

SOEN 390/F: SPRINT 2 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: February 23rd, 2022

1.0 INTRODUCTION	3
1.1 Purpose	3
1.1.1 System	3
1.1.2 Document	3
1.2 Targeted Users	5
1.3 Targeted Readers	5
2.0 PROJECT DESCRIPTION	6
3.0 REQUIREMENTS	7
3.1 Backlog	7
Table 2: Sprint 2 User Stories	7
3.2 User Stories	8
4.0 RELEASE PLANNING	11
4.1 Sprint 2 Summary	11
4.2 Sprint 3 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	20
5.3 Domain Model Diagram	23
5.4 Database Design	24
5.5 Physical Design of the Database	26
5.5 Sequence Diagrams	28
6.0 RISK MANAGEMENT	32
6.1 Purpose of the Risk Management Plan	32
6.2 Risk Management Procedure	32
6.2.1 Process	32
6.2.2 Risk Identification	33
6.3 Risk Analysis	35
6.3.1 Qualitative Risk Analysis	36

6.3.2 Quantitative Risk Analysis	36
6.4 Risk Response Planning	36
6.5 Risk Monitoring, Controlling and Reporting	37
7.0 USER INTERFACE DESIGN	38
7.1 Sprint 1 UI Mockups	38
7.1.1 Admin Portal Sprint 1	38
7.2 Sprint 2 UI Mockups	44
7.2.1 Admin Portal Sprint 2	44
7.2.2 Client Portal Sprint 2	46
7.3 Sprint 3 UI Mockups	53
7.3.1 Dark Theme Update	53
7.3.1.1 Dark Theme Update Admin Portal	53
7.3.1.2 Dark Theme Update ClientPortal	53
7.3.2 Admin Portal Sprint 3	60
7.3.3 Client Portal Sprint 3	61
The following figures are the UI mockups for the Client portal.	61
Figure 45: UI Client Patient for User Story	65
Figure 46: UI Client Patient for User Story	66
Figure 47: UI Client Patient for User Story	67
8.0 TESTING PLAN AND REPORT	68
8.1 Unit Testing	68
8.2 Code Coverage	68
8.3 Acceptance Testing	69
8.4 System Tests	71
8.5 Testing Procedure	72

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.

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. **7 user stories** have been elicited for a total of **41 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 2:

ID	Name	USP	Priority
Issue-3	As a doctor, health official or immigration officer, I want to raise flags on certain COVID-19 patients so that their updates are prioritized over others.	5	1
Issue-13	As a doctor or health official, I want to be able to contact my patients via email or chatbot within the application, and vice versa so that they can receive immediate help.	8	3
Issue-17	As an administrator, I want to know how many patients are assigned to each doctor so that no doctor is overloaded while some others do not have as many patients.	2	1
Issue-18	As an administrator, I want to assign doctors to incoming patients so the doctors are aware of which patients they are assigned.	5	1
Issue-20	As an administrator, I want to re-assign a patient to another doctor when his/her doctor has an emergency to attend to so that the doctor focuses on the patient in the emergency.	5	5
Issue-22	As a user, I want to have a barcode / QR code for each individual record, so that it can be shared / read by other media or other devices.	13	5

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.	3	1
Total		41	

Table 2: Sprint 2 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)

Issue-3	As a doctor, health official or immigration officer, I want to raise flags on certain COVID-19 patients so that their updates are prioritized over others.
USP	5
Priority	1
Description	<p>Acceptance criteria:</p> <p>Given: The user is logged in as a doctor, health official, or immigration officer and is on a patient profile.</p> <p>When: The user clicks on the "flag".</p> <p>Then: The application priorities the COVID-19 patient.</p>

Table 3: User Story 1

<u>Issue-13</u>	As a doctor or health official, I want to be able to contact my patients via email or chatbot within the application, and vice versa so that they can receive immediate help.
USP	8
Priority	3
Description	<p>Additonal details:</p> <p>Note1:</p> <p>The communication could be marked as “urgent” or “emergency” for priority review by the doctor.</p> <p>Note2:</p> <p>For the chatbot, we can use a 3rd party provider such as Messenger / Tidio to integrate them into our platform. Or, if we have time, we can build our own chat system.</p> <p>Acceptance criteria:</p> <p>Given: The user is logged in as a doctor or health official and is on the inbox page.</p> <p>When: The user clicks on the "message" button for a specific patient.</p> <p>Then: The application opens a messenger window with the patient.</p>

Table 4: User Story 2

<u>Issue-17</u>	As an administrator, I want to know how many patients are assigned to each doctor so that no doctor is overloaded while some others do not have as many patients.
USP	2
Priority	1
Description	<p>Acceptance criteria:</p> <p>Given: The user is logged in as an administrator.</p> <p>When: The user goes to the doctor and patients page.</p> <p>Then: The application displays all doctors and the number of patients they have been assigned.</p>

Table 5: User Story 3

<u>Issue-18</u>	As an administrator, I want to assign doctors to incoming patients so the doctors are aware of which patients they are assigned.
USP	5
Priority	3
Description	<p>Acceptance criteria:</p> <p>Given: User sees patient without doctor assigned</p> <p>When: User clicks on patient listed without doctor</p> <p>Then: User will be redirected to doctor assigning module</p>

Table 6: User Story 4

Issue-20	As an administrator, I want to re-assign a patient to another doctor when his/her doctor has an emergency to attend to so that the doctor focuses on the patient in the emergency.
USP	5
Priority	5
Description	<p>Additional Info:</p> <p>Linked to : Issue-18</p> <p>Acceptance criteria:</p> <p>Given: The user is logged in as an administrator and is on the patient list page.</p> <p>When: The user clicks on the dropdown under "Assigned Doctor".</p> <p>Then: The application shows a list of doctors that are available.</p>

Table 7: User Story 5

<u>Issue-22</u>	As a user, I want to have a barcode / QR code for each individual record, so that it can be shared / read by other media or other devices.
USP	13
Priority	5
Description	<p>Acceptance criteria:</p> <p>Given: The user is logged in.</p> <p>When: The user on on a patient profile.</p> <p>Then: The application displays a QR code which houses the patient information.</p>

Table 8: User Story 6

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.
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 navbar.</p> <p>Then: The user is redirected to their profile displaying a list of given details about the patient.</p>

Table 9: User Story 7

4.0 RELEASE PLANNING

4.1 Sprint 2 Summary

Sprint 2 focused on delivering a variety of features including prioritizing patient updates, contact between doctors and patients, displaying the frontend for identifying how many patients are assigned to each doctor, displaying the frontend for assigning doctors to patients, providing barcodes/QR codes for each record and displaying the frontend for client patient profile. In this sprint, work was performed in the frontend and backend, although the majority of the work was completed in the frontend. The problems we faced in this sprint included CSS conflicts due to generic classNames which is an issue that first started to surface in sprint 1. In response to this, we decided we should follow the convention of naming our classes specific to our component rather than simply giving a generic className such as "text".

During this sprint, issues 37, 2, 4 and 16 were brought forward to be completed in Sprint 3 due to time constraints. Moreover, issue 14 was brought forward to be completed in Sprint 4 as it was deemed better suited for a later sprint once the core features with greater priority are completed.

Story ID	Story Title	USP	Status
Issue-3	Prioritize Patient Updates	5	DONE
Issue-13	Contact between patients & doctors	8	DONE
Issue-17	Frontend display for "Identify how many patients are assigned to each doctor"	2	DONE
Issue-172*	Frontend display for "Assign doctors to patients"	3	DONE
Issue-20	[Additional] Re-assign a patient to another doctor	2	DONE
Issue-22	[Additional] Provide barcodes / QR codes or each record	13	DONE

<u>Issue-103</u>	Frontend Display for "Profile for patient"	3	DONE
<u>Issue-37</u>	Allow client to edit profile page (patients)	8	PUSHED TO SPRINT 3
<u>Issue-2</u>	Arrange Appointments	13	PUSHED TO SPRINT 3
<u>Issue-4</u>	Trace & notify COVID-19 patients	3	PUSHED TO SPRINT 3
<u>Issue-16</u>	Display / show that a patients' status updates have been reviewed	2	PUSHED TO SPRINT 3
<u>Issue-14</u>	Different visibility for different users	5	PUSHED TO SPRINT 4
Total		67	36

Table 10: Sprint 2 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 3 Planning

Sprint 3 will focus on delivering the next set of important features including tracing & notifying COVID-19 patients, notifying a doctor when a status is reupdated on the same day, arranging appointments, monitoring status and symptoms of confirmed and unconfirmed patients, showing that a patients' status updates have been reviewed, allowing admin to manage client account, allowing patients to update status everyday, allowing patients to edit their profile page, including a patient diary to update symptoms and locations, enabling patients to view the basic information of the assigned doctor, allowing patients to view history of his/her symptoms + diary and 3 other optional user stories.

In this sprint, our team is planning on covering 14 user stories which sum up to a total of 66 user story points. As compared to the previous sprint, the workload can be seen to have remained relatively the same for Sprint 3. 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 third sprint, we are also catching up on four user stories (Issue-37, Issue-2,

Issue-4 and Issue-14) 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>	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>	Arrange Appointments	13	
<u>Issue-11</u>	Monitor status & symptoms of confirmed and unconfirmed patients	2	
<u>Issue-16</u>	Display / show that a patients' status updates have been reviewed	2	
<u>Issue-19</u>	Allow admin to manage client account	5	
<u>Issue-29</u>	Patient updates status everyday	8	
<u>Issue-37</u>	Allow client to edit profile page (patients)	8	
<u>Issue-38</u>	Patient diary to update symptoms, locations	5	
<u>Issue-39</u>	Allow patient to view the basic info of the assigned doctor	2	
<u>Issue-41</u>	Allow patients view the history of his/her updated symptoms + diary	5	
<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		66	

Table 11: Sprint 3 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.

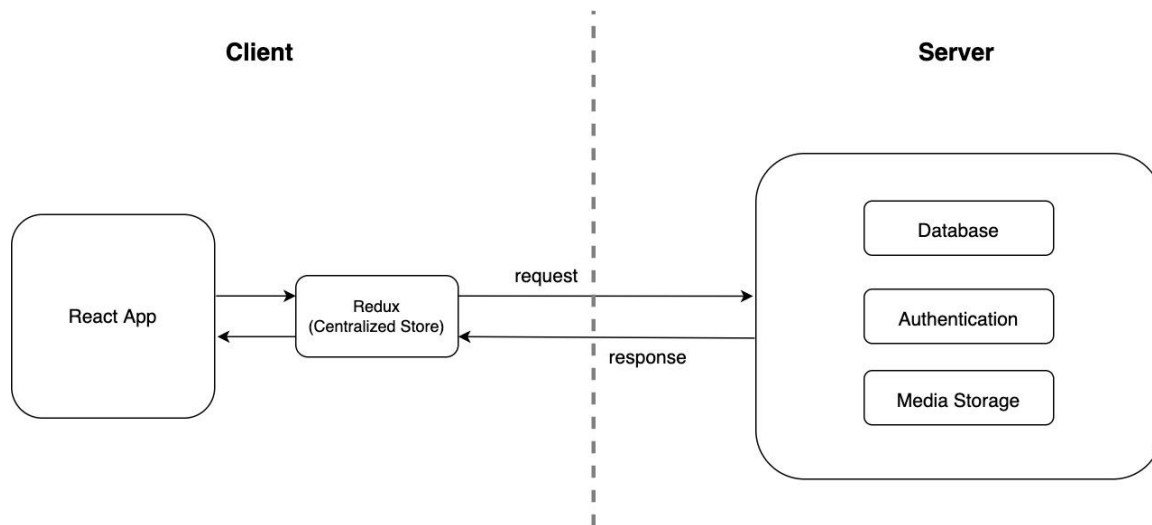


Figure 1: High-level overview system diagram

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.

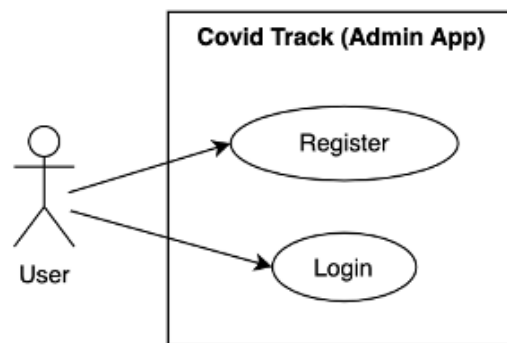


Figure 2: Use case diagram for authentication and registration on the Admin App.

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

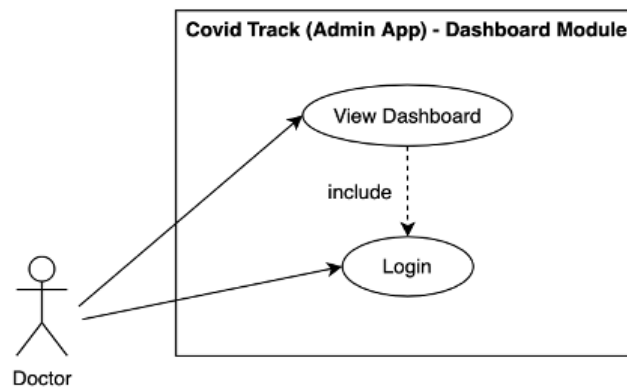


Figure 3: Use case diagram for dashboard view on the Admin App.

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

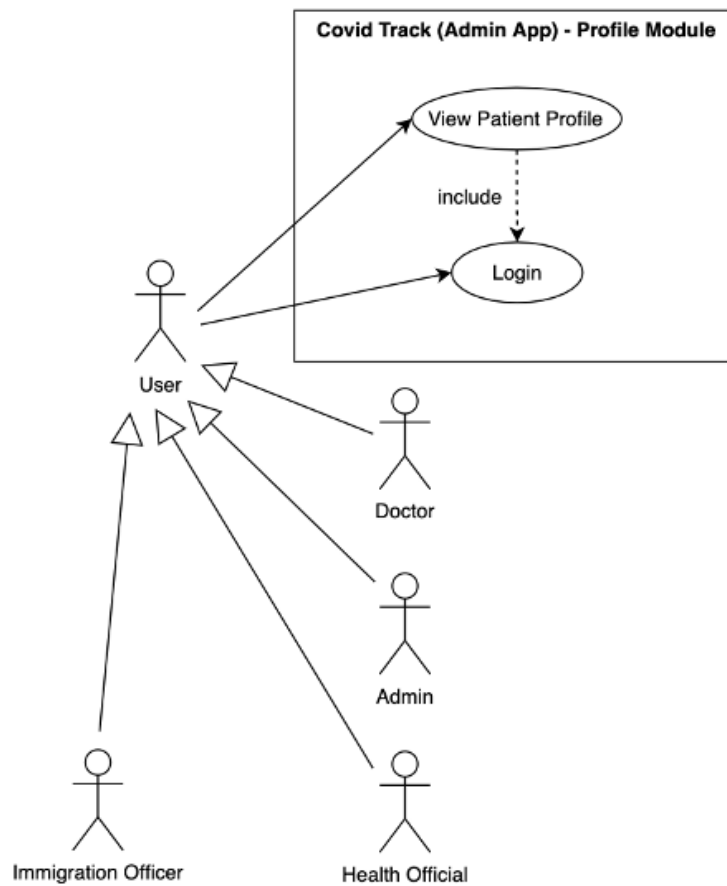


Figure 4: Use case diagram for patient profile view on the Admin App.

- **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.

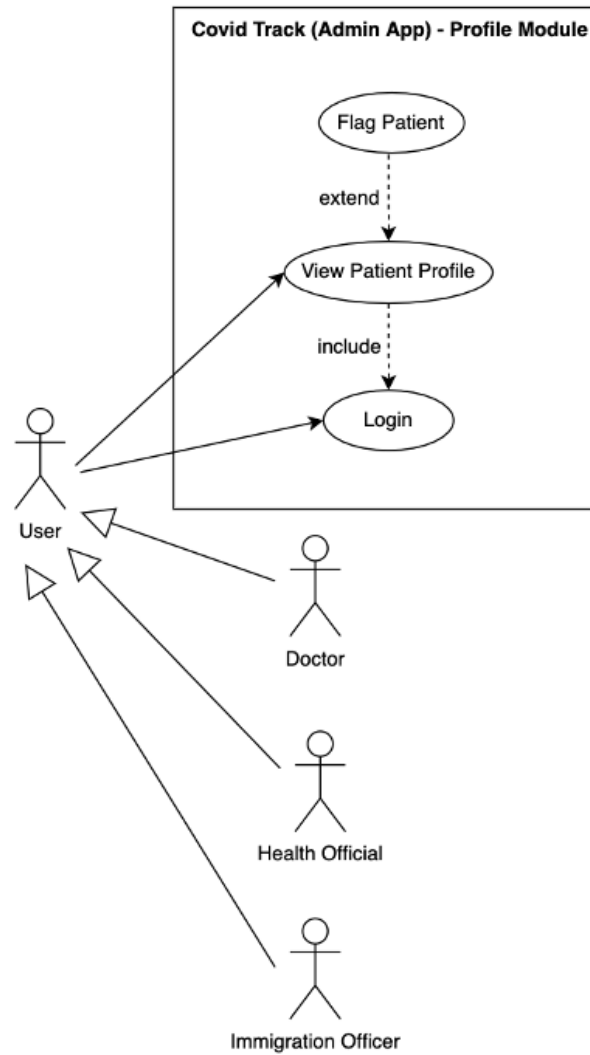


Figure 5: Use case diagram for prioritizing patient updates on the Admin App.

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

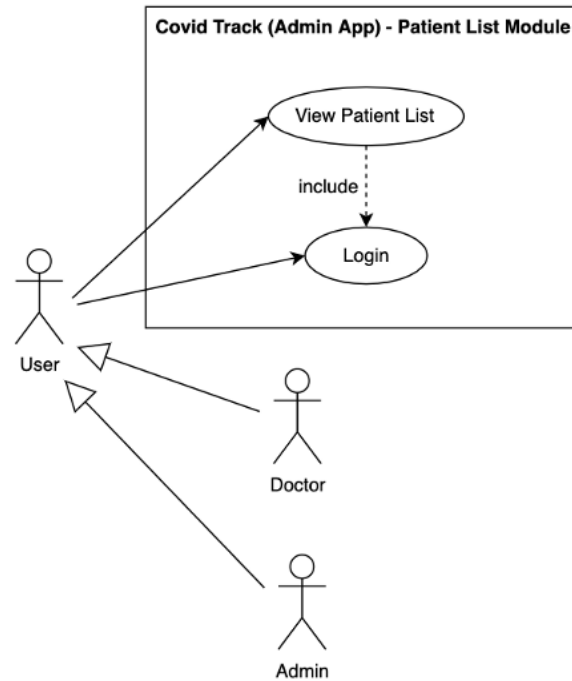


Figure 6: Use case diagram for patient list view on the Admin App.

- **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.

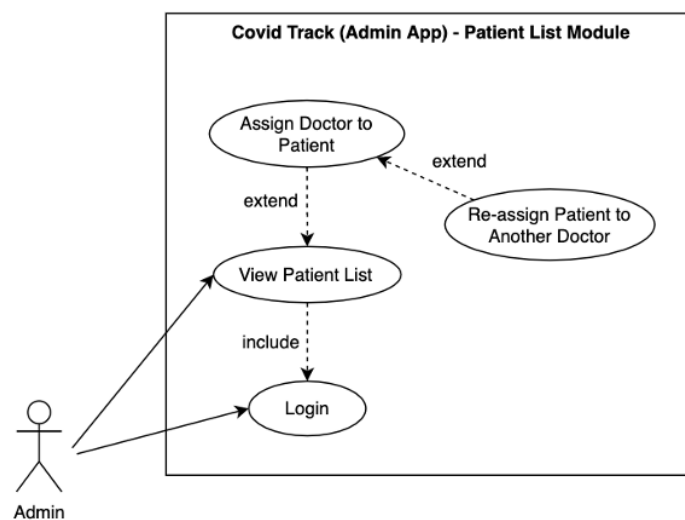


Figure 7: Use case diagram for assigning doctors to patients and re-assigning a patient to another doctor on the Admin App.

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

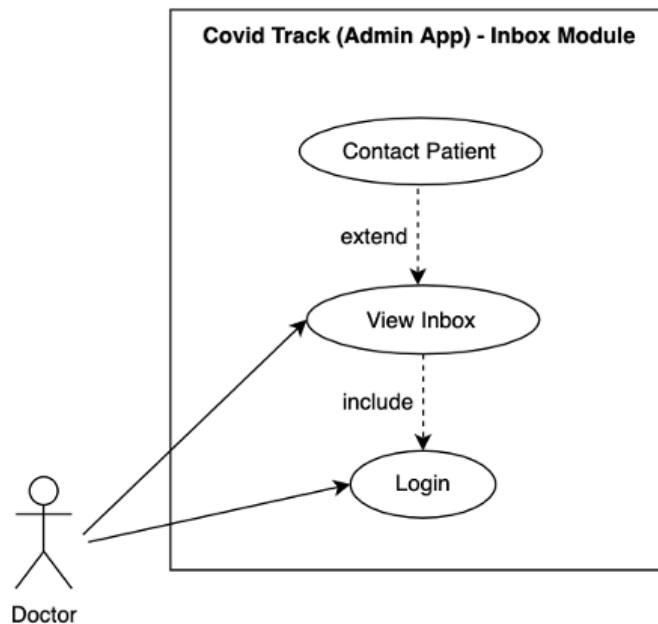


Figure 8: Use case diagram for contacting a patient on the Admin App.

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.

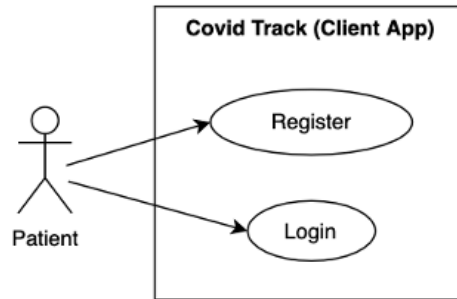


Figure 9: Use case diagram for authentication and registration on the Client App.

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

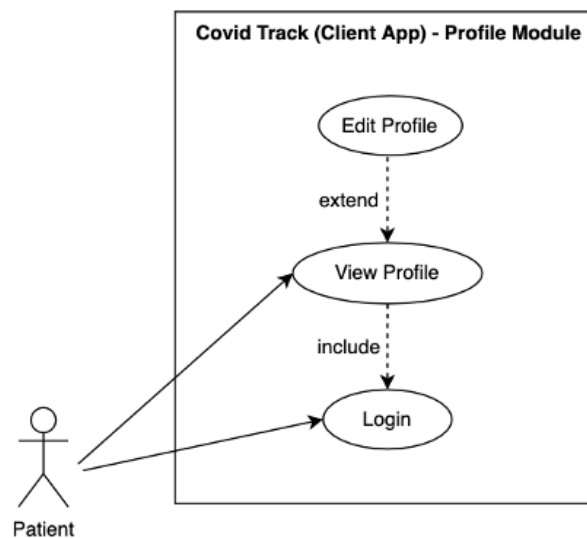


Figure 10: Use case diagram for profile view and edit on the Client App.

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

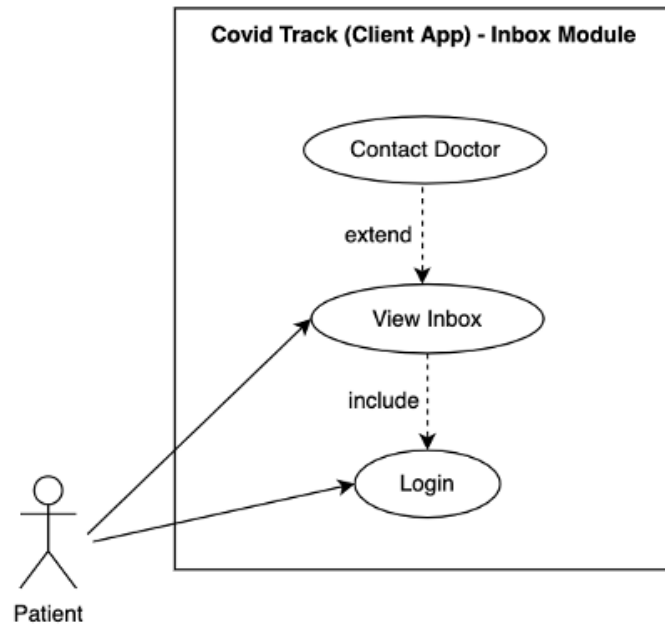


Figure 11: Use case diagram for contacting a doctor on the Client App.

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*.

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 appointment at a time to ensure that they do not overload the system. However, for doctors, they can have multiple appointments with their patients.

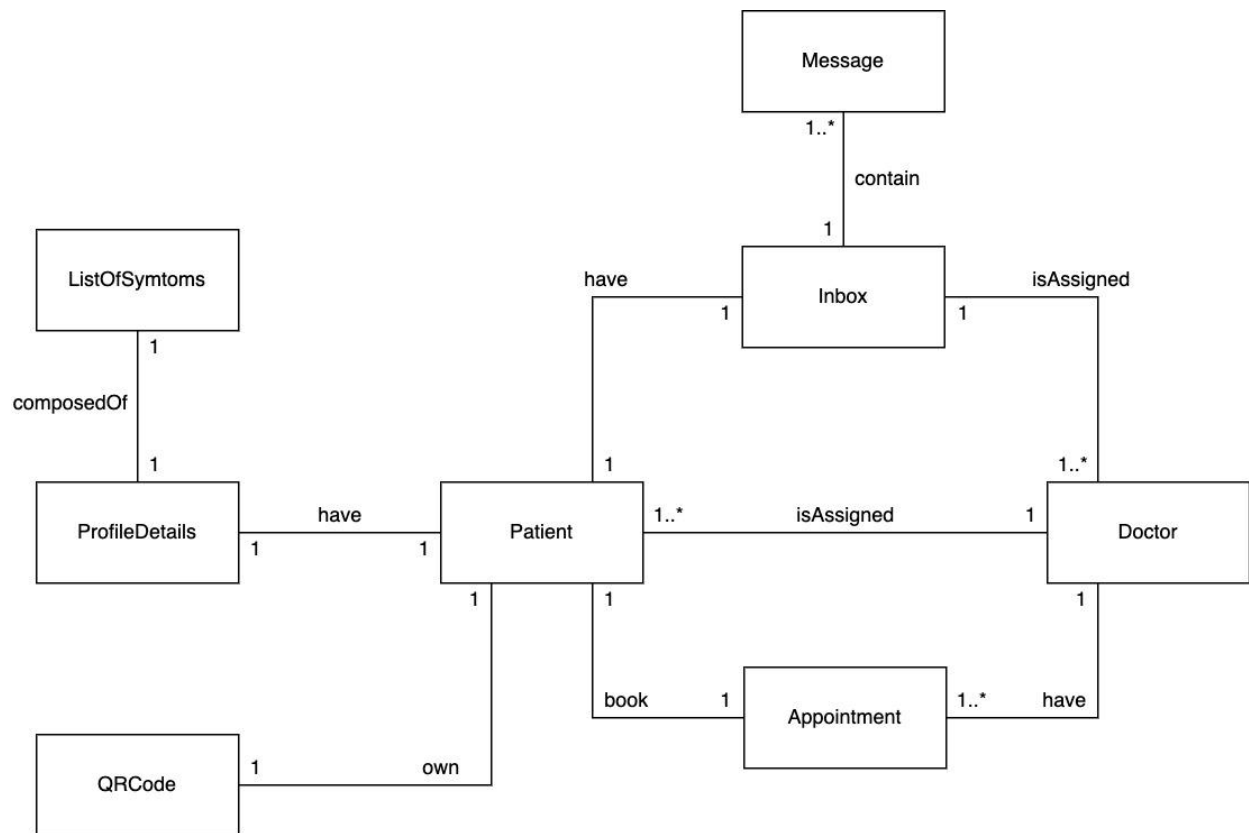


Figure 12: Domain model diagram

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*, and a unique *QRCode* that belongs to his/her profile. For the appointment, it makes sense to allow a doctor to have many *Appointments*, however, each patient should be allowed to make only one appointment at a time to avoid overwhelming the system.

