-

SOEN 390/F: SPRINT 1 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 2nd, 2022

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

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 **24 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 1:

ID	Name	USP	Priority
Issue-7	As a user, I want to have a login / signup system so that users with different roles can log into their dashboard to view the appropriate contents.	3	3
Issue-5	As a doctor, I want to have a dashboard view so that I can see an overview graph, my patients' status at a high level, and status updates.	8	1
Issue-9	As a doctor, I want to see the list of details about a patient so that I can find out his/her status, temperature, weight, list of symptoms, etc.	5	2
Issue-89	As an admin and doctor, I want to see the list of patients with details (assigned doctor, upcoming appointment, etc) so that I can monitor each patient.	8	3
Total		24	

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-7	**As a user, I want to have a login / signup system so that users with different roles can log into their dashboard to view the appropriate contents**
USP	3
Priority	3
Description	Additional details: Login / SignUp page for admin users (health officials, doctors, etc...) Login / SignUp page for clients (patients, guests) Acceptance criteria: Given: User type in his/her password correctly When: User clicks on the 'login' button Then: User will be redirected to the dashboard

Issue-5	**As a doctor, I want to have a dashboard view so that I can see an overview graph, my patients' status at a high level, and status updates.**
USP	8
Priority	1
Description	Example: All of a patient’s previously submitted status updates should be accessible and may include the last time the patient has made changes and, possibly, the device from which the changes were made. Acceptance criteria: Given: The doctor is logged in. When: The doctor clicks on the dashboard. Then: The doctor is shown the dashboard with graphs and pertinent information.

Issue-9	**As a doctor, I want to see the list of details about a patient so that I can find out his/her status, temperature, weight, list of symptoms, etc.**
USP	5
Priority	2
Description	Additional details: Create a similar "List of details" page for patients (on the patient app) #80 Acceptance criteria: Given: The user is logged in as a doctor. When: The user clicks on a patient profile. Then: The application shows a list of given details about the patient.

Issue-89	**As an admin and doctor, I want to see the list of patients with details (assigned doctor, upcoming appointment, etc) so that I can monitor each patient.**
USP	8
Priority	3
Description	As an admin and doctor, I want to see the list of patients with details (assigned doctor, upcoming appointment, etc) so that I can monitor each patient. Acceptance criteria: Given: The user is logged in as an admin or doctor. When: The patient clicks on patients. Then: The application shows a list of patients with details such as assigned doctor and more.

4.0 RELEASE PLANNING

4.1 Sprint 1 Summary

Sprint 1 focused on delivering a variety of features including login/signup, dashboard view for doctors, patient profile details, monitoring status & symptoms of confirmed/unconfirmed patients and list of patients with details. This sprint provided a kickstart to the project in which the core features of the application were started and substantial work in the frontend was performed. The problems we faced in this sprint included several merge conflicts due to multiple team members working on different branches simultaneously for a considerable amount of time and thus, merging the branch code to the main often resulted in inconsistencies (e.g., CSS conflicts due to a generic className). In response to this, we realized that Git merges should be performed incrementally at a faster pace rather than an accumulation of a big chunk of work and merged directly to main for very minor changes instead of creating a new branch and causing extra conflicts.

Additionally, we had to rename generic classNames to specific names in order to remove conflicts and prevent any future conflicts that may arise. During this sprint, the LogIn/SignUp feature (issue-7) was pushed back and was completed in Sprint 1 instead of Sprint 2 since our team realized it is a core feature that should rather be implemented early on and we also received this recommendation from our TA. On the other hand, issue-37 and issue-17 were brought forward to be completed in Sprint 2 as they were justified to be better suited for later sprints which also allowed to ensure the timely completion of Sprint 1. Similarly, issue-19 and issue-11 were also judged to be better suited for Sprint 3 allowing the team to focus on core features providing basic functionality first.

Story ID	**Story Title**	**USP**	**Status**
Issue-7	LogIn / SignUp function	3	**DONE**
Issue-5	Dashboard view for doctors	8	**DONE**
Issue-9	Patient profile details	5	**DONE**
Issue-89*	List of patients with details (Admin)	8	**DONE**
Issue-37	Allow client to edit profile page (patients)	8	**PUSHED TO SPRINT 2**
Issue-17	Identify how many patients are assigned to each doctor	2	**PUSHED TO SPRINT 2**

Issue-19	Allow admin to manage client account	5	PUSHED TO SPRINT 3
Issue-11	Monitor status & symptoms of confirmed and unconfirmed patients	2	PUSHED TO SPRINT 3
Total		**43**	**24**

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 2 Planning

Sprint 2 will focus on delivering the next set of importance features including tracing & notifying COVID-19 patients, prioritizing patient updates, providing a feature for contact between patients & doctors, enabling different visibility for different users, showing that a patients' status updates have been reviewed, identifying the number of patients assigned to each doctor, assigning/re-assigning doctors to patients, providing barcodes/QR codes for each record, allowing patients to edit their profile page and creating a similar "List of details" page for patients on the patient app.

In this sprint, our team is planning on covering 11 user stories which sum up to a total of 58 user story points. As compared to the previous sprint in which we completed a total of 26 user story points, the workload can be seen to have doubled for Sprint 2. The problem we might possibly face is to have to push forward some user stories to later sprints due to time constraints. In this second sprint, we are also catching up on two user stories (Issue-17 and Issue-37) 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**
Issue-4	Trace & notify COVID-19 patients	3	
Issue-3	Prioritize patient updates	2	
Issue-13	Contact between patients & doctors	8	

Issue-14	Different visibility for different users	5	
Issue-16	Display / show that a patients' status updates have been reviewed	2	
Issue-17	Identify how many patients are assigned to each doctor	2	
Issue-18	Assign doctors to patients	5	
Issue-20	Re-assign a patient to another doctor	5	
Issue-22	Provide barcodes / QR codes for each record	13	
Issue-37	Allow client to edit profile page (patients)	8	
Issue-80	Create a similar "List of details" page for patients (on the patient app)	5	
Total		**58**	

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.

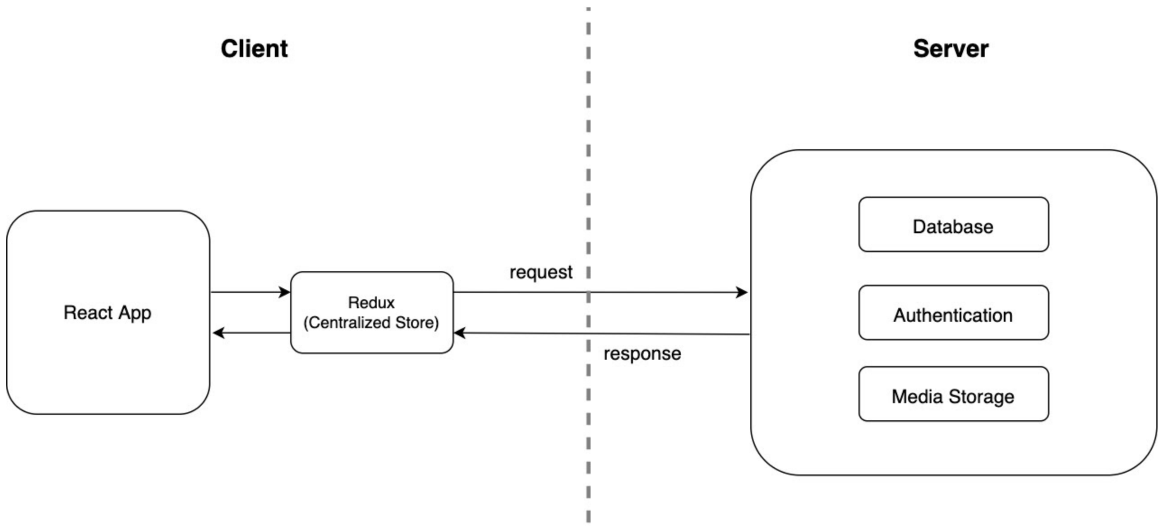


Figure 1: High-level overview system diagram

5.2 Use Case Diagram

The diagram below provides a complete graphical illustration of all possible interactions of a user with the COVID-19 application system. In this case, the four different users include patients, doctors, health officials, and immigration officers.

