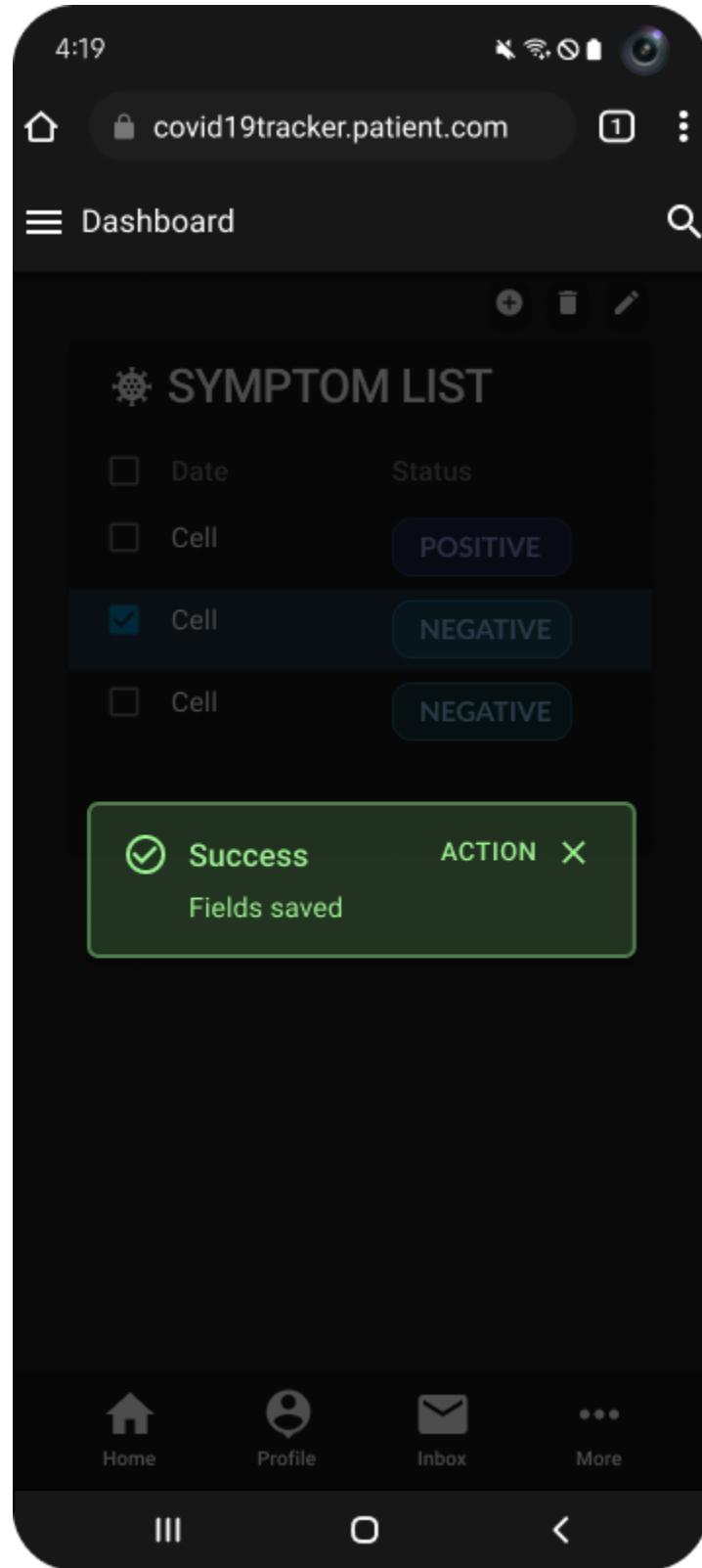


Figure 63: UI Symptoms Entry Success for User Story [Issue-41](#)

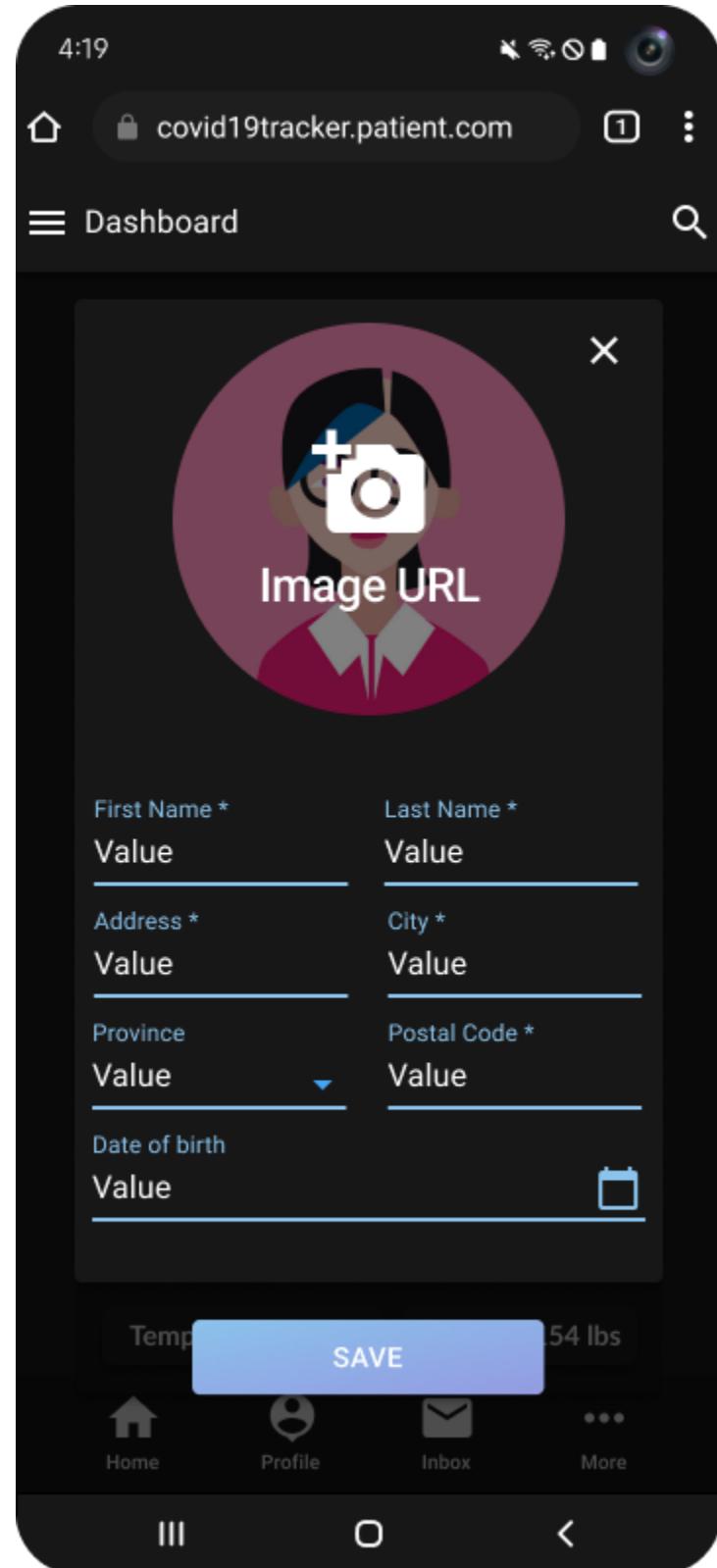


Figure 64: UI Client Patient Edit for User Story [Issue-37](#)