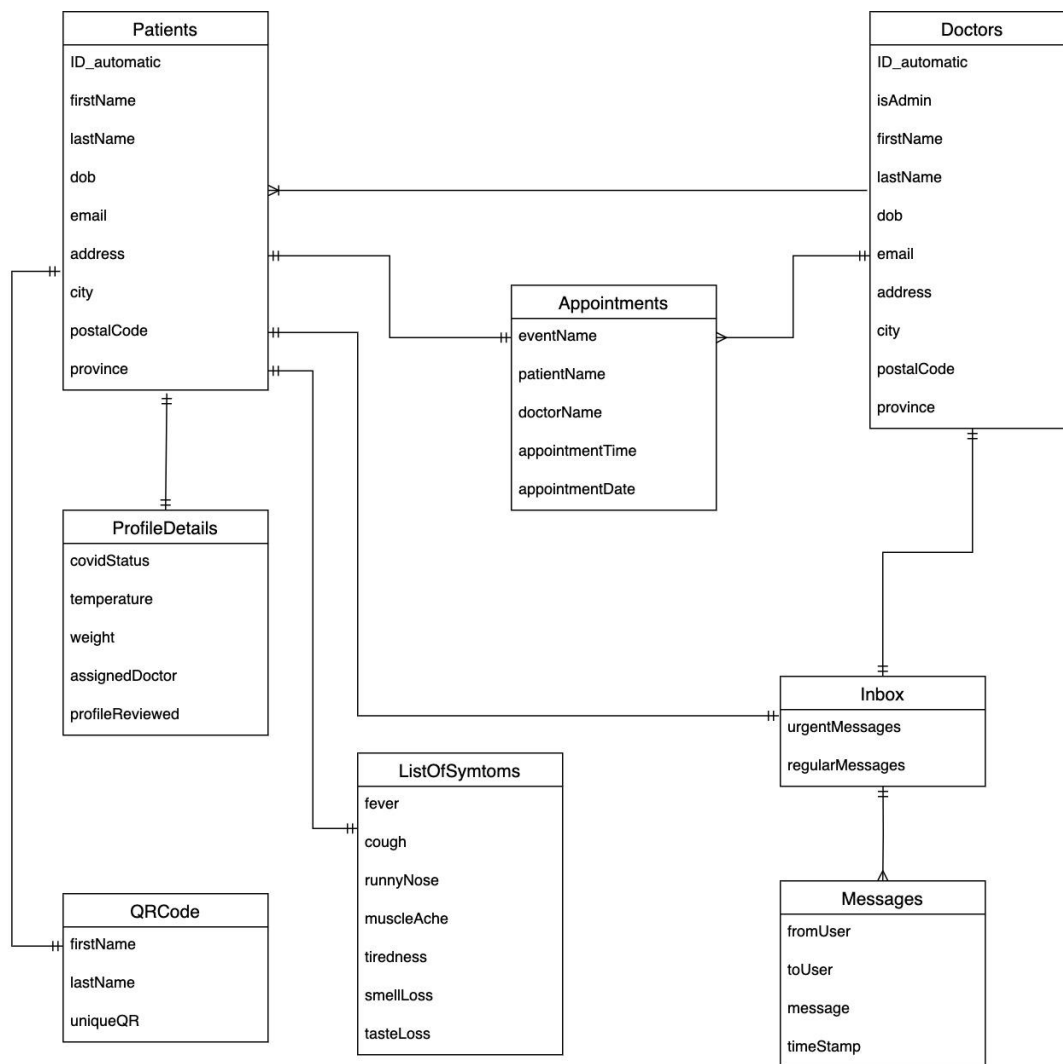


Figure 13: Database design

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.

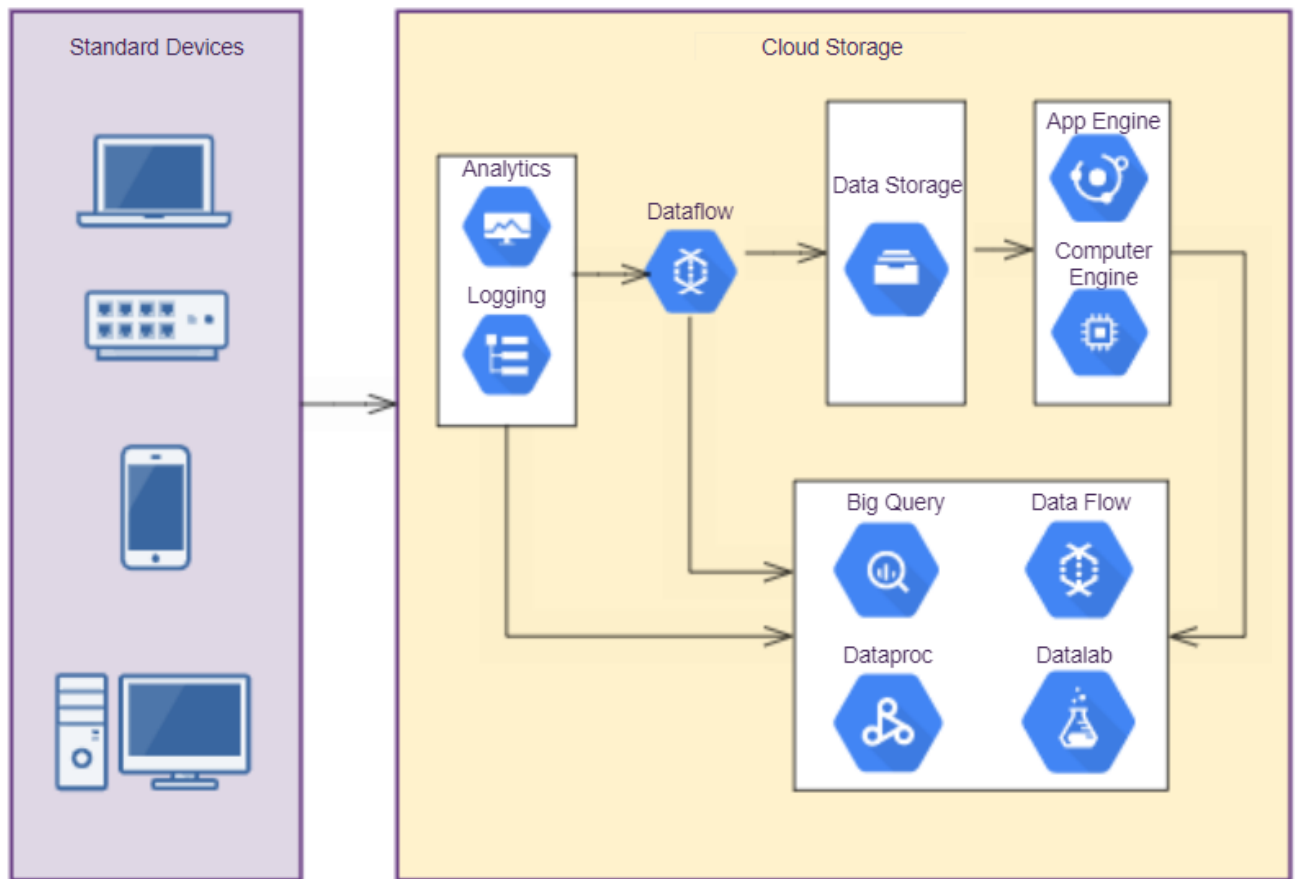


Figure 14: Physical Design of the Database

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.

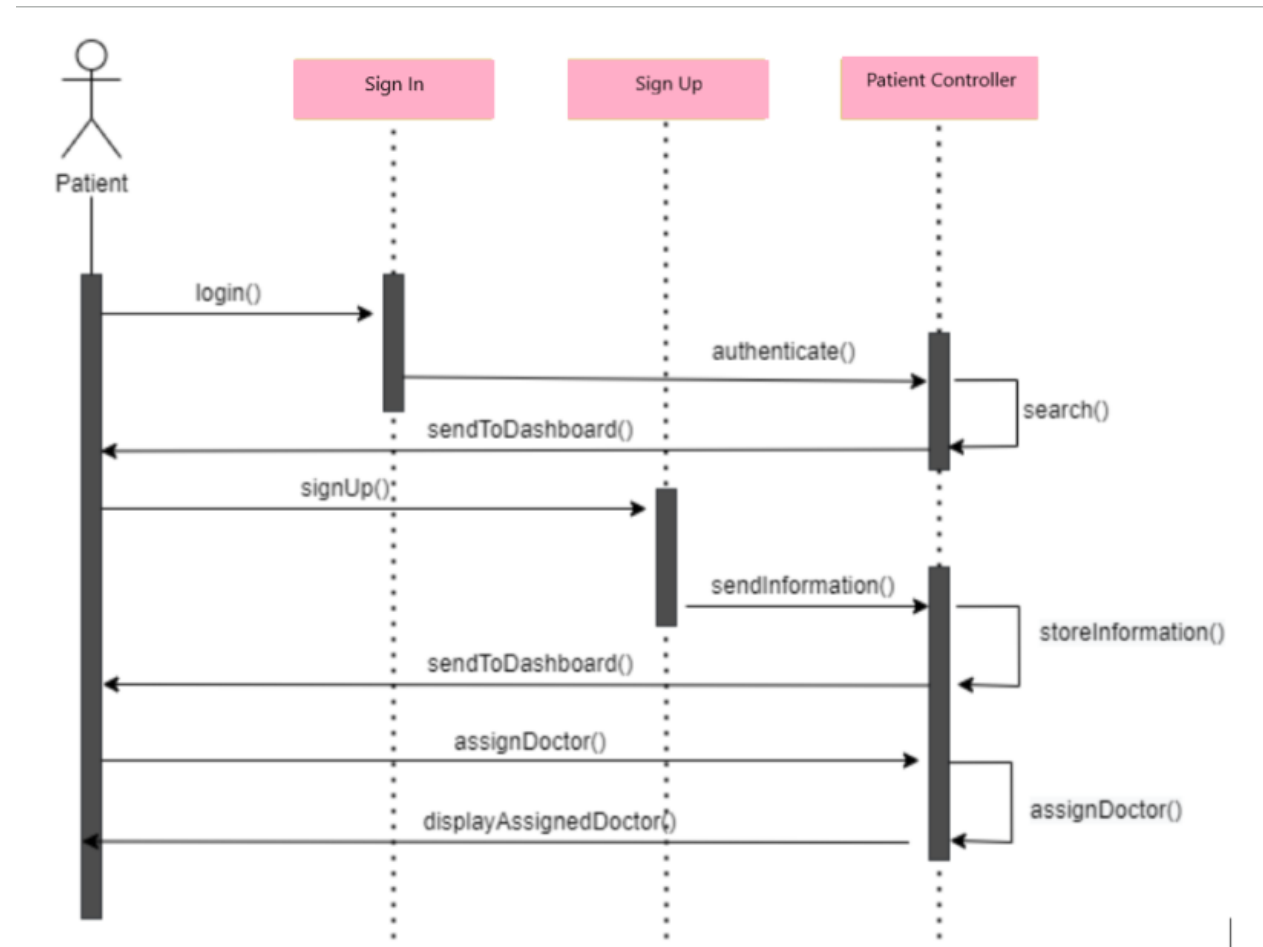


Figure 15: Sign In/ Sign Up for Patient Sequence Diagram

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.

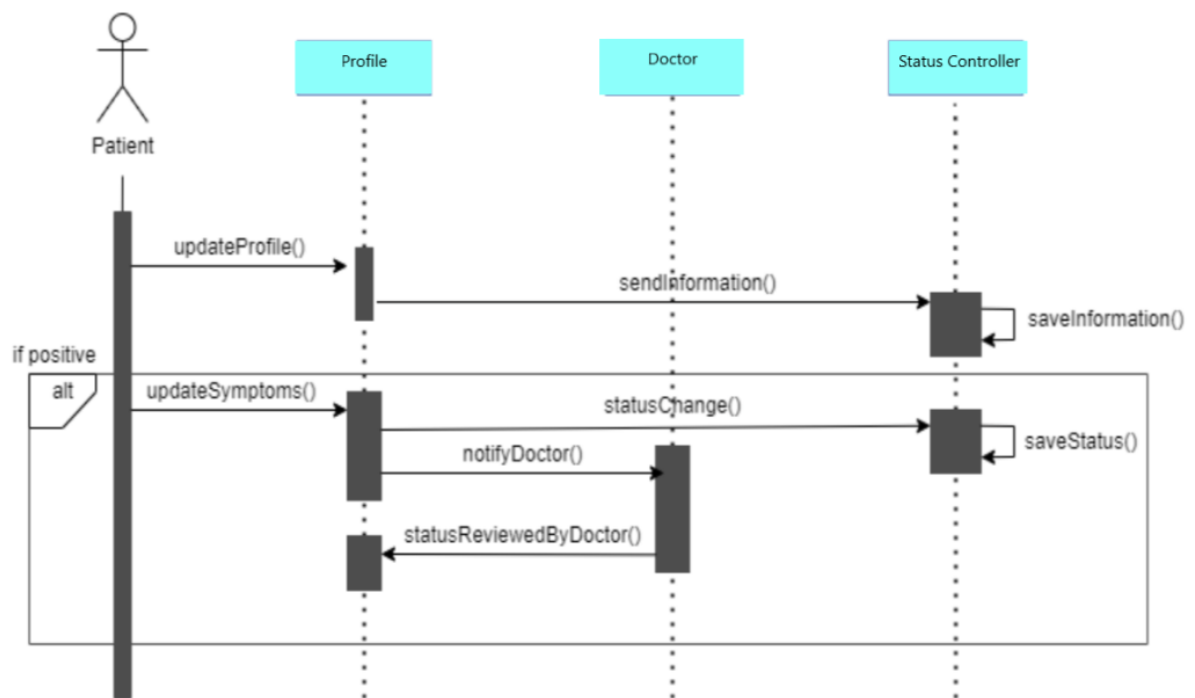


Figure 16: Status Update for Patients Sequence Diagram

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.

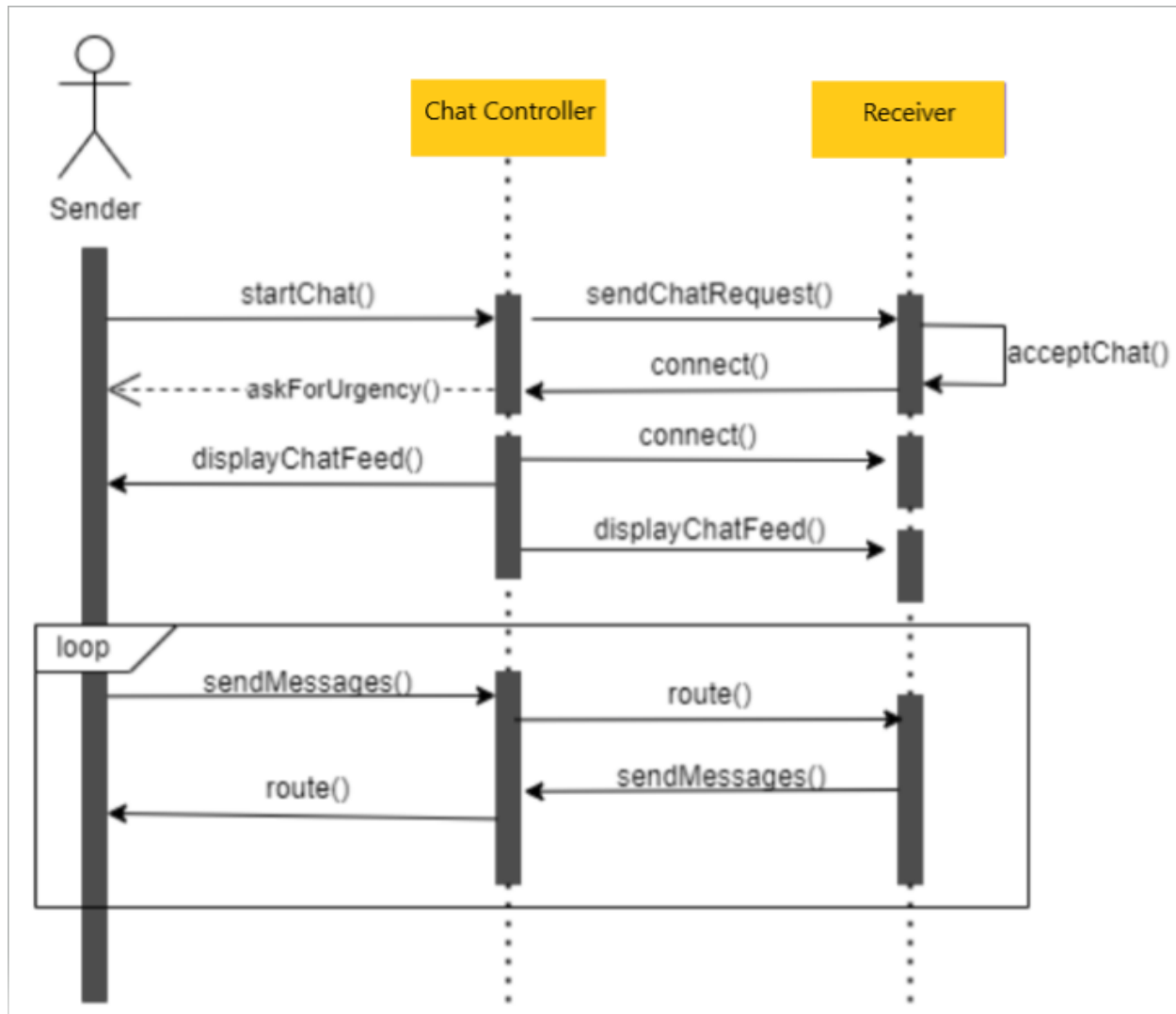


Figure 17: Chatting System Sequence Diagram

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.

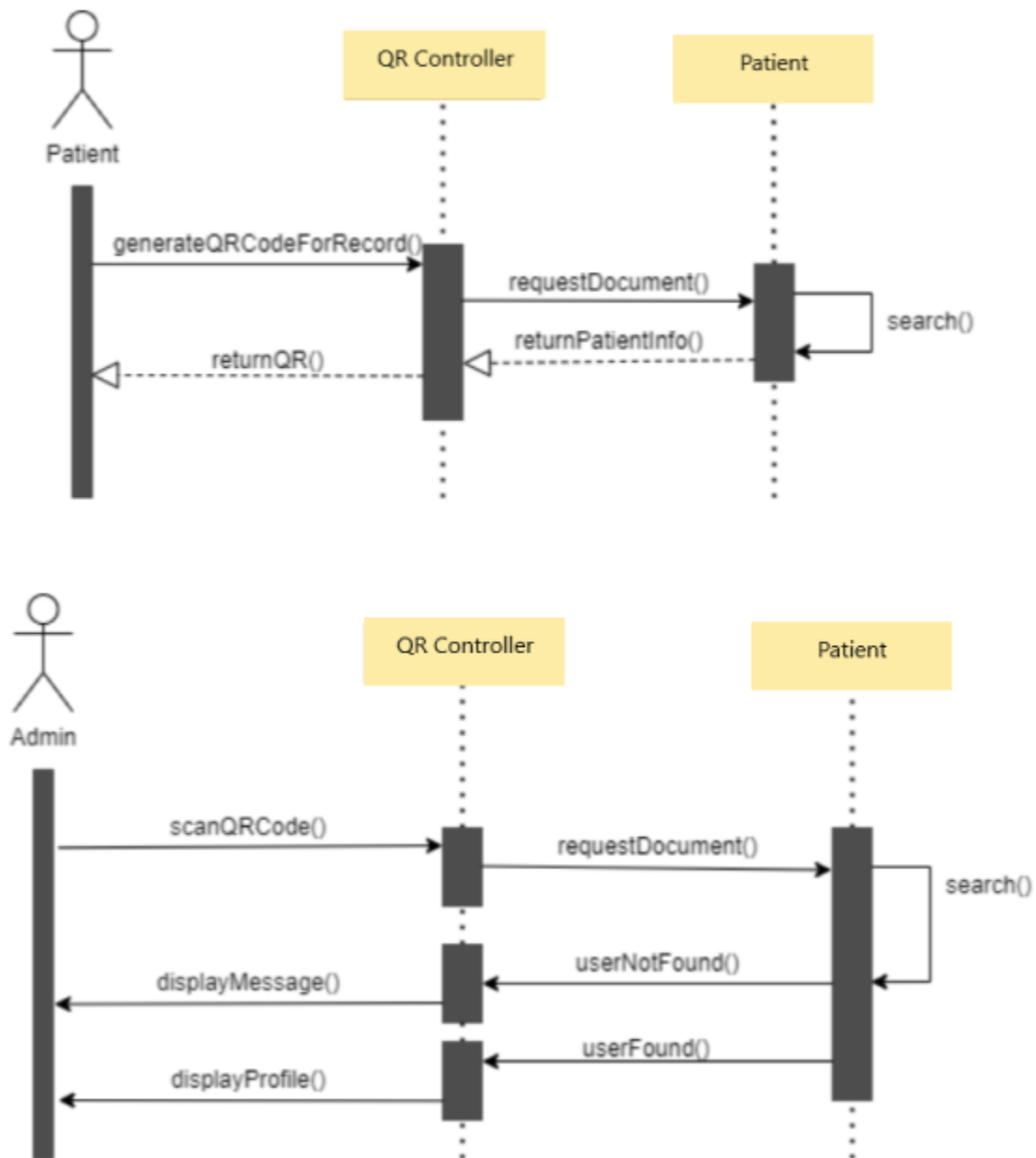


Figure 18: QR Feature Sequence Diagram for Patients and Admins

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

Version #	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				
High				R5	
Moderate			R6, R7, R8	R1, R2, R3, R4	
Low		R9, R10			
	Very low	Low	Moderate	High	Very High
	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))

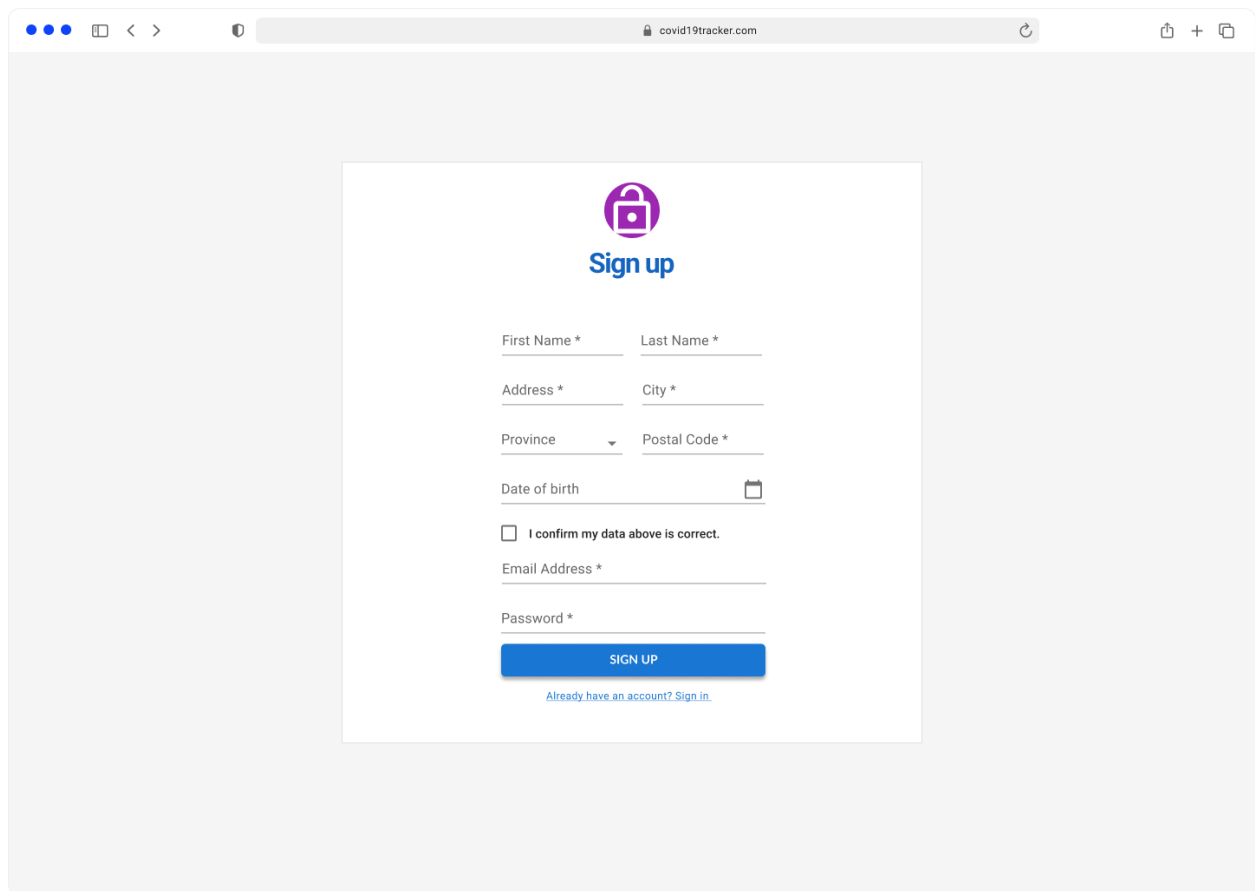
In this section, the **UI mockup** for each feature (user story) completed in Sprint 1 and the ones we intend to complete in Sprint 2.

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.



The image shows a web browser window with the URL "covid19tracker.com". The main content is a sign-up form titled "Sign up" with a purple padlock icon. The form fields are: First Name *, Last Name *, Address *, City *, Province (dropdown), Postal Code *, Date of birth (with a calendar icon), a checkbox for "I confirm my data above is correct.", Email Address *, and Password *. A blue "SIGN UP" button is at the bottom, with a link "Already have an account? Sign in" below it.

