SOEN 390 Team 16

CovidTracker

Sprint 5 Report

Brandon Takli	40124336
Liam Burchell-Reyes	40131010
Matteo Gisondi	40121525
Peter Delis	40130063
Dimitrios Kirbizakis	40124471
Daniel Macicasan	40133711
David Seleznev	40132314
Andrei Grosuliac	40133614
Ineke Nguyen	40095491

Table of Contents

1. Introduction

CovidTracker is a responsive web-application for doctors, patients, health officials and immigration officers to view the status of relevant patients (confirmed or unconfirmed) with COVID-19. This project will include various high level, easy to read data on patients health that an assigned doctor can read and also allow convenient contact between doctors and patients.

The purpose of this project, as assigned by the professor of the SOEN 390 course at Concordia University, is to develop an application that allows doctors, health officials, patients and immigration officers to view the status of COVID-19 patients during this ongoing pandemic. The problem we are trying to resolve is the grand issue of doctors/health officials trying to properly and conveniently track the symptoms and progress of all their patients, as well as to see the status in the province as a whole. Patients themselves also struggle with contacting doctors/health officials, thus, this project should solve this issue and provide much needed help to the medical and patient community in this time of crisis.

This document for the second Sprint of development aims to inform the various stakeholders (users, PO & the professor) about the project and the work done thus far. Various important details such as risk management plans, product backlog, architectural design and testing will be available, allowing for detailed communication between the developers of the system and the stakeholders. The goal of this document is to communicate the plan and work done, so that any necessary changes can be performed earlier in the process in order to avoid issues later. This document is an iteration on the *Sprint 1* document, taking into account feedback and new changes.

2. Project Description

This project takes advantage of various industry standard tools and libraries such as Django, react and MUI, all of which will be mentioned in *Section 5* of this report.

This project will be developed using the *Agile* methodology for software development. *Agile* is an iterative development approach that focuses on delivering value to the customers sooner rather than later. This is done by performing work in smaller iterations called *Sprints*, and by the end of each sprint, a working prototype is required, thus showing the customer tangible value and allowing for more concrete views of the work done so far. This allows for a more responsive development cycle, where customer feedback is constantly taken into account and changes can be made earlier in the development cycle. This is the major difference between Agile and the traditional Waterfall model: Agile does not require full planning before any coding has been done, thus it allows for far more flexibility and modifications to the plan.

The sprint schedule will be as follows:

Sprint	Date
1	12/01/2022 - 2/02/2022
2	02/02/2022 - 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. Requirements

3.1 Product Backlog

Below is the product backlog, encompassing all the features to be delivered by the final sprint:

Story-ID	Epic-ID	Feature	Priority (MoSCoW)	Effort (USP)	Status
1	1	Login	M	3	**Done**
2	1	Sign Up	M	5	**Done**
3	2	Account Creation Request	M	5	**Done**
4	2	Assign Doctors to Patients	M	2	**Done**
5	3	Patient Status Update	M	3	**Done**
6	4	Monitor Status & Symptoms of Patients	M	5	**Done**
7	4	High Level Patient Dashboard	S	3	**Done**
8	3	Contact Doctor/Patient	S	3	**Done**
9	1	Receive Notifications	M	5	**Done**
10	4	Prioritize Patient	M	3	**Done**
11	1	Contact Tracing	M	5	**Done**

12	2	Report System	S	2	**Done**
13	4	Arrange Appointment	S	3	**Done**
14	1	Infected in Area	C	3	**Done**
15	3	Rapid Test Confirmation	C	2	**Done**
16	3	Request Help	S	3	**Done**
17	2	Doctor Emergency Reassign Patients	M	3	**Done**
18	2	API Interface to Other COVID-19 Applications	M	5	**Done**
19	2	QR Code for Individual Patient Records	M	5	**Done**
20	5	Immigration Officer Monitoring Immigrant Patients	M	3	**Done**
21	5	Immigration Officer Flagging Immigrant Patients	M	3	**Done**
Total				74	

3.2 Epics

Below are the *Epic* cards which encompass functionality that is divided into *User Stories* which will be elaborated on in *Section 3.3*. Each epic focuses on a category of user (or a general "User") and is divided into user stories to be done during each sprint. Each epic contains a list with the IDs of the user stories which it contains.

ID	1
Title	User Functionalities
Description	As a User, I want to be able to access information relevant to me and receive relevant notifications in a convenient and clear manner, so that I can help myself and others during this pandemic.
User Stories	1,2,9,11,14

ID	2
Title	Administrator Functionalities

Description	As an Administrator, I want to be able to manage and monitor contact between users so that I can ensure that all users have a role that is appropriate to them.
User Stories	3,4,12,17,18,19

ID	3
Title	Patient Functionalities
Description	As a Patient, I want to be able to access helpful services and communicate to my assigned Doctor by always updating my current status to them, so that I can understand more about if I am taking care of myself properly
User Stories	5,8,15,16

ID	4
Title	Doctor/Health Official Functionalities
Description	As a Doctor/Health Official, I want to be able to have full control of the data of all my Patients, so that I am able to monitor their situation, and am able to contact them at any time.
User Stories	6,7,10,13

ID	5
Title	Immigration Officer Functionalities
Description	As an Immigration Officer, I want to be able to monitor relevant information about immigrant users, so that I can take appropriate measures fitting of their status and the situation.
User Stories	20,21

3.3 User Stories

The following requirements were elicited from the product owner from the requirements document and have been turned into user stories approved by the product owner. **21 user stories** have been elicited for a total of **74 user story points**.