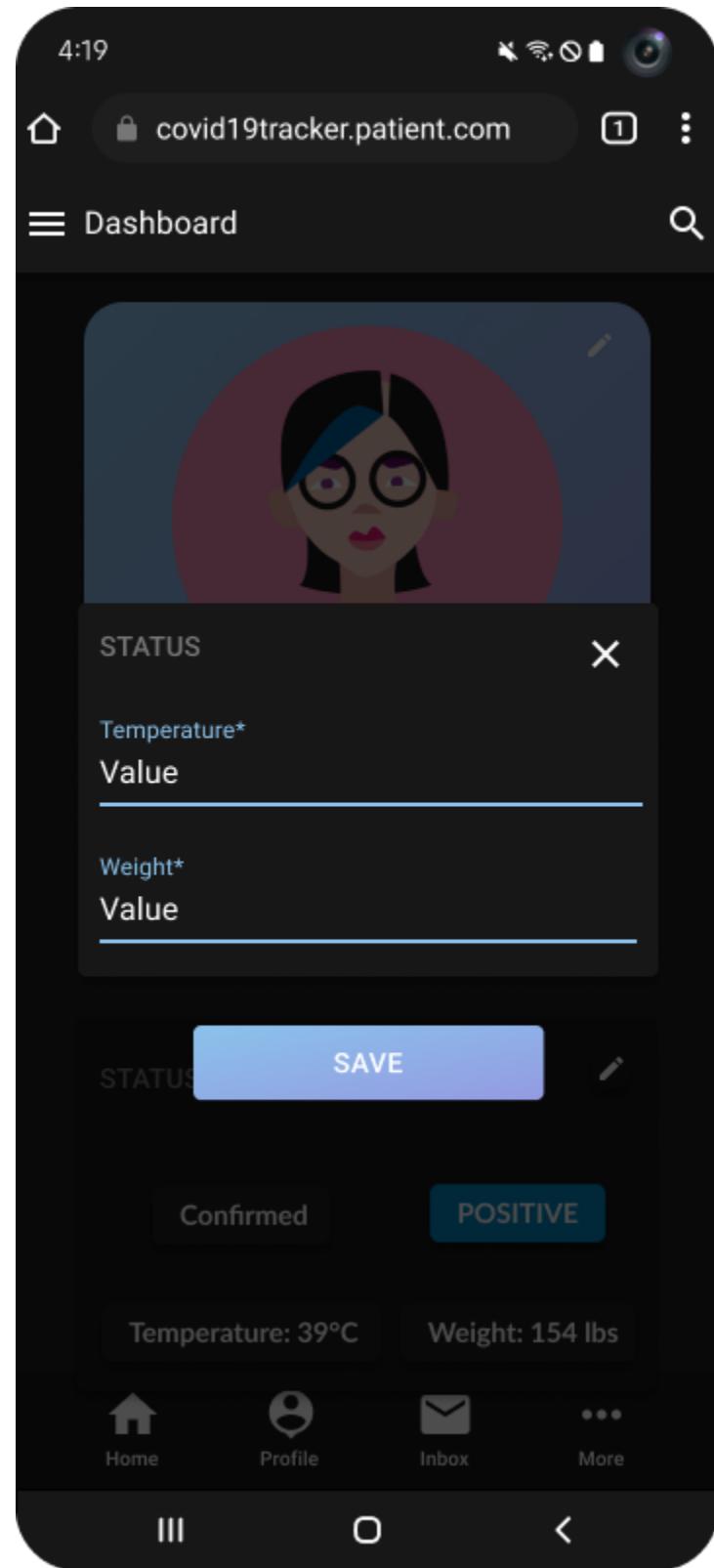


Figure 65: UI Client Patient Edit for User Story [Issue-37](#)

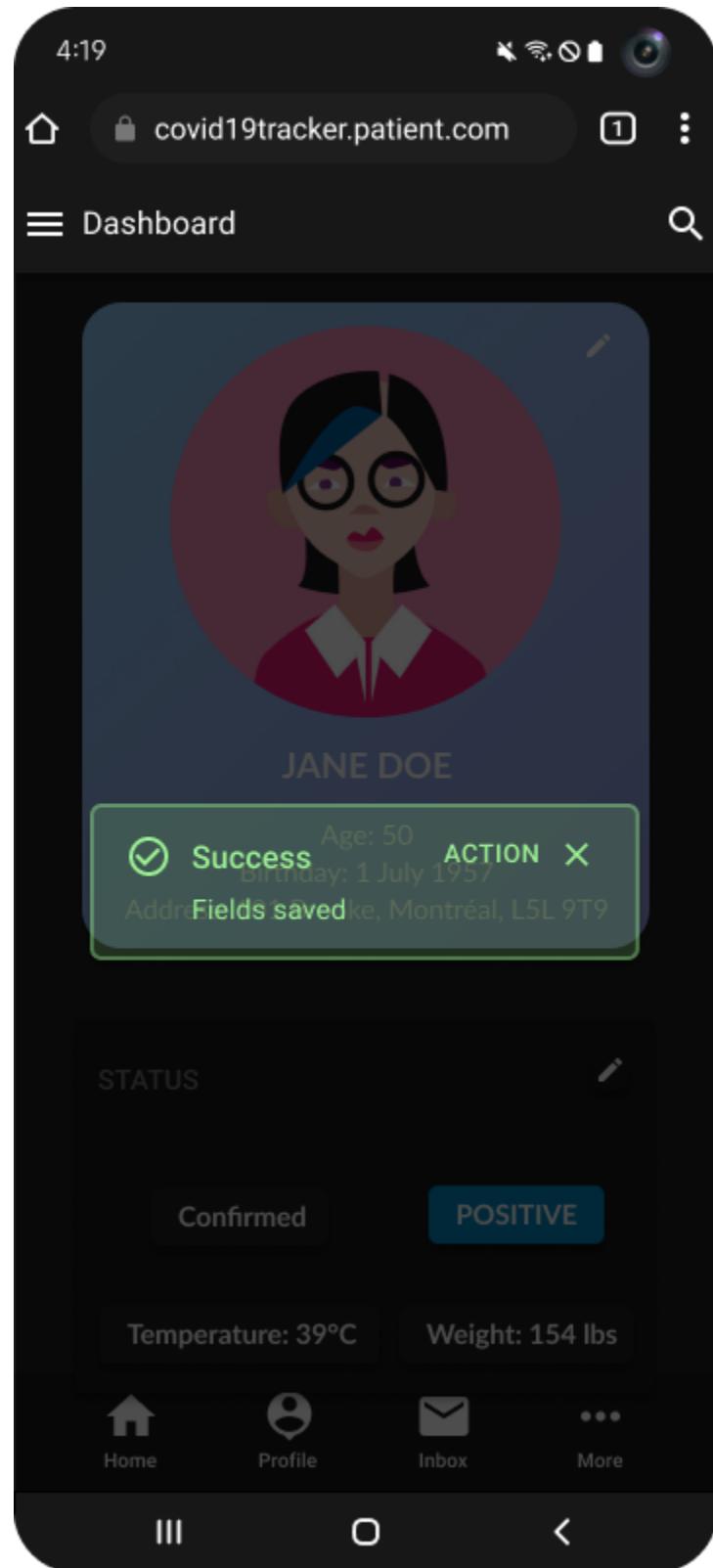


Figure 66: UI Client Patient Success for User Story [Issue-37](#)

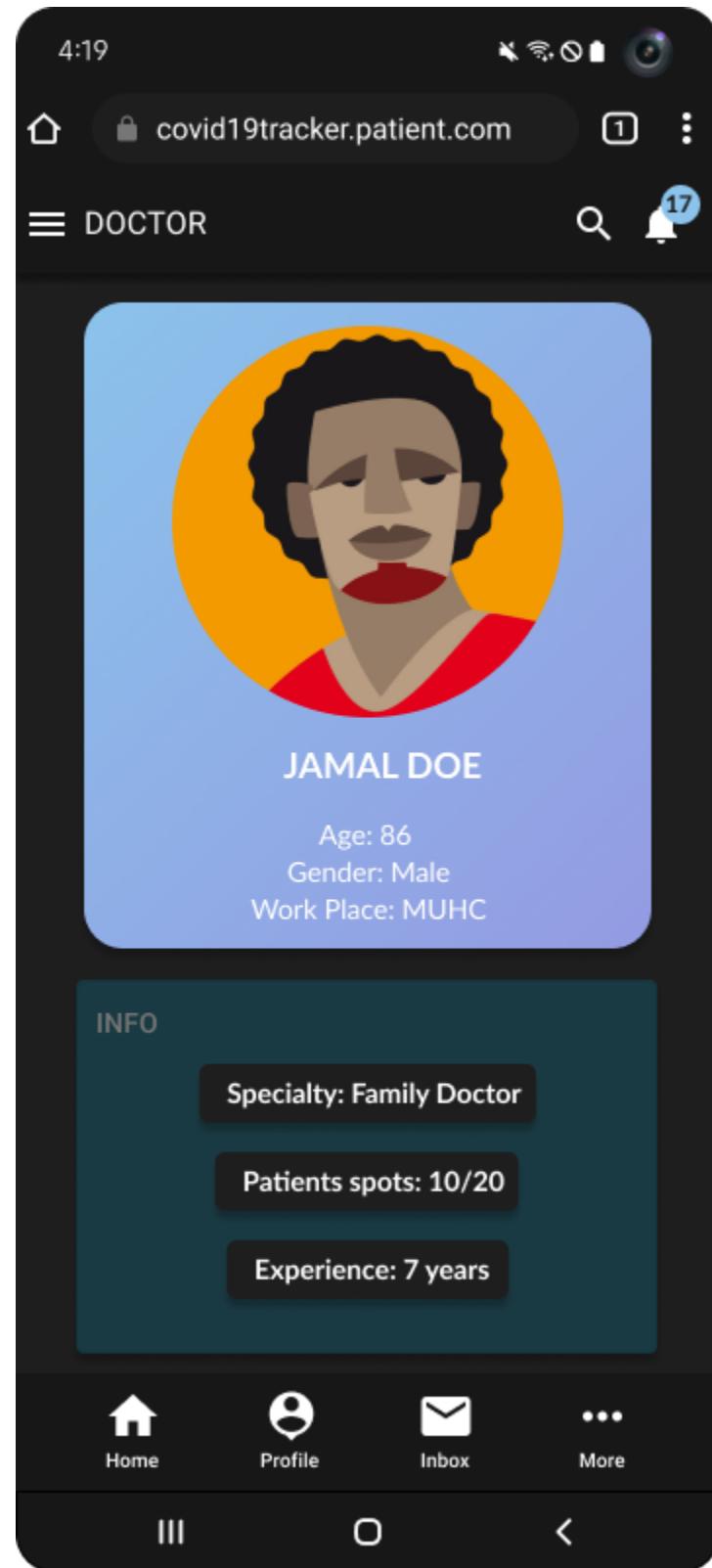


Figure 67: UI Doctor Info for User Story [Issue-39](#)