Figure 19: UI Admin Sign-up for User Story [Issue-7](#)

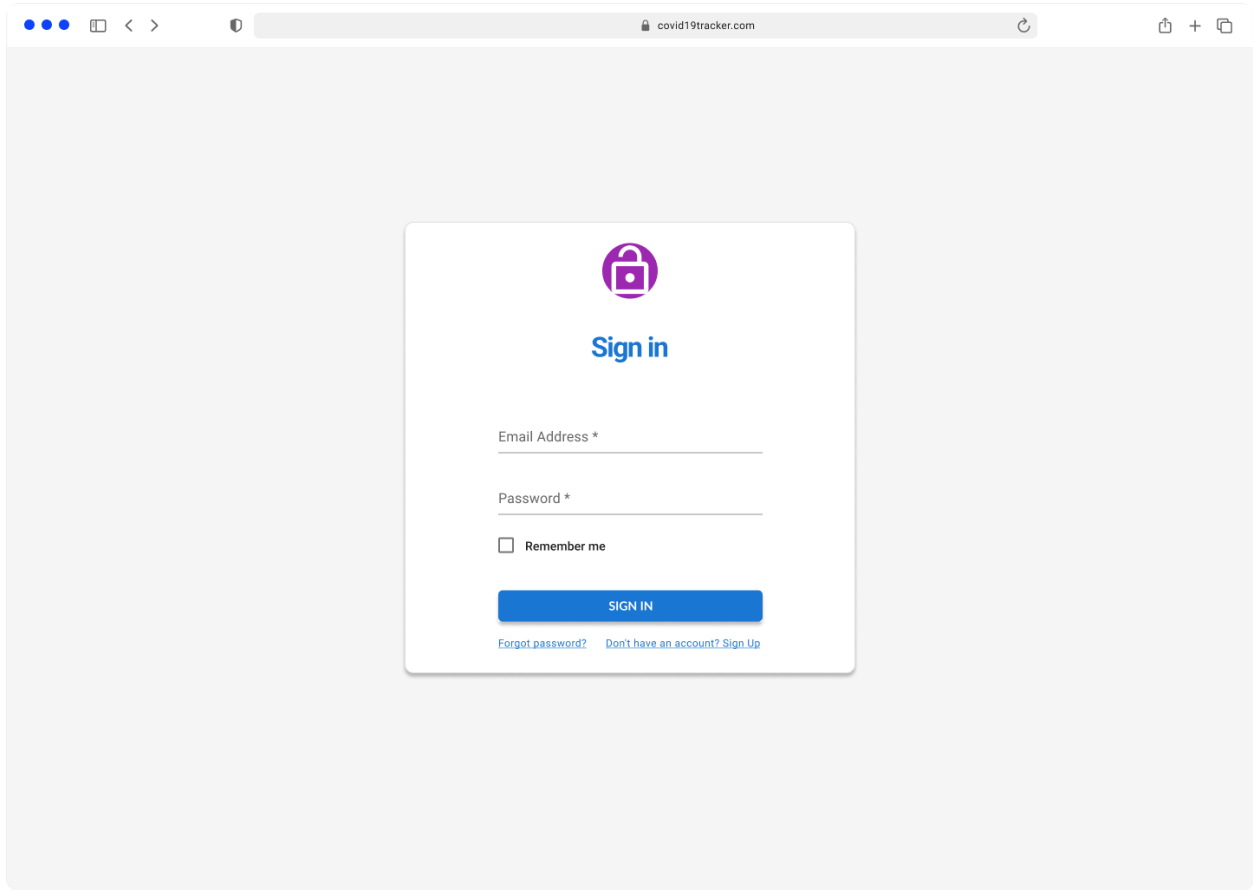


Figure 20: UI Admin Sign-in for User Story [Issue-7](#)

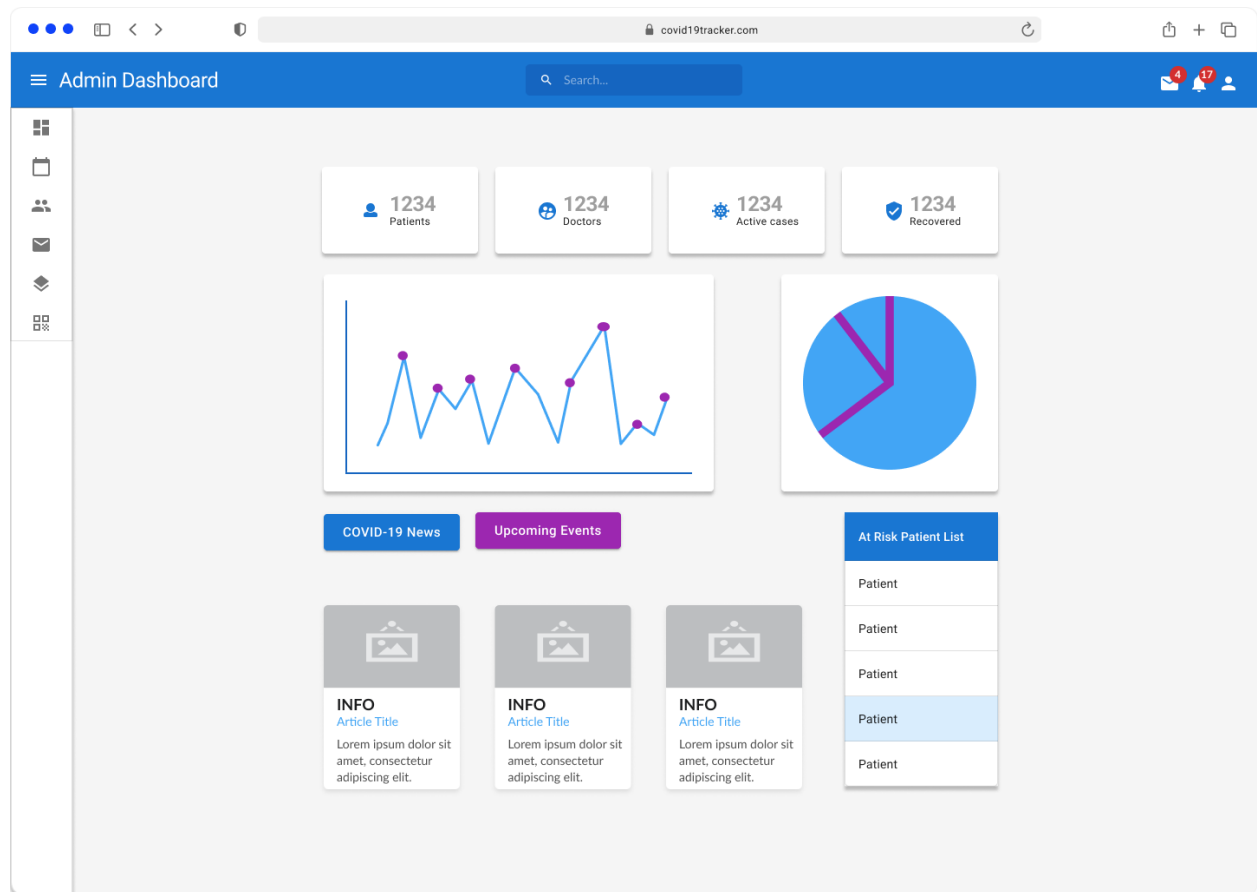


Figure 21: UI Admin Dashboard for User Story [Issue-5](#)

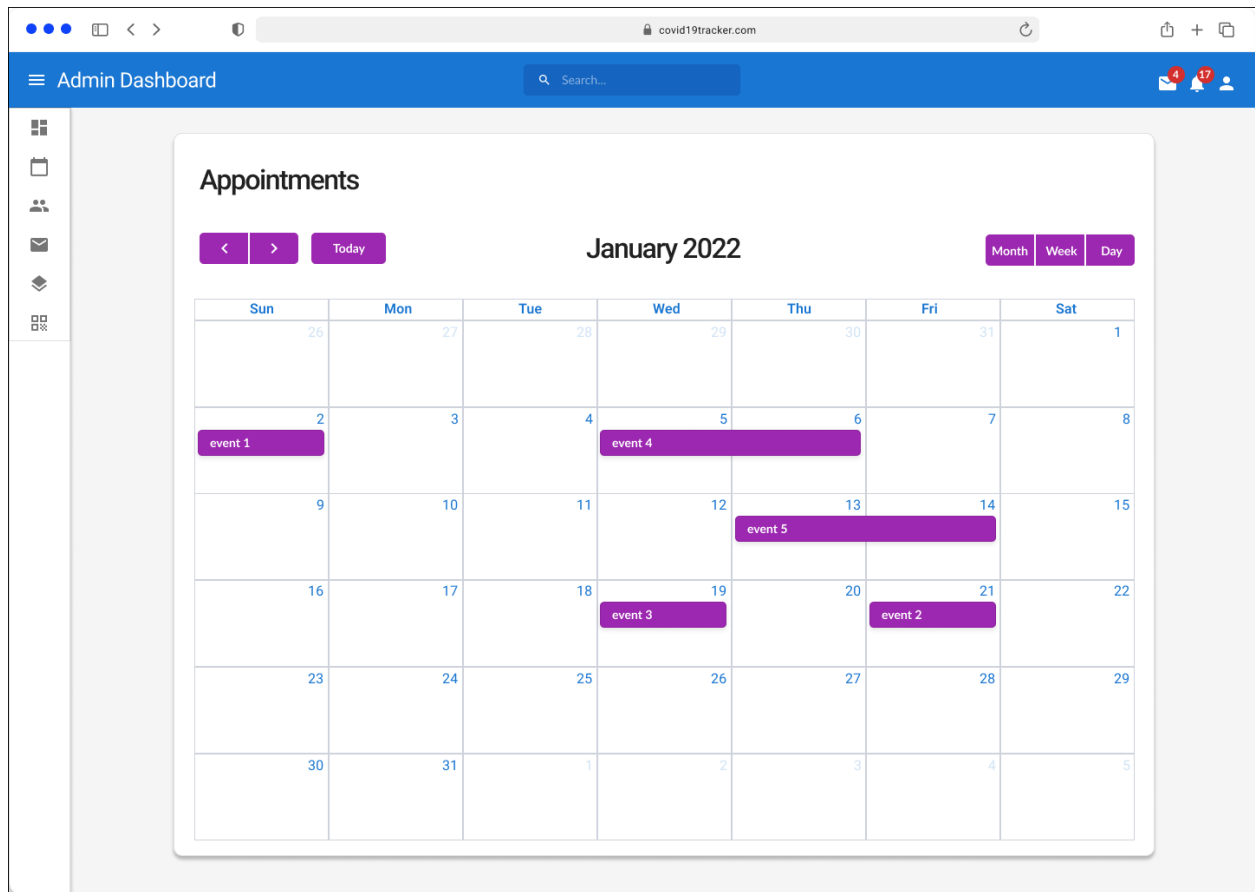


Figure 22: UI Admin Appointments for User Story [Issue-5](#)

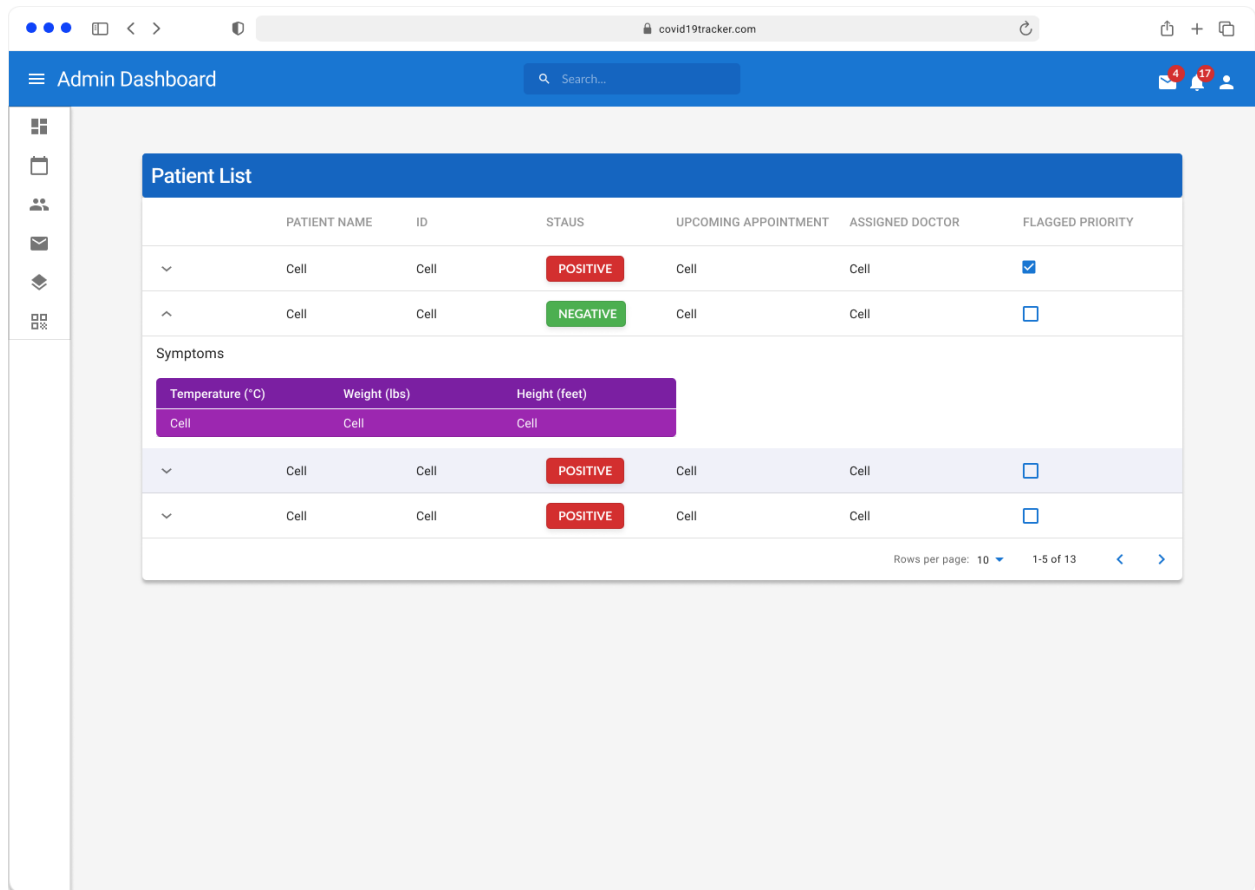


Figure 23: UI Admin Patient List for User Story [Issue-89](#)

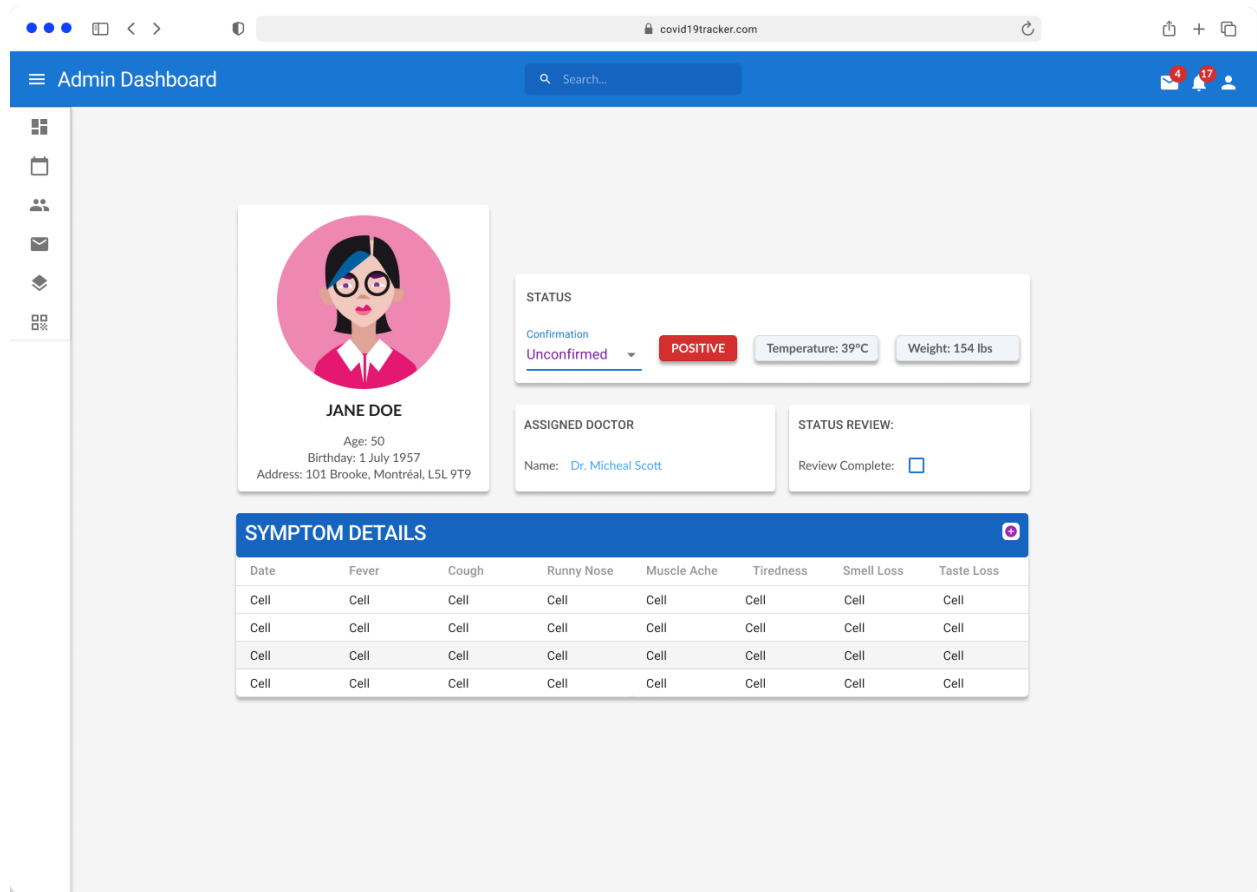


Figure 24: UI Admin Patient Details for User Story [Issue-9](#)

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.

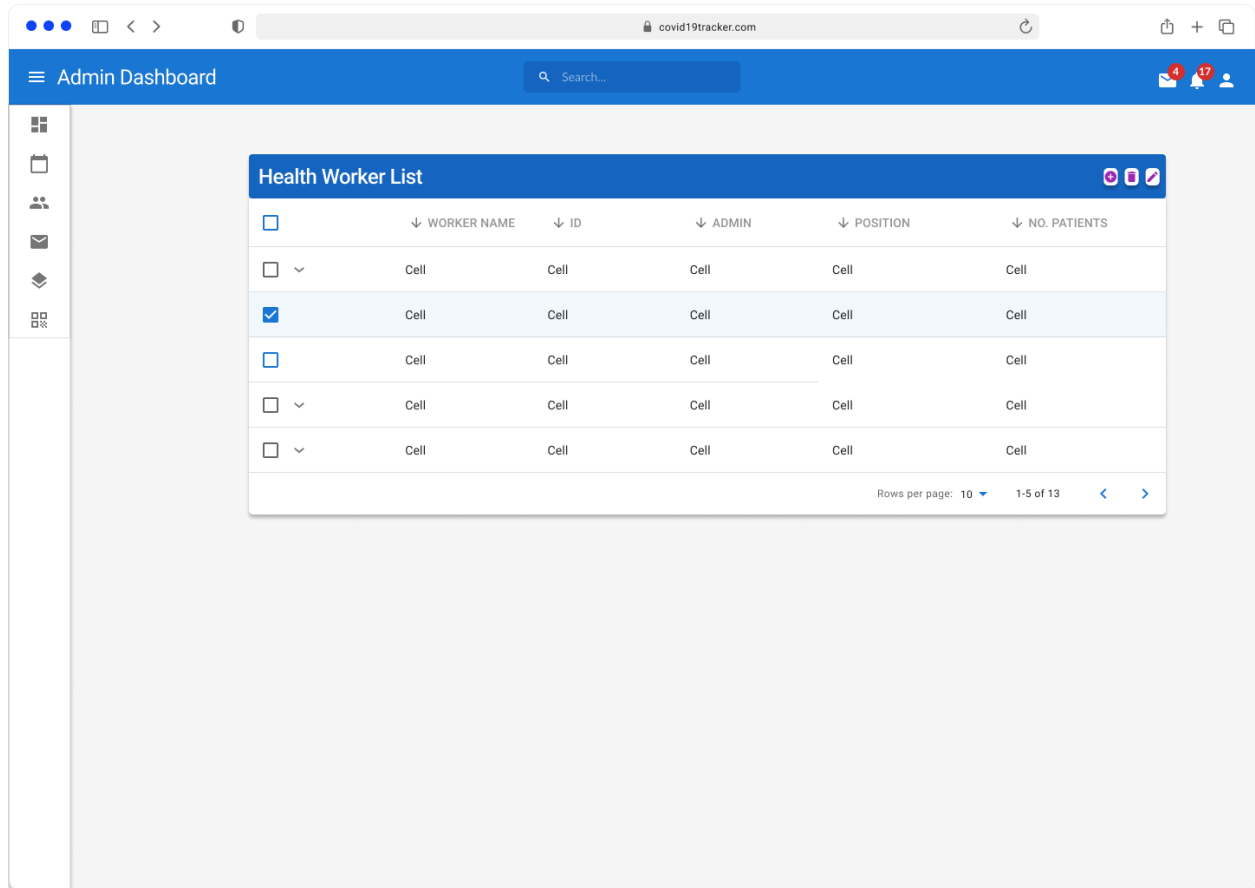


Figure 25: UI Admin Health Worker List for User Story [Issue-17](#) and [Issue-19](#)

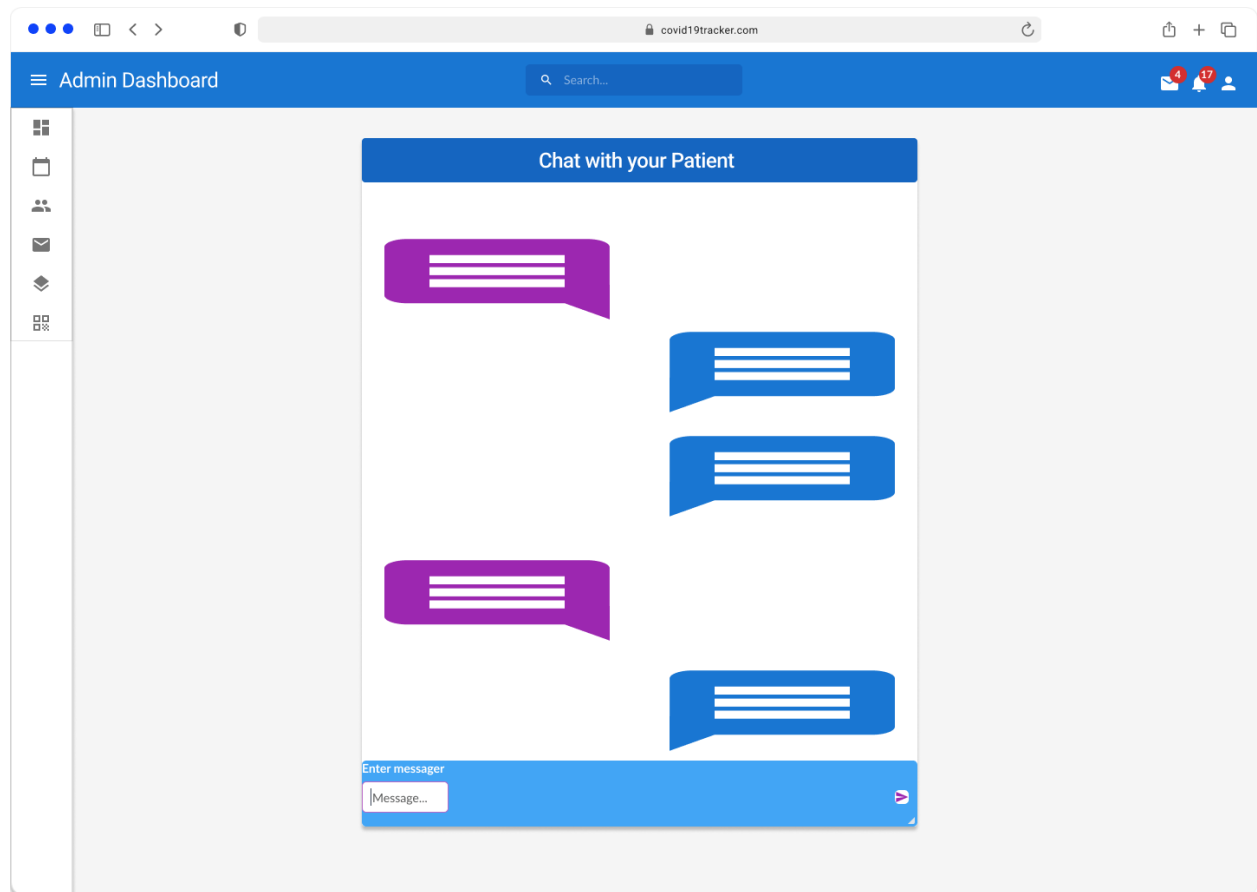


Figure 26: UI Admin Chat with patient for User Story [Issue-13](#)

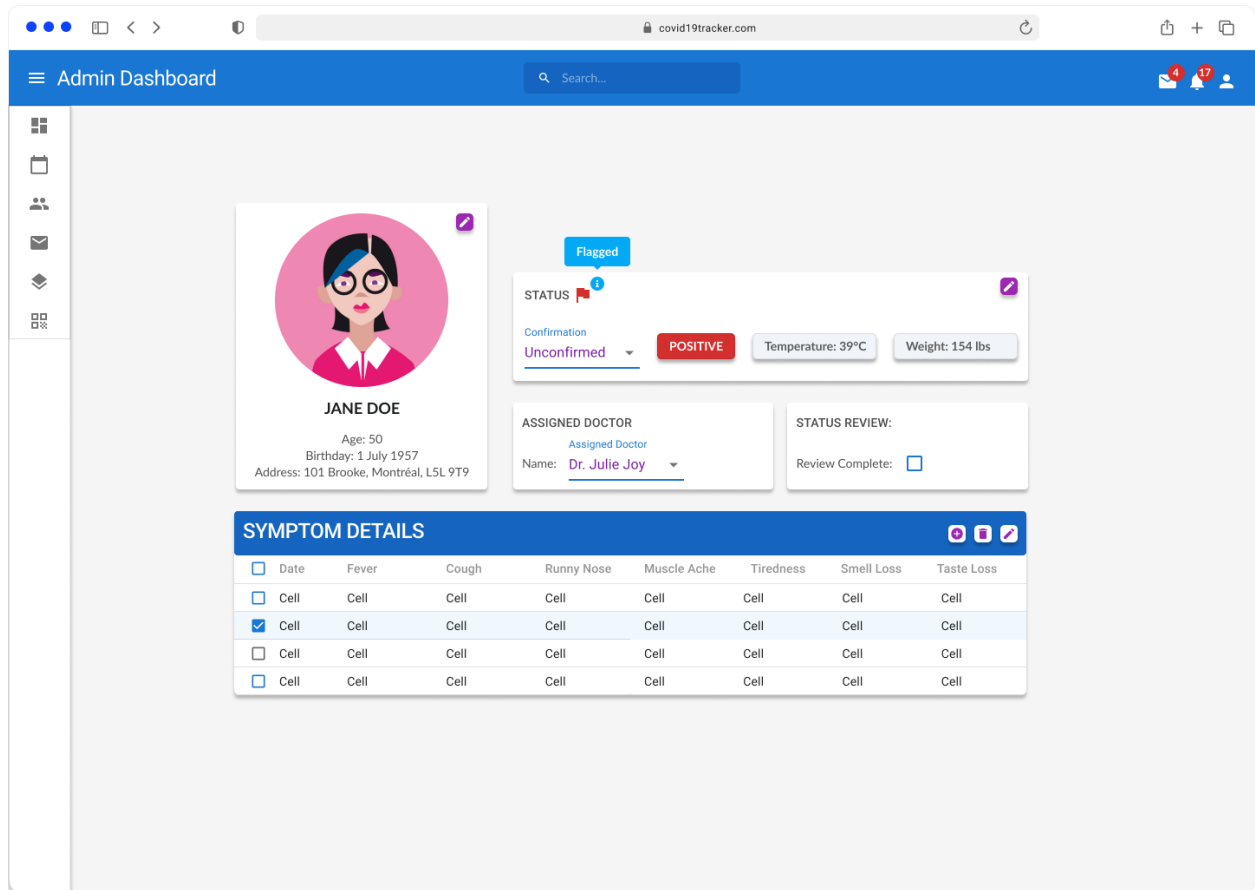


Figure 27: UI Patient Profile where you can edit symptoms, re-assign doctor, view if patient is flagged, review status, and edit patient info, for User Stories [Issue-3](#), [Issue-16](#), [Issue-18](#), [Issue-20](#).