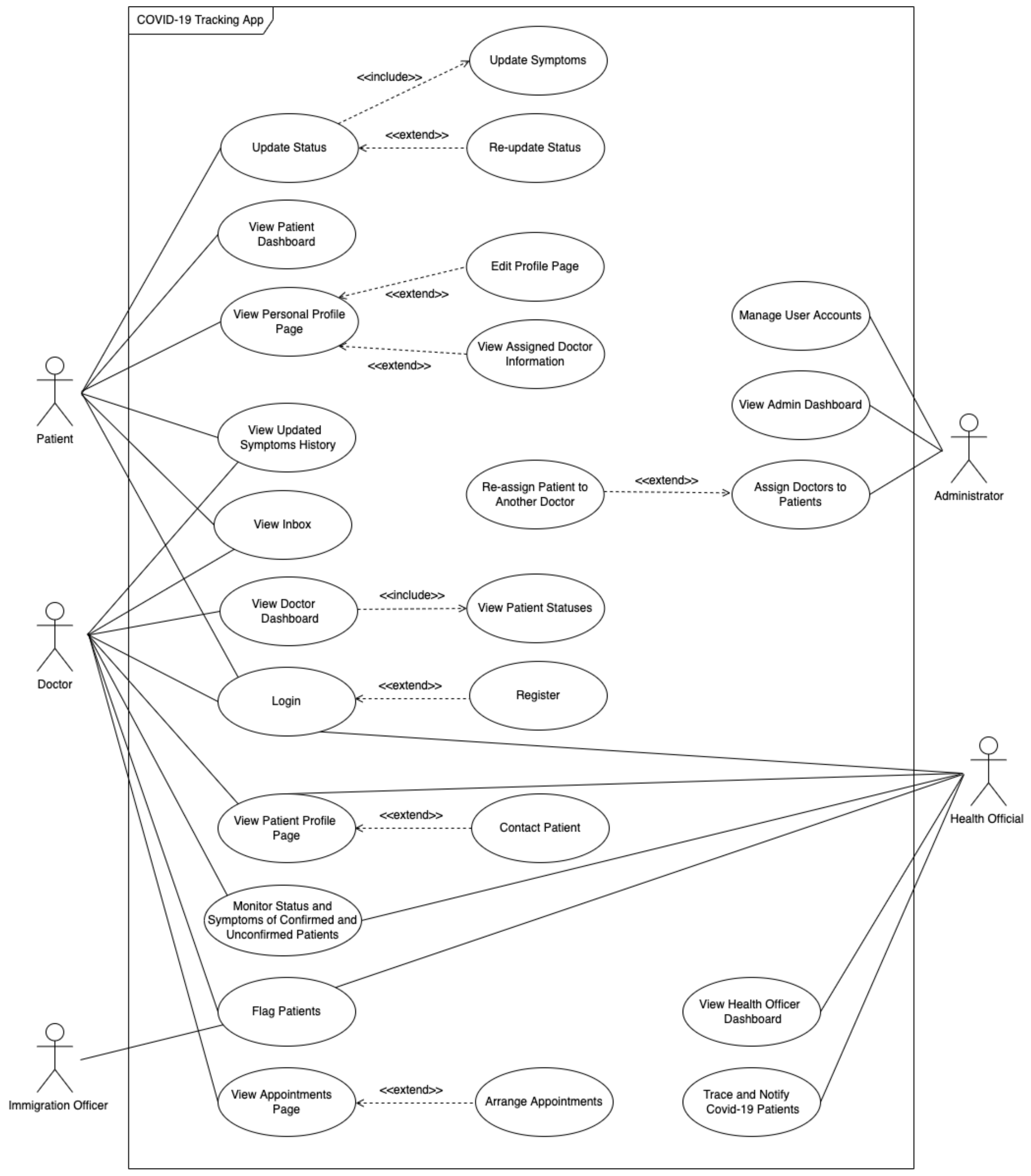


Figure 2: Use case diagram

5.3 Domain Model Diagram

The idea of a centralized storage system was derived to simplify the data flow of the system in general. In other words, passing data in any form (either from parent->child or child->parent) inside the component tree can get complicated very quickly as the application's complexity grows. To make a good separation of concerns, the system needs to have a centralized storage system outside of the component tree to manage all of its states effectively.

In this second diagram, the App component is the entry point of the application. Thus, it makes sense to "wrap" the Store Provider service around this root component so that the rest of the component tree has access to all states from the centralized store.

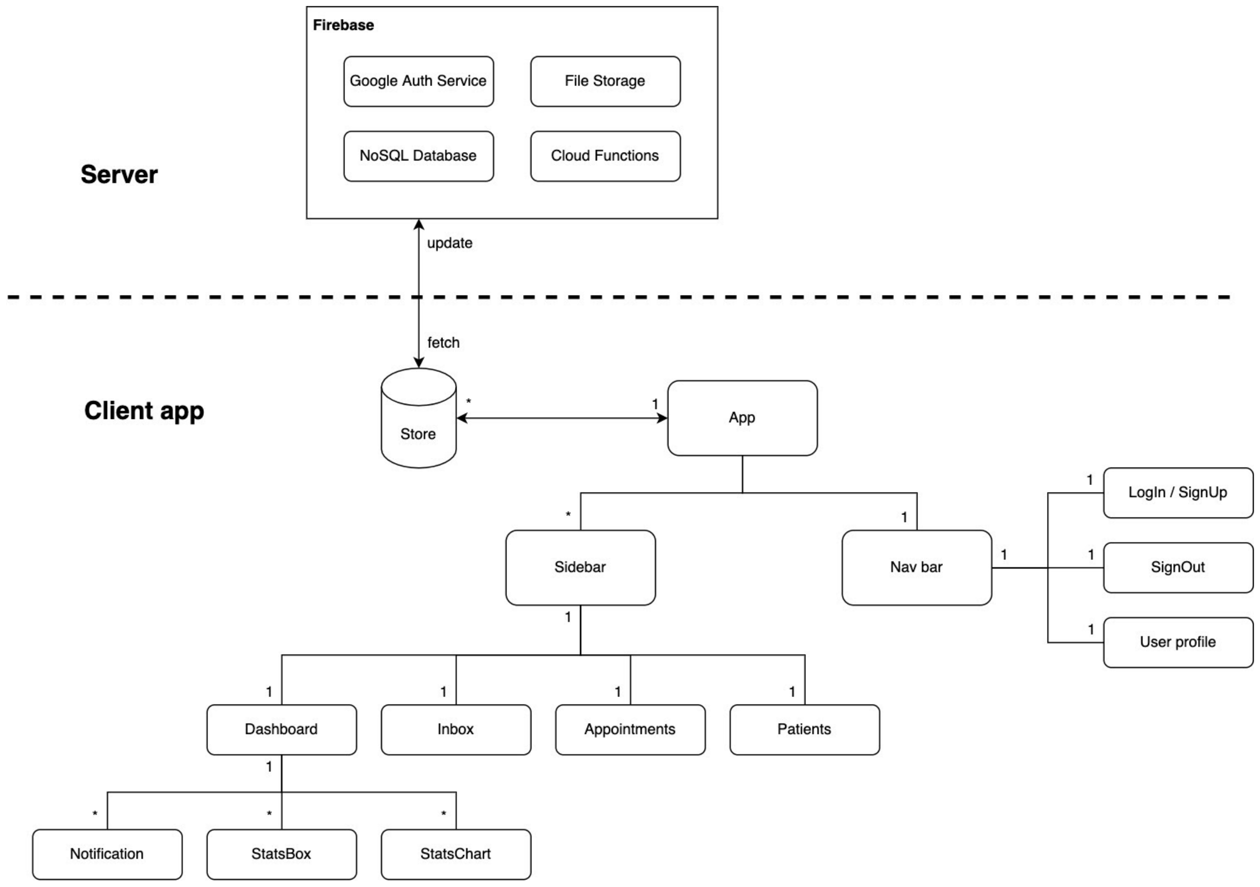


Figure 3: Domain model diagram

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.

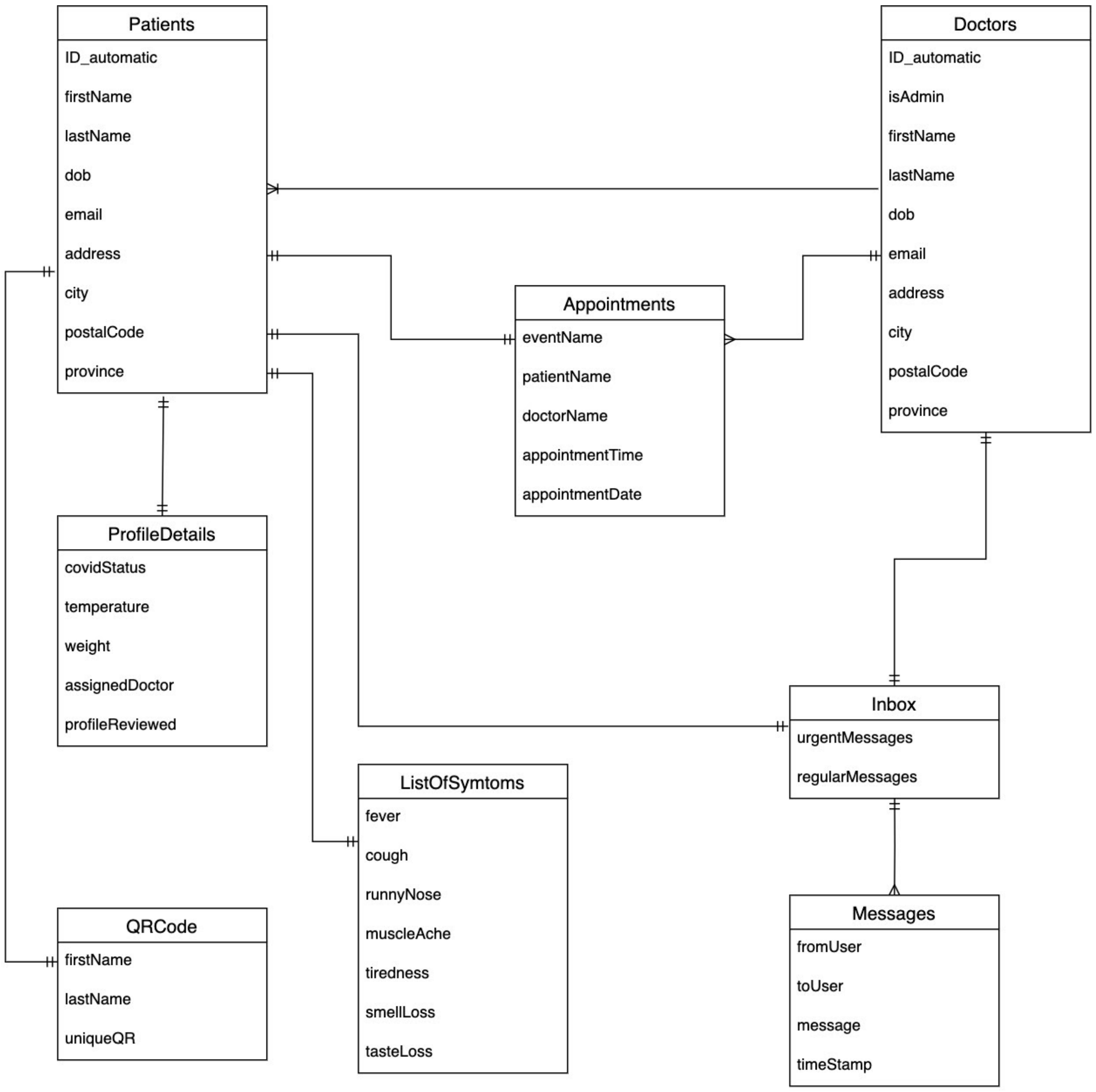


Figure 4: Database design

6.0 RISK MANAGEMENT

In this section, we present the purpose of the risk management plan, it’s 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

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.