ID	1
Title	Login
Epic ID	1
Description	As a User, I want to be able to login, so that I can access my personal information.
Tasks	1. Allow the User to access features, personal information, and services for logging in. 2. Allow the User to save account data into the system and edit it.
Acceptance Criteria	A User is able to log in to the application by entering their account credentials (if they have created an account prior) which allows them access to many of the application's features and their personal home page.
MoSCoW	M
Business Value	XL
Risk	6
Effort	3

ID	2
Title	Sign Up
Epic ID	1
Description	As a User, I want to be able to create an account and specify my role in the application so that I can make use of the application.
Tasks	1. Allow the User to create an account by entering account information. 2. Allow the User to select their role. 3. If the role is not "Patient," the User may supply proof of their role and submit an account creation request. 4. If required, allow an Administrator to verify the proof provided by the User to finish the account creation process.
Acceptance Criteria	A User is able to submit their account credentials, which are automatically validated. The User may then select their role, and, if it is not "Patient," they must provide proof and wait for verification. An Administrator may accept non-"Patient" account creation requests.
MoSCoW	M

Business Value	XL
Risk	6
Effort	5

ID	3
Title	Account Creation Request
Epic ID	2
Description	As an Administrator, I want to be able to see, accept, and reject account creation requests for non-“Patient” roles so that each User may use the application according to their role.
Tasks	1. Have any non-“Patient” account creation send an account creation request notification to all Administrators. 2. Have all pending account creation requests visible to Administrators in a tab. 3. Allow an Administrator to view the account information and the attached proof of role for any account creation request. 4. Allow an Administrator to accept or reject any account creation request.
Acceptance Criteria	The attempted creation of a non-“Patient” account sends an account creation request as well as a notification to all Administrators. All pending account creation requests may be accepted or rejected by an Administrator. Accepted account creation requests results in the creation of that account, and that User may now log in.
MoSCoW	M
Business Value	XL
Risk	6
Effort	5

ID	4
Title	Assign Doctors to Patients
Epic ID	2
Description	As an Administrator, I want to be able to assign a Doctor to a Patient, so that communication and information sharing may be established between the two.
Tasks	1. Allow the Admin to view a list of patients without a Doctor 2. Allow the Admin to view a list of available Doctors 3. Allow the Admin to assign a Patient to a Doctor, notifying both Users
Acceptance Criteria	An Administrator is able to see a list of all available Doctors, as well as a list of Patients without a Doctor assigned to them. An Administrator is able to assign a Doctor to a Patient, with the Patient being able to see their assigned Doctor and the Doctor being able to see their newly assigned Patient
MoSCoW	M
Business Value	XL
Risk	4
Effort	2

ID	5
Title	Patient Status Update
Epic ID	3
Description	As a Patient, I want to be able to update my status within a predefined time frame, so that I can inform my Doctor of my condition.
Tasks	1. Allow the Doctor to see the list of details that the Patient must provide. 2. Allow the Patient to see the list, and be able to appropriately update their status. 3. Allow the Patient to modify their sent status update, in case of an error or change in condition during the day.
Acceptance Criteria	A Patient receives a list of details to fill out for their Doctor, and their responses to each field are sent to the Doctor, who can view them.

MoSCoW	M
Business Value	XL
Risk	3
Effort	3

ID	6
Title	Monitor Status & Symptoms of Patients
Epic ID	4
Description	As a Doctor/Health Official, I want to be able to monitor the status and symptoms of confirmed and unconfirmed COVID-19 patients, so that I can take appropriate measures to ensure their safety and the safety of others.
Tasks	1. Doctor should be able to see all their Patients on their home screen 2. Doctor should be able to see the list of Patient symptoms as well as their status (*Healthy, Infected, Recovered*) for each Patient 3. Each Patient should be able to view their status & symptoms that they reported and change them if need be.
Acceptance Criteria	The Doctor/Health Official is able to see the relevant status of the Patients (confirmed as having COVID-19 or not), allowing them to take action if need be.
MoSCoW	M
Business Value	XL
Risk	4
Effort	5

ID	7
Title	High Level Patient Dashboard
Epic ID	4
Description	As a Doctor, I want to be able to see an easily digestible visual summary of the status of my patients, so that I can quickly grasp the situation the patient is in.

Tasks	1. Doctor should be able to see a chart showing the number of currently infected Patients under his/her care 2. Doctor should be able to see a chart showing the number of recovered Patients under his/her care.
Acceptance Criteria	The Doctor/Health Official is able to view their Patient dashboard, which includes nicely formatted graphs and tables showing their status in a format more easily digestible at a glance.
MoSCoW	S
Business Value	L
Risk	3
Effort	3

ID	8
Title	Contact Doctor/Patient
Epic ID	3
Description	As a Patient, I want to be able to contact my Doctor within the application, so that I can communicate with them about my condition/other relevant topics.
Tasks	1. Allow a Patient to have easy access to communicate with their assigned doctor. 2. Allow a Patient and Doctor to communicate through a messaging system. 3. Allow a Doctor to show status (Available, Busy, Etc.) to avoid overburdening Doctor through messages.
Acceptance Criteria	A Patient contacts their Doctor through the application and is able to message them via email. The Doctor will receive an email from their patient and can respond appropriately.
MoSCoW	S
Business Value	L
Risk	4
Effort	3