7.2.2 Client Portal Sprint 2

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

4:19


covid19tracker.patient.com

Sign-in Sign-up




Sign in

Email Address *

 emailaddress@gmail.com

Password *

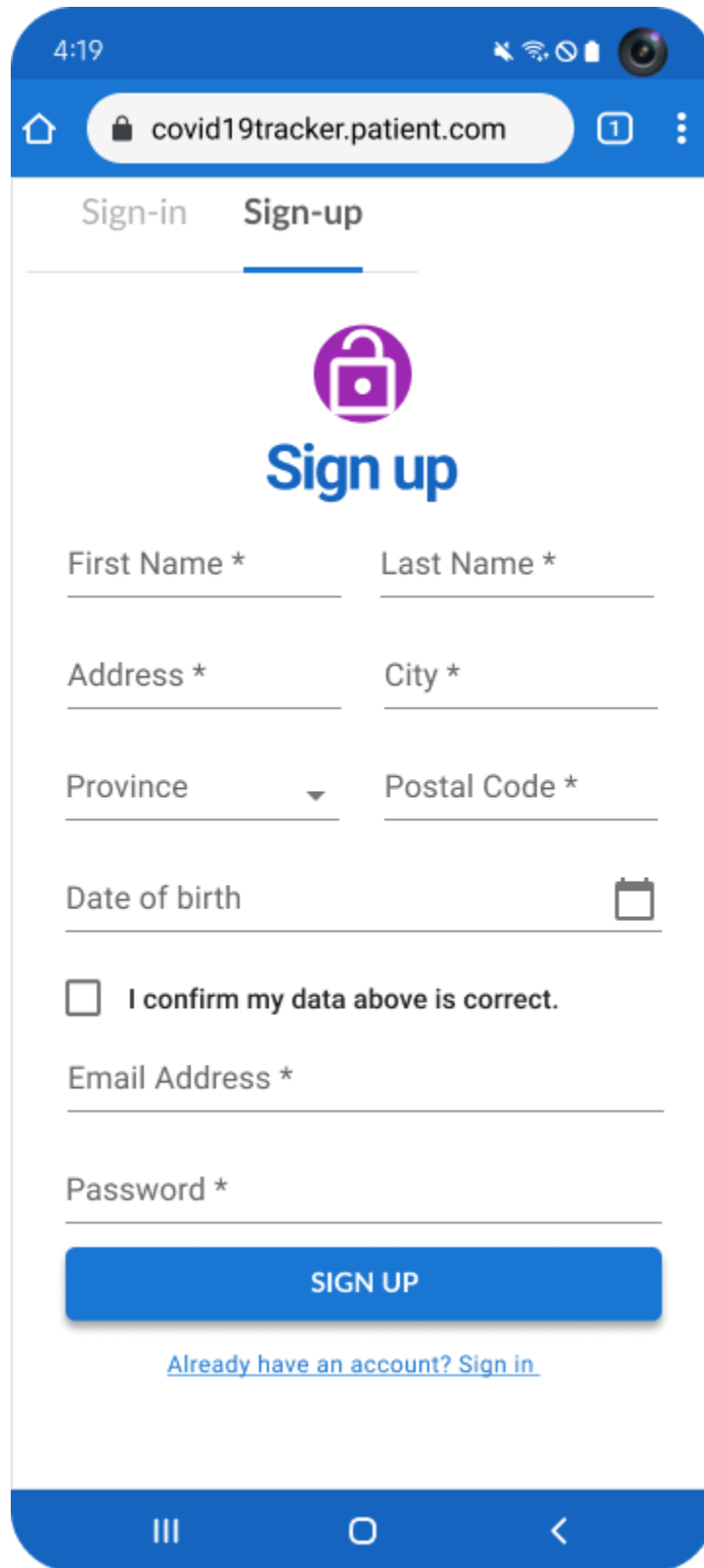
 password123

☐ Remember me

SIGN IN

[Forgot password?](#) [Don't have an account? Sign Up](#)

Figure 28: UI Client Sign-in for User Story [Issue-7](#)




The image shows a mobile application interface for a COVID-19 tracker. At the top, a blue header bar contains the time '4:19' and various status icons. Below this is a white browser-like address bar with a home icon, a lock icon, the URL 'covid19tracker.patient.com', a tab indicator with the number '1', and a menu icon. The main content area has a white background. At the top of this area are two tabs: 'Sign-in' and 'Sign-up', with 'Sign-up' being the active tab. Below the tabs is a purple padlock icon with a white 'S' inside, followed by the text 'Sign up' in a large, bold, blue font. The form consists of several input fields: 'First Name *' and 'Last Name *' (two columns), 'Address *' and 'City *' (two columns), 'Province' with a dropdown arrow and 'Postal Code *' (two columns), and 'Date of birth' with a calendar icon. Below these is a checkbox labeled 'I confirm my data above is correct.' followed by 'Email Address *' and 'Password *' (two columns). A large blue button with the text 'SIGN UP' in white is positioned below the password field. At the bottom of the form is a link that reads 'Already have an account? Sign in'. The entire interface is framed by a blue border, and at the very bottom is a blue navigation bar with three white icons: a hamburger menu, a circle, and a back arrow.

4:19

covid19tracker.patient.com

Sign-in Sign-up




Sign up

First Name * Last Name *

Address * City *

Province Postal Code *

Date of birth 

☐ I confirm my data above is correct.

Email Address *

Password *

SIGN UP

[Already have an account? Sign in](#)

Figure 29: UI Client Sign-up for User Story [Issue-7](#)

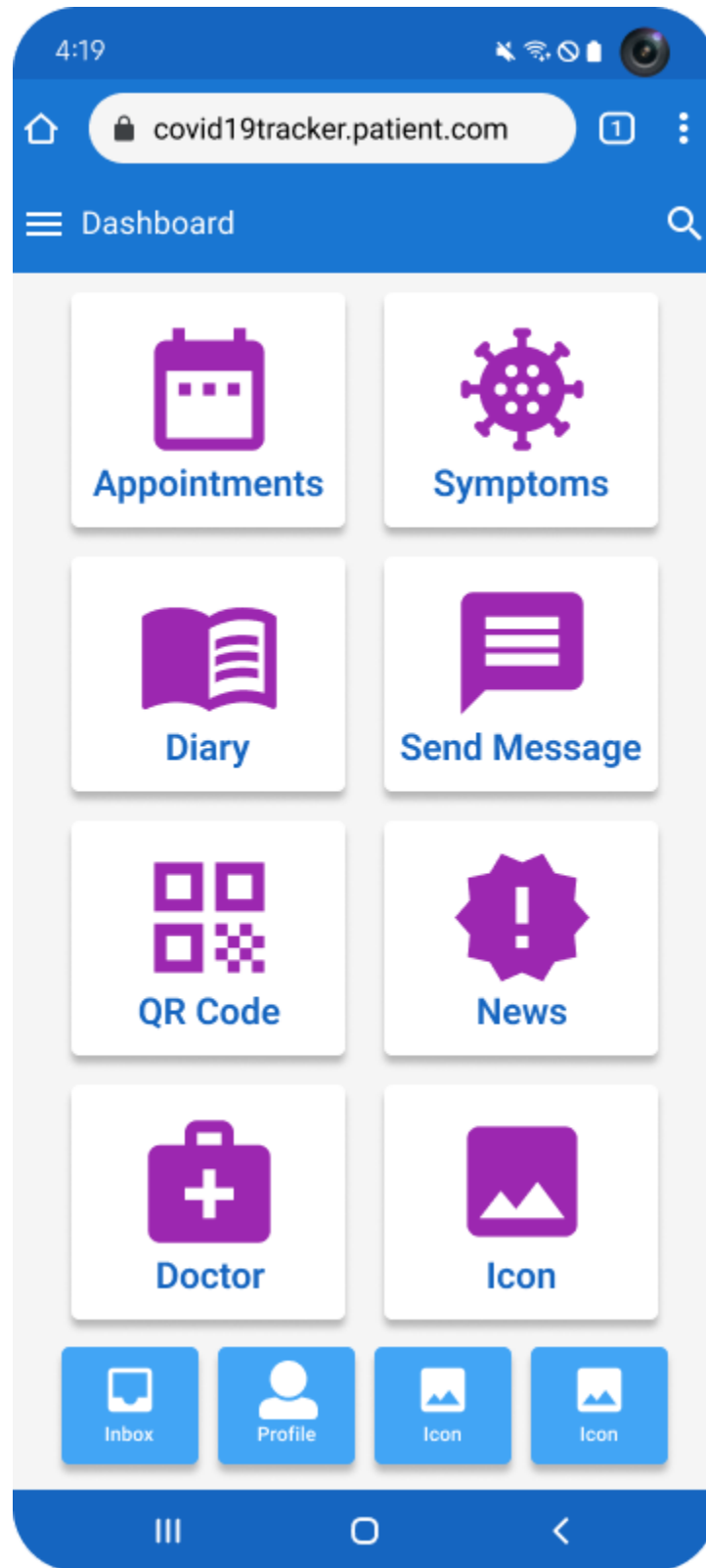


Figure 30: UI Client Dashboard

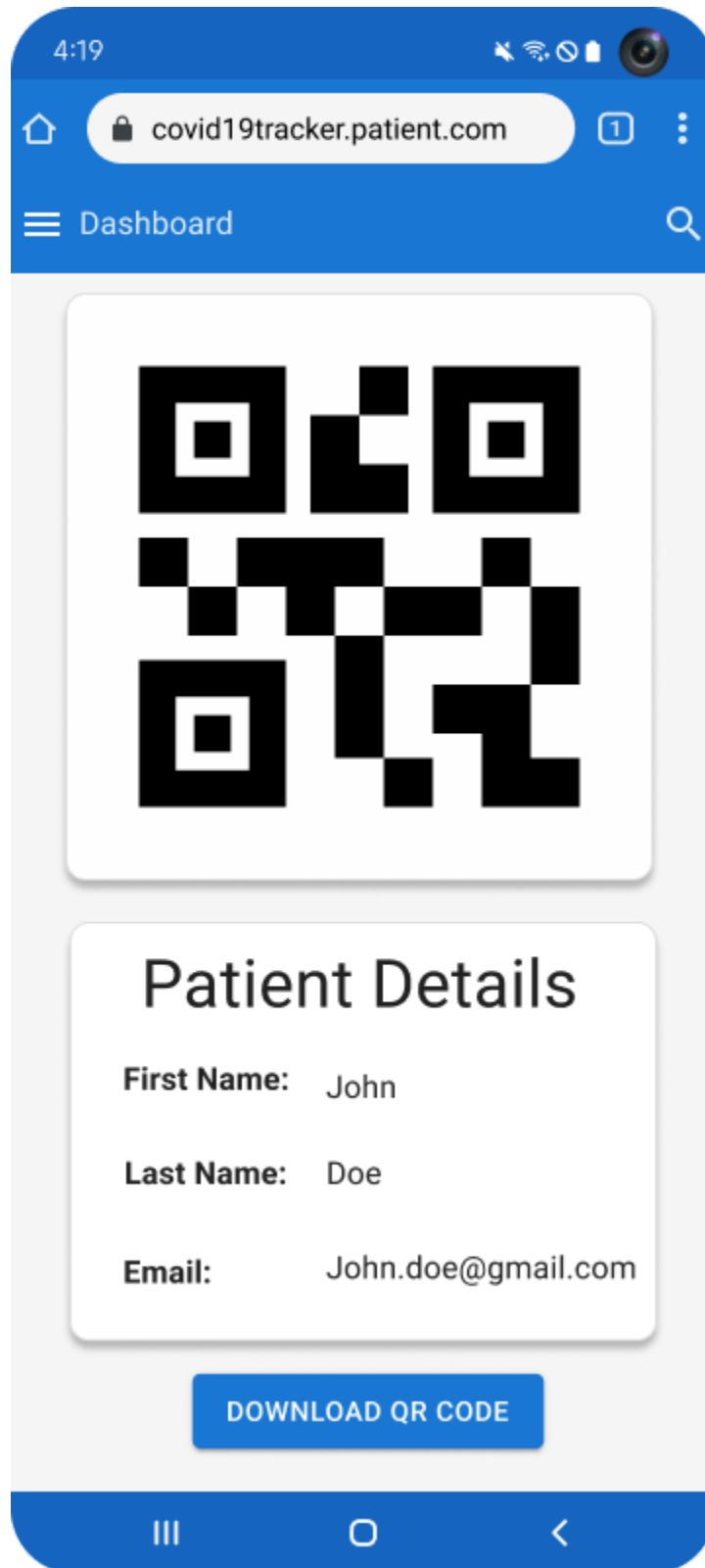


Figure 31: UI Client QR code for User Story [Issue-22](#)

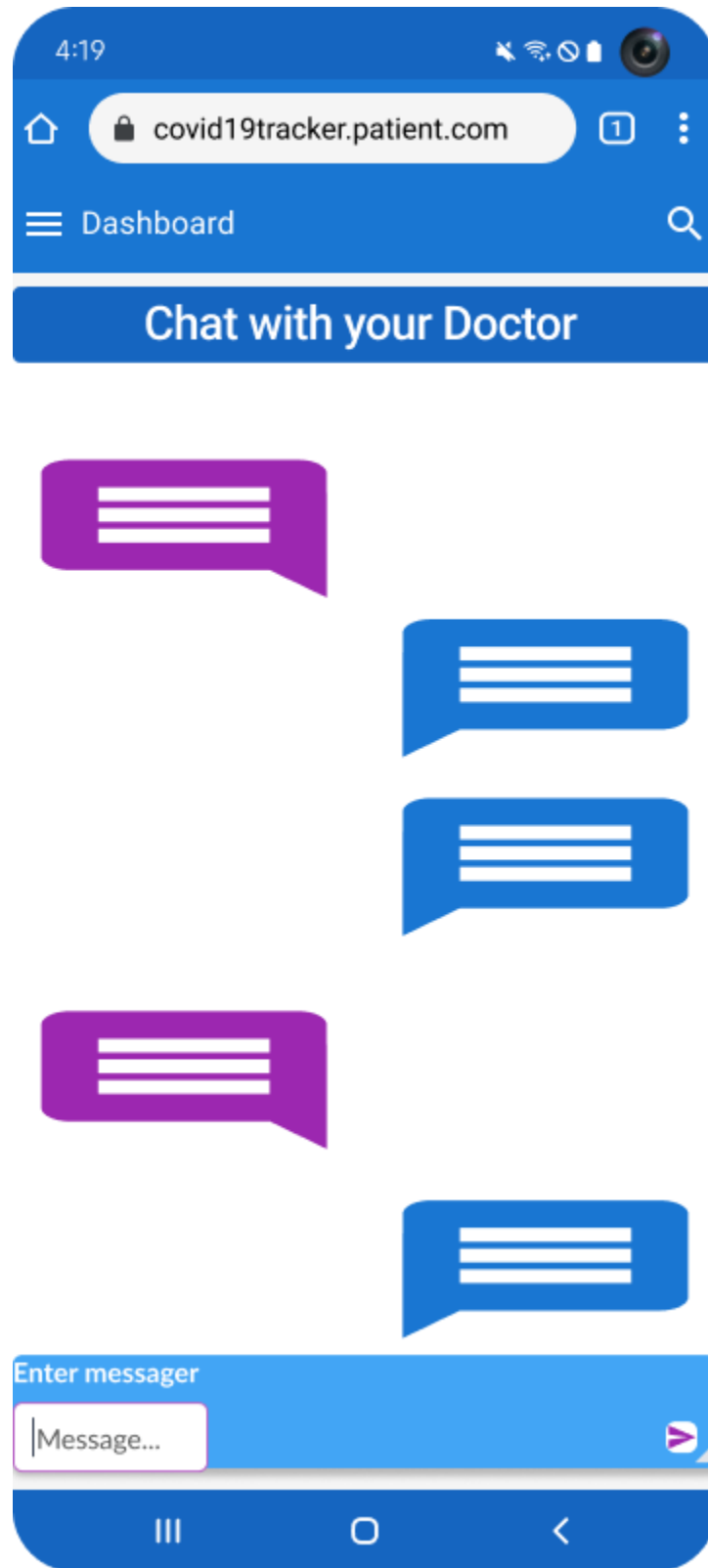


Figure 32: UI Client Chat with patient for User Story [Issue-13](#)

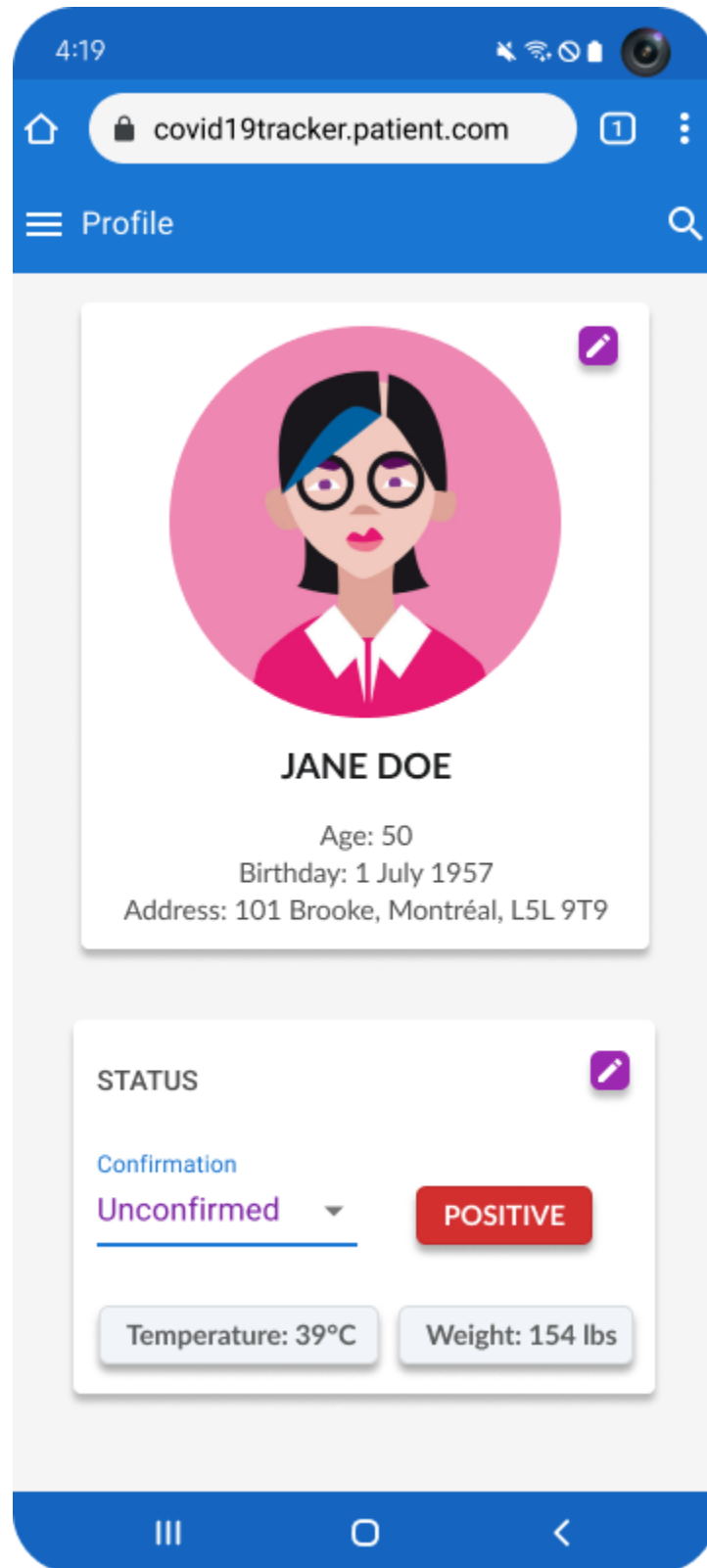


Figure 33: UI Client Patient for User Story [Issue-37](#)

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

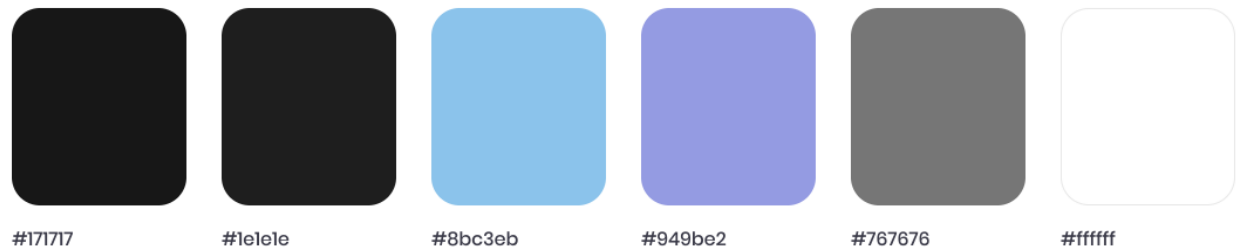


Figure 34: Theme Palette

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

7.3.1.1 Dark Theme Update Admin Portal

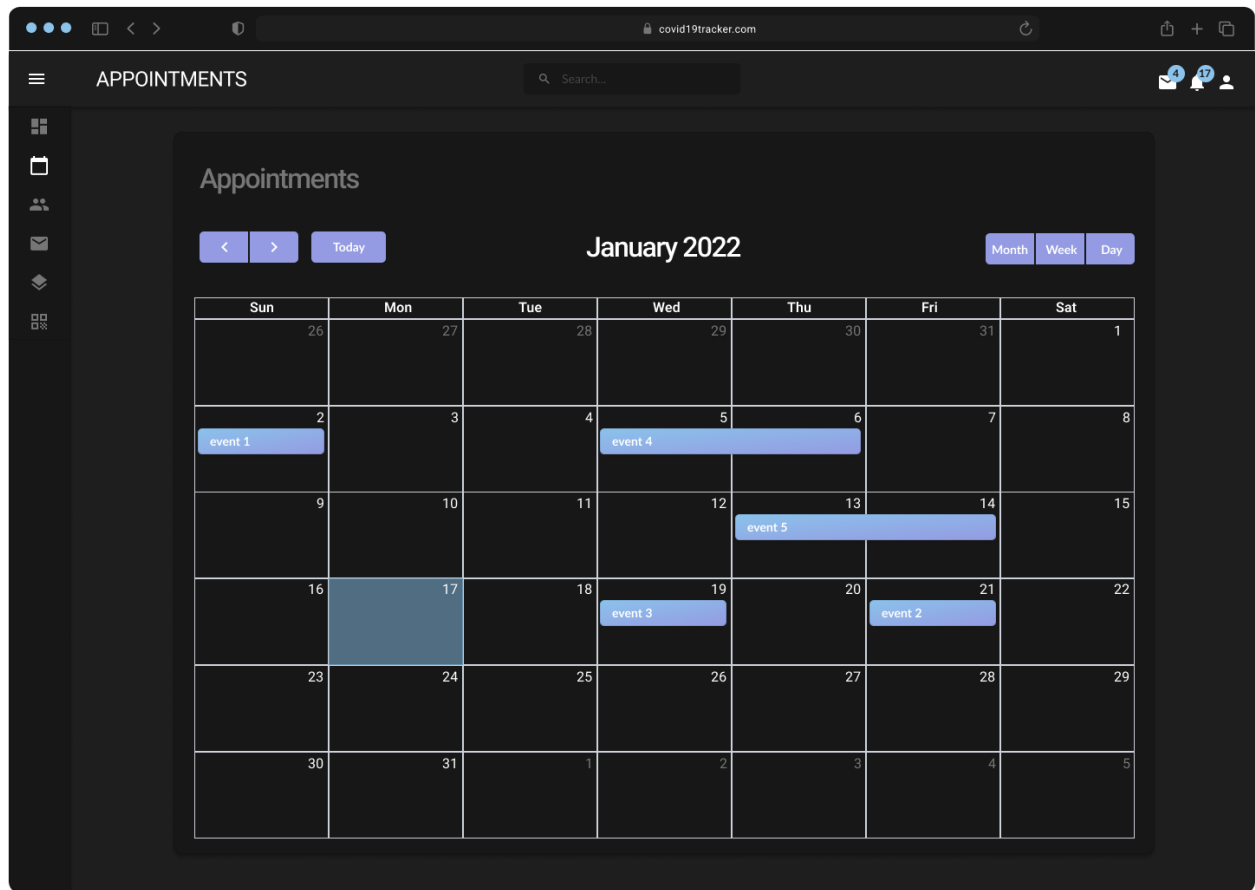


Figure 22: UI Admin Appointments for User Story [Issue-5](#)

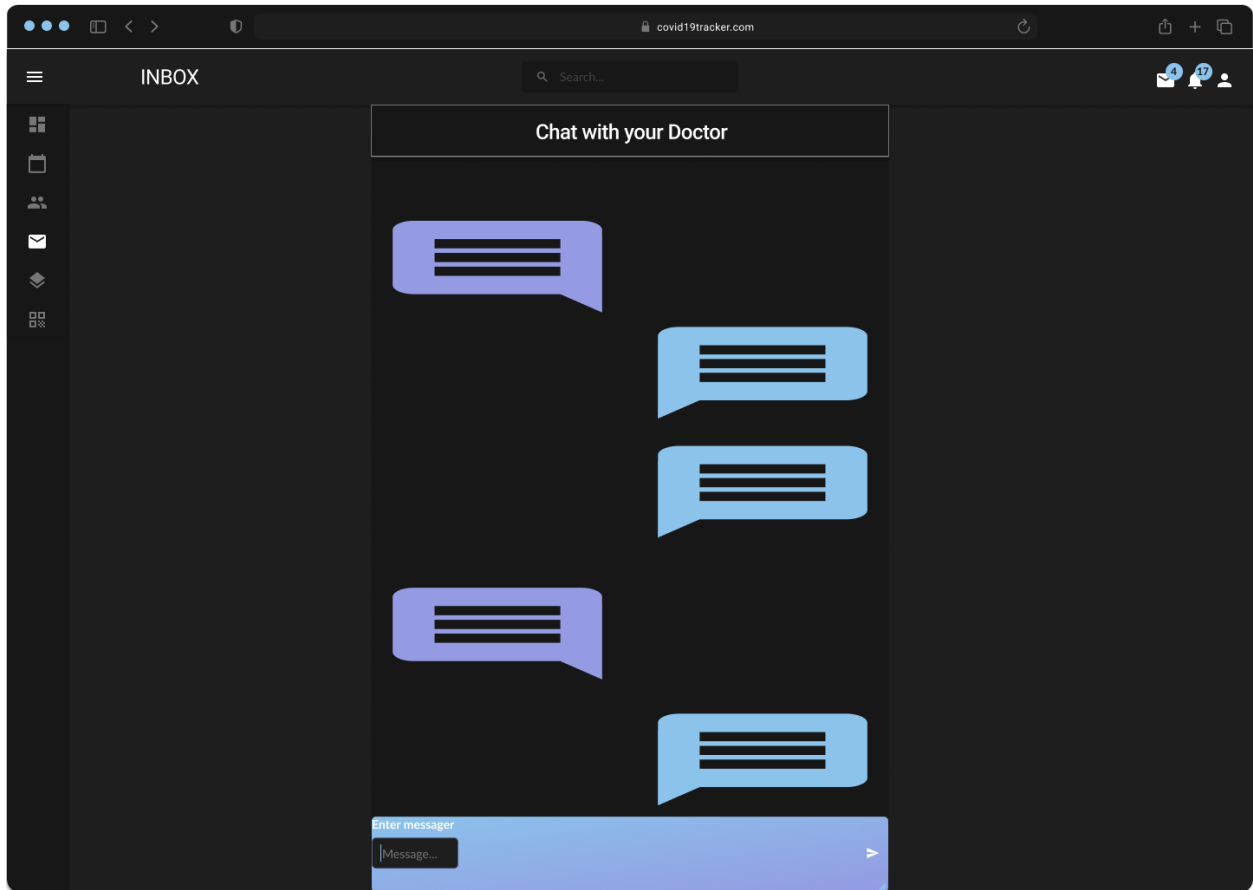


Figure 26: UI Admin Chat with patient for User Story [Issue-13](#)