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				

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

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

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

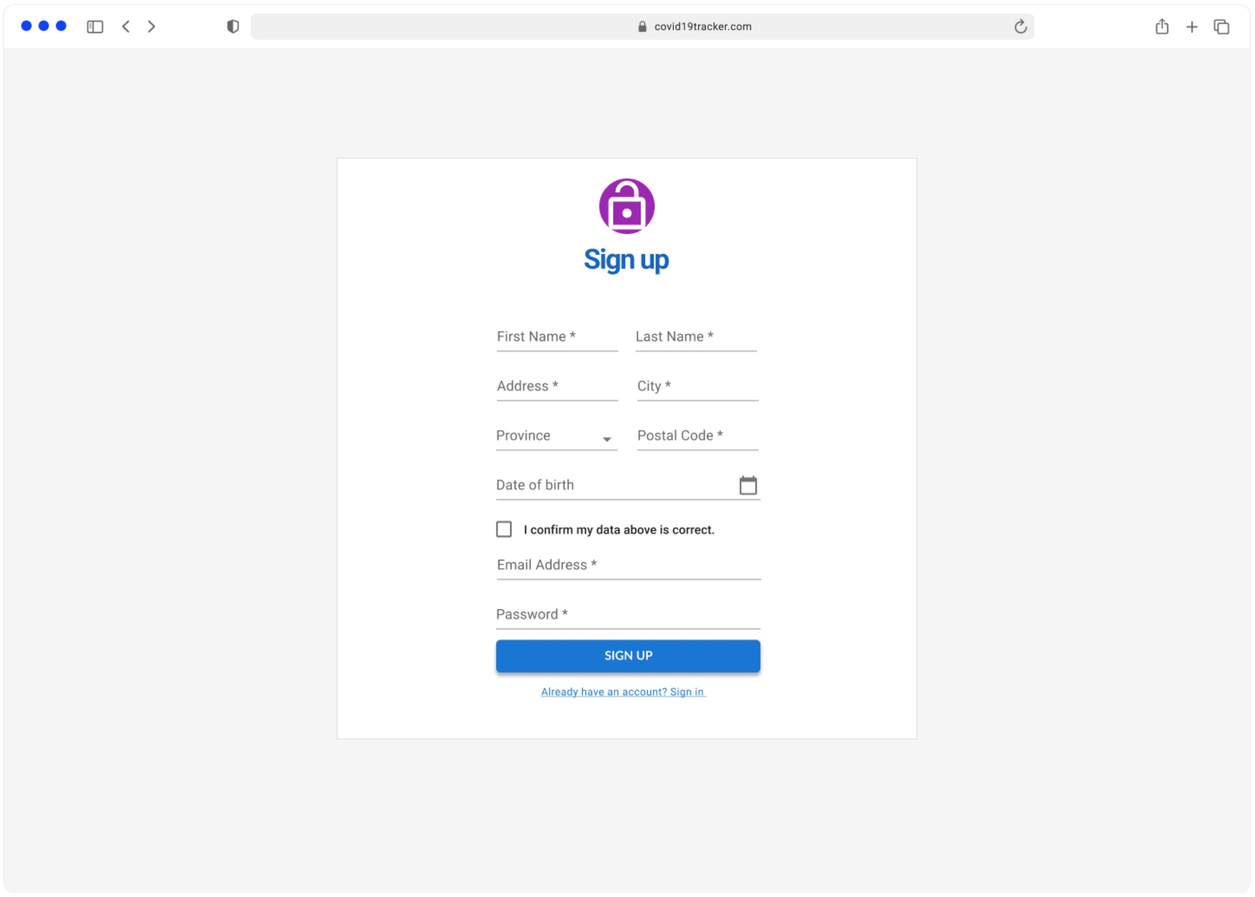


Figure 1: UI Admin Sign-up for User Story Issue-7

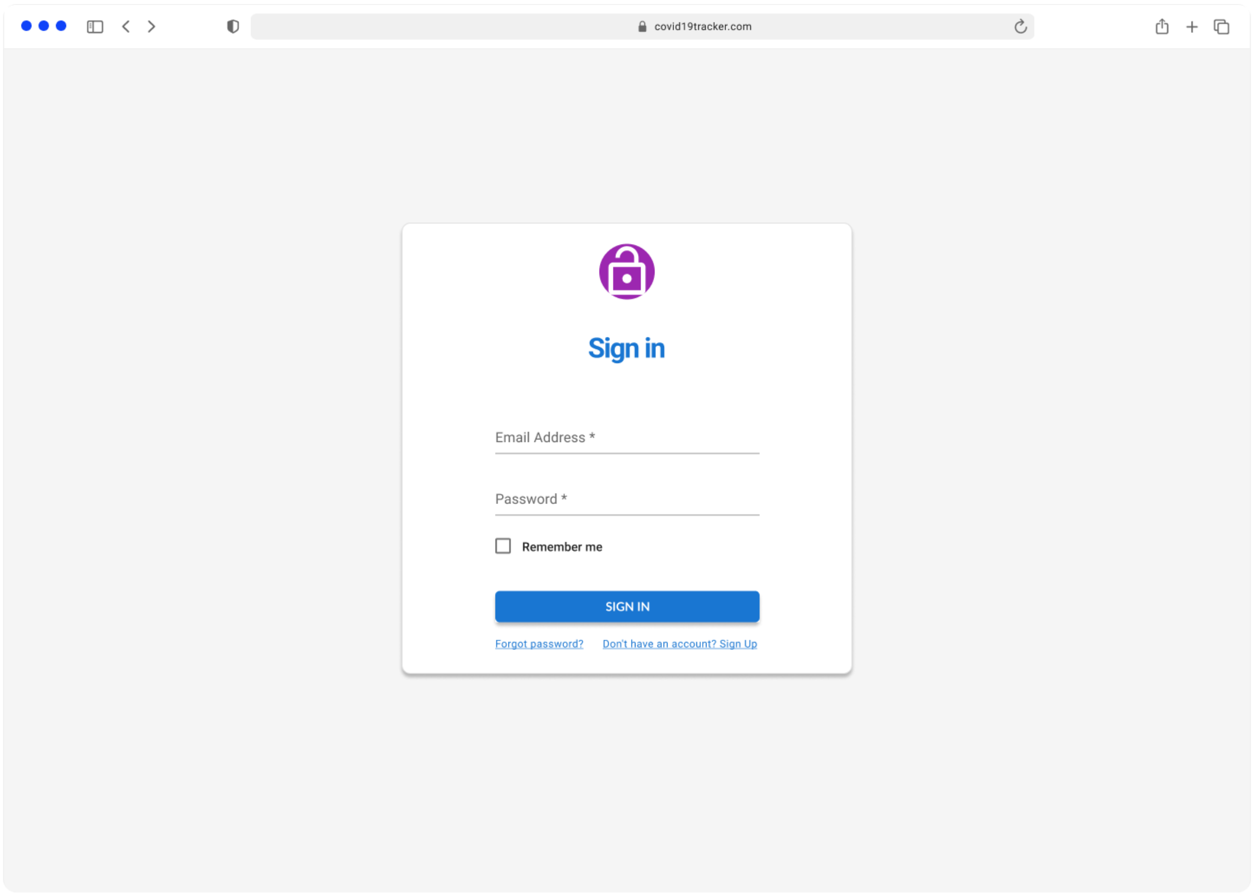


Figure 2: UI Admin Sign-in for User Story [Issue-7]