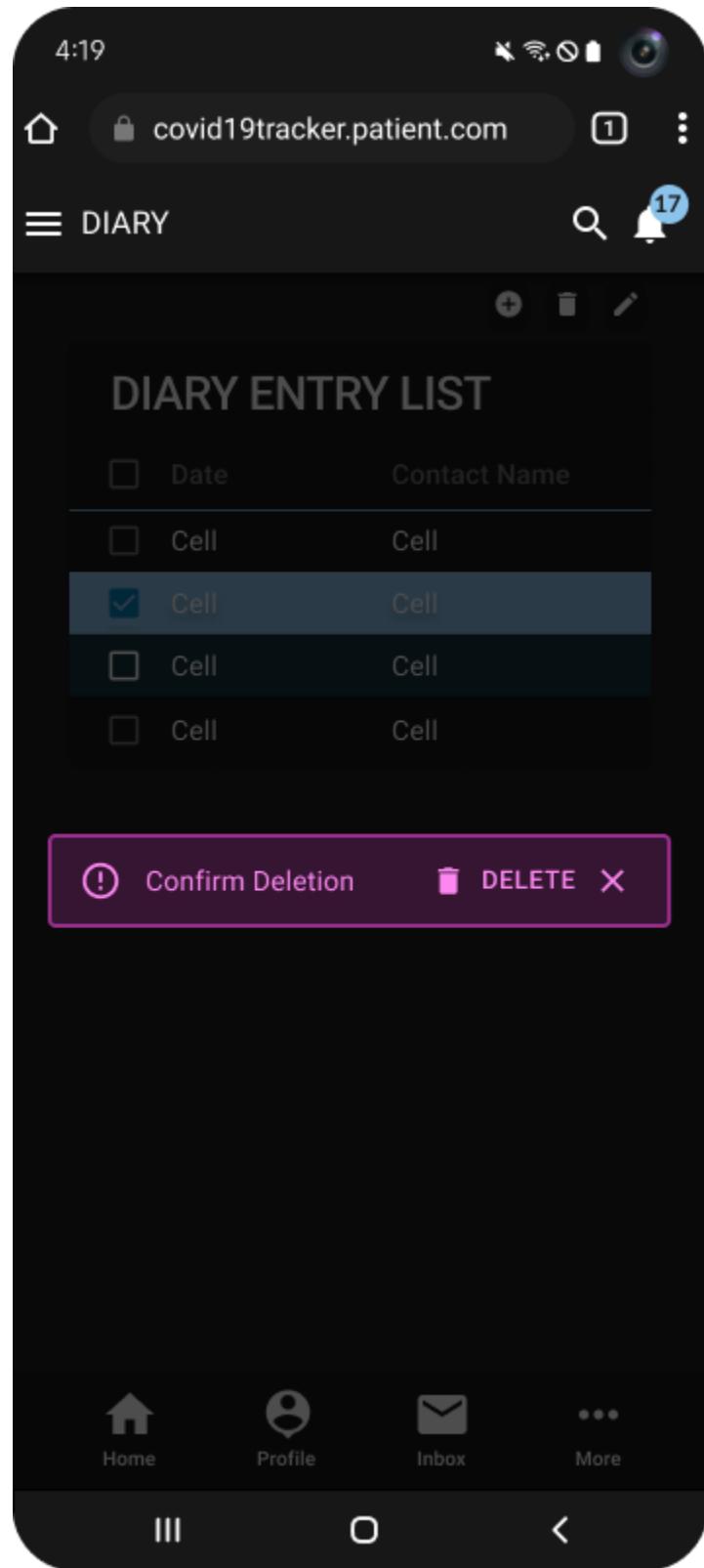


Figure 68: UI Diary delete

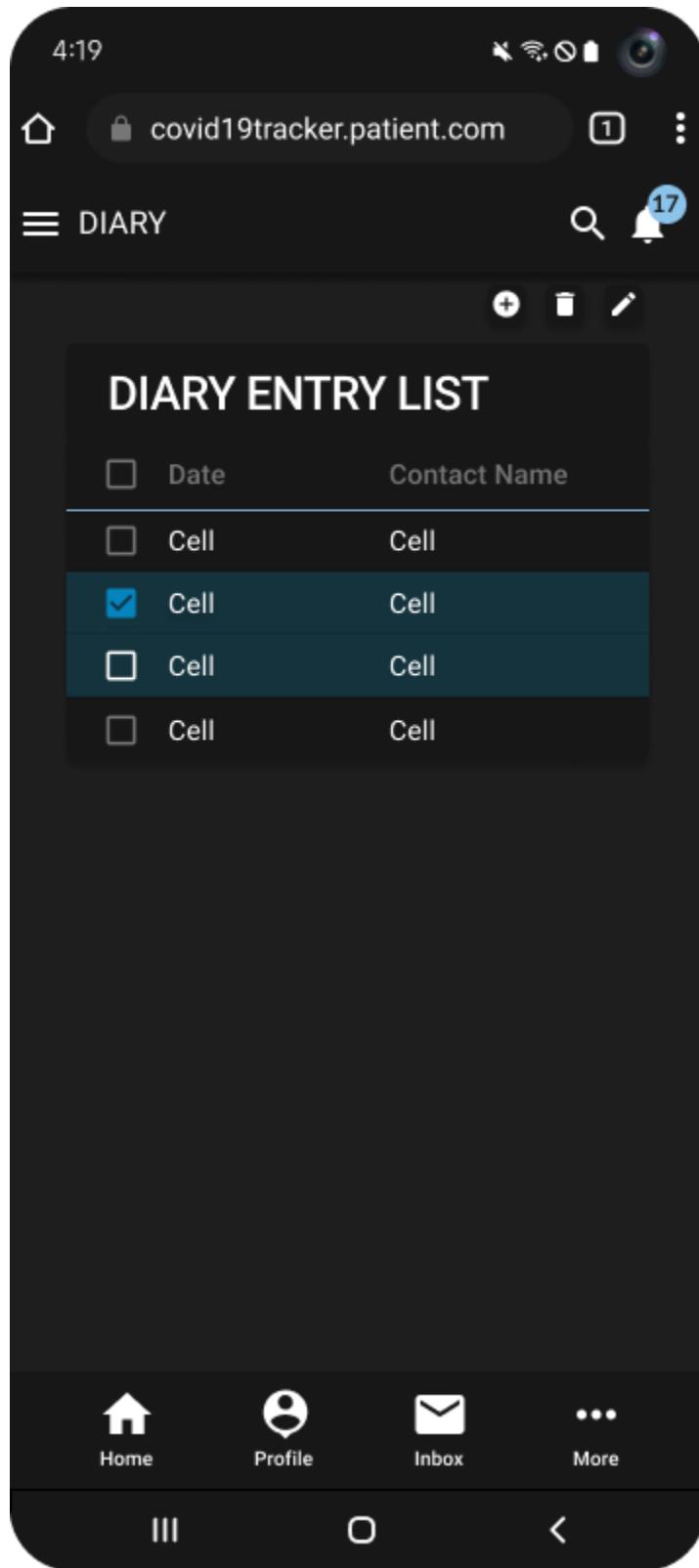


Figure 69: UI Diary list

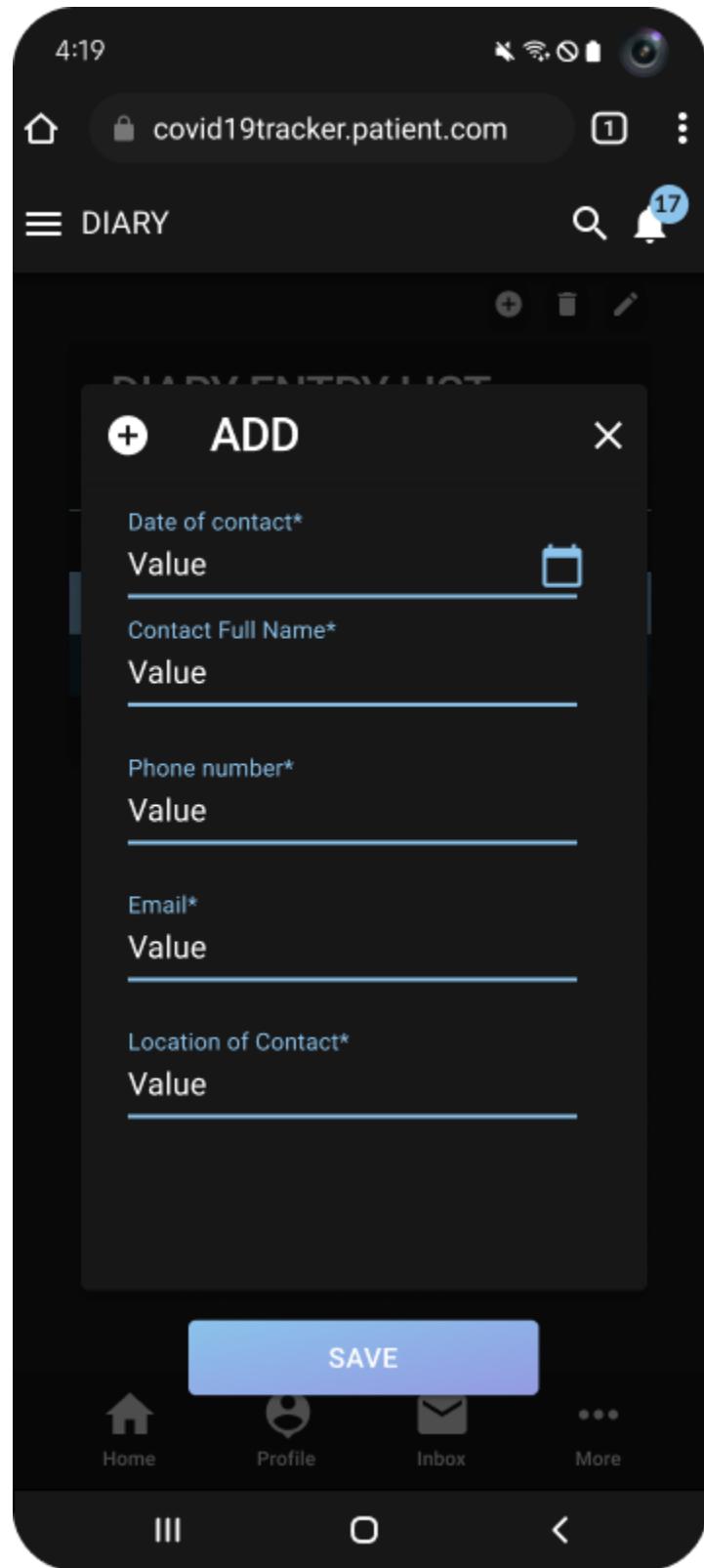