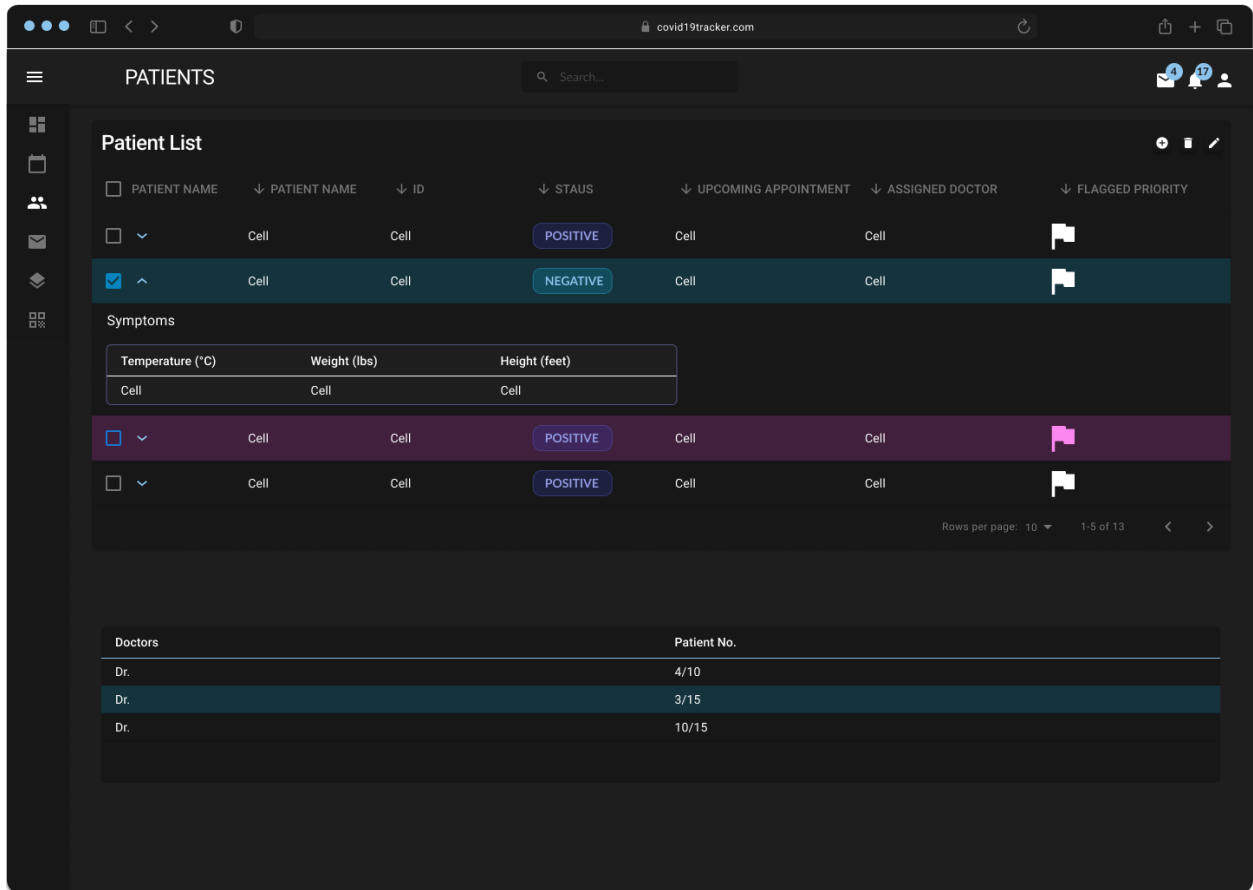


Figure 23: UI Admin Patient List for User Story [Issue-89](#)

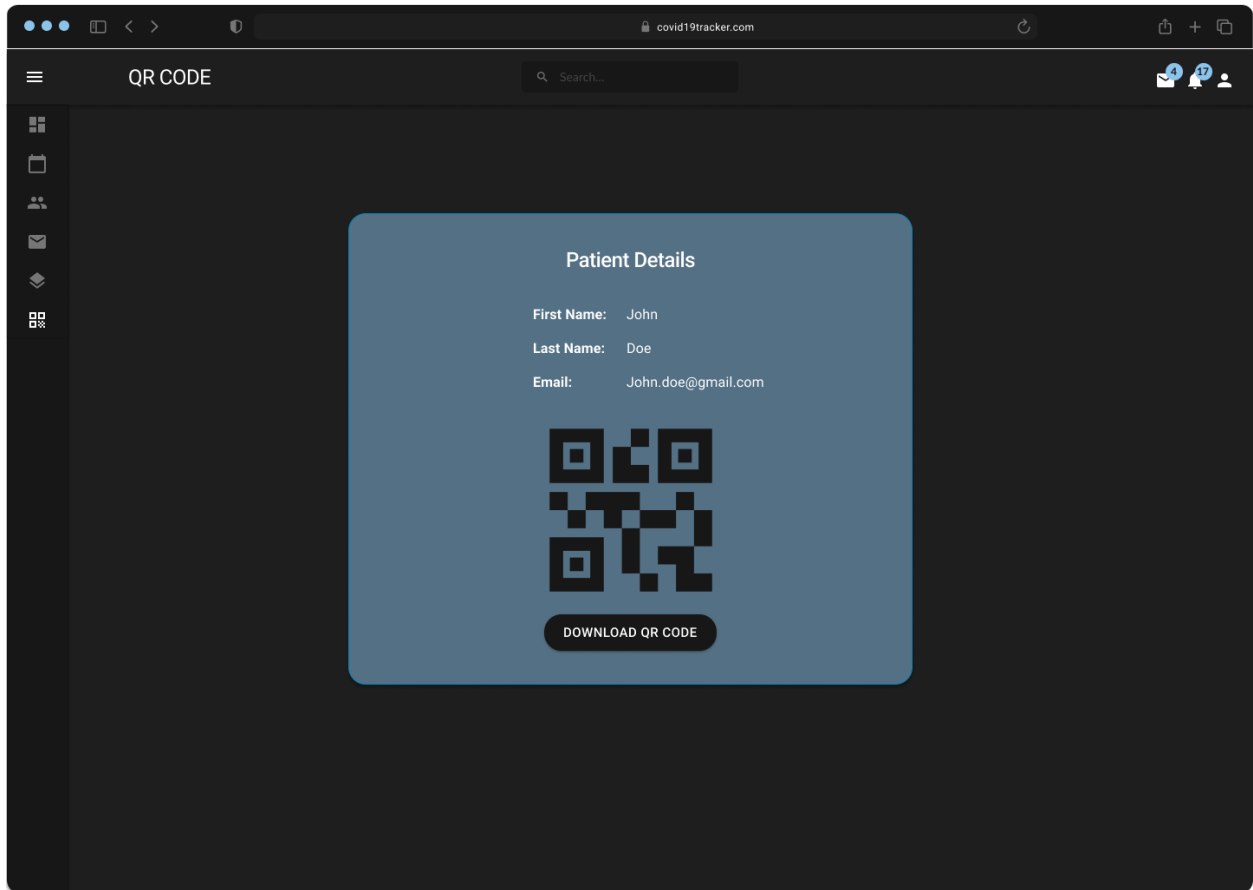
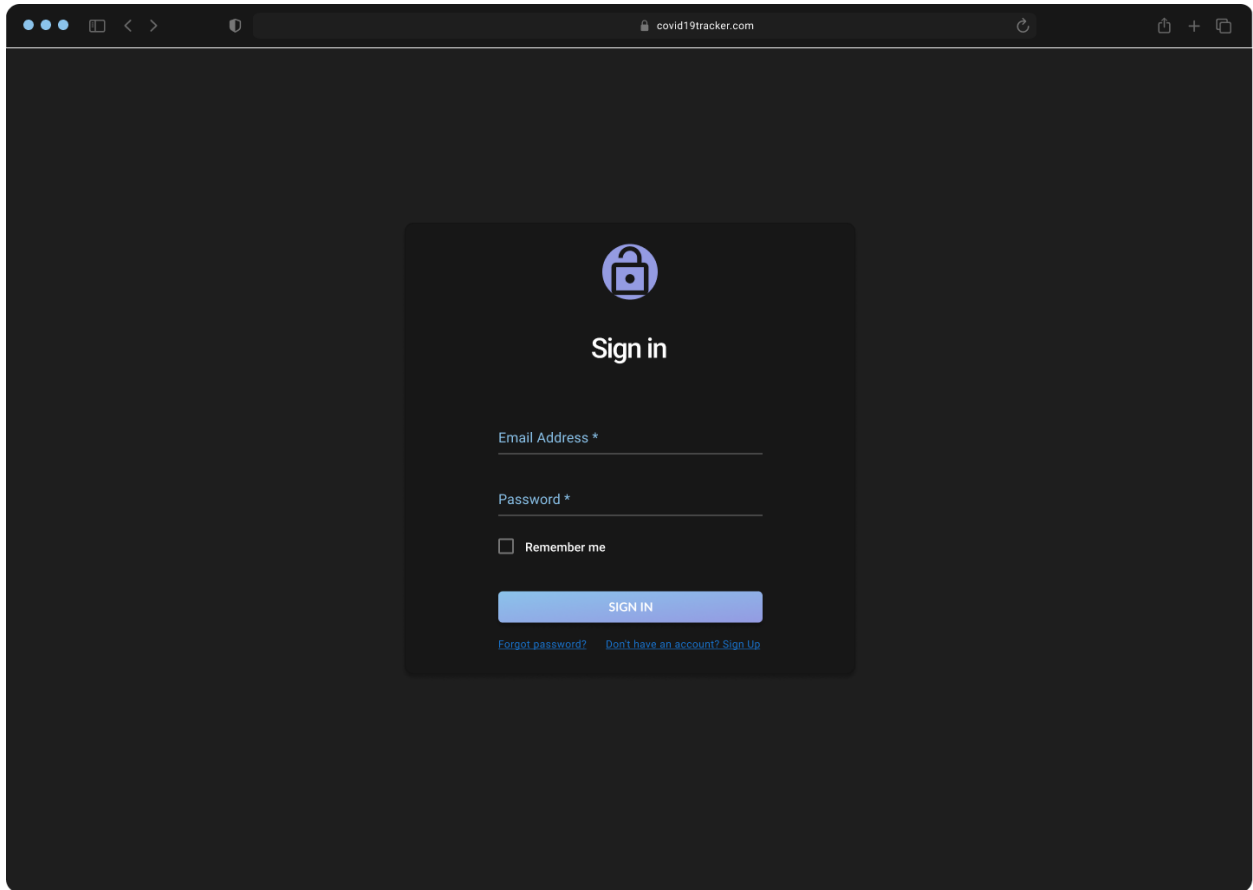



Figure 31: UI Client QR code for User Story [Issue-22](#)



covid19tracker.com



Sign up

First Name *

Last Name *

Address *

City *

Province

Postal Code *

Date of birth

☐

I confirm my data above is correct.

Email Address *

Password *

SIGN UP

[Already have an account? Sign in](#)

61

7.3.1.2 Dark Theme Update Client Portal

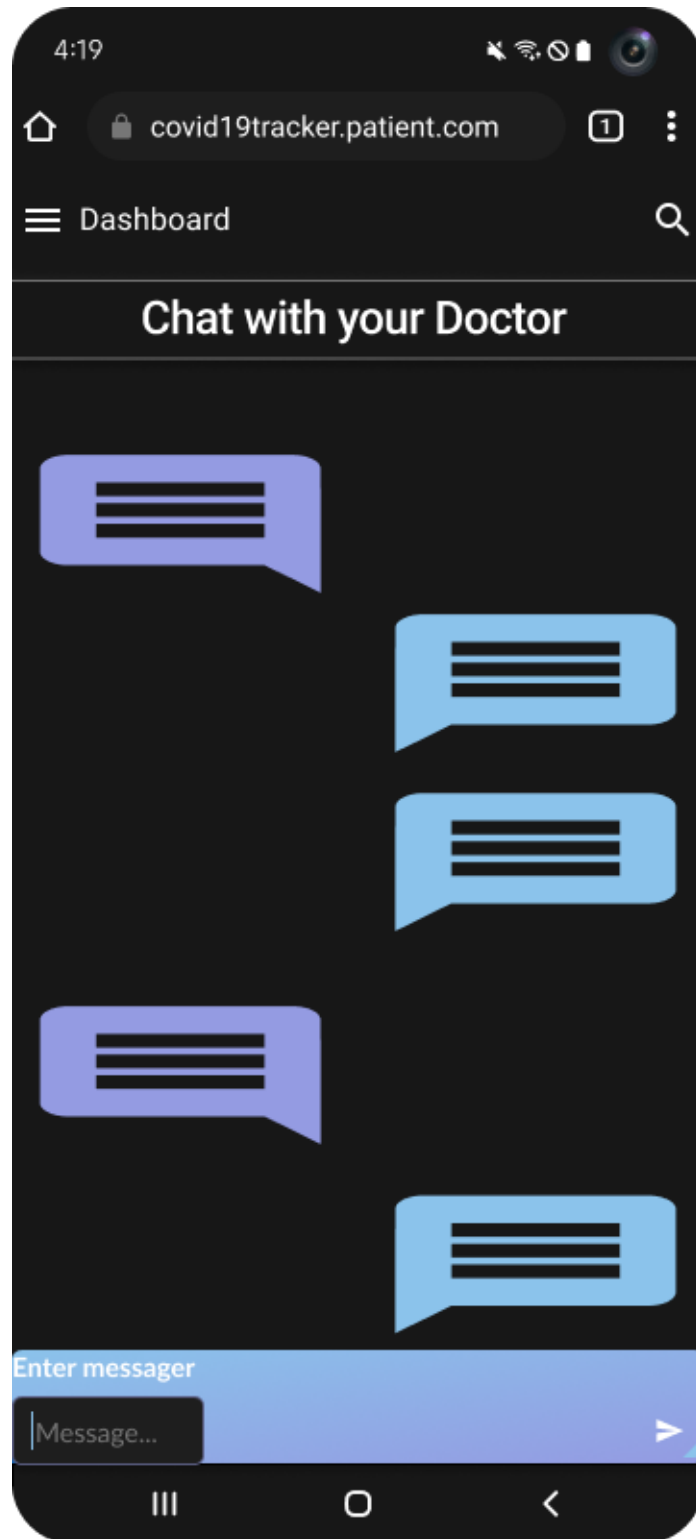


Figure 35: UI Patient Chat

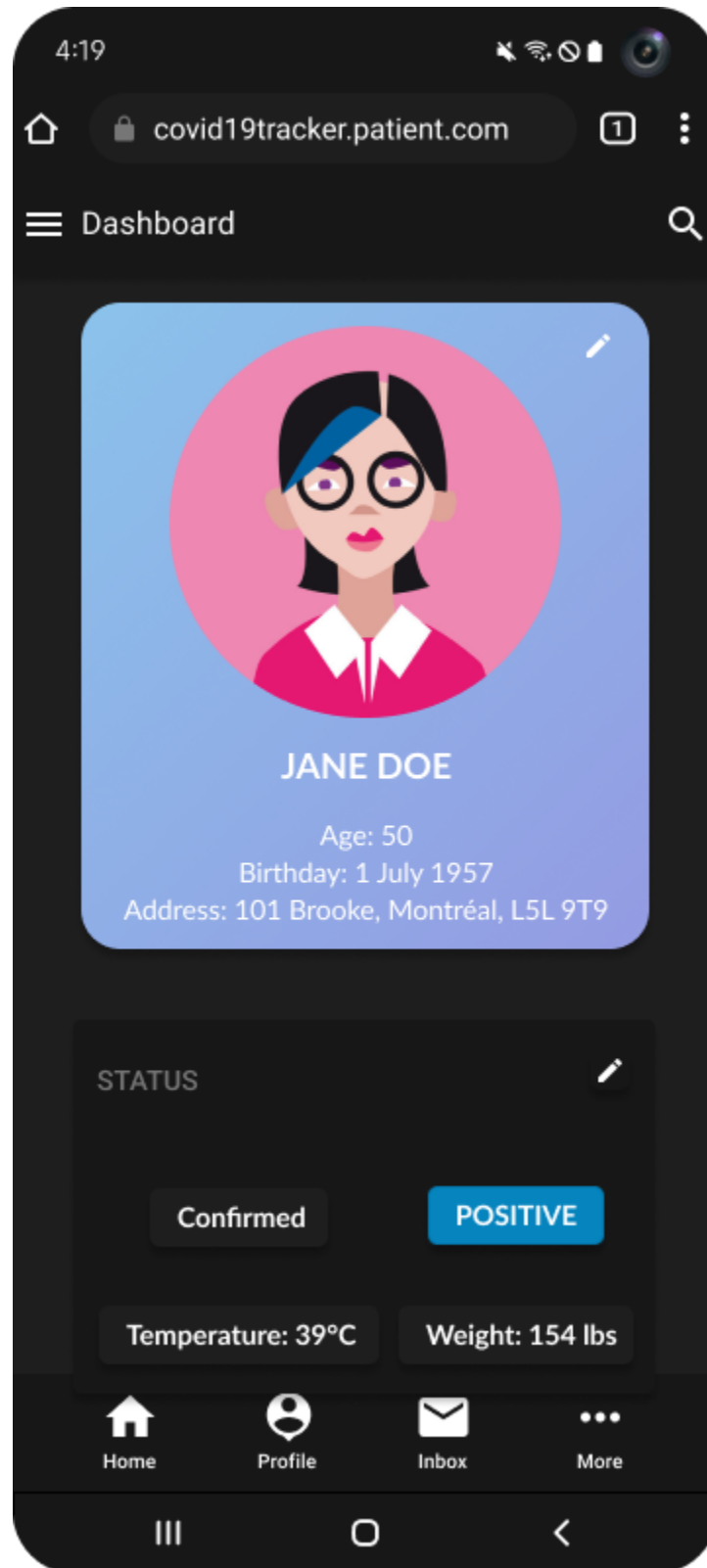


Figure 36: UI Patient Profile

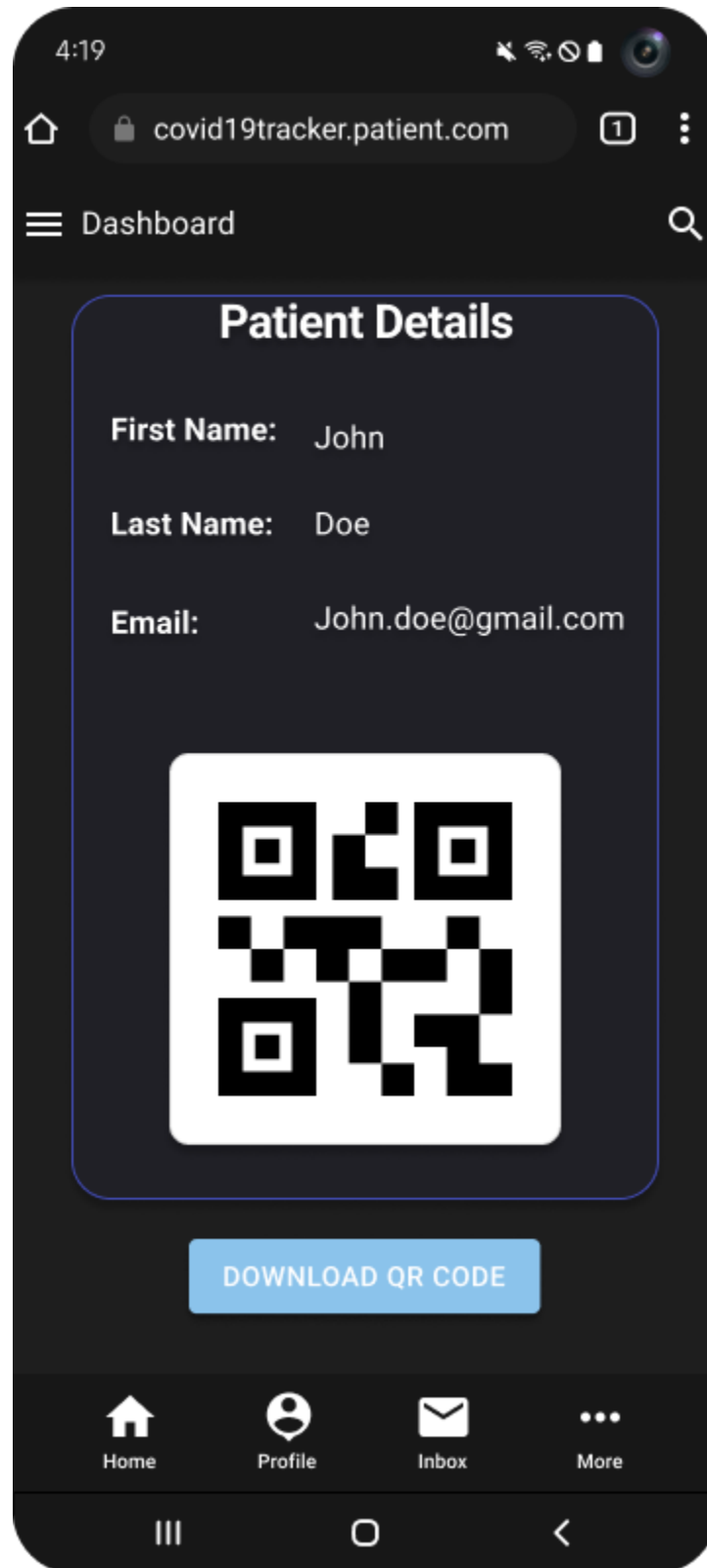


Figure 37: UI Patient QR Code

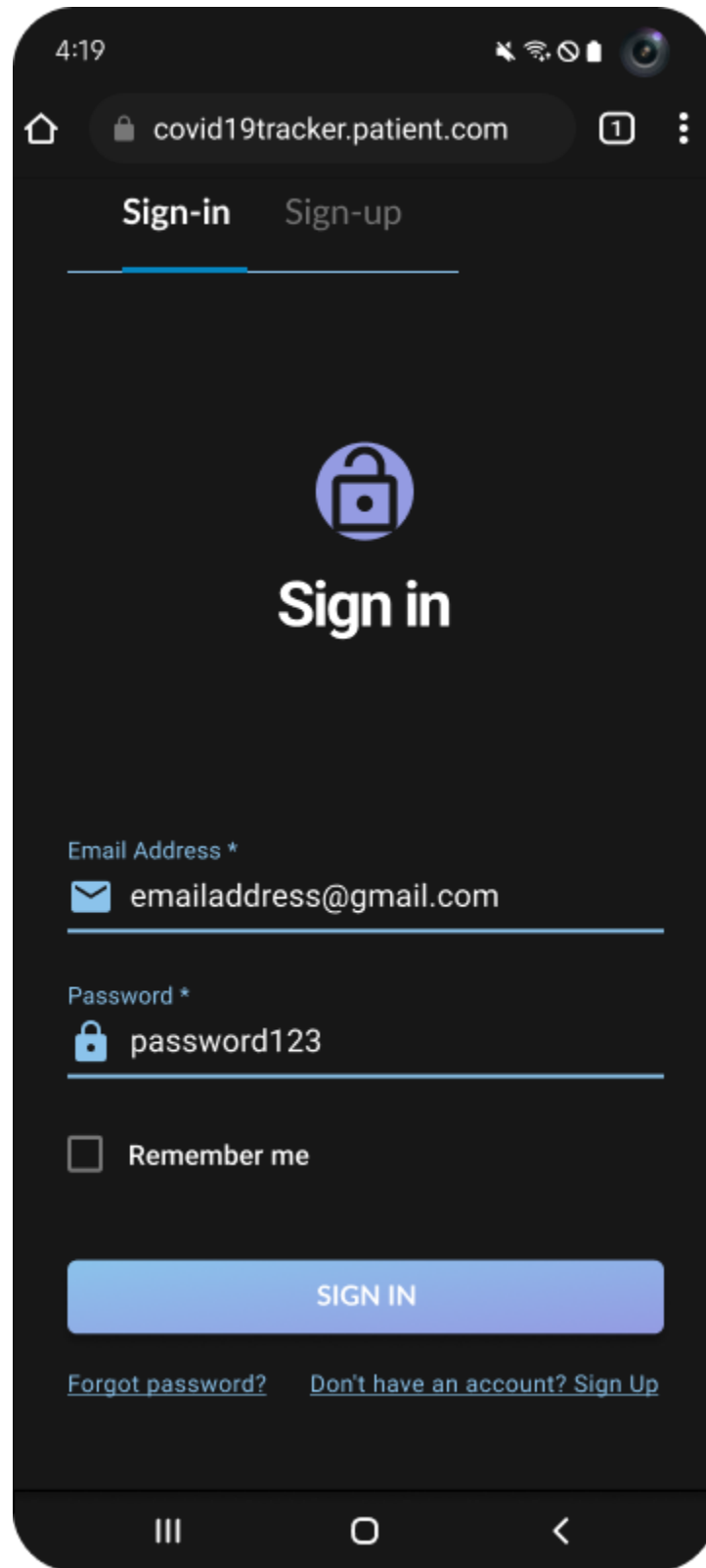



Figure 38: UI Client App Sign In

4:19

covid19tracker.patient.com

Sign-in Sign-up




Sign up

First Name * Last Name *

Address * City *

Province Postal Code *

Date of birth 

☐ I confirm my data above is correct.