ID	9
Title	Receive Notifications
Epic ID	1
Description	As a User, I want to be able to see notifications pertaining to me so that I may act upon them.
Tasks	1. Allow the User to see a notification tab listing all their notifications. 2. Allow non-“Patient” Users to see prioritized notifications in a separate tab. 3. Allow the User to see the amount of pending notifications. 4. Show unviewed notifications as bolded. 5. Allow the User to dismiss notifications.
Acceptance Criteria	A User receives a notification, which increases the number of pending notifications next to the notification tab. Upon opening the tab, the User sees all pending notifications, which the User may dismiss.
MoSCoW	M
Business Value	XL
Risk	5
Effort	5

ID	10
Title	Prioritize Patient
Epic ID	4
Description	As a Doctor, I want to be able to prioritize any of my Patients so that I can see their updates more efficiently.
Tasks	1. Allow the Doctor to toggle a flag for each Patient. 2. List the notifications for flagged Patients on a separate list above their assigned Doctor’s standard notification tab.
Acceptance Criteria	A Doctor flags a set of their Patients. Any notification sent pertaining to any of those Patients is listed in the Important Notifications tab.
MoSCoW	M

Business Value	M
Risk	3
Effort	3

ID	11
Title	Contact Tracing
Description	As a User, I want to be informed of any contact I might have had with someone who is infected. Implemented in a log of frequently visited locations. This database can be used to inform others if someone else who visits those locations has tested positive recently.
Tasks	1. Allow a User to input all their frequently-visited addresses. 2. When a User is diagnosed with COVID-19, all Users who share addresses receive a notification which also pings their assigned Doctor.
Acceptance Criteria	A User will be notified if another User who frequents their listed frequent locations is declared as infected. The User's Doctor (if applicable) will be notified as well.
MoSCoW	M
Business Value	L
Risk	4
Effort	5

ID	12
Title	Report System
Epic ID	2
Description	As an Administrator, I want to resolve all conflicts between users and ensure that all Users are respecting the Terms and Services
Tasks	1. Allow a User to report another User and explain why they are reporting to the Administration. 2. Allow Admin to restrict a User's privileges on the application.

Acceptance Criteria	An Administrator gets notified of a User reporting someone else and their reason they are reporting. The Administrator can access their conversation and can decide any further action that is necessary to either User. The Administrator will then store the report and the actions taken into the system's database.
MoSCoW	S
Business Value	M
Risk	4
Effort	2

ID	13
Title	Arrange Appointment
Epic ID	4
Description	As a Doctor, I would like to be able to schedule a time with my patient to speak to them.
Tasks	1. Allow the Doctor to send availability to Patient. 2. Allow the Doctor to book an appointment with the Patient within the application
Acceptance Criteria	A Doctor asks their Patient if they could book an appointment, where they decide when they could do this. Once decided, the Doctor can register in the system an appointment between the two at some time.
MoSCoW	S
Business Value	L
Risk	2
Effort	3

ID	14
Title	Infected in Area

Epic ID	1
Description	As a User, I want to check if the area that I am currently in contains many people who tested positive or not to ensure my safety.
Tasks	1. Provide users with a map showing the number of infected individuals in their area. 2. Ensure that the reporting is not too granular, in order to avoid singling people out.
Acceptance Criteria	All users are able to see a map showing the number of infected people in an area in order to take proper precautions when interacting with others in the neighbourhood.
MoSCoW	C
Business Value	L
Risk	5
Effort	3

ID	15
Title	Rapid Test Confirmation
Epic ID	3
Description	As a Patient, I would like to confirm with my Doctor how likely it is that I have COVID-19 after doing the rapid test.
Tasks	1. A Patient can send their Doctor their rapid test results (with a picture if requested) 2. The Doctor can see the results of the rapid test and inform the patient of next steps to take.
Acceptance Criteria	A Patient is able to send their Doctor their rapid test results, and the Doctor is able to see those results in order to decide the next steps to take with that Patient.
MoSCoW	C
Business Value	M
Risk	6
Effort	2

ID	16
Title	Request Help
Epic ID	3
Description	As a Patient who tested positive for COVID-19, I would like the ability to notify my Doctor of potential emergencies and I would like some help getting food since I cannot go shopping myself.
Tasks	1. Allow sick Patient to signal Doctor of emergencies. 2. Allow sick Patient to have access to resources on what to do and delivery services.
Acceptance Criteria	When a Patient who has tested positive requests help, they will be able to contact their Doctor, who can see the emergency status of their communication. When the Patient sets themselves as having tested positive, they will be shown resources and tips to help them during their time in quarantine (such as when it is safe to go out again, places to order food from, emergency numbers, etc.).
MoSCoW	S
Business Value	L
Risk	5
Effort	3

ID	17
Title	Doctor Emergency Reassign Patients
Epic ID	2
Description	As an Administrator, I want to be able to see if a Doctor has declared an emergency, so that I can reassign their Patients temporarily to another Doctor.
Tasks	1. A Doctor can declare an emergency they need to attend to as well as an expected timeframe. 2. An Administrator is notified that a specific Doctor has an emergency and is given the timeframe for that emergency. 3. An Administrator can temporarily assign the Patients of the Doctor with the emergency to another Doctor until the emergency is over.