Figure 70: UI Diary Add

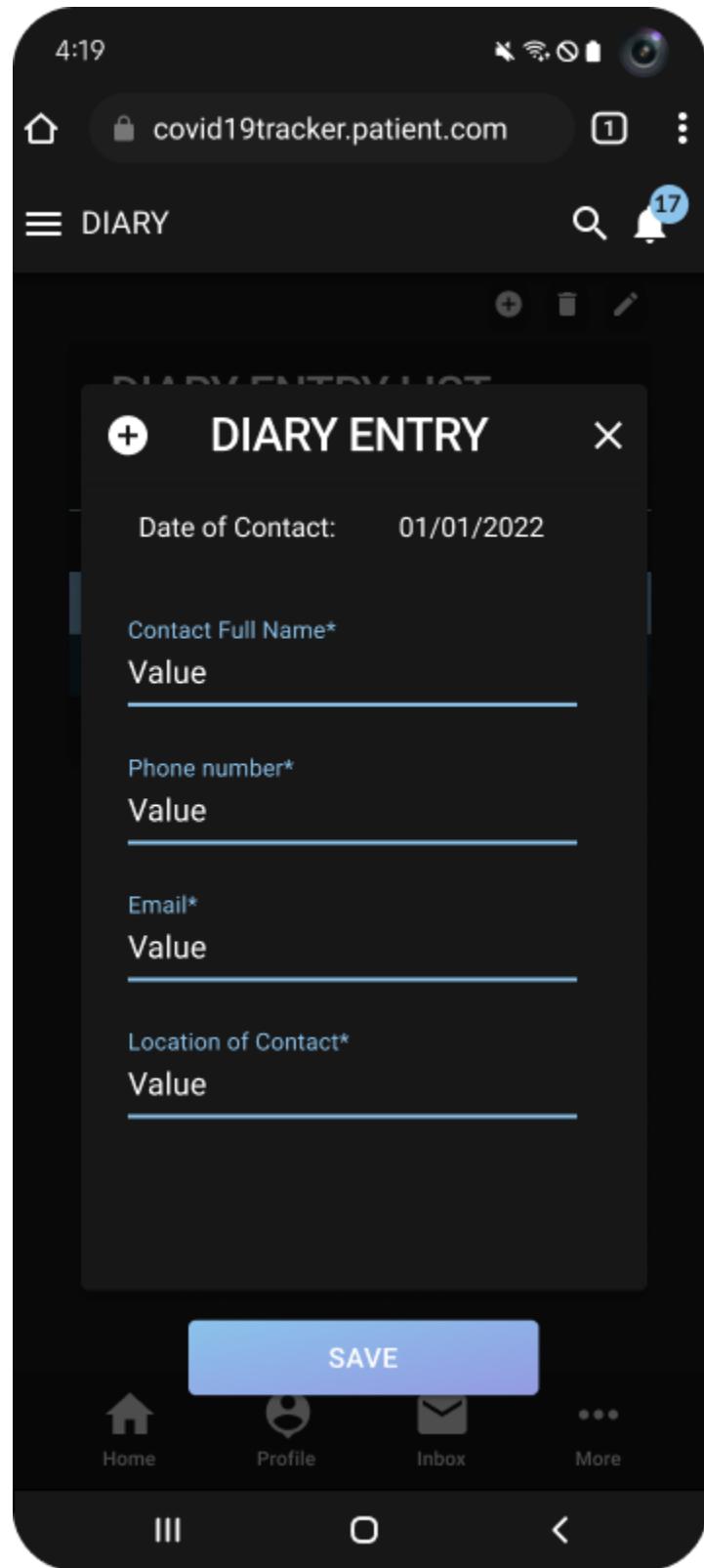


Figure 71: UI Diary Entry

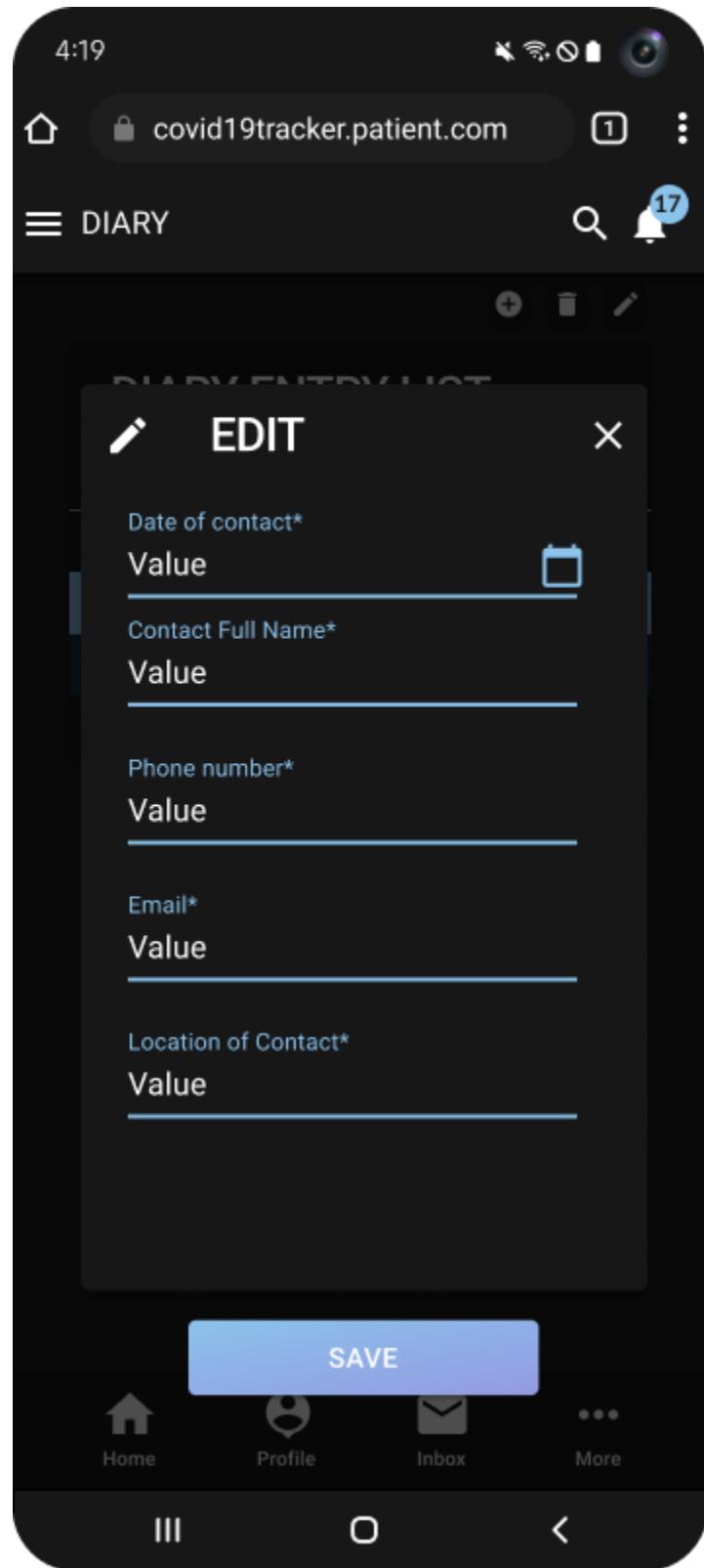


Figure 72: UI Diary Edit

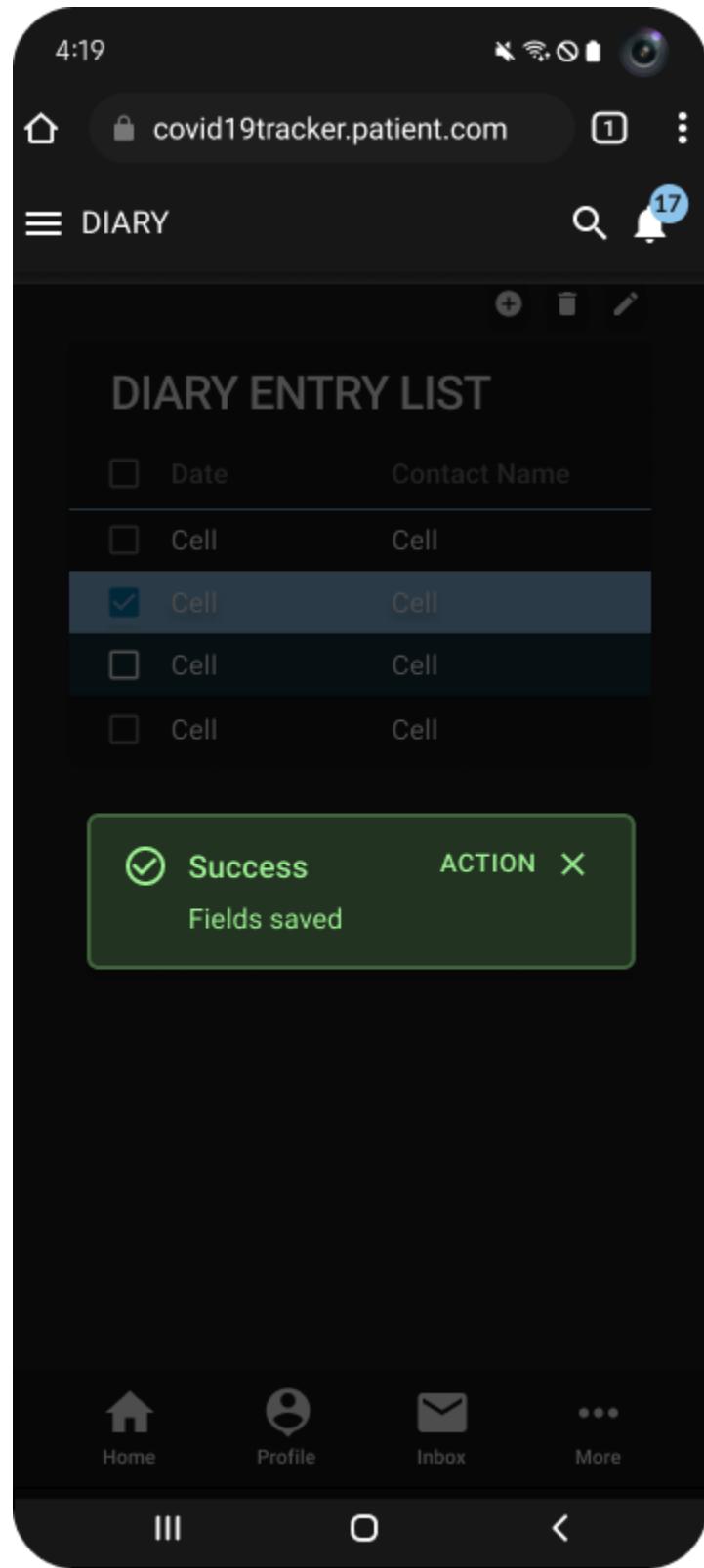