Email Address *

Password *

SIGN UP

[Already have an account? Sign in](#)

Figure 39: UI Client App Sign In

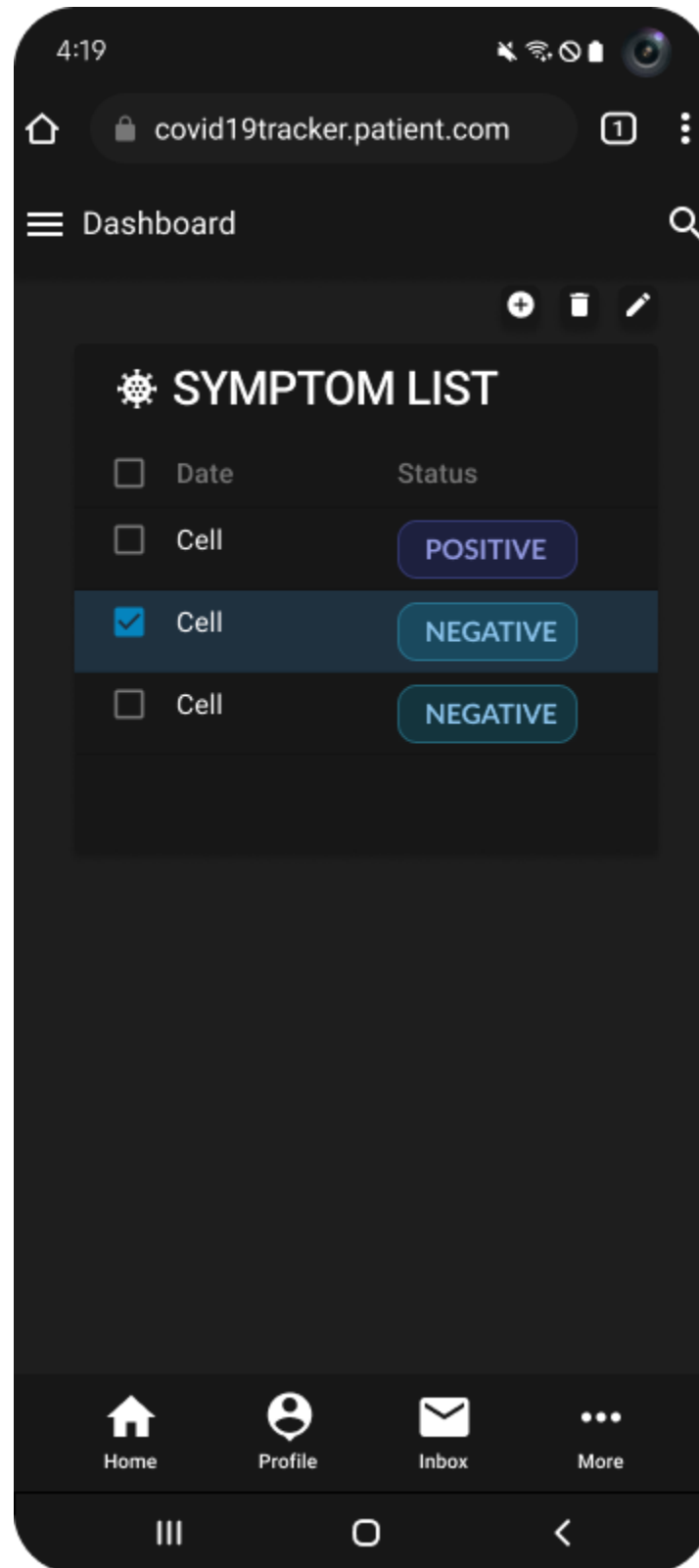


Figure 40: UI Client Symptoms List

7.3.2 Admin Portal Sprint 3

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

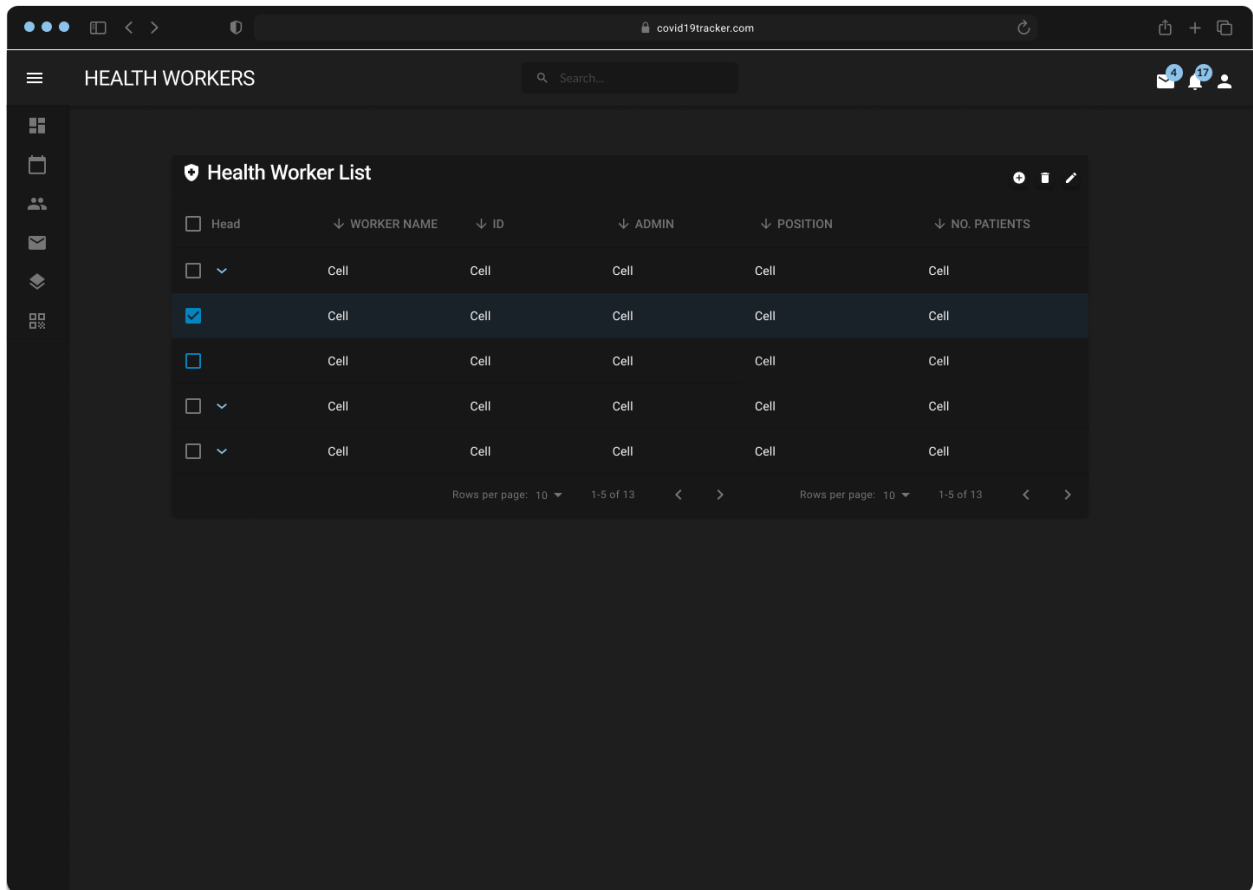


Figure 41: UI Admin Health Worker List for User Story [Issue-19](#)

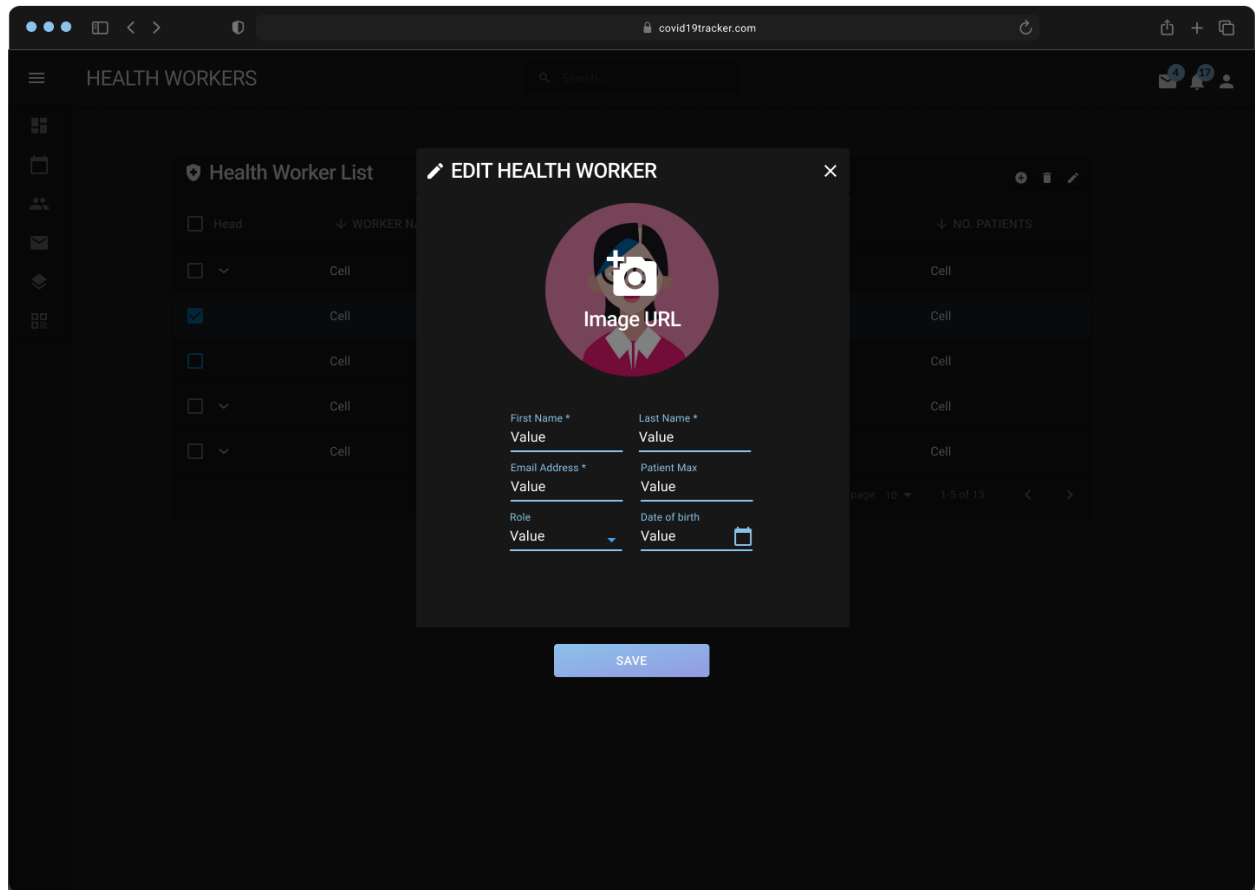


Figure 41: UI Admin Health Worker Edit for User Story [Issue-19](#)

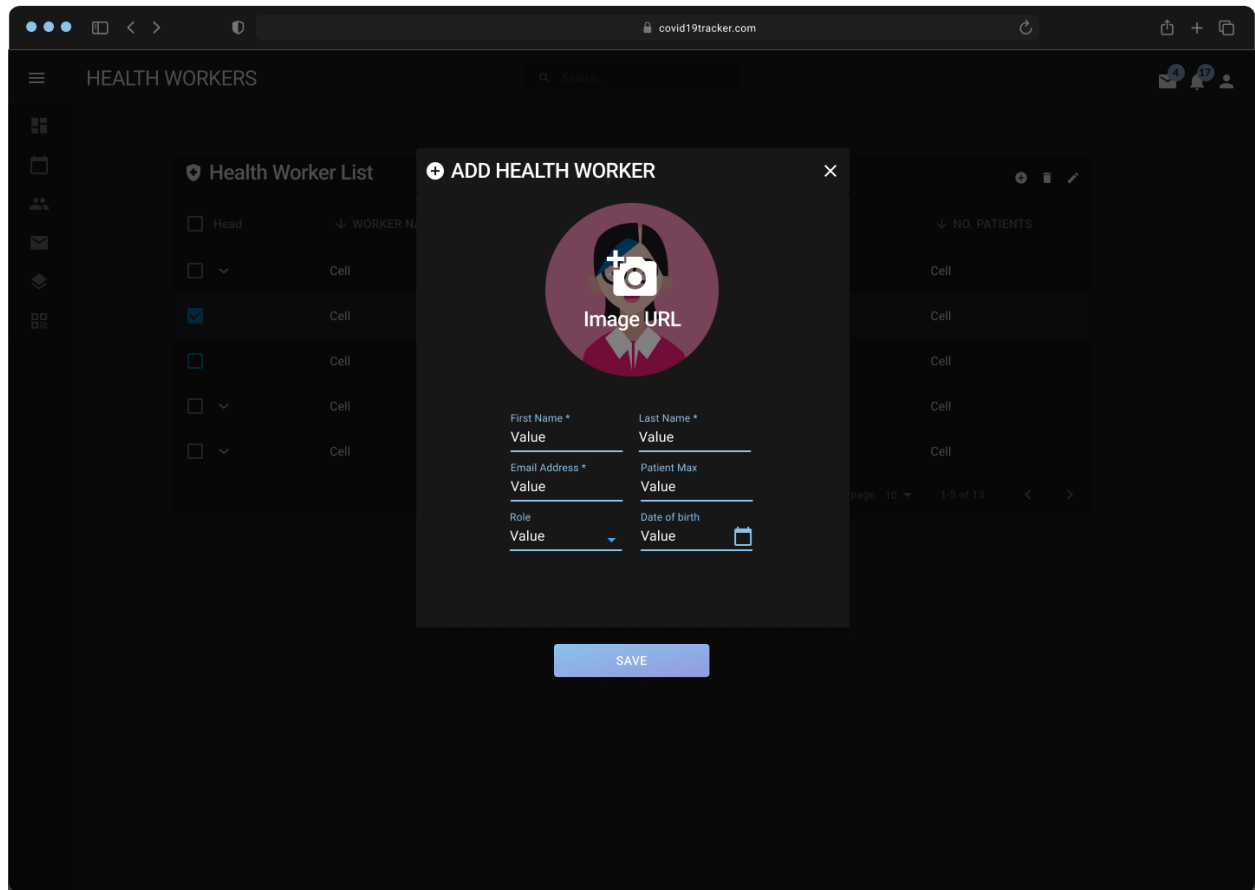


Figure 41: UI Admin Health Worker Add for User Story [Issue-19](#)

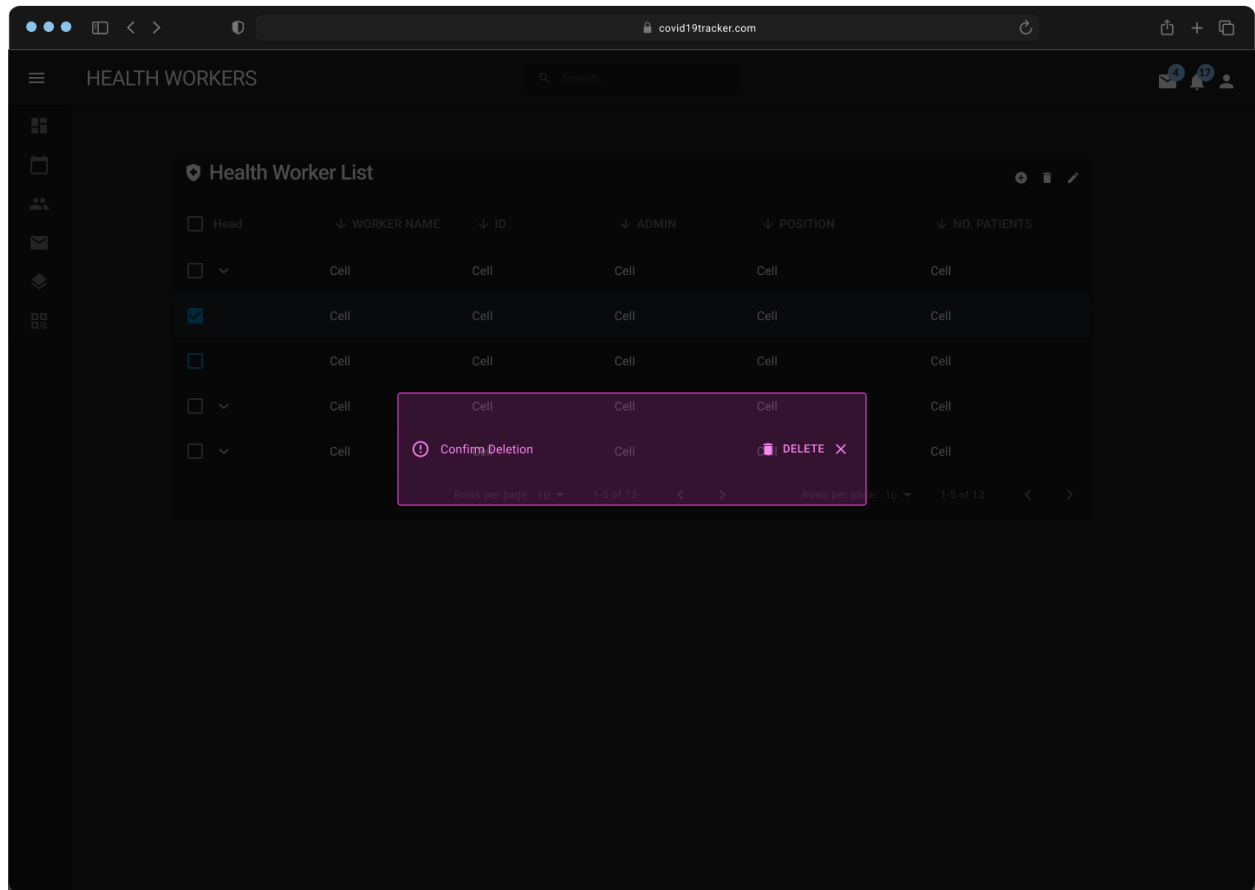


Figure 41: UI Admin Health Worker Delete for User Story [Issue-19](#)

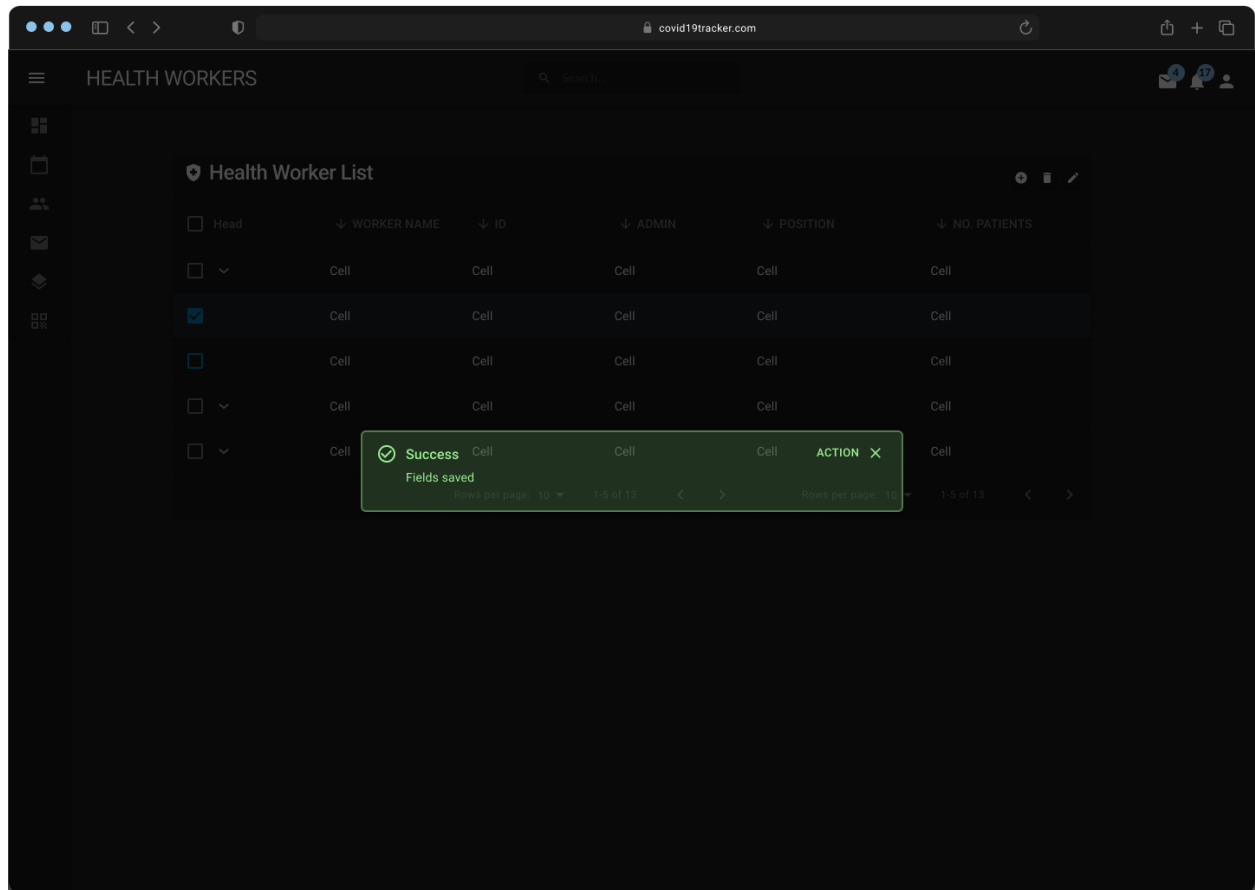


Figure 41: UI Admin Health Worker Success for User Story [Issue-19](#)



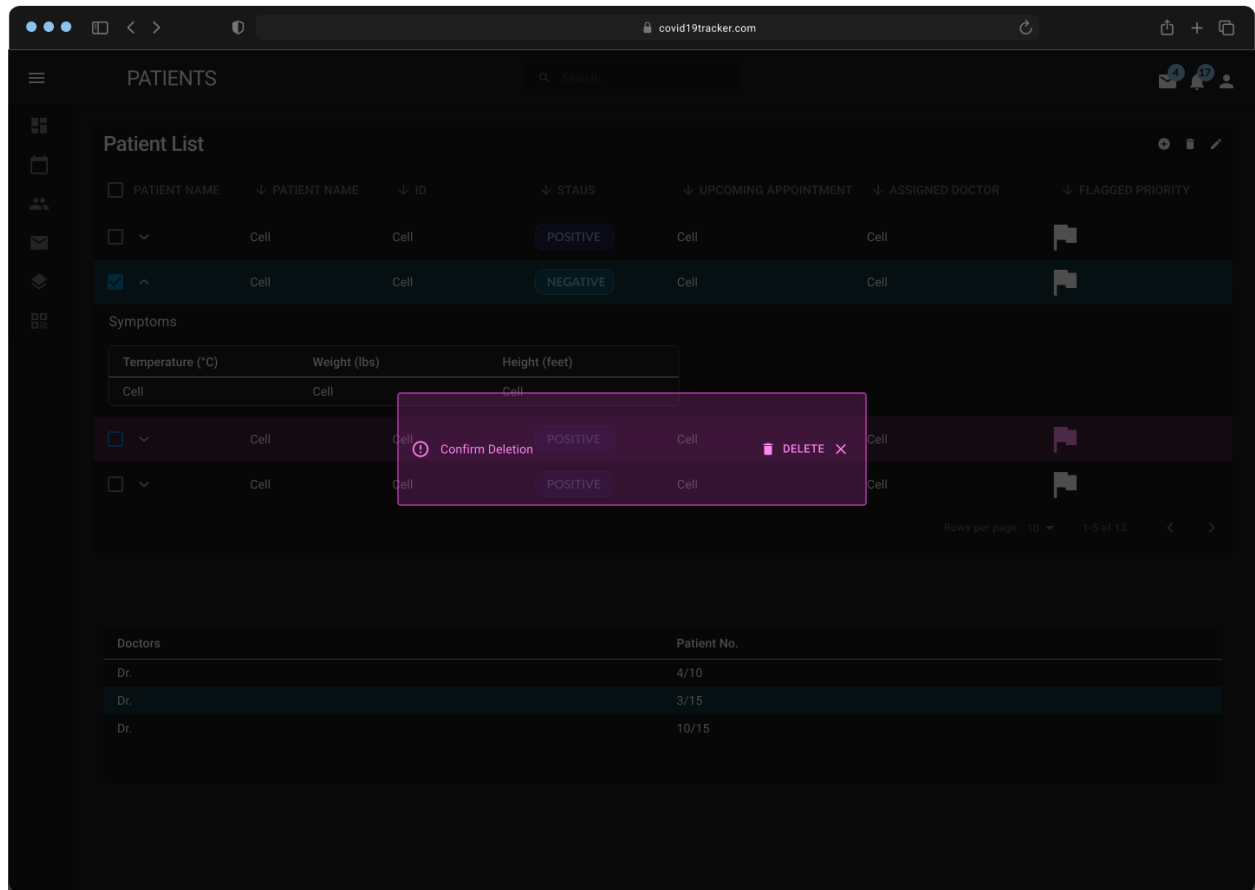


Figure 41: UI Patient Delete Modal

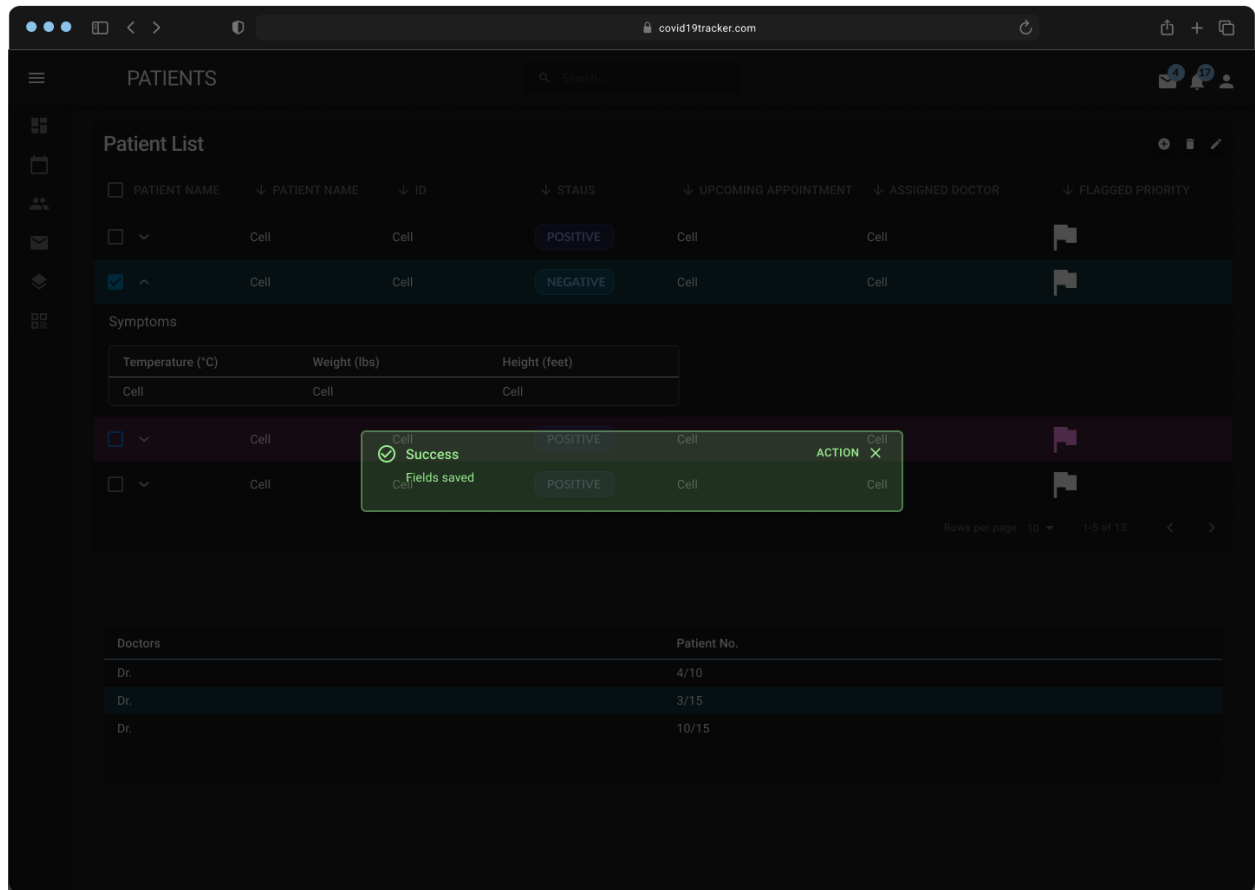


Figure 41: UI Patient Success Modal

7.3.3 Client Portal Sprint 3

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

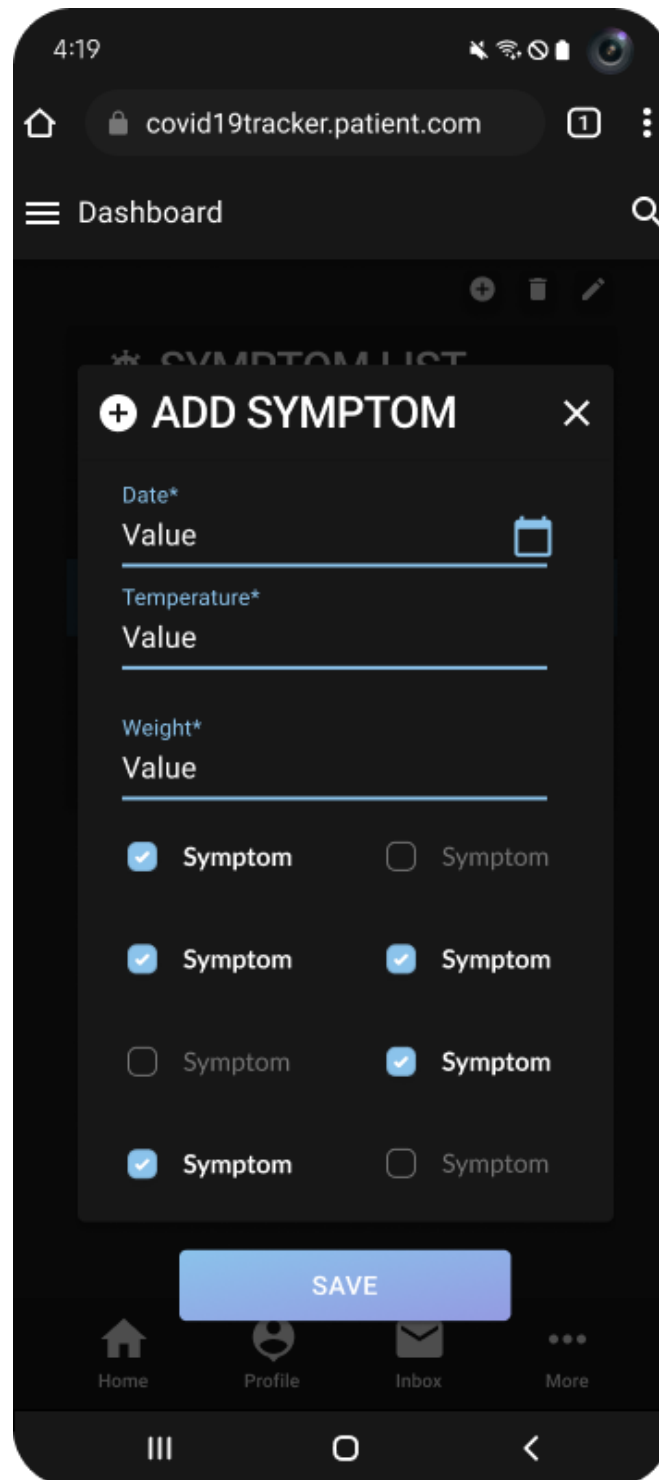


Figure 42: UI Symptoms Entry Add for User Story [Issue-41](#)