Acceptance Criteria	A Doctor can declare an emergency and describe the expected duration. All Patients assigned to the Doctor are temporarily reassigned to other Doctors by an Administrator. When the Doctor returns, they are able to reassign all their previous Patients back to themself.
MoSCoW	M
Business Value	L
Risk	3
Effort	3

ID	18
Title	API Interface to Other COVID-19 Applications
Epic ID	2
Description	As a User, I want to have an interface to other COVID-19 applications, so that I can have more useful data and resources at my disposal.
Tasks	1. Integrate other relevant COVID-19 applications with an available API interface into CovidTracker. 2. View relevant information via the interface that are properly cited.
Acceptance Criteria	Users have access to information/tools from different COVID-19 applications and thus do not need to switch between multiple applications.
MoSCoW	M
Business Value	L
Risk	6
Effort	5

ID	19
Title	QR Code for Individual Patient Records
Epic ID	2

Description	As a Patient, I want to be able to generate a QR code of my record containing relevant information, so that relevant officials can easily see my information at a glance without revealing all personal details.
Tasks	1. Allow Patient to generate a QR code showing current status, their Doctor and previous status changes 2. Provide a nice format for viewing the information provided by the QR code.
Acceptance Criteria	A relevant official is able to scan the QR code and view all the relevant information about that Patient. The information is presented nicely and clearly.
MoSCoW	M
Business Value	L
Risk	3
Effort	5

ID	20
Title	Immigration Officer Monitoring Immigrant Patients
Epic ID	5
Description	As an Immigration Officer, I want to be able to monitor Immigrant Patients under my jurisdiction, so that I can take appropriate measures based on their status.
Tasks	1. A Patient must declare if they are an immigrant or permanent resident. 2. An Immigration Officer may see a list of immigrant Patients and their status 3. An Immigration Officer may be notified of changes in the status of immigrant Patients
Acceptance Criteria	Only Patients who are listed as immigrants are visible to Immigration Officers, who can see their status and will be notified of changes to their status.
MoSCoW	M
Business Value	L
Risk	6
Effort	3

ID	21
Title	Immigration Officer Flagging Immigrant Patients
Epic ID	5
Description	As an Immigration Officer, I want to be able to flag certain Immigrant Patients, so that I can monitor them more closely.
Tasks	1. Allow Immigration Officer to flag specific Immigrant Patients so that they can be monitored more closely.
Acceptance Criteria	An Immigration Officer is able to flag an Immigrant Patient if they would like to pay special attention to changes in their status. Thus receiving a priority in terms of notifications of status changes.
MoSCoW	M
Business Value	L
Risk	3
Effort	3

4. Release Planning

4.1 Sprint 1 Summary

This sprint focused on planning and clarifying the vision of the project. In terms of development, the Login feature was finished and the Sign Up feature was started ahead of schedule. Due to this being the initial sprint, the major difficulty was the task division phase, as the team does not have much experience working with each other as of yet. In response, a task division document was made and, for the first sprint, team members signed themselves up for several tasks. The Sign Up story was brought forward as it was deemed feasible to begin during the first sprint.

Story ID	Task Name	Responsible	USP	Duration	Status
1	Login	Peter Delis	3	4	**Complete**
2	Sign Up*	Daniel Macicasan	5	6	**In Progress**
Total	**Sprint 1**		**3**	**22**	**Complete**

* denotes a user story added to the sprint.
Project velocity after 1 sprint: 3

4.1.1 Sprint 1 Retrospective

Keep doing:

- Weekly meetings
- Working in sub-teams and communication within and between them
- Peer reviewing code
- Status updates on work done
- Descriptive git commits

Start doing:

- Require reviewers for Git pull requests
- Use Git tags for pull requests (enhancement, bug fix, etc.)
- Using tags on Jira to differentiate between tag types (programming, documentation, etc).

Stop doing:

- Procrastinating
- Overwhelming one's self

Do more of:

- Updating JIRA tasks.
- Taking meeting notes.

Do less of:

- Taking long breaks
- Starting work session late

4.2 Sprint 2 Summary

This sprint focused on delivering the core functionality of the application. Accounts under different roles will be able to be created and approved, patient statuses may be updated and monitored, patients may view their dashboard, and communications may be made between doctor and patient. A lot of work has been done setting up the backend in preparation for future sprints and expandability. Many prototypes have been done on the frontend as well, helping the team gain a shared vision of how the features will take shape in the coming sprints. Despite an unprecedented amount of work from other courses, progress was good and we managed to cover the needed features for this story-packed sprint.

Story ID	Task Name	Responsible	USP	Duration	Status
2	Sign Up	Daniel Macicasan	5	6	**Complete**
3	Account Creation Request	Peter Delis	5	9	**Complete**
4	Patient Status Update	Andrei Grosuliac	3	4	**Complete**
5	Monitor Status & Symptoms of Patients	Andrei Grosuliac	5	8	**Complete**
6	High Level Patient Dashboard	Daniel Macicasan	3	4	**Complete**
7	Contact Doctor/Patient	Matteo Gisondi	5	6	**Complete**
Total	**Sprint 2**		**26**	**20**	

Sprint velocity: 26

Story points completed after 2 sprints: 29

4.2.1 Sprint 2 Retrospective

Keep doing:

- Meet multiple times a week.
- Open dialogue and feedback.

Start doing:

- Create a Discord chat for daily tasks/updates for the whole team.

Stop doing:

- Over/underestimating the workload.
- Being afraid to ask for help.

Do more of:

- Update everyone on tasks done in detail.
- Knowledge sharing sessions: If you know a lot about something, teach others!
- Update the Jira as you're doing tasks.
- Read over the report document periodically.

Do less of:

- Working late.
- Leaving testing to the last minute.