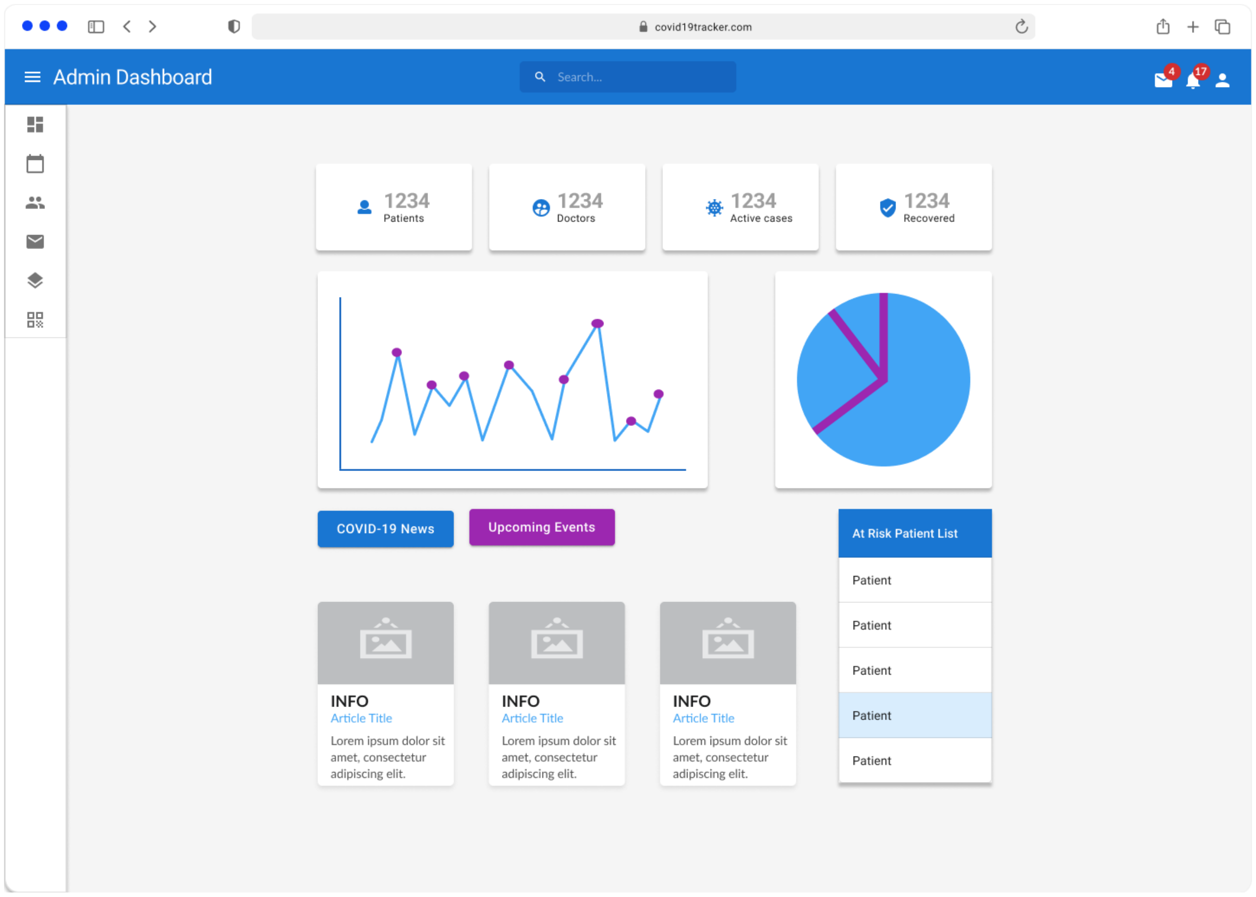


Figure 3: UI Admin Dashboard for User Story Issue-5

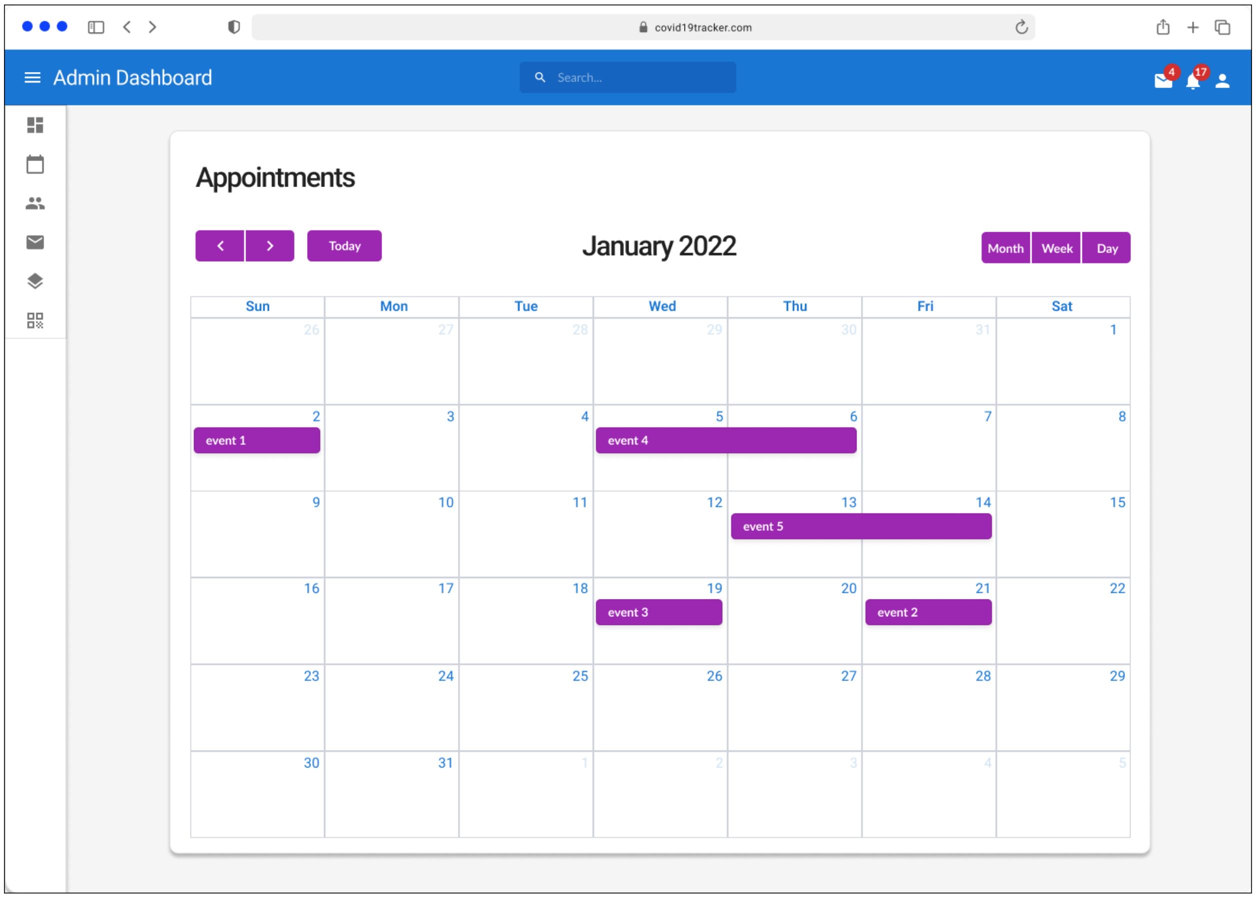
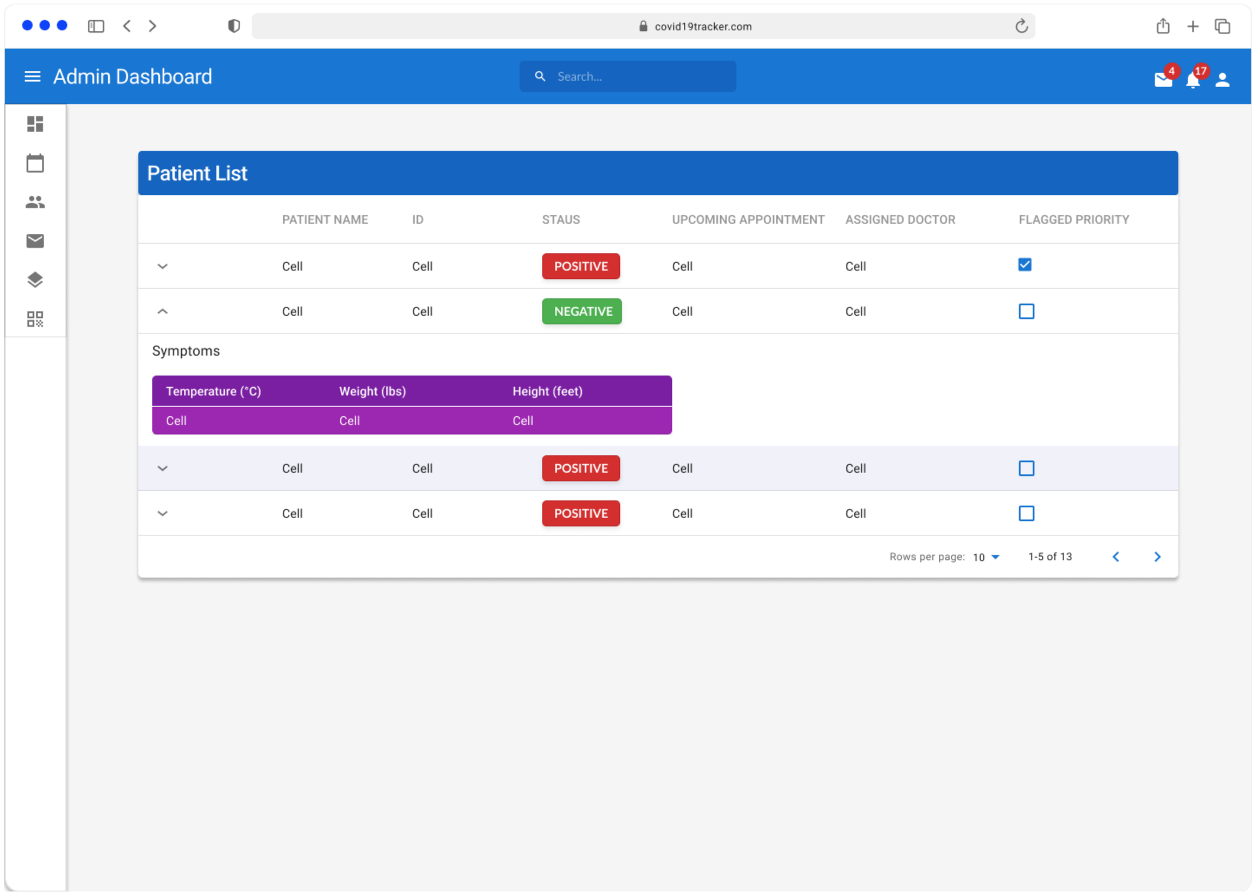