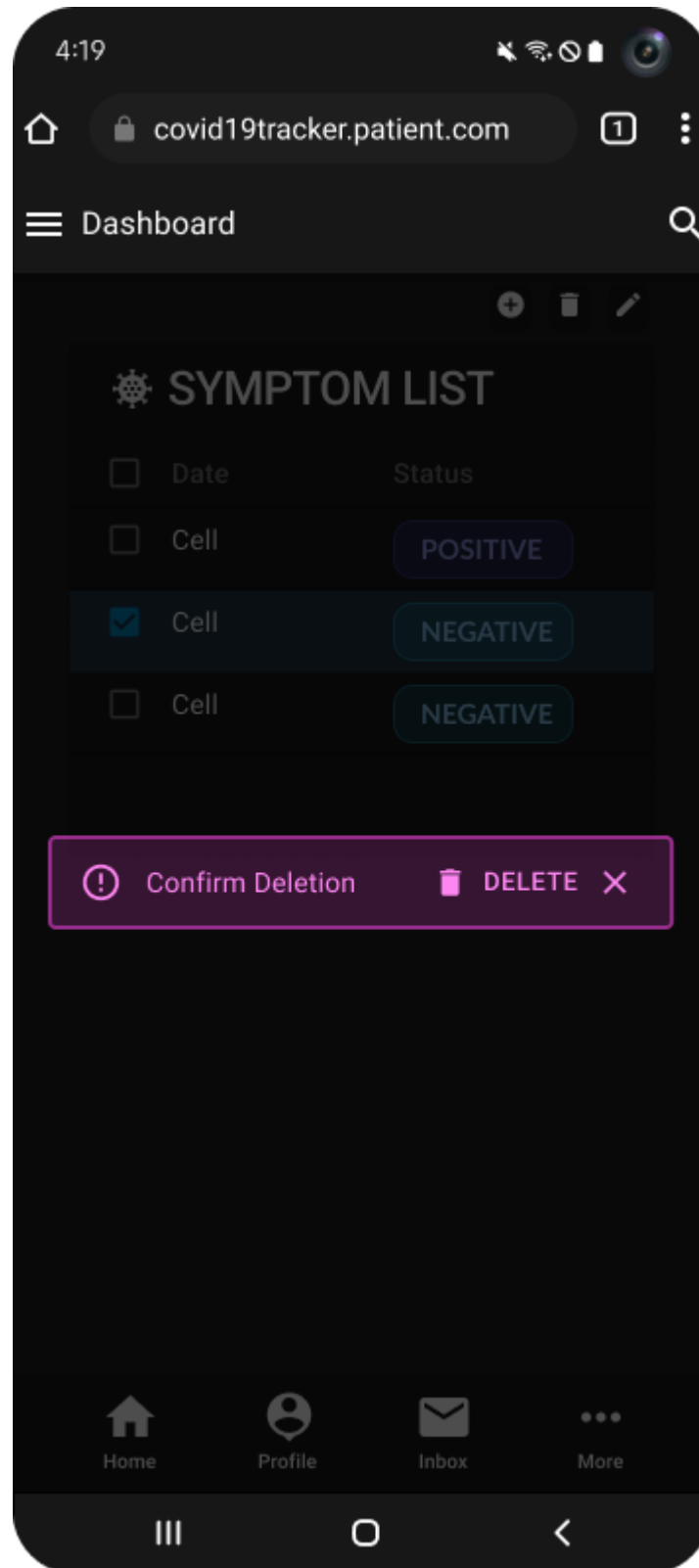


Figure 43: UI Symptoms Entry Delete for User Story [Issue-41](#)

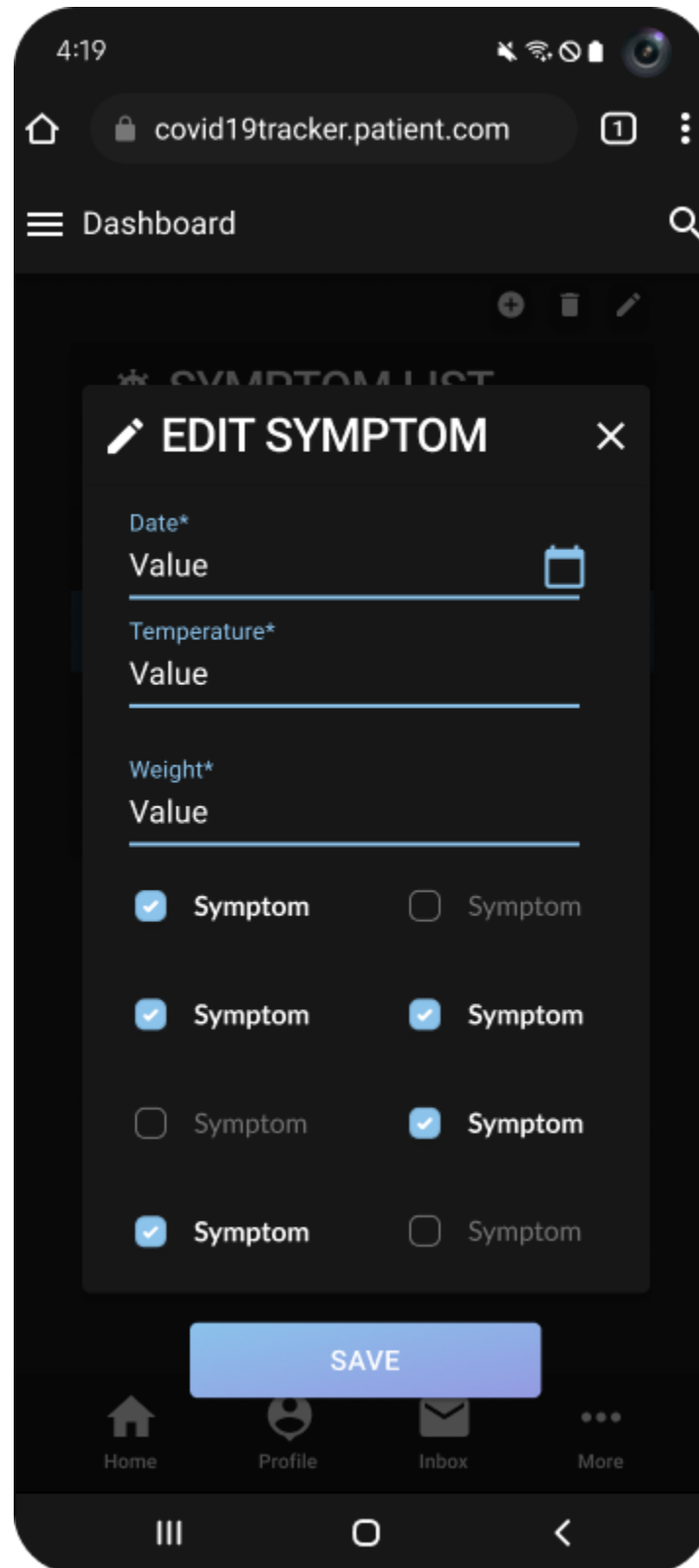


Figure 44: UI Symptoms Entry Edit for User Story [Issue-41](#)

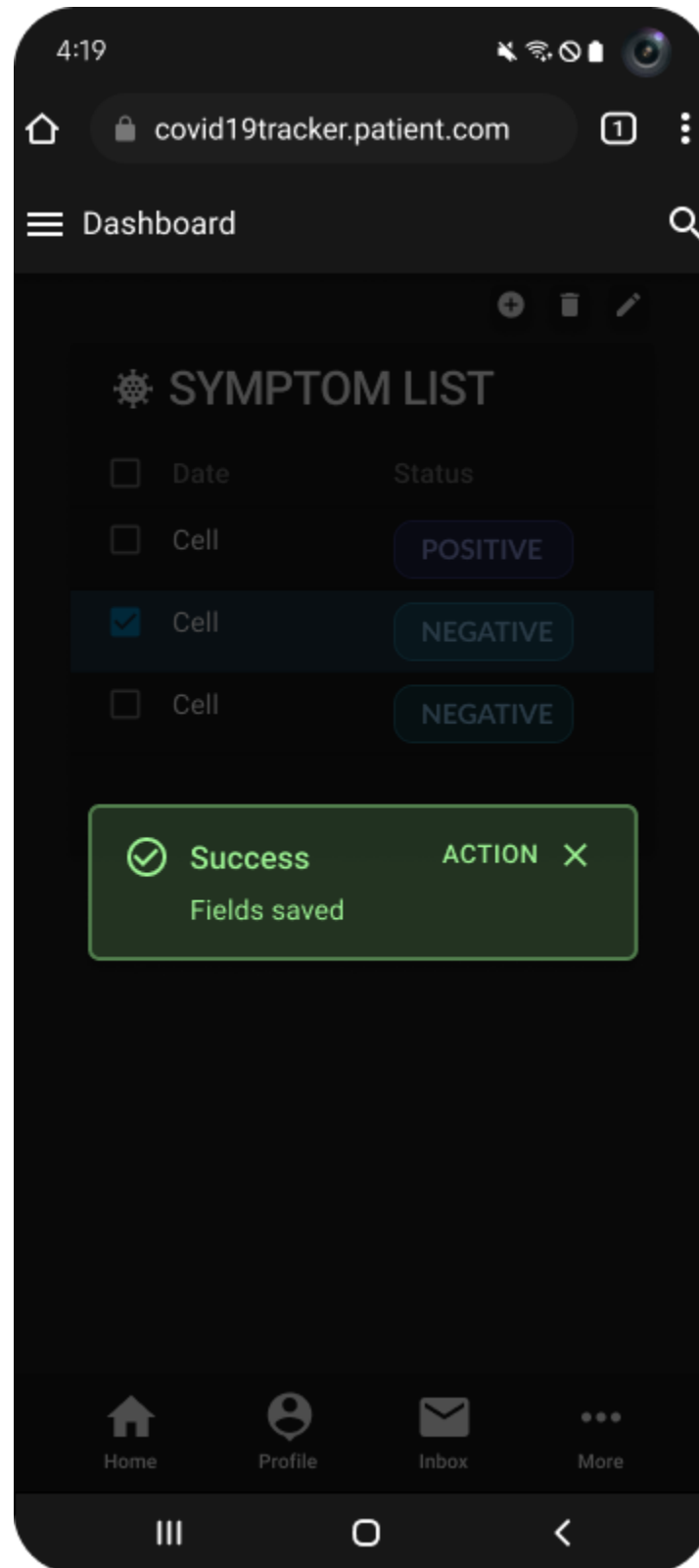


Figure 45: UI Symptoms Entry Success for User Story [Issue-41](#)

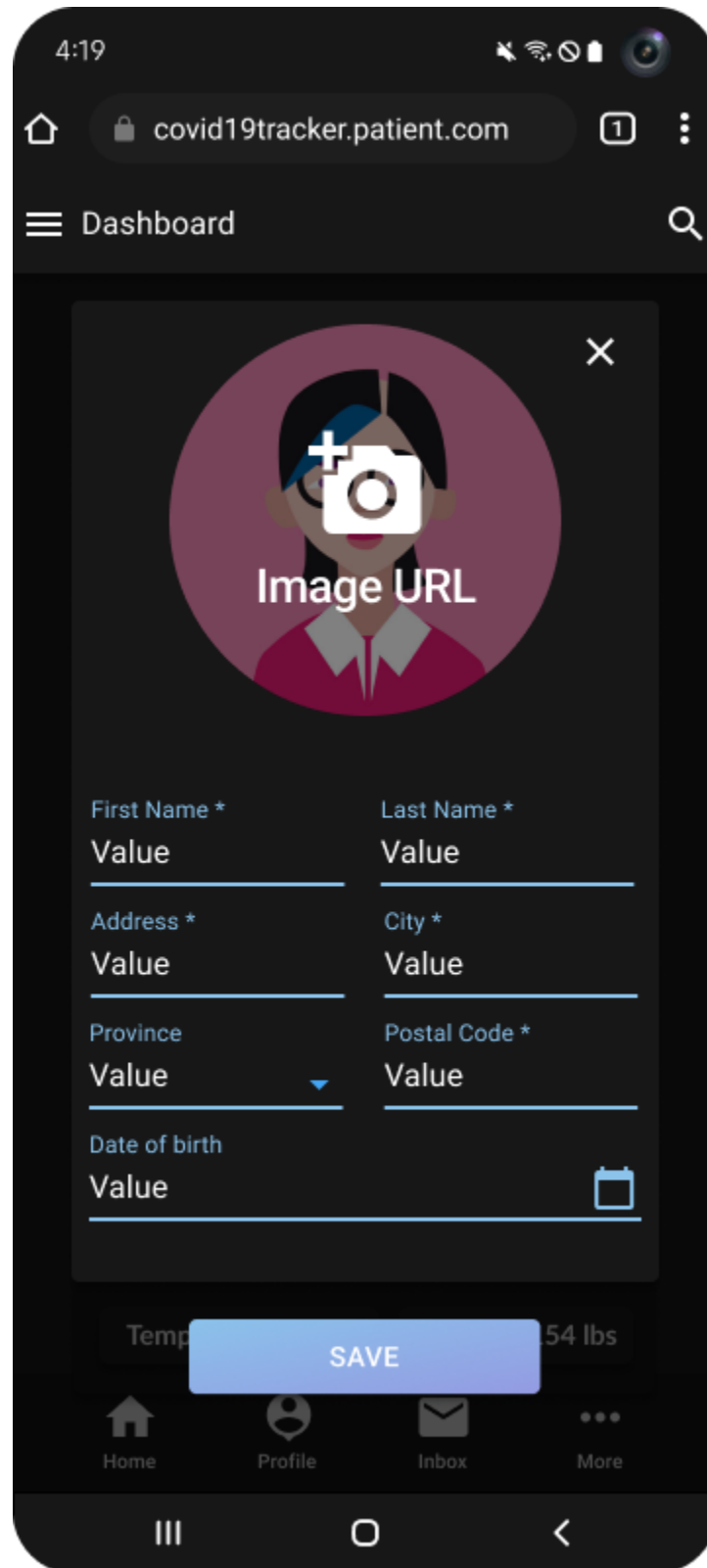


Figure 46: UI Client Patient Edit for User Story [Issue-37](#)

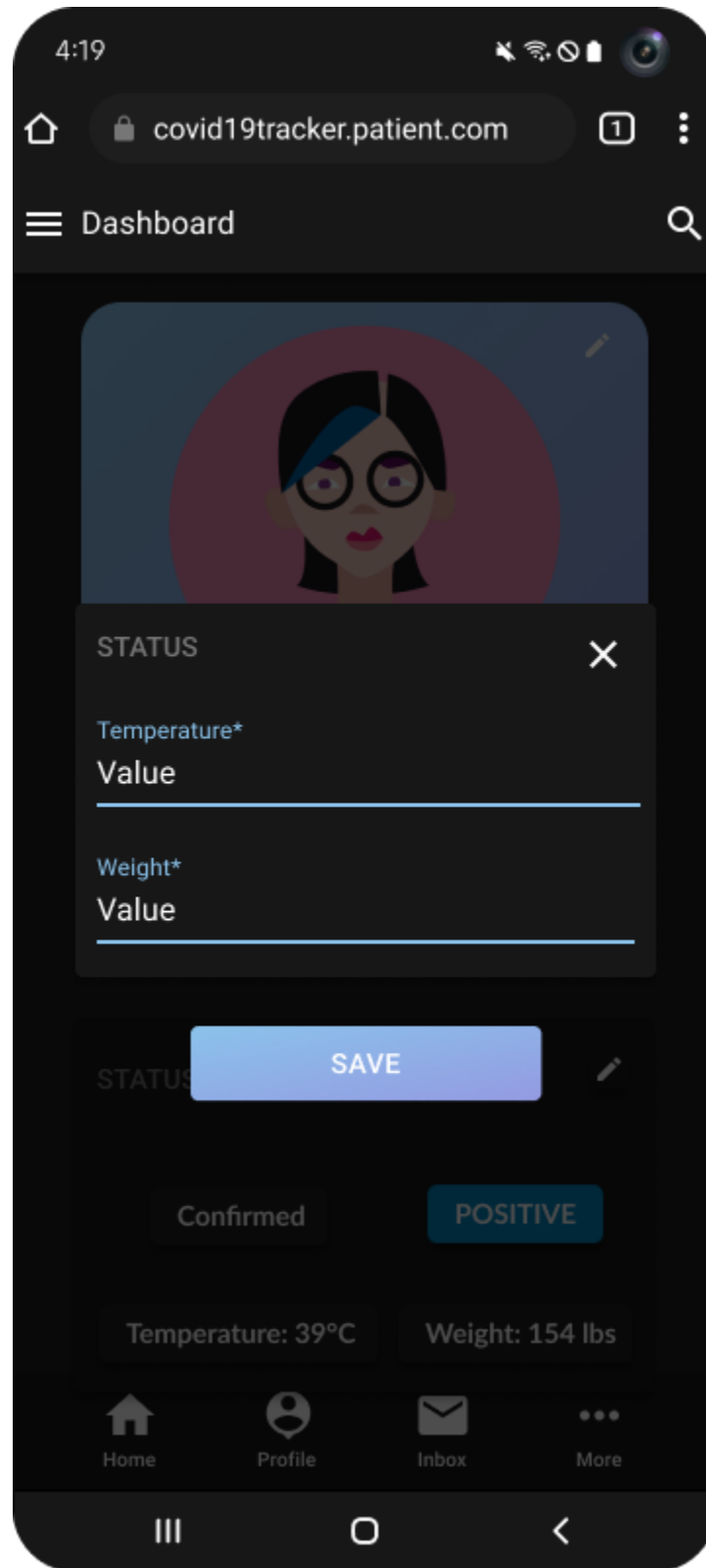


Figure 47: UI Client Patient Edit for User Story [Issue-37](#)

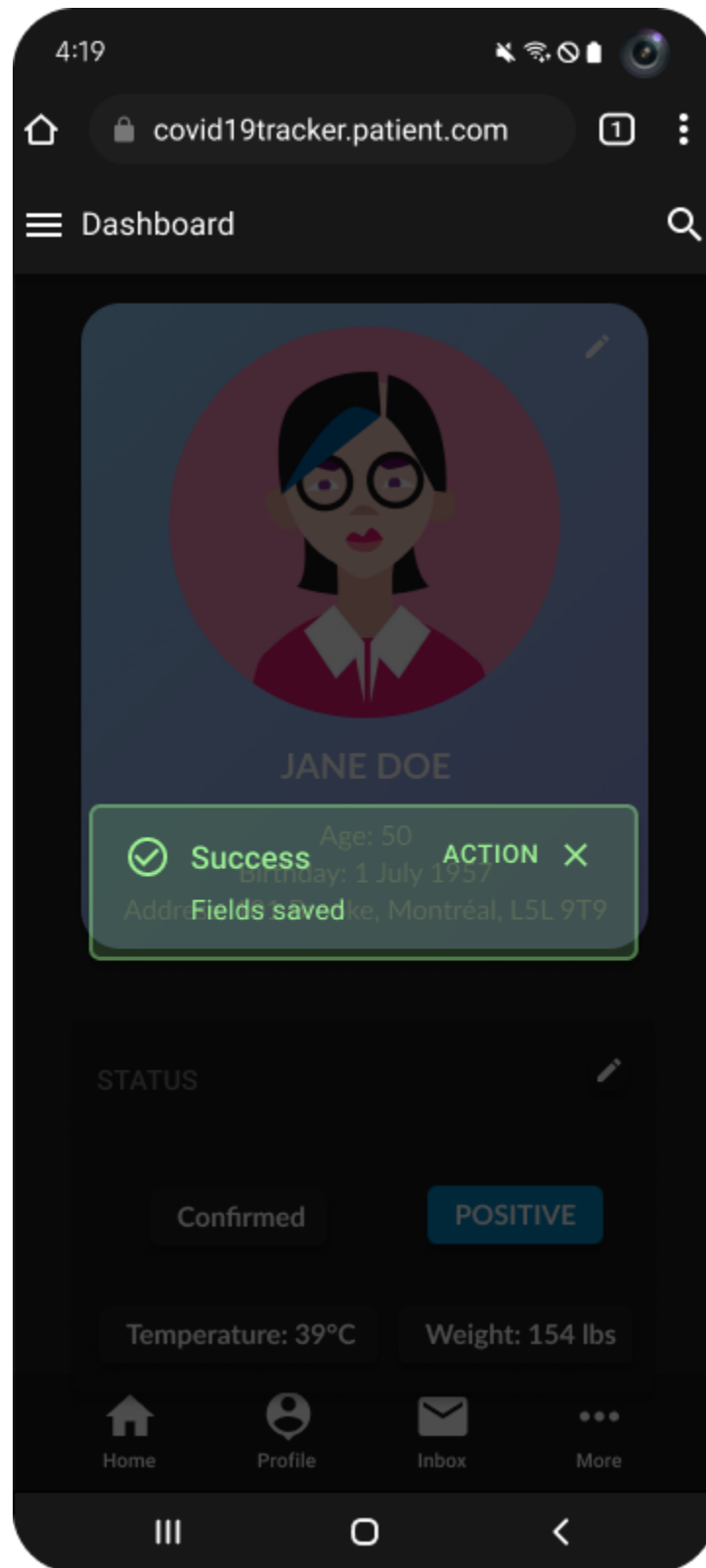


Figure 48: UI Client Patient Success for User Story [Issue-37](#)

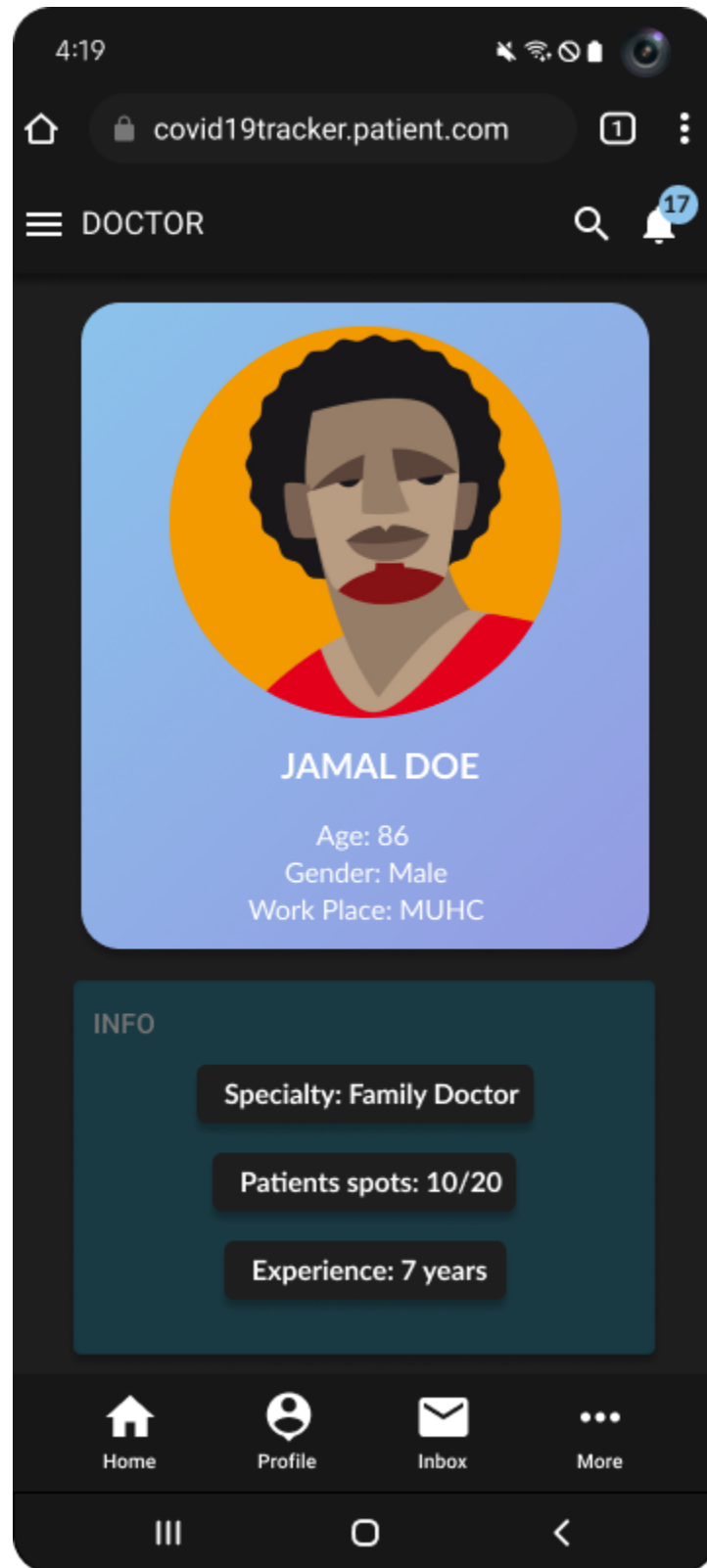


Figure 49: UI Doctor Info for User Story [Issue-39](#)