Figure 73: UI Diary Success

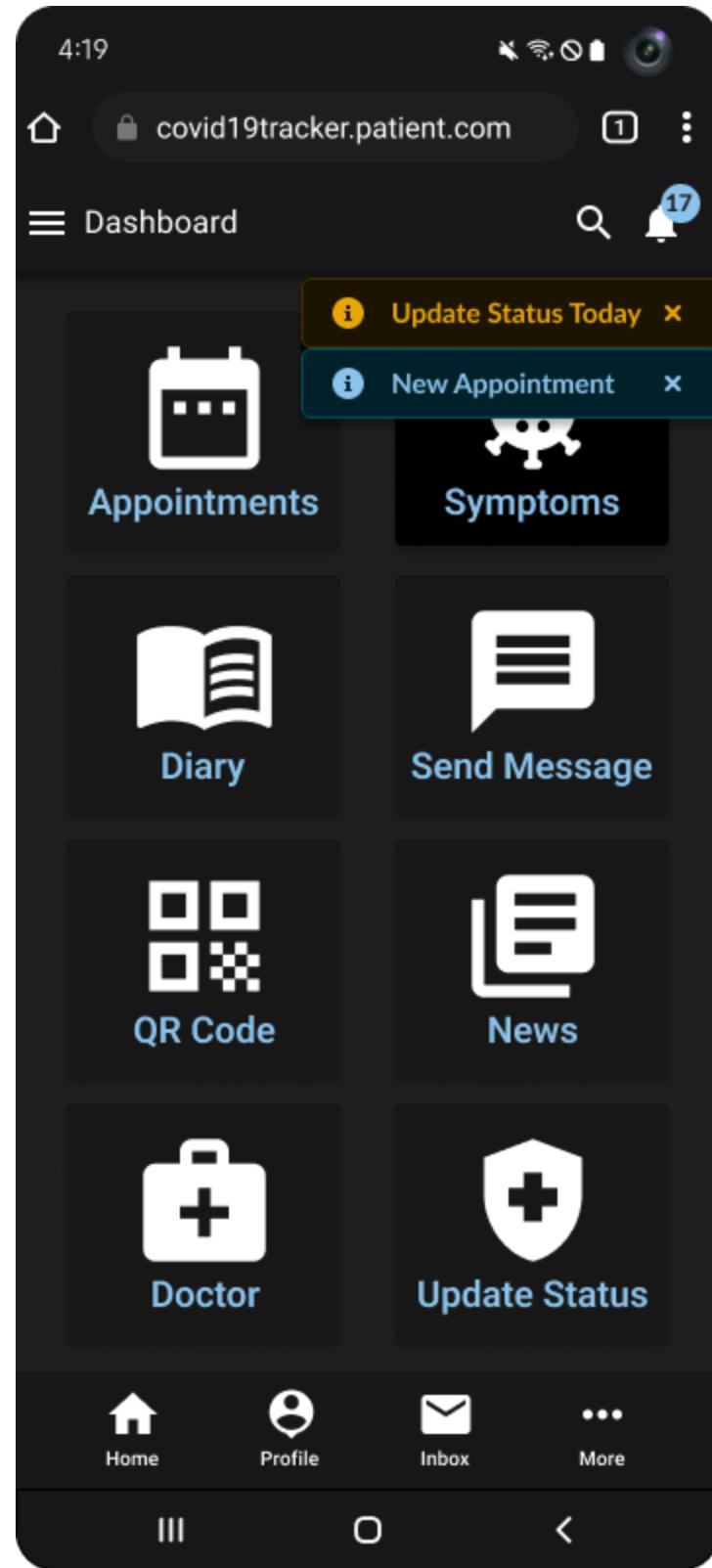


Figure 74: UI Notifications

7.4 Sprint 4 UI Mockups

Here are the UI mockups of user stories completed in Sprint 4.