Figure 4: UI Admin Appointments for User Story Issue-5


covid19tracker.com

Patient List

	PATIENT NAME	ID	STAUS	UPCOMING APPOINTMENT	ASSIGNED DOCTOR	FLAGGED PRIORITY
	Cell	Cell	POSITIVE	Cell	Cell	☑
	Cell	Cell	NEGATIVE	Cell	Cell	☐

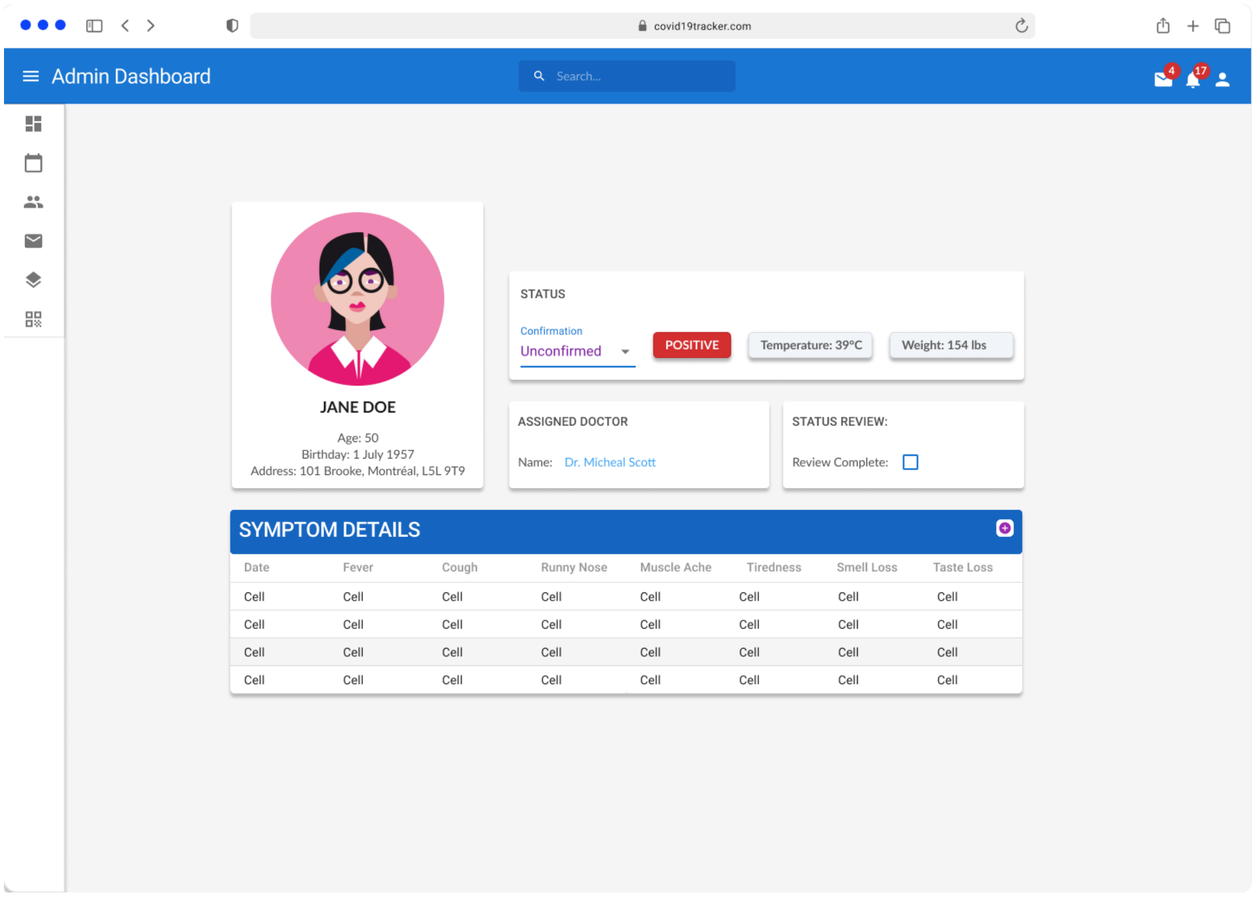
Symptoms

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

	Cell	Cell	POSITIVE	Cell	Cell	☐
	Cell	Cell	POSITIVE	Cell	Cell	☐

Rows per page: 10 1-5 of 13

Figure 5: UI Admin Patient List for User Story Issue-89


covid19tracker.com

Admin Dashboard

Search...

STATUS

Confirmation: Unconfirmed

POSITIVE

Temperature: 39°C

Weight: 154 lbs

JANE DOE

Age: 50
Birthday: 1 July 1957
Address: 101 Brooke, Montréal, L5L 9T9

ASSIGNED DOCTOR

Name: 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

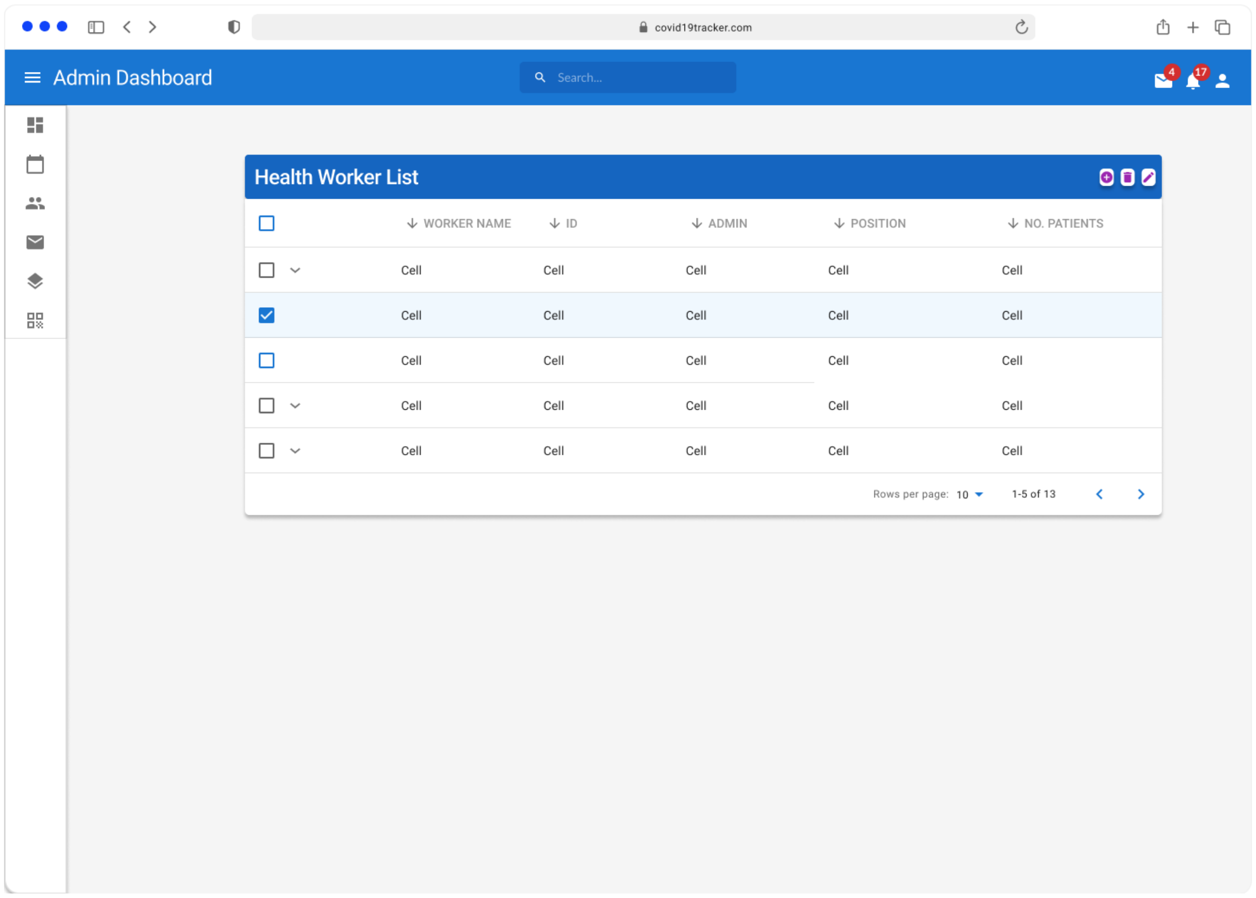
Figure 6: UI Admin Patient Details for User Story Issue-9

7.2 Sprint 2 UI Mockups

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

7.2.1 Admin Portal

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



☐	WORKER NAME	ID	ADMIN	POSITION	NO. PATIENTS
☐	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 6: UI Admin Health Worker List for User Story Issue-17 and Issue-19