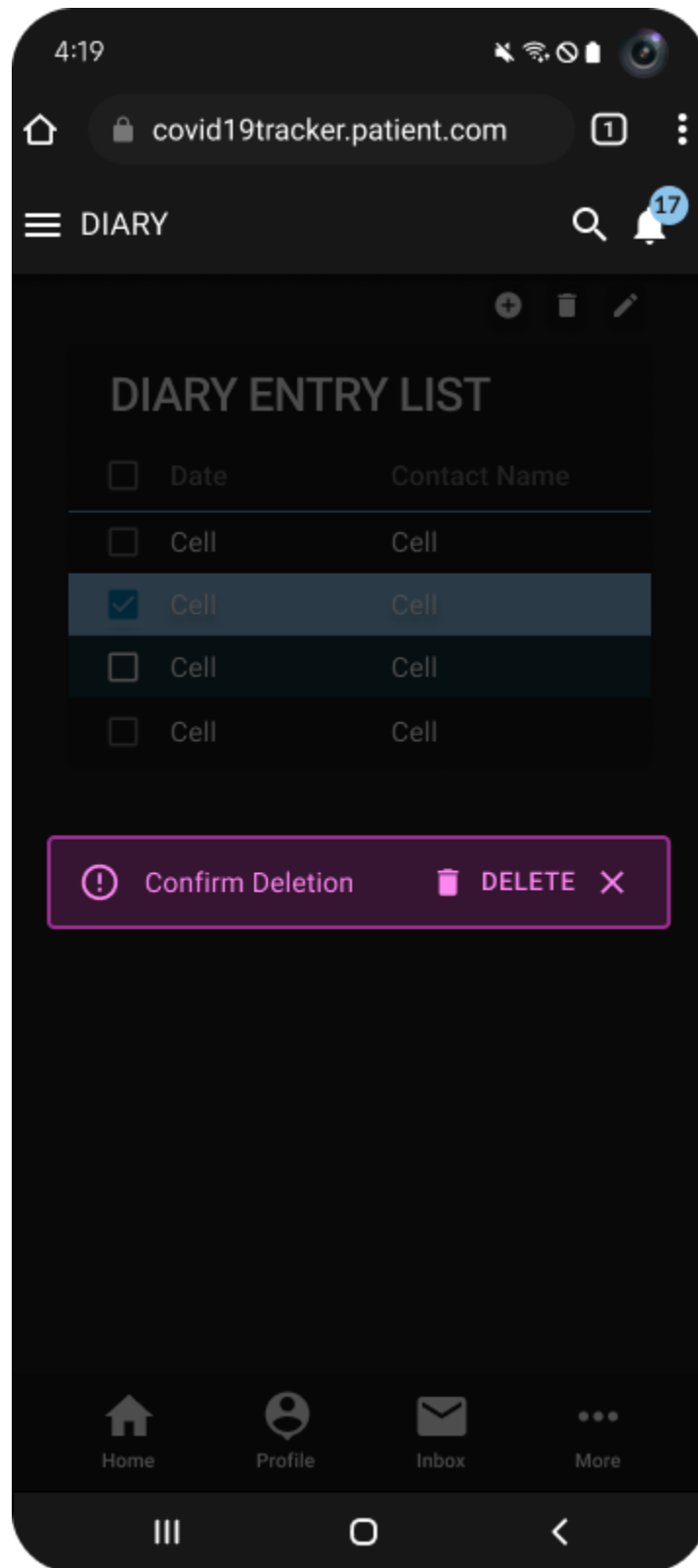


Figure 49: UI Diary delete

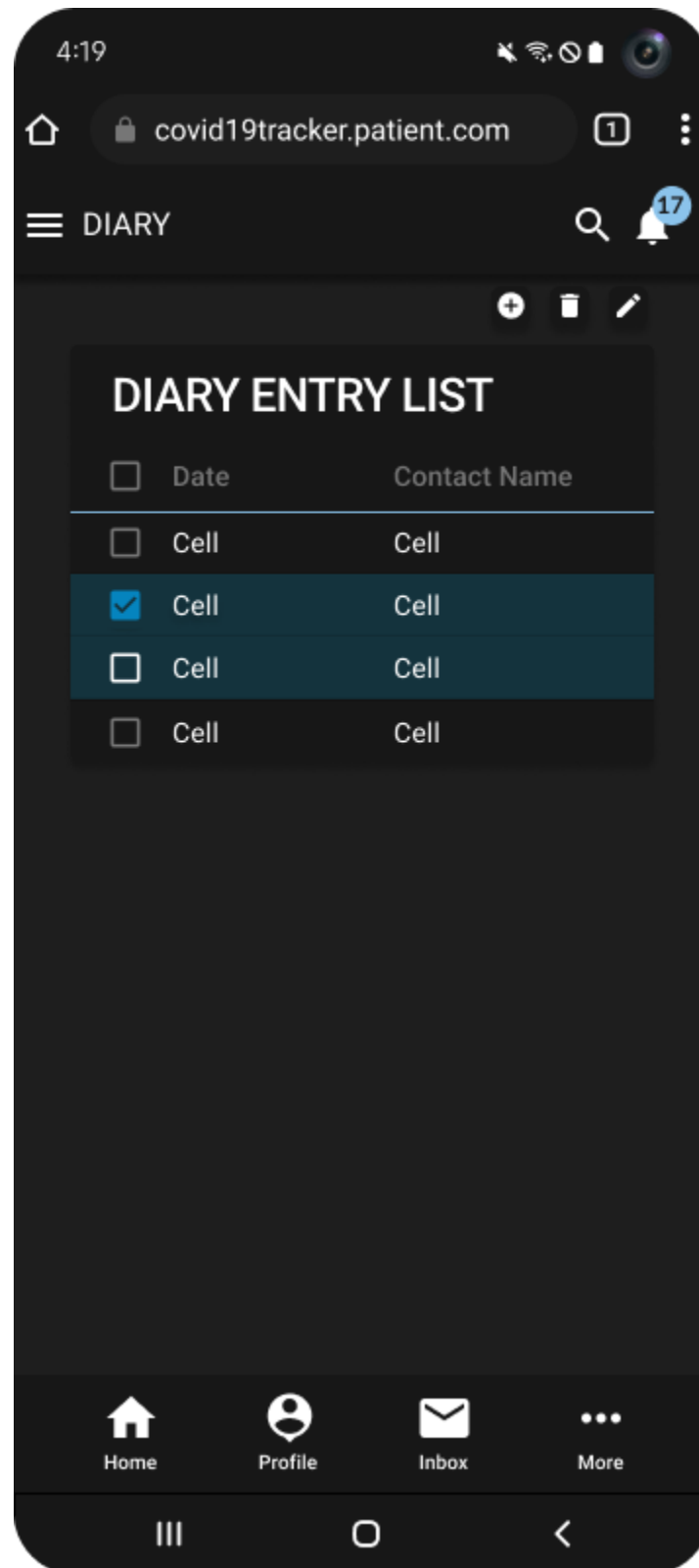


Figure 49: UI Diary list

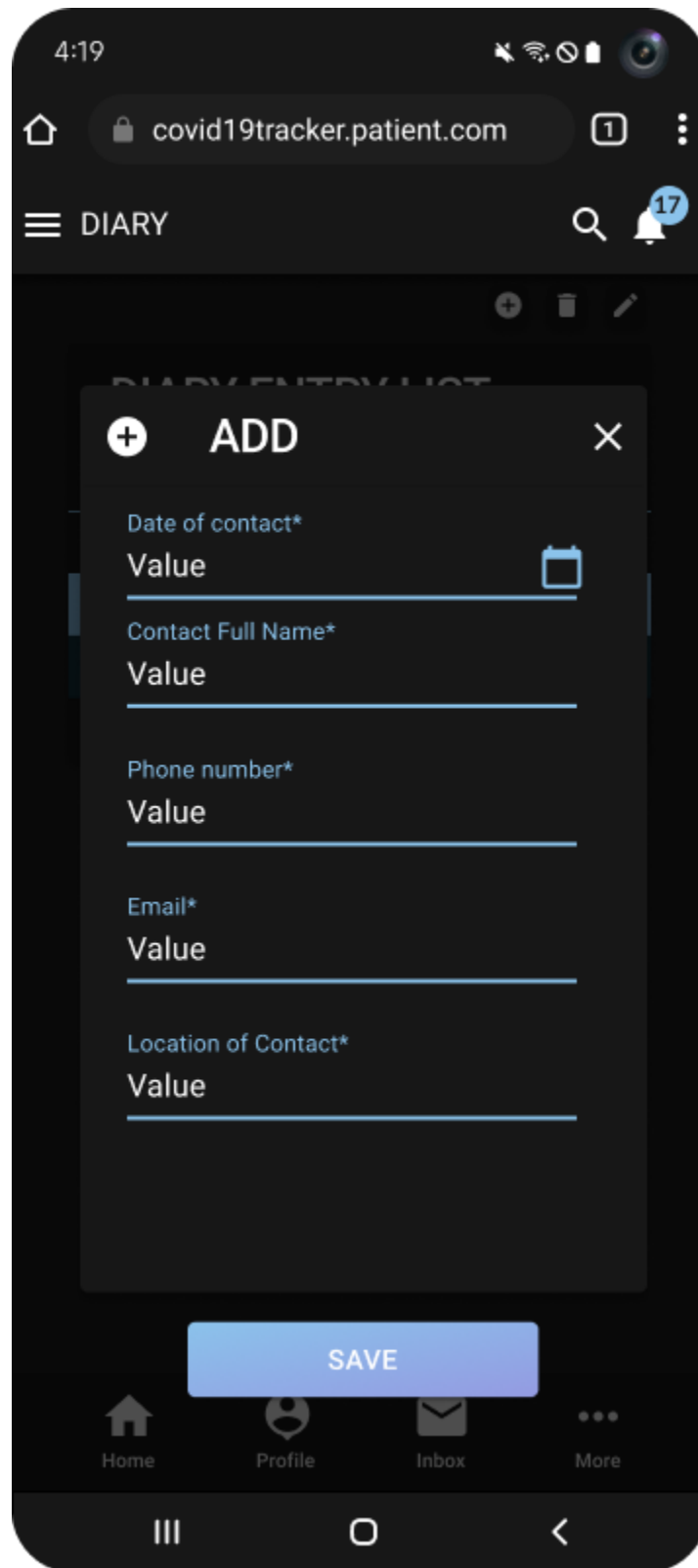


Figure 49: UI Diary Add

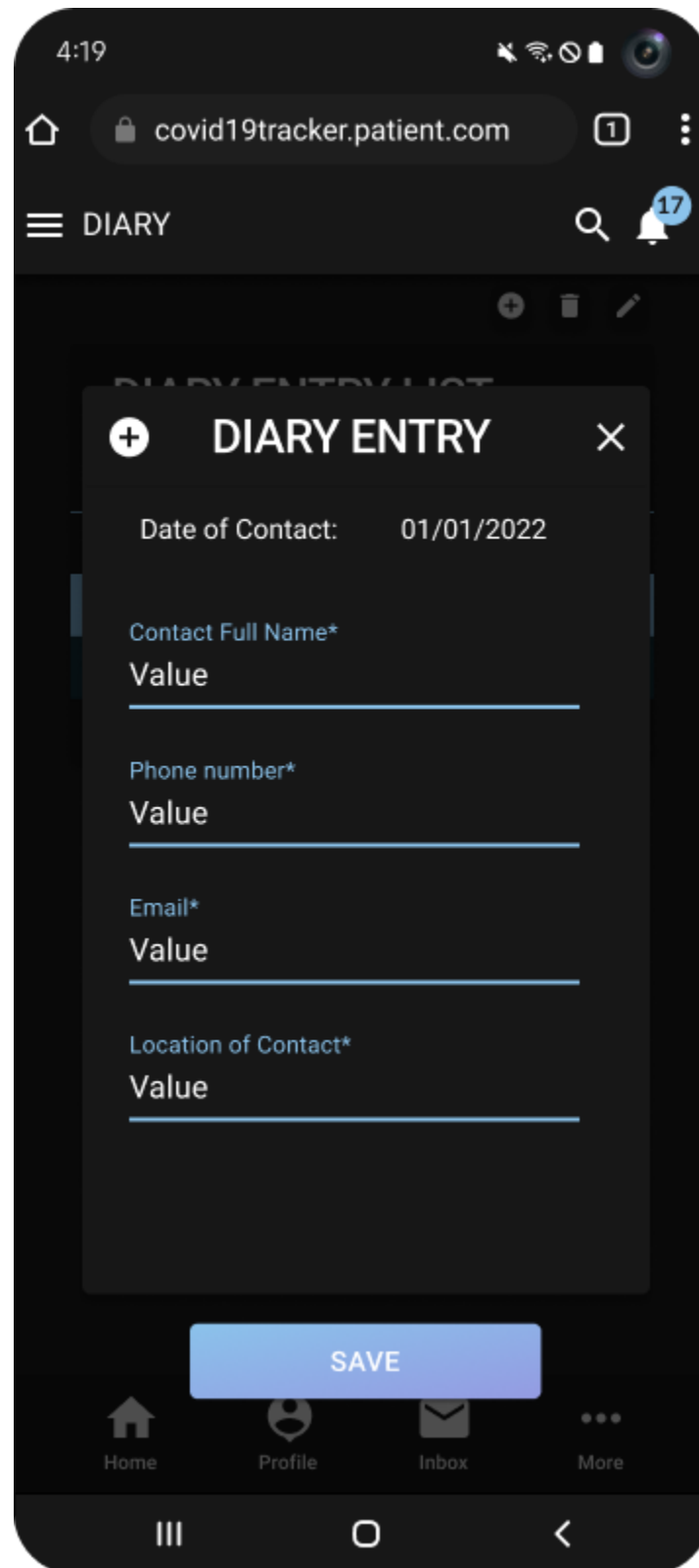


Figure 49: UI Diary Entry

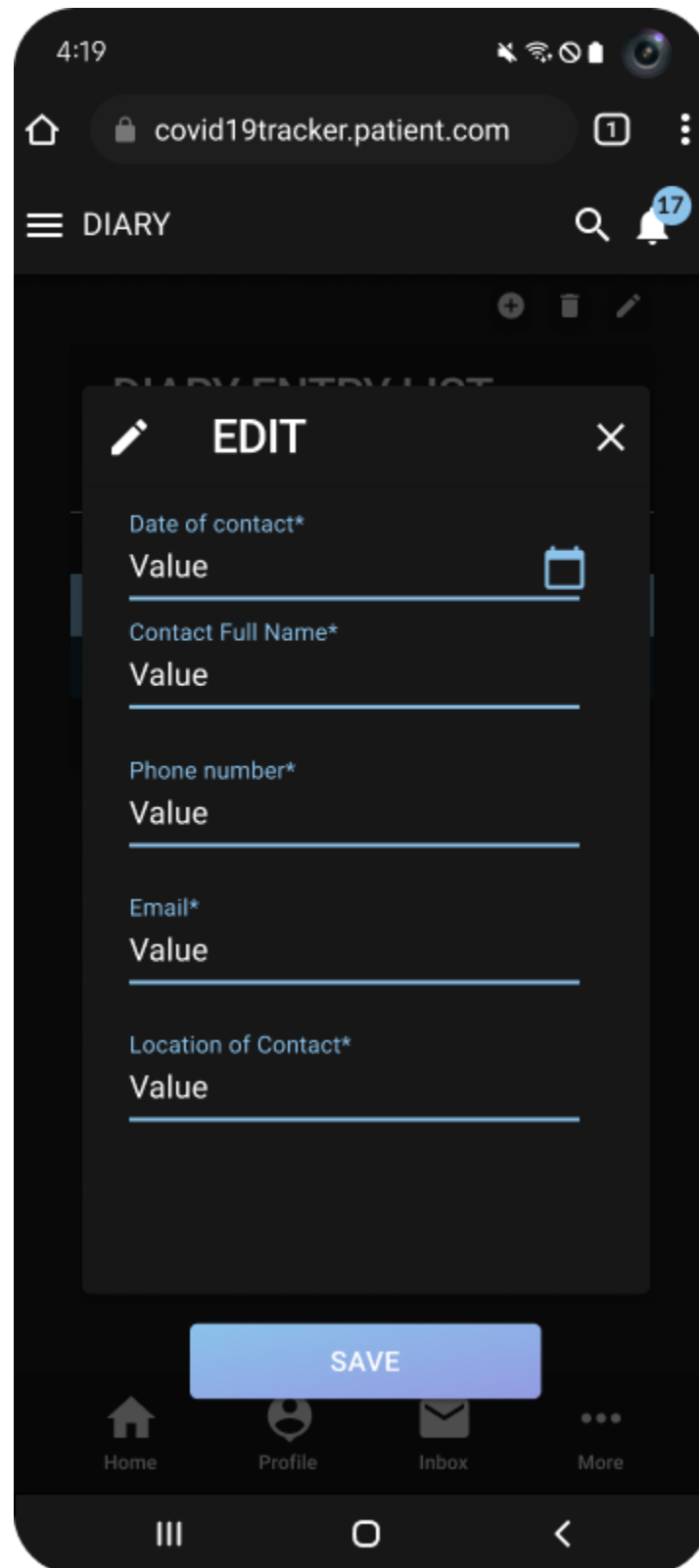


Figure 49: UI Diary Edit

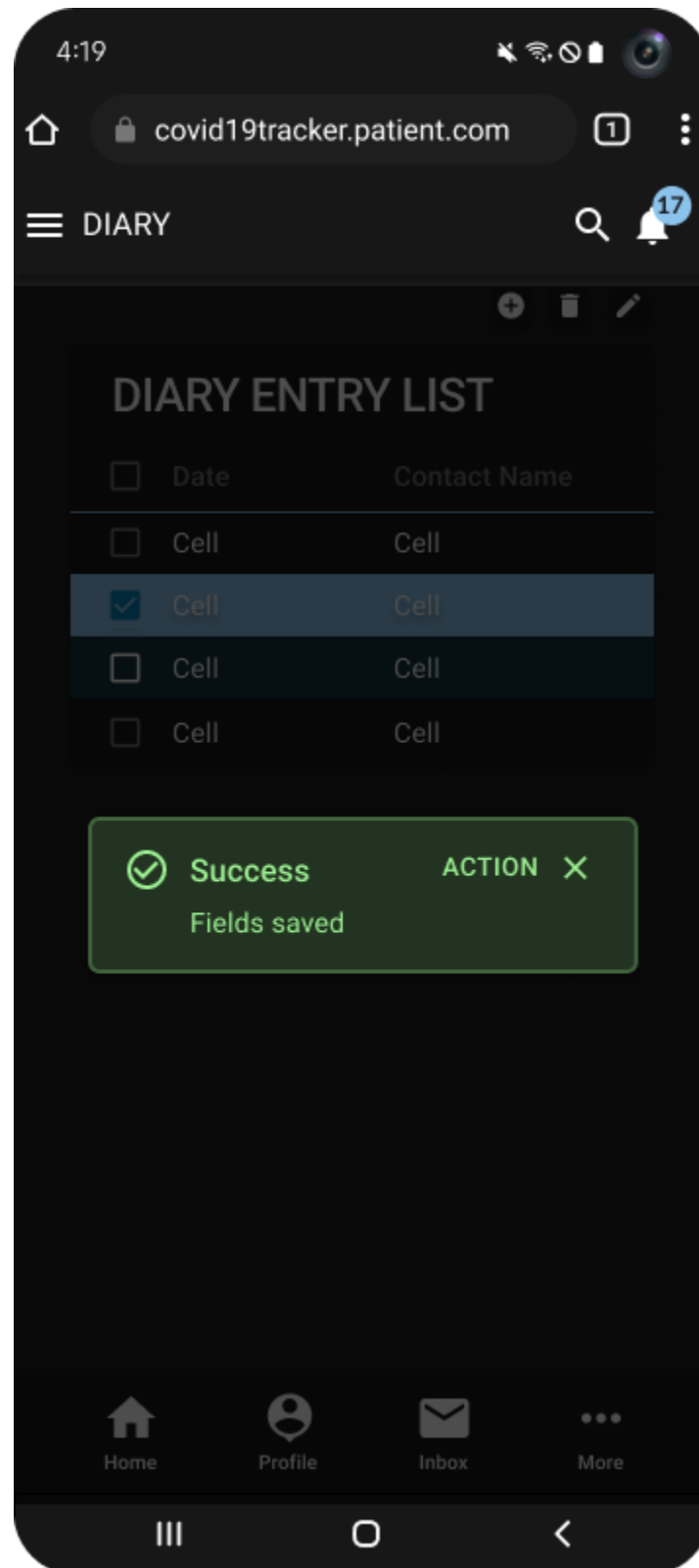


Figure 49: UI Diary Success

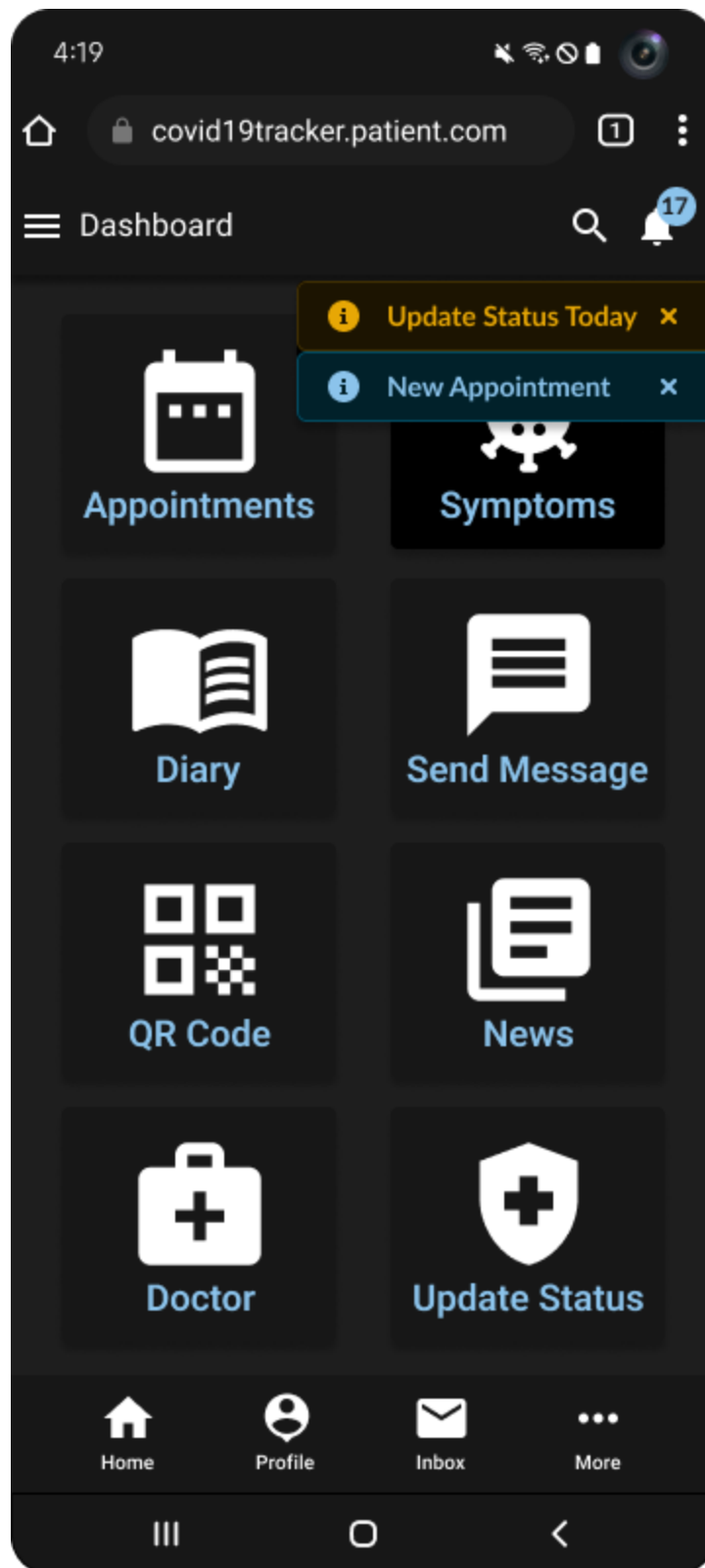


Figure 49: UI Notifications

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.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	87.5	100	80	87.5	
infiniteTimerGame.js	80	100	66.67	80	12
timerGame.js	100	100	100	100	

Test Suites: 2 passed, 2 total
 Tests: 4 passed, 4 total
 Snapshots: 0 total
 Time: 0.878s, estimated 1s
 Ran all test suites.

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	As a doctor, health official or immigration officer, I want to raise flags on certain COVID-19 patients so that their updates are prioritized over others
Acceptance Criteria	Given that I am logged in as a doctor, health official or immigration officer,
	and that I am on the patient's profile,
	and that I click on the flag,
	then I should see the patient's updates are prioritized.

Result	PASS
Comments	NONE

AT-2	Contact between patients & doctors
Acceptance Criteria	Given that I am logged in as a doctor or health official,
	and that I am on the inbox page,
	and that I click on the message button for a specific patient,
	then I should be redirected to a messenger window with the patient.
Result	PASS
Comments	NONE

AT-3	Frontend Display for “Identify how many patients are assigned to each doctor”
Acceptance Criteria	Given that I am logged in as an admin,
	and that I am on the Patients Page
	then I should be displayed all doctors and the number of patients they have been assigned.
Result	PASS
Comments	NONE

AT-4	Frontend display for "Assign doctors to patients"
Acceptance Criteria	Given that I am logged in as an admin, and that I am on the Patients Page, then I should be able to assign a doctor to a patient.
Result	PASS
Comments	NONE

AT-5	[Additional] Provide barcodes / QR codes for each record
Acceptance Criteria	Given that I am logged in as a doctor, admin, health official or immigration officer, and that I am on the QR code page, then I should be able to see a QR code.
Result	PASS
Comments	NONE

AT-6	[Additional] Frontend Display for "Re-assign a patient to another doctor"
Acceptance Criteria	Given that I am logged in as an admin, and that I am on the Patients Page,

	then I should be able to re-assign a patient to another doctor.
Result	PASS
Comments	NONE

AT-7	Frontend display for "Profile for patient"
Acceptance Criteria	Given that I am logged in as a patient,
	and that I click on my profile,
	then I should be able to see my profile page.
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.

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 49: Screenshot from actual test results for Admin app

Unit & integration test for Client app

Test file name	Description	Result
Dropdown.test.js	Test if the DropdownConfirmation component is present in the document.	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 to the screen.	PASSED
BottomNav.test.js	Test if the bottom navigation bar contains the 'bottomNav-iconTitle' class and contains the	FAILED

	'Profile' text content.	
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
SingUp.test.js	Test if the Signin page is rendered correctly to the screen.	PASSED

```

PASS src/components/DropdownConfirmation/Dropdown.test.js
PASS src/components/Dashboard/Dashboard.test.js
PASS src/components/Navbar/NavBar.test.js
PASS src/components/SymptomsTable/SymptomsTable.test.js
FAIL src/components/BottomNav/BottomNav.test.js
  ● Test suite failed to run

    Cannot find module 'material-ui-popup-state' from 'src/components/BottomNav/index.js'

    Require stack:
      src/components/BottomNav/index.js
      src/components/BottomNav/BottomNav.test.js

       9 | import Typography from "@mui/material/Typography";
      10 | import Popover from "@mui/material/Popover";
    > 11 | import PopupState, { bindTrigger, bindPopover } from "material-ui-popup-state";
        | ^

```

Figure 50: Screenshot from actual test results for Client app

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

System test for Client app (Cypress)

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

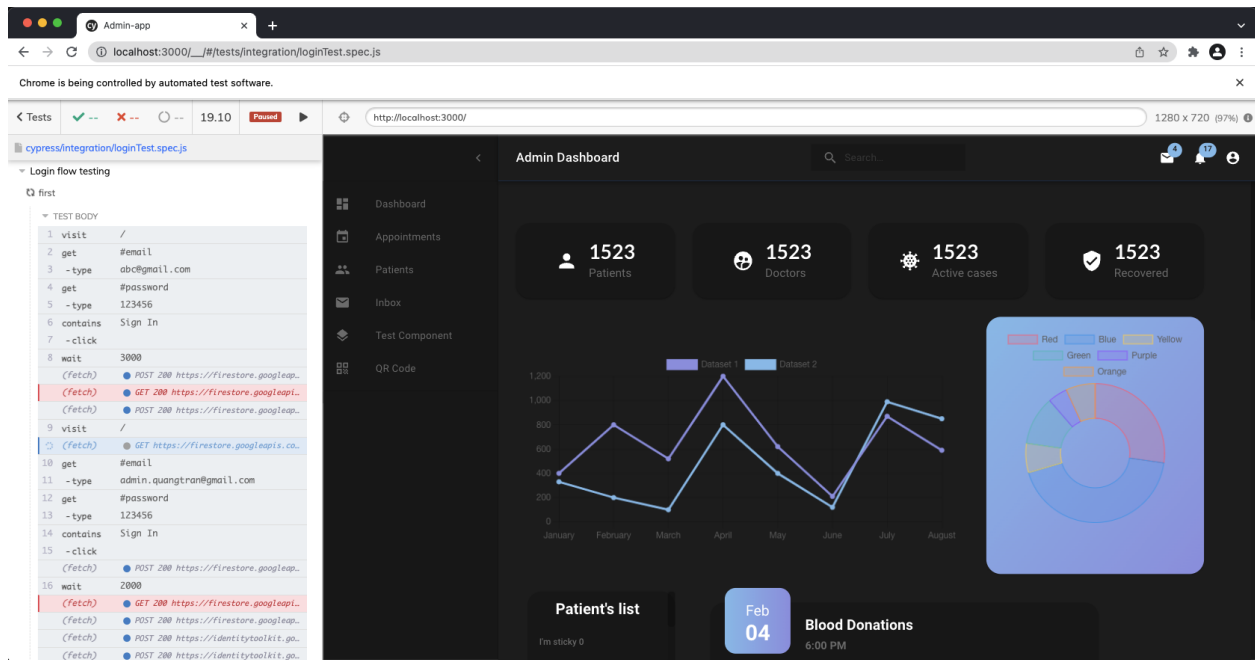


Figure 51: Screenshot from Cypress system test results