7.4.1 Admin Portal Sprint 4

The following figures are the UI mockups for the Admin portal.

The screenshot shows the Admin Dashboard with the title "ADMIN DASHBOARD". Below it is the "Patient List" section. The header of the table includes columns: PATIENT NAME, ID, STATUS, UPCOMING APPOINTMENT, ASSIGNED DOCTOR, FLAGGED PRIORITY, and DISABLED. The first row has a checkbox, a dropdown, and three "Cell" entries. The "STATUS" column contains a red button labeled "POSITIVE". The second row has a checked checkbox, a dropdown, and three "Cell" entries. The "STATUS" column contains a green button labeled "NEGATIVE". The third row has an unchecked checkbox, a dropdown, and three "Cell" entries. The "STATUS" column contains a green button labeled "UNCONFIRMED". Below the table, there is a "Doctors" section with a table:

Doctors	Patient No.
Dr.	4/10
Dr.	3/15
Dr.	10/15

At the bottom right of the dashboard, there are buttons for "Rows per page: 10", "1-5 of 13", and navigation arrows.

Figure 75: UI Patient List with disabled patient column Issue #45

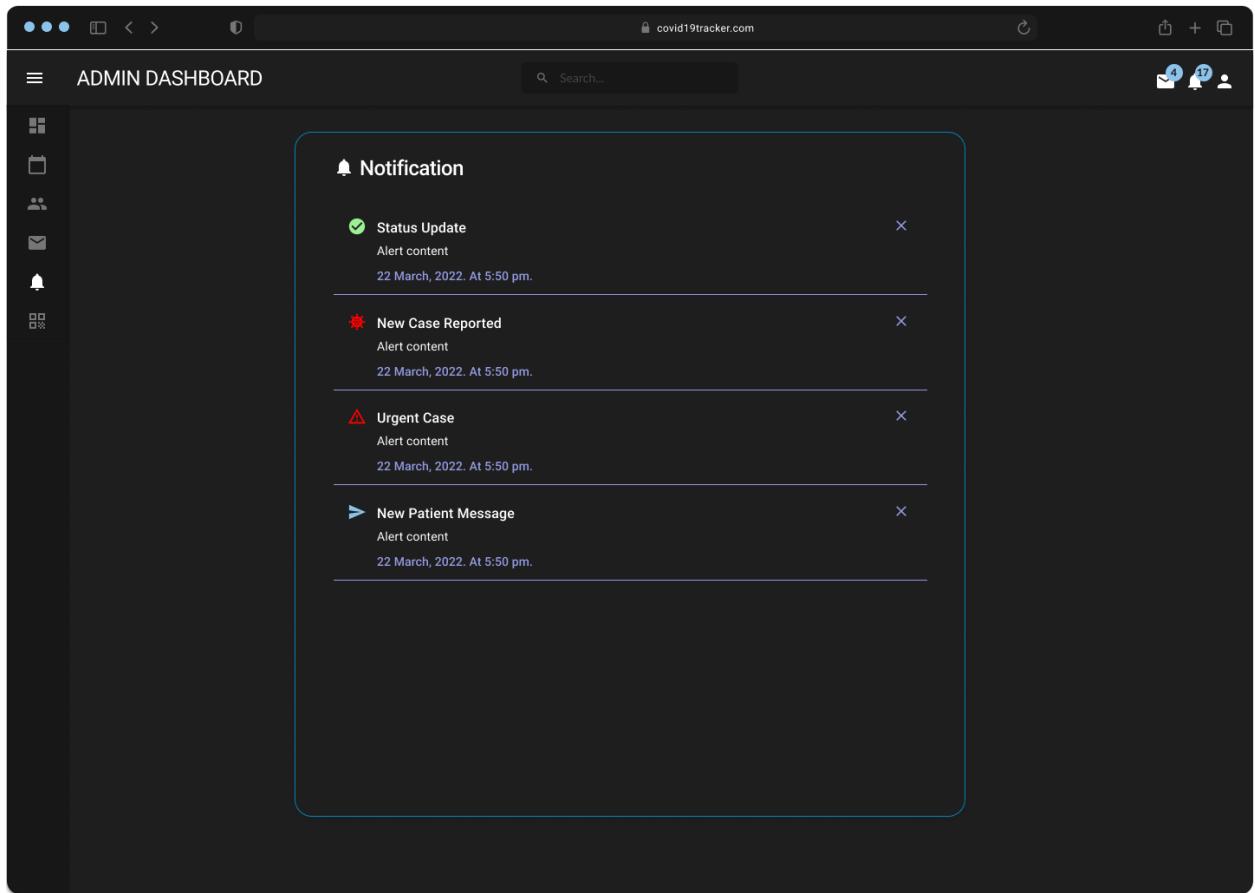


Figure 76: UI Notifications Issue #2 #10 #13

7.4.2 Client Portal Sprint 4

The following figures are the UI mockups for the Client portal.

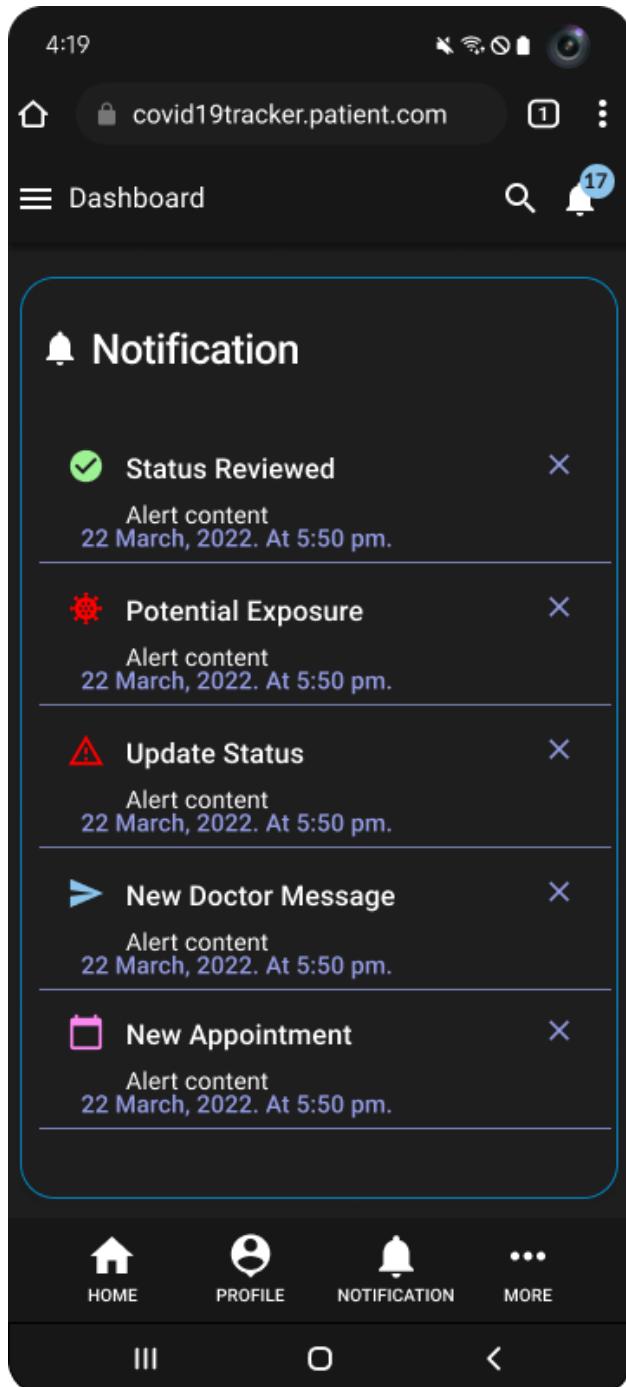


Figure 76: UI Notifications Issue #4 #12 #42

8.0 TESTING PLAN AND REPORT

In this section, we detail the unit testing, the code coverage, acceptance testing and system tests.

8.1 Unit Testing

Unit testing is the most fundamental type of test that guarantees a unit of code function properly on its own. A unit test is the smallest testable part of an application. The main purpose is to test each function or component as soon as they are constructed. A unit test usually has one or more inputs and produces a single output.

Jest's main focus is on the simplicity of the provided functions and good support for large web applications. It works with web applications using Babel, TypeScript, Node.js, React, Angular, Vue.js, Svelte, and more. We chose Jest since it is an already established framework in the JavaScript ecosystem. In addition, their documentation is very clear and concise. Thus, the learning curve for developers would be less compared to other unit testing frameworks.