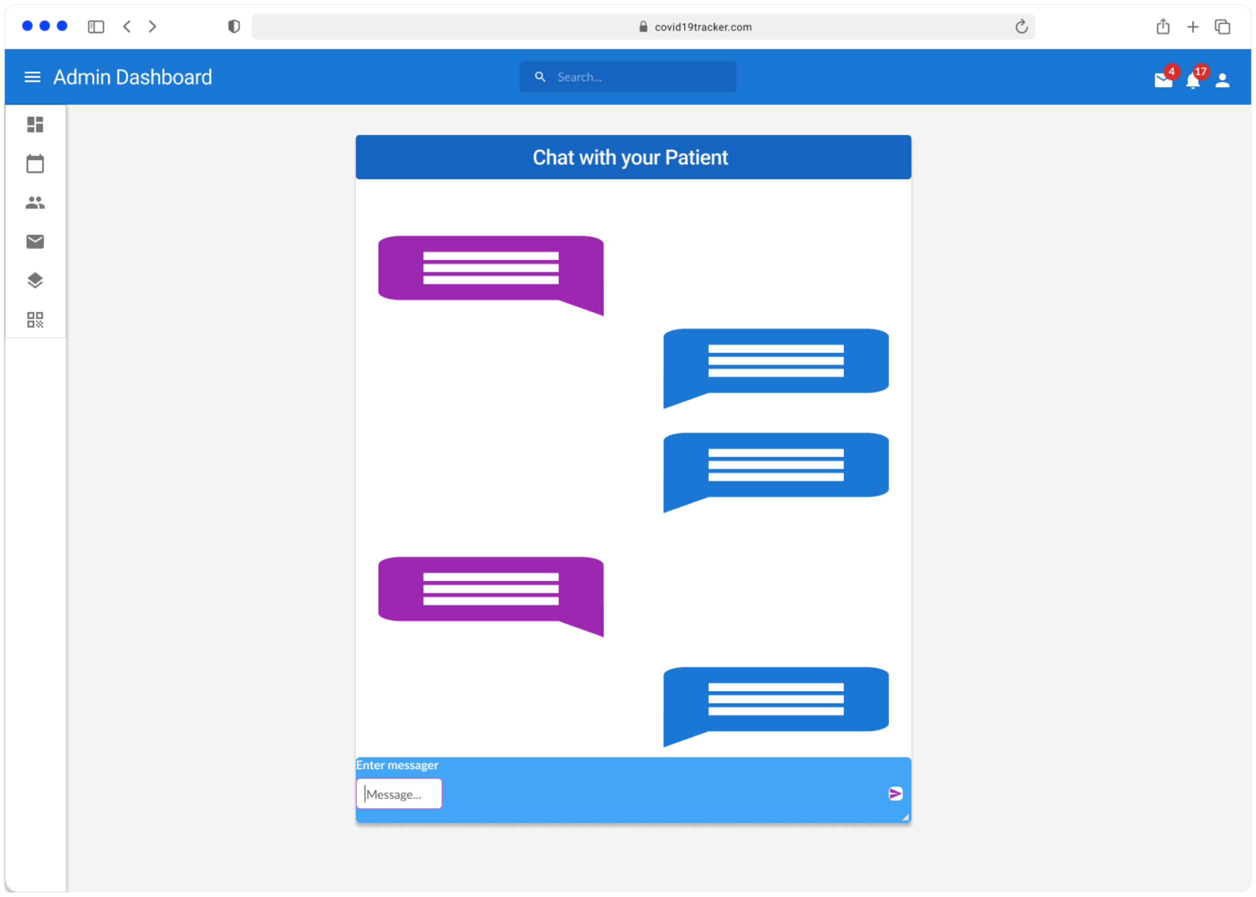
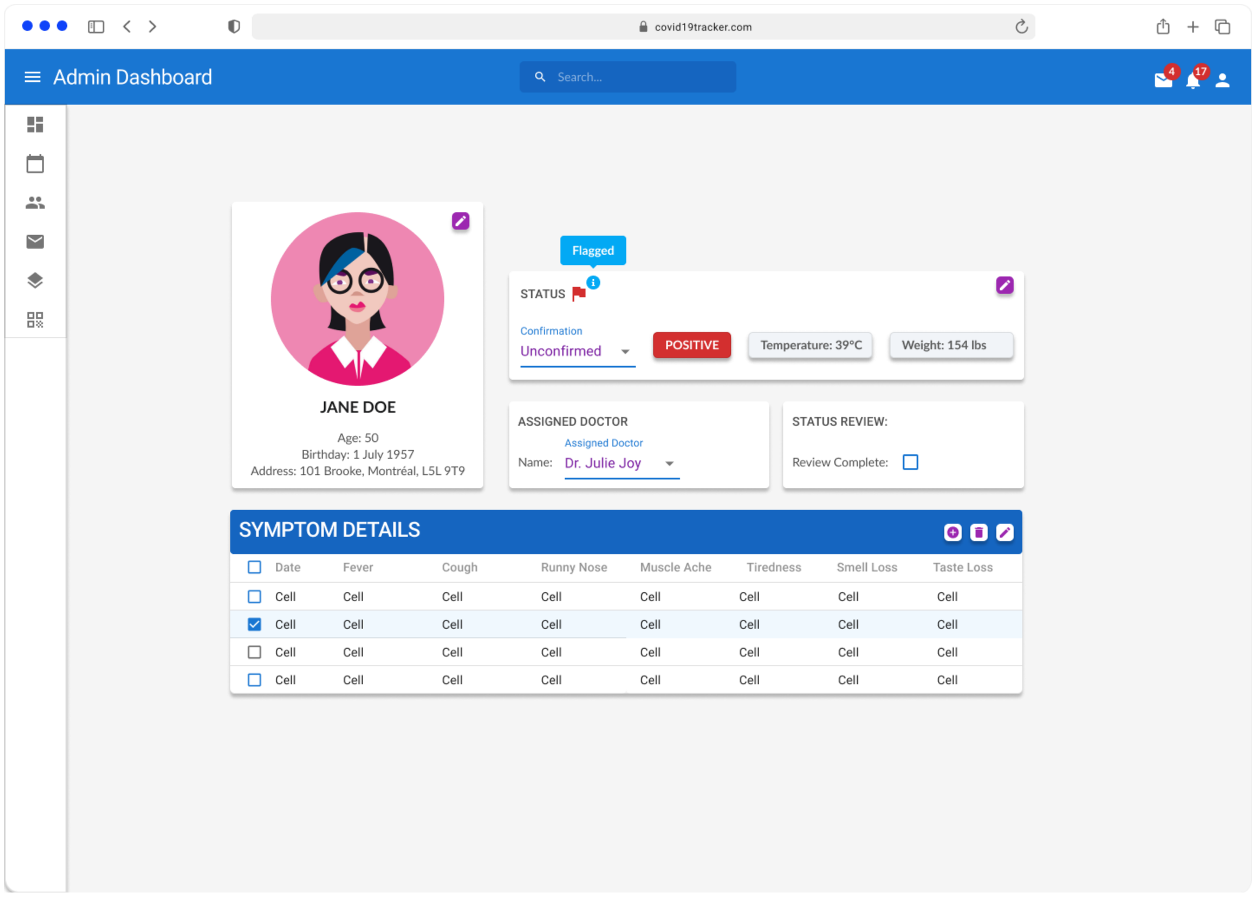


Figure 6: UI Admin Chat with patient for User Story Issue-13


covid19tracker.com

Admin Dashboard

Search...

JANE DOE

Age: 50
Birthday: 1 July 1957
Address: 101 Brooke, Montréal, L5L 9T9

Flagged

STATUS

Confirmation: Unconfirmed

POSITIVE

Temperature: 39°C

Weight: 154 lbs

ASSIGNED DOCTOR

Name: Assigned Doctor: Dr. Julie Joy

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 6: 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