Summary:

This sprint was very difficult on the team due to an unprecedented amount of assignments and tests due in other classes. Furthermore the story point load for this sprint was disproportionately large compared to the planned story point load for the other sprints. We took the feedback into account based on our feedback from *Sprint 1* in order to revise and improve our report and demo. Despite the hurdles, we managed to make a great deal of progress and are still on track for an unhindered *Sprint 3*.

Team Member	**Work Done**
Brandon Takli	- Worked on reformatting report based on Sprint 1 feedback. - Prepared presentation for the demo - Verified work done and checked against requirements - Verified PRs, minor code changes/updates.
Liam Burchell-Reyes	- Translated User Stories into programming tasks on JIRA - Prepare the Backlog for the following Sprint - Divided JIRA tasks amongst members to evenly split workload - Assisted with the documentation
Matteo Gisondi	- Admin Page frontend - Patient/Doctor contact form - Frontend Tests and coverage report - Verified PRs

Peter Delis	- Designed scalable Custom User Models - Implemented User, Patient and Doctor views, serializers and admin filtering - Introduced Account Creation Request, Verification, and Confirmation - Added Registration, Login and Logout API calls - Registration, Login and Logout Unit Tests for all User types - Made Redux actions for scalable and maintainable API consumption
Dimitrios Kirbizakis	- Assisted with SAD. - Designed UI Prototype/Wireframe for sprint 3 features. - Worked on Front-End files for the website.
Daniel Macicasan	- Designed UI Prototype/Wireframe for sprint 3 features. - Designed the main layout of the frontend. - Participated in the Patient/Doctor contact form. - Pre-login redirecting depending on user type. - Added frontend High level Patient Dashboard.
David Seleznev	- Updated Domain Model - Updated class Diagram - High level Patient Dashboard front & back end
Andrei Grosuliac	- Created the Patient Status Page in the frontend with the status and symptoms for a patient and the ability to modify it - Added the Status model in the Backend along with multiple views to retrieve the statuses and latest status of any patient - Added the ability for each doctor to see the status of any of his patients
Ineke Nguyen	- Frontend for Doctor’s dashboard that displays the list of patients assigned specifically to them, a bar chart detailing an updated number of patients infected, recovered and healthy. - Updated risk analysis

4.3 Sprint 3 Summary

This sprint aimed to complete all primary functionality of the application. While after Sprint 2, the application ran sufficiently, it lacked many useful features that make it relevant compared to the alternatives. A superior notification system, patient prioritization, and the ability for doctors to arrange for appointments with their patients, as implemented in Sprint 3, allow for more interactions through the application. Contact tracing and being able to view the number of infected individuals in one's area allows users to be proactive in maintaining their health. Moderation was implemented with the report system. Unfortunately we were unable to deploy the website this sprint and make it publicly accessible, but we plan to do this as soon as possible in the next sprint.

Story ID	Task Name	Responsible	USP	Duration	Status
9	Receive Notifications	Dimitri Kirbizakis	5	7	**Complete**
10	Prioritize Patient	Ineke Nguyen	3	6	**Complete**
11	Contact Tracing	Andrei Grosuliac	5	8	**Complete**
12	Report System	Peter Delis	2	5	**Complete**
13	Arrange Appointment	Ineke Nguyen	3	7	**Complete**
14	Infected in Area	Daniel Macicasan	3	7	**Complete**
Total	**Sprint 3**		**21**	**20**	

Sprint velocity: 21

Story points completed after 3 sprints: 50

4.3.1 Sprint 3 Retrospective

Keep doing:

- Constant communication between team members/subteams
- Individual work updates via messaging
- Peer review
- Weekly/frequent meetings

Start doing:

- Commit early
- Frequent commits

Stop doing:

- Procrastination

Do more of:

- Testing frontend/backend
- System tests for a PR
- Check the Slack

Do less of:

- Gold plating

Summary:

This sprint was more difficult than anticipated: We each had other obligations during the break and thus lost a good chunk of time for this sprint. Furthermore other assignments came up for different courses as well, which slowed us down and diverted our focus. However we are still proud of what we did this sprint and managed to make good progress, keeping us on the right track to finishing on time. We wish we could have deployed the website for this sprint, but were focused on getting the features done, so this is our first priority for the next sprint.

Team Member	**Work Done**
Brandon Takli	- Verified PRs to make sure features worked on conform with requirements - Verified testing progress and examined areas to improve - Prepared presentation - Worked on updating documentation - Wrote basic tests
Liam Burchell-Reyes	- Continued to translate User Stories into programming tasks on JIRA - Prepared the Backlog for the following Sprint - Divided JIRA tasks amongst members to evenly split workload - Assisted with updating the documentation

Matteo Gisondi	- Deployment - Frontend Tests and coverage report - Verified PRs
Peter Delis	- Redux pattern for Notifications and Status. - Refactor Header, Navigator, ContactForm, and Page routing. - Development revision of overall work and best practices - Unit Testing for Status and Patient revtrieval API
Dimitrios Kirbizakis	- Designed UI Prototype/Wireframe for sprint 4 features. - Designed the frontend and backend for the notification tab. - Verified PRs to make sure features worked on conform with requirements
Daniel Macicasan	- Verified PRs to make sure features worked on conform with requirements - Designed UI Prototype/Wireframe for sprint 4 features. - Designed the Infected in Area page by implementing a map API. - Designed the frontend for the notification tab. - Created address table for patients in the frontend.
David Seleznev	- Modeled the appointment booking with a sequence diagram - Modeled the sequence diagram for the contact tracing - Assisted in the client notification design - Modeled the base Use Cases of the system - Assisted with Updating documentation
Andrei Grosuliac	- Verified PRs to make sure features worked on conform with requirements - Created the Address model in the Backend with the necessary api calls as well as the Redux calls for the frontend - Created a complex query to be able to send notifs of infected alert to any patients that have matching addresses

	with the infected person
Ineke Nguyen	- DoctorDashboard showcasing patient prioritization toggle page and appointment page - Created priority toggle switch/page for patients assigned to the doctor - Created the appointment booking for patients. Doctor availability is shown on selected weeks. Patients are unable to select restricted slots that are considered unavailable, can only select one time slot, and can deselect choice to pick another slot.