Some of the obvious benefits of using Jest for unit testing are:

- Improving the overall quality of the codebase.
- Ensure code reusability and reliability.
- Help seamless integration with other modules/components.

8.2 Code Coverage

Code coverage tools may not guarantee to find all defects hidden in the current codebase but it allows developers to see how complete the test cases cover the code base. Thus, having code coverage results will help gaining confidence in developers. Code coverage tool that comes with Jest can be enabled by simply adding the “`--coverage`” flag in the test run command. No other additional setup required, Jest will automatically run through the project to collect all the written test files.

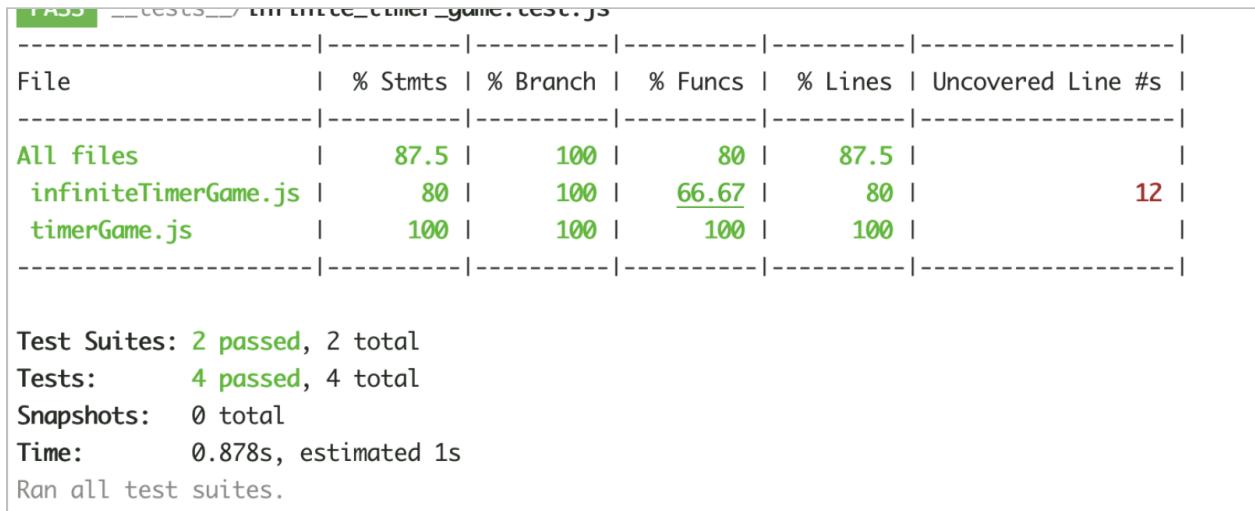


Figure 48: Sample code coverage screenshot using Jest

8.3 Acceptance Testing

An acceptance test is a formal description of certain behaviors of a system when some conditions have been met. An acceptance test is usually expressed as a scenario statement or a group of statements. Like unit tests, acceptance tests also have a binary result (either passed or failed). A passed acceptance test is the base indicator of a correctly implemented function/feature that met the requirements.

In our project management backlog, every user story has its Acceptance Test written under the description section so that we can refer to it at the end of the Sprint to ensure that the feature has met the requirements.

AT-1	Monitor status & symptoms of confirmed and unconfirmed patients
Acceptance Criteria	<p>Given: The user is logged in as an Admin (doctor or health official).</p> <p>When: The user is on the Profile of the Patient page.</p> <p>Then: The user can select the patient's status from the dropdown.</p>
Result	PASS

Comments	NONE
-----------------	------

AT-2	Display / show that a patients' status updates have been reviewed
Acceptance Criteria	Given: The user is logged in as a doctor and is on the patient page. When: The user updates and reviews a patient profile. Then: The application shows profiles that have not been reviewed yet.
Result	PASS
Comments	NONE

AT-3	Allow Patient to edit profile page
Acceptance Criteria	Given: Patient is logged in successfully. When: Patient clicks on profile page. Then: All the basic fields should be editable & updatable.
Result	PASS
Comments	NONE

AT-4	Profile for patient
-------------	----------------------------

Acceptance Criteria	Given: The user is logged in as a patient on the client app. When: The user clicks on the profile button on the bottom navbar. Then: The user is redirected to their profile displaying a list of given details about the patient.
Result	PASS
Comments	NONE

8.4 System Tests

System testing (or end-to-end testing) is a common methodology used in the software development process to test if an application works well under product-like circumstances and with data that replicates live settings. The goal is to create a realistic experience that follows the steps of a real user scenario. To fully validate the system, testing should not only be performed on the system under test, but also on any other subsystems that are related.

We chose Cypress because it is one of the most famous frameworks for system (end-to-end) testing in the JavaScript ecosystem. Compared to other similar frameworks such as Selenium or Splinter, Cypress takes much less time to learn and much less code to write in order to achieve the same results. In addition, Cypress also comes with extra functionalities to help developers observe the testing steps visually which is very convenient to pinpoint exactly where the test failed (if it happened).

Some of the "best-selling" features that are packed with Cypress:

- Time travel: Cypress can take snapshots as it runs the test suite.
- Debugging: Readable and traceable errors.
- Automatic waiting: Waits for commands and assertions before moving on.
- Spies, stubs, and clocks: Verify and control the behavior of functions, server responses, or timers.
- Network Traffic Control: Control, stub, and test edge cases without involving the server.

- Screenshots and videos: View screenshots were taken automatically on failure.
- Cross-browser Testing: Run tests within Firefox or Chrome browsers locally

8.5 Testing Procedure

Using automated workflow is the best way to save the team's time on testing. Writing tests is often time-consuming, running them on the local machine every time a new function is added (repeatedly) can be tedious. This is the reason why we planned to continuously run tests and deploy code on the remote server. This way our team can save some more time focusing on development.

Travis CI already comes with every Github project, it takes a minimal setup time in exchange for a full pack of benefits:

- Quick setup.
- Live build views.
- Pull request support.
- Pre-installed database services.
- Auto deployments on passing builds.
- Clean VMs for every build.
- Mac, Linux, and iOS support.

8.6 Test Results

This section contains the information about our actual test results (Unit test, Integration test and System test) from both Admin app and Client app. The details for each are presented below.

8.6.1 Unit & integration test for Admin app

Test file name	Description	Result
Dashboard.test.js	Test if the patient list and some card components are rendered correctly on Dashboard.	PASSED
PatientProfile.test.js	Check if some basic patient info (such as age, status), the assigned doctor, and the symptom details show up in the patient profile.	PASSED
Dropdown.test.js	Check if certain select options are visible inside this component.	PASSED
Calendar.test.js	Check if the month-view button, week-view button, and day-view button are contained in the rendered calendar element.	PASSED
EventTest.test.js	Integrate the EventDetails component with the MemoryRouter component.	PASSED
DoctorList.test.js	Verify whether the DoctorList component is being attached to the document (the DOM tree).	PASSED
UpcomingEvents.test.js	Test if certain classes and text contents can be found in the rendered component. Integrate the UpcomingEvents component with the MemoryRouter component.	PASSED
NewsTest.test.js	Integrate the NewList component with the MemoryRouter component.	PASSED
CovidButton.test.js	Check if the COVID19Button component is rendered to the DOM tree.	PASSED
Notifications.test.js	Check if the Notifications component has an expected text content.	PASSED
SmallStatBox.test.js	Test if one of the small statistic boxes contains a certain class.	PASSED
DoughnutChart.test.js	Test if the DoughnutChart component is rendered out correctly.	PASSED
LineChart.test.js	Test if the LineChart component is rendered out correctly.	PASSED
PatientList.test.js	Test if the PatientList component is rendered out correctly.	PASSED

SignIn.test.js	Test if the SignIn component is rendered out correctly.	PASSED
SignUp.test.js	Test if the SignUp component is rendered out correctly.	PASSED

```

PASS  src/screens/Dashboard/Dashboard.test.js
PASS  src/components/PatientProfile/PatientProfile.test.js
PASS  src/components/DropdownConfirmation/Dropdown.test.js
PASS  src/components/Calendar/Calendar.test.js
PASS  src/components/Event/EventTest.test.js
PASS  src/components/DoctorList/DoctorList.test.js
PASS  src/components/UpcomingEvents/UpcomingEvents.test.js
PASS  src/components/News/NewsTest.test.js
PASS  src/components/COVID-19Button/CovidButton.test.js
PASS  src/components/Notifications/Notifications.test.js
PASS  src/components/SmallStatBox/SmallStatBox.test.js
PASS  src/components/Charts/DoughnutChart.test.js
PASS  src/components/Charts/LineChart.test.js
PASS  src/components/PatientList/PatientList.test.js
PASS  src/components/SignIn/SignIn.test.js
PASS  src/components/SignUp/SignUp.test.js

Test Suites: 16 passed, 16 total
Tests: 21 passed, 21 total
Snapshots: 0 total
Time: 4.565 s