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

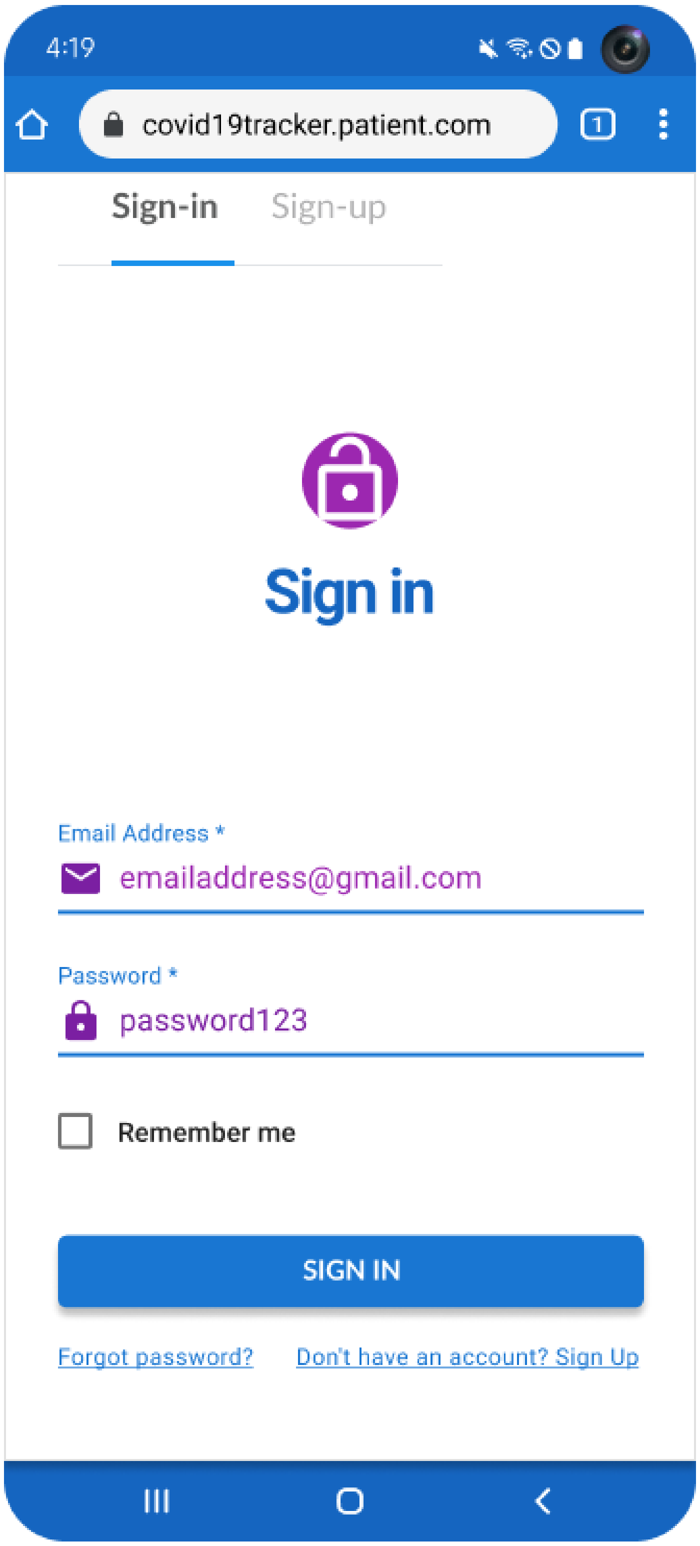


Figure 7: UI Client Sign-in for User Story Issue-7

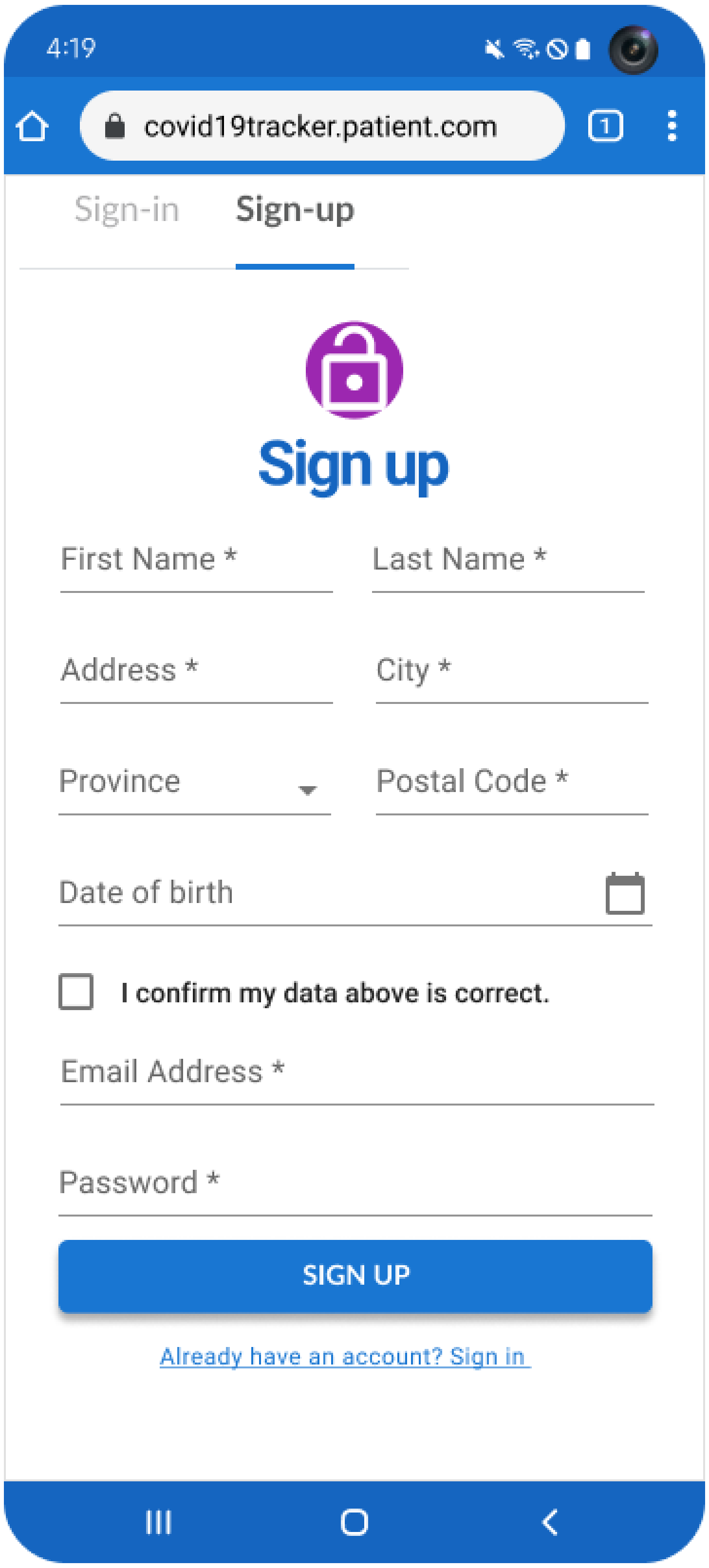


Figure 8: UI Client Sign-up for User Story Issue-7

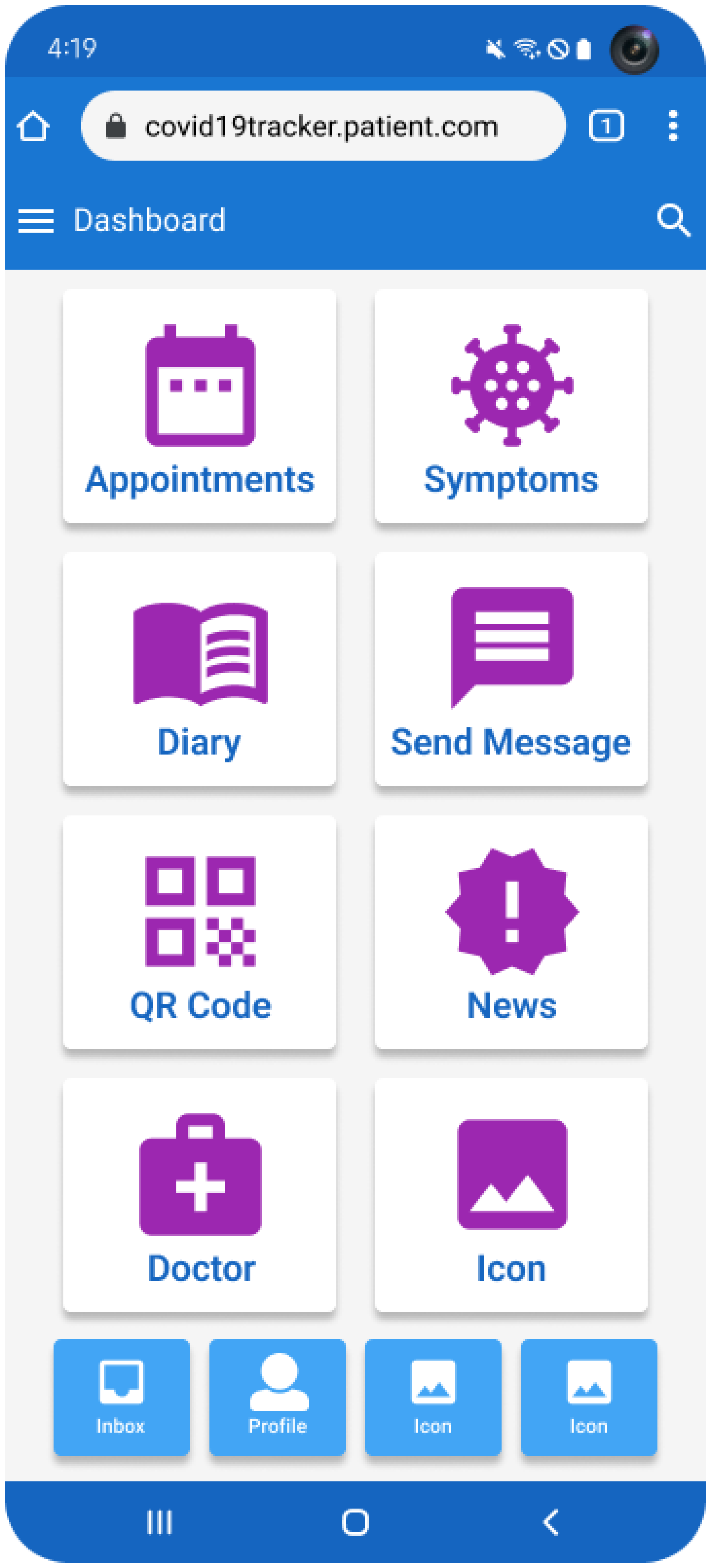
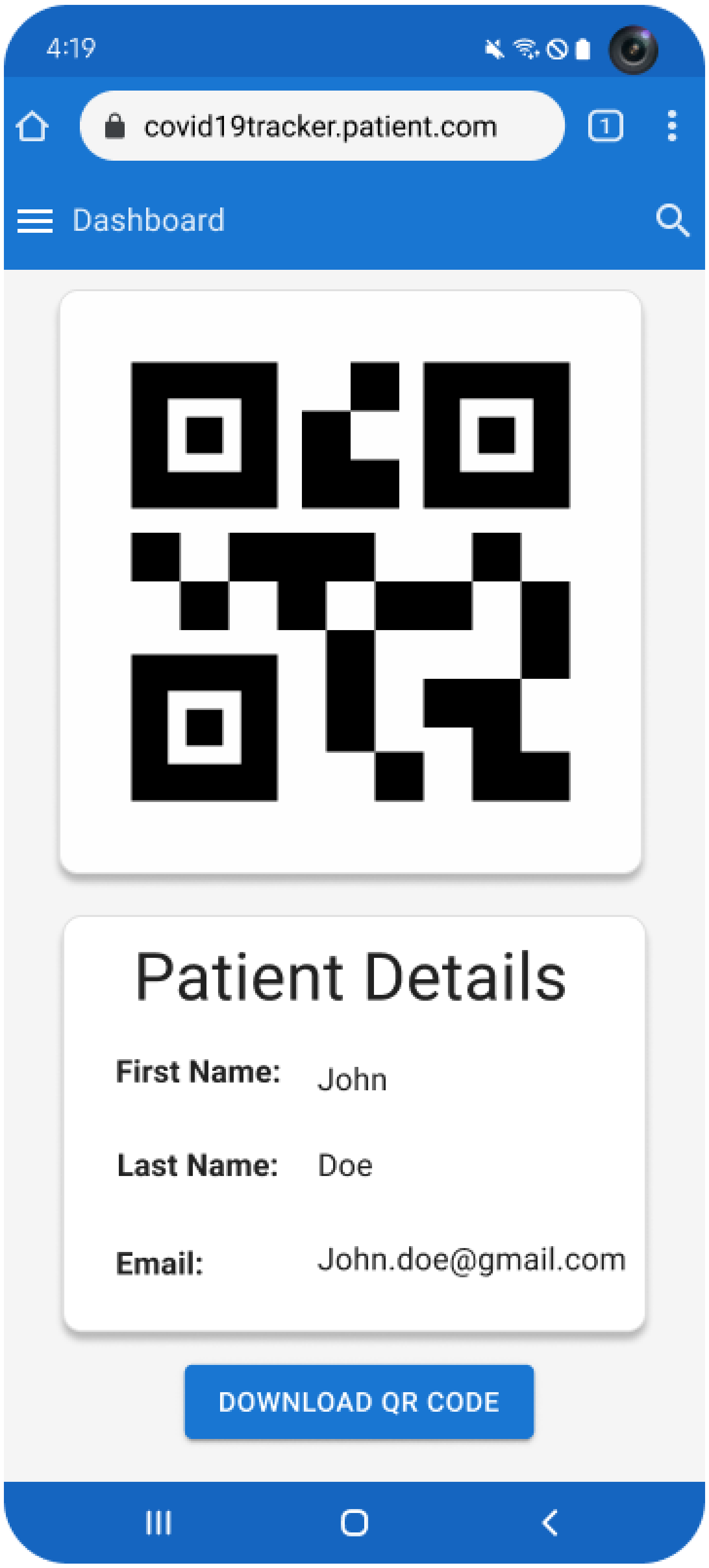