4.4 Sprint 4 Summary

Sprint 4 focused on more features for the application. While these features are not necessarily as critical as the ones introduced in the first 3 sprints, they incorporate the Immigration Officer role for the first time. This is significant as it now presents a whole new set of permissions and views that are required and will be built off of the existing backend/frontend work already done. New information will need to be stored for patients (if they are immigrants or not) and an Immigration Officer must present their credentials, similar to our case for Doctors. Furthermore, an API to other COVID-19 applications is a significant addition, providing Users with more data from other sources and allowing us to make our application far more useful and connected with existing sources.

QR codes also provide an interesting, mobile-friendly way of sharing patient records with relevant parties, and should prove to be a simple and secure way of sharing relevant patient information.

Story ID	Task Name	Responsible	USP	Duration	Status
15	Rapid Test Confirmation	Daniel Macicasan	2	5	**Complete**
16	Request Help	Ineke Nguyen	3	7	**Complete**
17	Doctor Emergency Reassign Patients	Andrei Grosuliac	3	5	**Complete**
18	API Interface to Other COVID-19 Applications	Ineke Nguyen	5	9	**Complete**
19	QR Code for Individual Patient Records	Matteo Gisondi	5	8	**Complete**

20	Immigration Officer Monitoring Immigrant Patients	Dimitri Kirbizakis	3	8	**Complete**
21	Immigration Officer Flagging Immigrant Patients	Dimitri Kirbizakis	3	7	**Complete**
Total	**Sprint 4**		**24**	**20**	

Sprint velocity: 24

Story points completed after 4 sprints: 76

4.4.1 Sprint 4 Retrospective

Keep doing:

- Frequent team meetings
- Consulting other team members who specialized in specific parts of the project

Start doing:

- Ensuring everything is finalized
- Ensuring the team is aware of all facets of the project

Stop doing:

- Procrastination

Do more of:

- Constant communication
- Updating the rest of the team on one's work

Do less of:

- Merging all pull requests at the last minute to avoid merge conflicts

Summary:

This sprint involved some larger features such as QR code generation and doctor emergency reassignment, thus special attention needed to be paid to time management and feature correctness. Many of the features related to the *Immigration Officer* were similar to features for the *Doctor* and thus

reuse of existing code was possible. For *Sprint 5* we plan on refactoring and improving test coverage in order to finalize our work.

Team Member	**Work Done**
Brandon Takli	- Worked on general documentation (update diagrams, risk analysis, etc.) - Prepared presentation - Verified work being done against promised features in User Stories - Monitored testing and updated coverage
Liam Burchell-Reyes	- Expanded upon acceptance test format and updated existing tests. - Assisted updating the documentation. - Clarified design with members to ease the implementation process.
Matteo Gisondi	- Deployment - QR Code - Frontend Tests and coverage report - Verified PRs
Peter Delis	- Backend testing - Verifying PRs - Redux - Refactoring code for legibility and better design
Dimitrios Kirbizakis	- Added option for patients to confirm if they are an immigrant. - Implemented more functionality for immigration officer.
Daniel Macicasan	- Added the Rapid Test page and designed a form to upload rapid test pictures and send them to a doctor - Configured a new EmailJS service that allows for attachments when sending emails
David Seleznev	- Implementing Patient QR code front end page - Added Patient information frontend page for QR code link
Andrei Grosuliac	- Verified PRs to make sure features worked on conform with requirements

	- Emergency absence for doctor and reassigning patients both frontend and backend - Created necessary automatic notifications for doctor change
Ineke Nguyen	- Request Help feature for patient page consisting of an emergency alert button to notify/email doctor, grocery delivery options, sources for additional information on covid protocols - Covid API

4.5 Sprint 5 Summary

Sprint 5 was the final sprint of our project and was very short in duration. All story points were completed as of the end of *Sprint 4* and thus we focused on refining and testing for this sprint. The plan was to increase test coverage and refactor where possible. Furthermore, we focused on finalizing the documentation for this project as well in order to best represent all work done on the system. In terms of team responsibilities, we worked on having all team members become familiar with every facet of the project to at least an acceptable degree.

4.5.1 Sprint 5 Retrospective

Keep doing:

- Communication between teammates
- Verifying documentation

Start doing:

- Understand the whole project and work done instead of just the part assigned to you

Stop doing:

- Procrastinating

Do more of:

- Ask for help when you need it
- Volunteer to help when you have the time

Do less of:

- Waiting until the last minute

Summary:

This final sprint did not involve any new features, but was a refinement of existing features and focused on refactoring and testing. This sprint marks the end of the project and a bigger focus on finalizing the documentation was done for this sprint.