```

Figure 75: Screenshot from actual test results for Admin app

8.6.2 Unit & integration test for Client app

Test file name	Description	Result
DoctorInfo.test.js	Test if the DoctorInfo component is present in the document, has the text content “Jamal Doe”, and has class “doctorInfo-card__profileName”	PASSED
Dashboard.test.js	Check if the ‘Appointment’ is visible through the Dashboard.	PASSED
NavBar.test.js	Check if the NavBar component integrates well with the MemoryRouter component. Also verify that the client app title is visible.	PASSED
SymptomsTable.test.js	Check if the SymptomsTable is rendered correctly	PASSED

	to the screen.	
BottomNav.test.js	Test if the bottom navigation bar contains the 'bottomNav-iconTitle' class and contains the 'Profile' text content.	PASSED
Signin.test.js	Check if the Signin page is rendered correctly to the screen.	PASSED
ClientProfile.test.js	Check if the avatar component is visible and contains the 'profile-name' class.	PASSED
SignUp.test.js	Test if the Signin page is rendered correctly to the screen.	PASSED
DiaryTable.test.js	Test if DiaryTable component is rendered correctly	PASSED
Chat.test.js	Test if Chat component is rendered correctly	PASSED

```

PASS src/components/DoctorInfo/DoctorInfo.test.js
PASS src/components/BottomNav/BottomNav.test.js
PASS src/components/Chat/Chat.test.js
PASS src/components/ClientProfile/ClientProfile.test.js
PASS src/components/Dashboard/Dashboard.test.js
PASS src/components/Diary/DiaryTable.test.js
PASS src/components/NavBar/NavBar.test.js
PASS src/components/DoctorInfo/DoctorInfo.test.js
PASS src/components/Diary/DiaryTable.test.js
PASS src/components/BottomNav/BottomNav.test.js
PASS src/components/Chat/Chat.test.js
PASS src/components/ClientProfile/ClientProfile.test.js
PASS src/components/NavBar/NavBar.test.js
PASS src/components/SignUp/SignUp.test.js
PASS src/components/Dashboard/Dashboard.test.js
PASS src/components/SignIn/Signin.test.js
PASS src/components/SymptomsTable/SymptomsTable.test.js

Test Suites: 10 passed, 10 total
Tests:       10 passed, 10 total

```

Figure 76: Screenshot from actual test results for Client app

8.6.3 System test for Admin app (Cypress)

Test file name	Description	Result
dashboard.spec.js	Test if user could visit the Dashboard page and click on the 'Appointment for vaccination' link	PASSED
loginTest.spec.js	Mimics the user login flow (sign-in and click on different pages) with different sign-in credentials.	PASSED

8.6.4 System test for Client app (Cypress)

Test file name	Description	Result
dashboard.spec.js	Test if user could visit the Dashboard page	PASSED

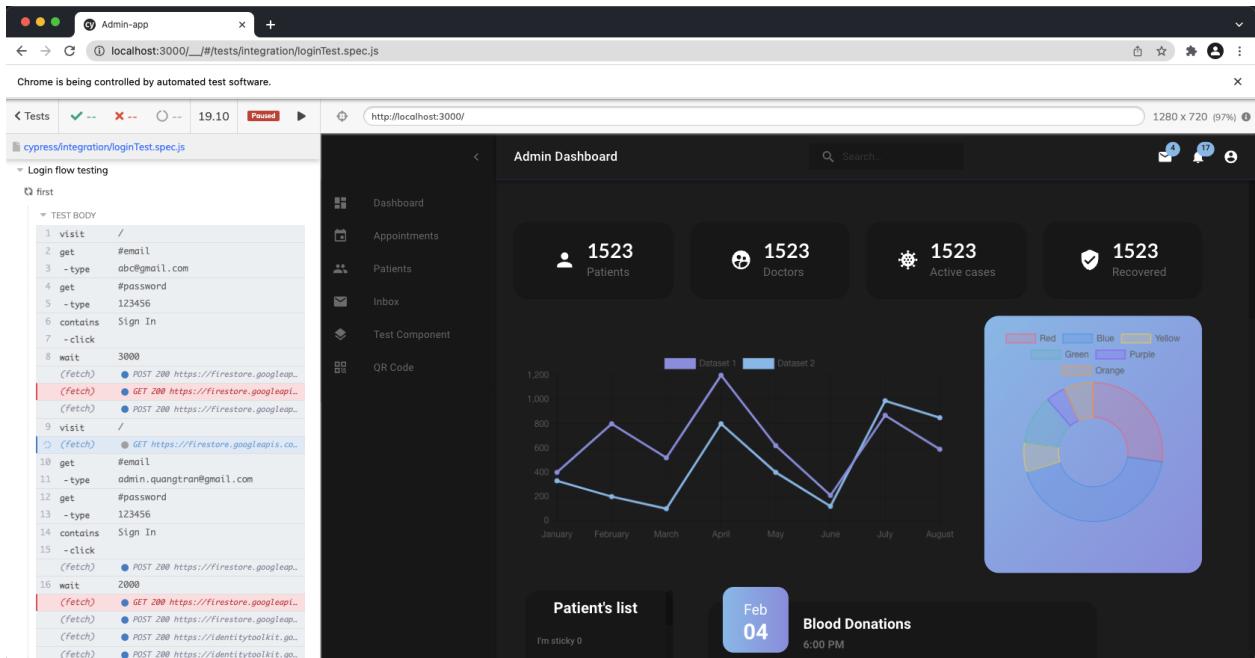


Figure 77: Screenshot from Cypress system test results

9.0 SPRINT RETROSPECTIVES

9.1 What went wrong

- **Lack of commenting on the code early:** During the sprint, we had a problem where we did not comment out our code as regularly as we should have. This sometimes created a bit of confusion amongst the team where we could look at another team member's code and would have to message them to understand it. This had no impact at all as we fixed this as soon as we did our first team meeting.
- **Struggle to adhere to consistent agile methodology:** During the sprint, a lot of team members had many other exams and assignments for other courses. This meant that many of them could not attend the two weekly meetings that we had amongst ourselves to discuss our progress of the project. Another reason for this was also that the sprint coincided during our one-week break, and a lot of the team was studying/relaxing during that time. Overall, while this did have an impact on the overall scrum meetings, and to see what the overall progress of the team was, we did have a written description of what we did, and we will be doing during the meetings in our team server.
- **The application appears differently among devices:** During the sprint, sometimes the application when we would run it, especially the client application looked different for different people. The code would be similar for everyone since we mostly dealt with pull requests and needed approval before making big changes, however, the layout sometimes and some functions would be different for some people. To fix this, we usually ended up having a call with other members of the team and went through step-by-step to solve the problems. Overall, this could have had a large impact on the project, but the issue was still solved due to teamwork we displayed.

9.2 What went well

- **Separate branches to reduce merge conflicts:** After sprint 2, we knew that we would be dealing a lot more with the backend in sprint 3. Because of this, we decided after the last sprint, that it would be a good idea to have a develop branch, where we could create other different branches from, and then later create pull requests to merge the code in. This was easier for the team to handle, instead of always merging pull requests to the main branch and then merging it to their specific branch. Overall, this had a big positive aspect on the workflow since nobody had to wait until something was merged from the main branch to continue working on their own branch.
- **Everyone doing more on the project:** In sprint 2, we only assigned one tester to write most of the tests. This meant one teammate had to write many of the tests, which also

meant that he had to understand how the other code worked to do it. While this was still not a bad thing, as we were able to do the tasks very well for the last sprint, in this sprint, everyone was able to write tests based on their own code, a code they would understand, and could hence write better tests for it. Overall, this also had a good impact on the team since we were able to learn more stuff to do, which at the end of the day is the goal of this project.

- **Team communication:** As in the other sprints, our team communication was again on point for this sprint. Multiple times in the sprint, we encountered bugs on the application which one teammate could not figure out how to solve. This prompted someone else from the team who could fix the issue to step in, and either call the person, or even screen share to help figure out the issue at hand. Another example of this would be the implementation of the backend. As most of the team are more comfortable with the backend, there was always bound to be some hiccups transitioning to that aspect, but once again, if someone was stuck in the team, another teammate who could help always stepped in to help the person in need. Overall, this had a very good impact on the project, as it did not derail anyone in the team, as well as letting us work on our tasks in a timely manner, which allowed us to finish the sprint stories in a timely manner as well.