Figure 8: UI Client Dashboard



4:19

covid19tracker.patient.com

Dashboard

Patient Details

First Name:	John
Last Name:	Doe
Email:	John.doe@gmail.com

Figure 8: UI Client QR code for User Story Issue-22

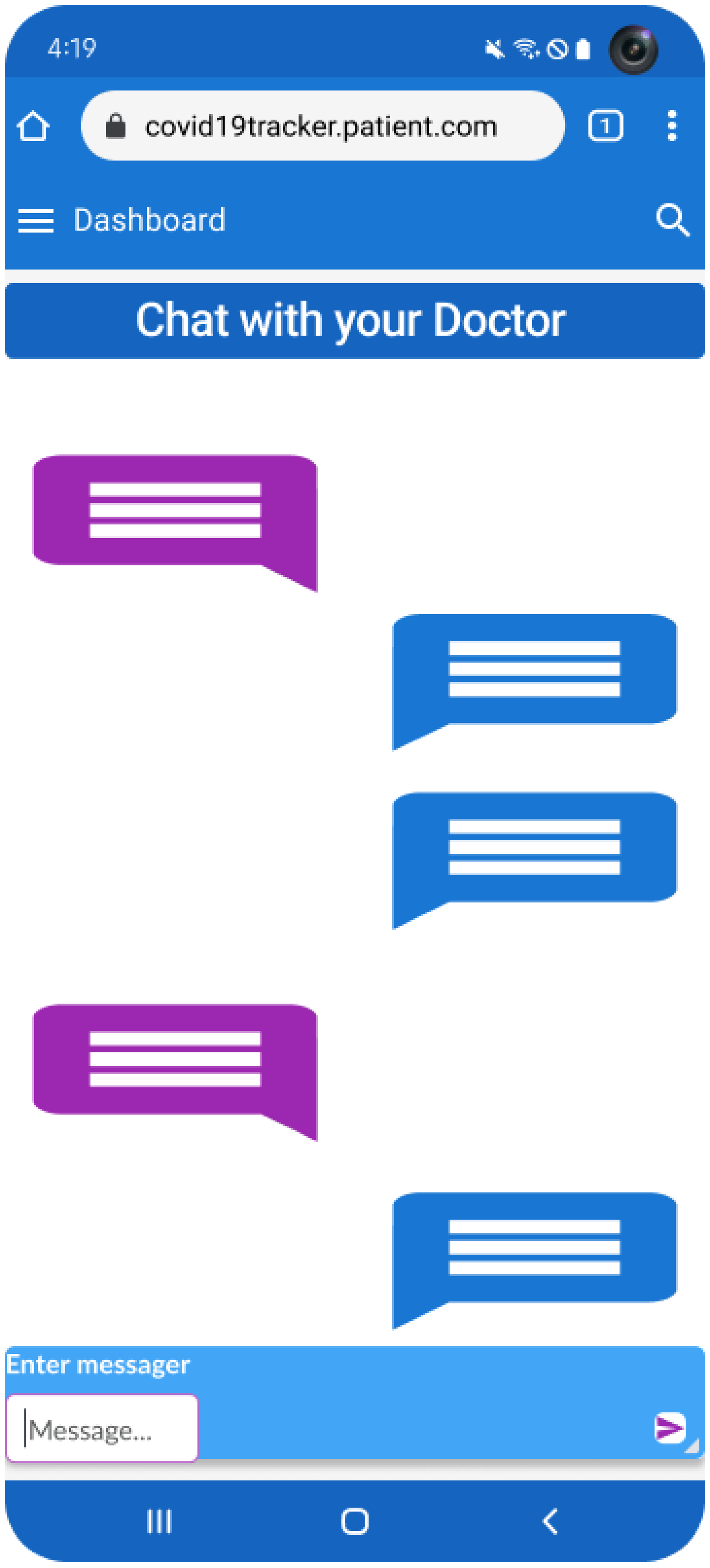


Figure 6: UI Client Chat with patient for User Story Issue-13

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.

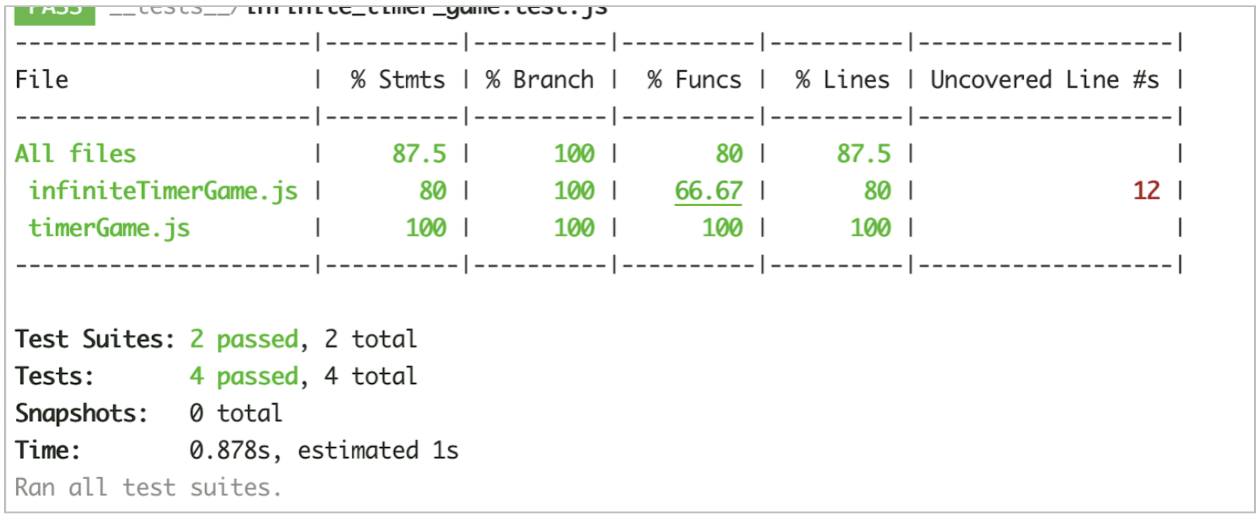
Jest's main focus is on the simplicity of the provided functions and good support for large web applications. It works with web applications using Babel, TypeScript, Node.js, React, Angular, Vue.js, Svelte, and more. We chose Jest since it is an already established framework in the JavaScript ecosystem. 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.



```
 PASS  __tests__/infinite_timer_game.test.js
----------------------|----------|----------|----------|----------|-------------------|
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 9: 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 user, I want to have a login / signup system so that users with different roles can log into their dashboard to view the appropriate contents**
Acceptance Criteria	Given that I am on the login page, and that I input my email address, and that I input my password, and that I clicked LOGIN, then I should see the dashboard.

Result	**PASS**
Comments	NONE

AT-2	**As a doctor, I want to have a dashboard view so that I can see an overview graph, my patients' status at a high level, and status updates.**
Acceptance Criteria	Given that I am logged in as a doctor, and that I click on the dashboard then I should be redirected to the dashboard which displays graphs and pertinent information.
Result	**PASS**
Comments	NONE

AT-3	**As a doctor, I want to see the list of details about a patient so that I can find out his/her status, temperature, weight, list of symptoms, etc.**
Acceptance Criteria	Given that I am logged in as a doctor, and that I am on the Patients Page and that I click on the name of a patient, then I should be redirected to the profile of that patient displaying a list of given details about the patient.
Result	**PASS**
Comments	NONE

AT-4	**As an admin or doctor, I want to see the list of patients with details (assigned doctor, upcoming appointment, etc) so that I can monitor each patient**
Acceptance Criteria	Given that I am logged in as an admin or doctor, and that I click on patients, then I should be redirected to the Patients Page which shows a list of patients with details such as assigned doctors and more.
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.