Team Member	**Work Done**
Brandon Takli	- Adding to and checking over the documentation - Presentation - Monitoring test coverage - Added several unit tests
Liam Burchell-Reyes	- Checking over and assisting with the documentation
Matteo Gisondi	- Cleaned up backend code base - Helped About page + T&C page - Updated backend metrics - Refactored frontend tests - Deployed system
Peter Delis	- Completed Notification Management (read notif, see how many there are, order by most recent) - Allow Patient Appointment Bookings - Patient and Doctor availability viewing and updating - Email Verification when registering any user account - Doctor is Away page redirect - Doctor reassign patient API data passing - Backend Testing - Dynamic Emailing for all forms - Updated the EmailJs Templates - Settings Page - About Page - Terms and Conditions Page - Profile Page - Dynamic Alert Messages and Notifications - Refactoring
Dimitrios Kirbizakis	- Reviewed frontend and backend code to make sure it respects the requirements

	- Reviewed documentation
Daniel Macicasan	- Reviewed frontend code to make sure it respects the requirements - Reviewed PRs on Github and solved some merge conflicts - Tested final version of the application to verify everything works as intended - Reviewed/added documentation in Sprint 5 report
David Seleznev	- Updating Class Diagram to better fit final product - Updating ER Diagram to better fit final product - Updating ER Diagram Documentation
Andrei Grosuliac	- Reviewed frontend and backend code to make sure it respects the requirements - Reviewed PRs on Github and solved some merge conflicts - Assisted with the documentation
Ineke Nguyen	- Added the api page for Canada's covid statistics for relevancy

4.6 Summary of Project (All 5 Sprints)

Over the past 5 sprints, we accomplished 21 user stories for a total of 76 story points. We managed to finish all necessary items on time as we had initially scheduled despite instances where we were caught up with other obligations (work from other courses, personal obligations, etc.). However, in the end we managed to consistently deliver the features we promised. As time progressed, we became more comfortable with the workflow and, with feedback from the PO, managed to improve and refine our reports and deliver better presentations and demos. Furthermore, we also gained valuable experience working in a large team following formal methodologies, which is something we experienced briefly in other courses, but not to this extent.

We employed many new methods and technologies that not many of us had not used prior to this project (GitHub Actions, unittest, Django framework, etc.) and thus learned quickly how to adapt and use these tools to better serve our project and team.

One consistent difficulty we had with the project was integration of components: Everyone writes their frontend code slightly differently or with different assumptions, thus integrating this code with the backend proved a difficult endeavor that required modification on both sides. However in the

end everything was integrated as it should be and improvements were made in this respect as the sprints progressed.

We consider this project to be a great success and are proud of the work we accomplished. This project has helped prepare us for the format of the final capstone project and has given us valuable experience working with the latest development tools and languages. Furthermore, we learned to adapt to properly communicate with the PO and take advice to improve our project and steer it in a direction that is appropriate for the PO.

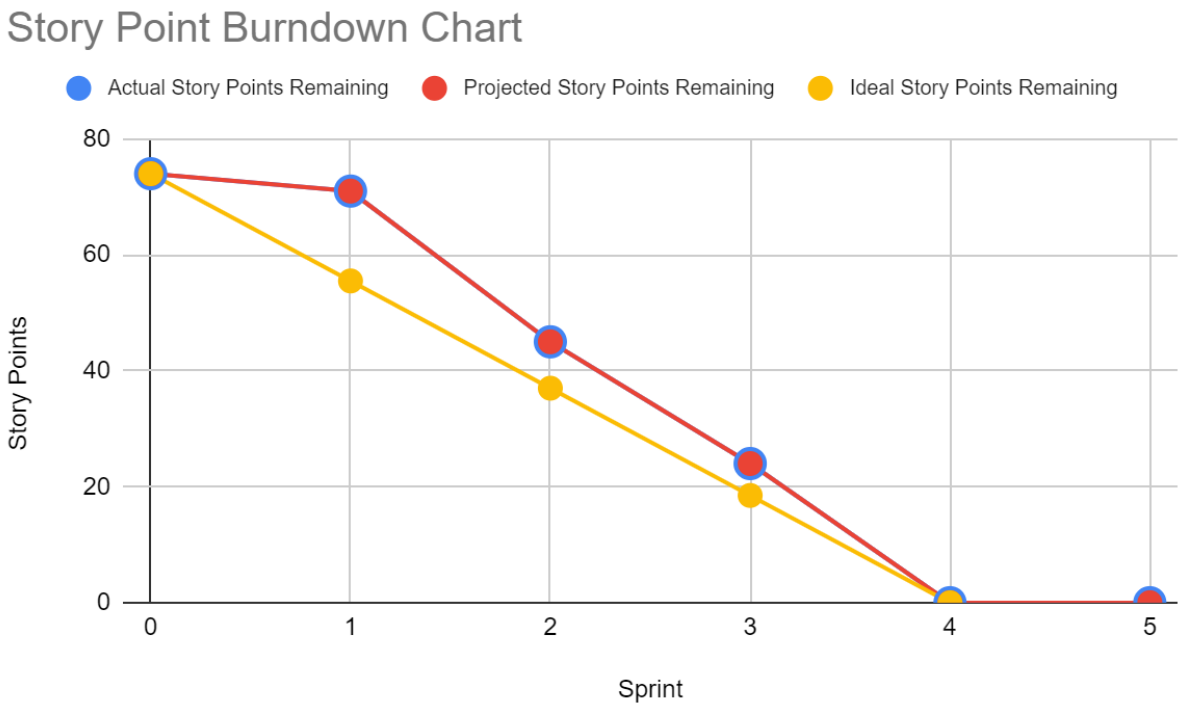


Figure 4.6.1: Story Point Burndown Chart

4.6.1 Project Retrospective

What went wrong:

- Interruptions from other classes (assignments, tests, other projects, etc.) slowed down progress and diverted focus
- Occasional miscommunications (particularly near the beginning of the project) resulted in some difficulties creating features as promised
- Lack of proper presentation during Sprint 1

What went right:

- Properly set up CI pipeline for testing since the beginning of the project with GitHub Actions
- Frequent team meetings to resolve miscommunications

- Adaptation of documentation and presentation based on PO feedback
- Organization of teammates into sub-teams based on desired roles and skills

Takeaway for future projects:

- Work early to avoid unforeseen issues
- Avoid procrastination
- Break off work into sub-sections based on expertise (backend, frontend, documentation, testing, etc.)
- Communicate constantly with the PO and teammates to ensure that the right product is being made
- Make sure to share information amongst all teammates so everyone is on the same page and has at least a basic understanding of the whole project, so that they can answer questions if need be

Summary:

This project served as a great tool to help us learn how to work in larger teams and follow Software Engineering protocols and work with the SCRUM/AGILE methodology. Sprint 1 helped us learn what was expected of us and allowed us to improve in subsequent sprints. This allowed us to deliver a better product with each sprint and helped bring us closer to the expectations set forth by the PO. Furthermore, as time went on we learned to manage work from other classes and plan ahead to avoid scheduling conflicts. Improvements in communication and documentation also helped ensure that development went smoothly and that the features were delivered on time.

5. Architecture

5.1 Use Case Diagram

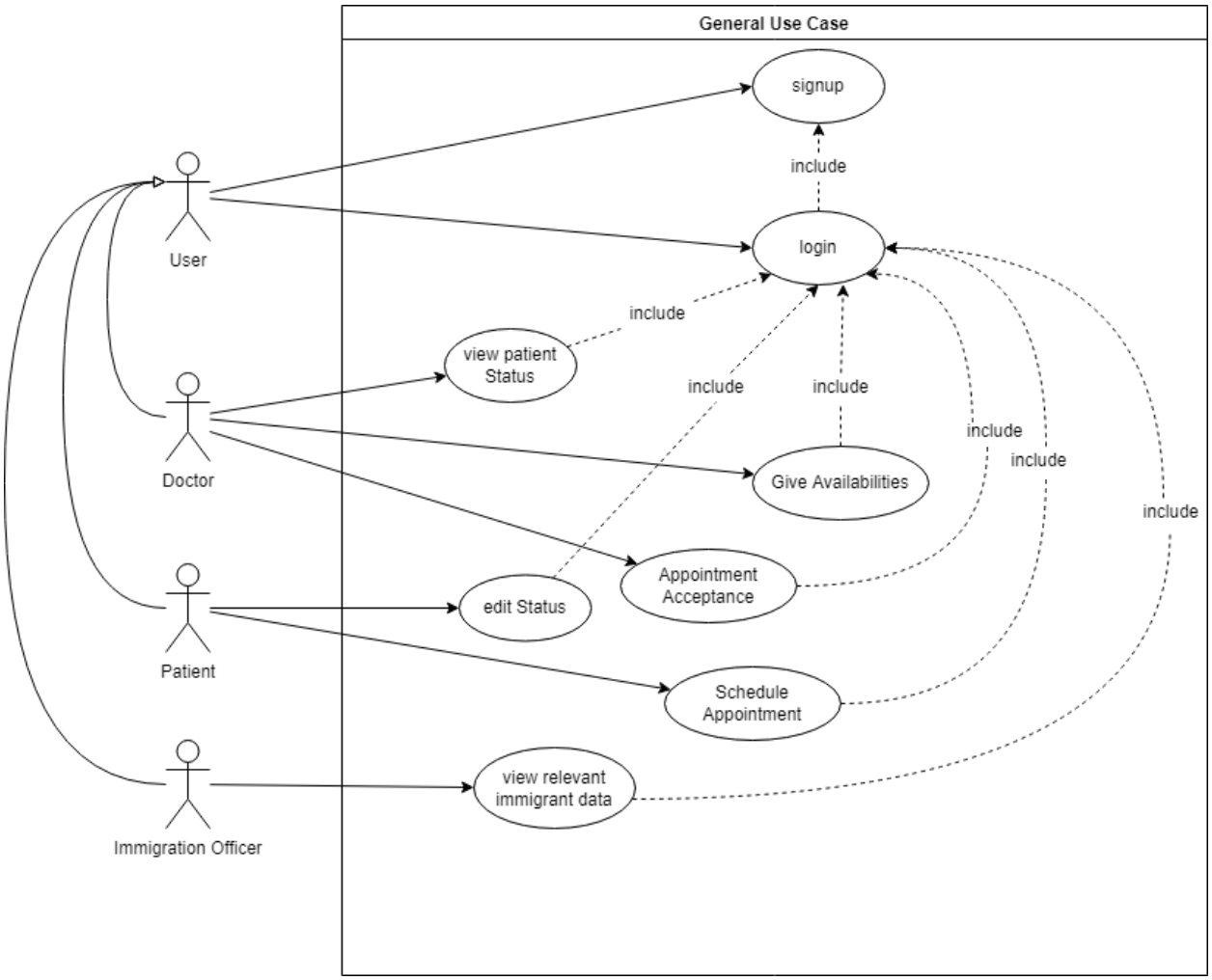


Figure 5.1.1: General Use Case Diagram

Figure 5.1.1 depicts the general use cases present in the system. All users must sign up to be able to login. After which, all other functionalities of the system are locked behind the login.

5.2 Entity-Relationship Diagram

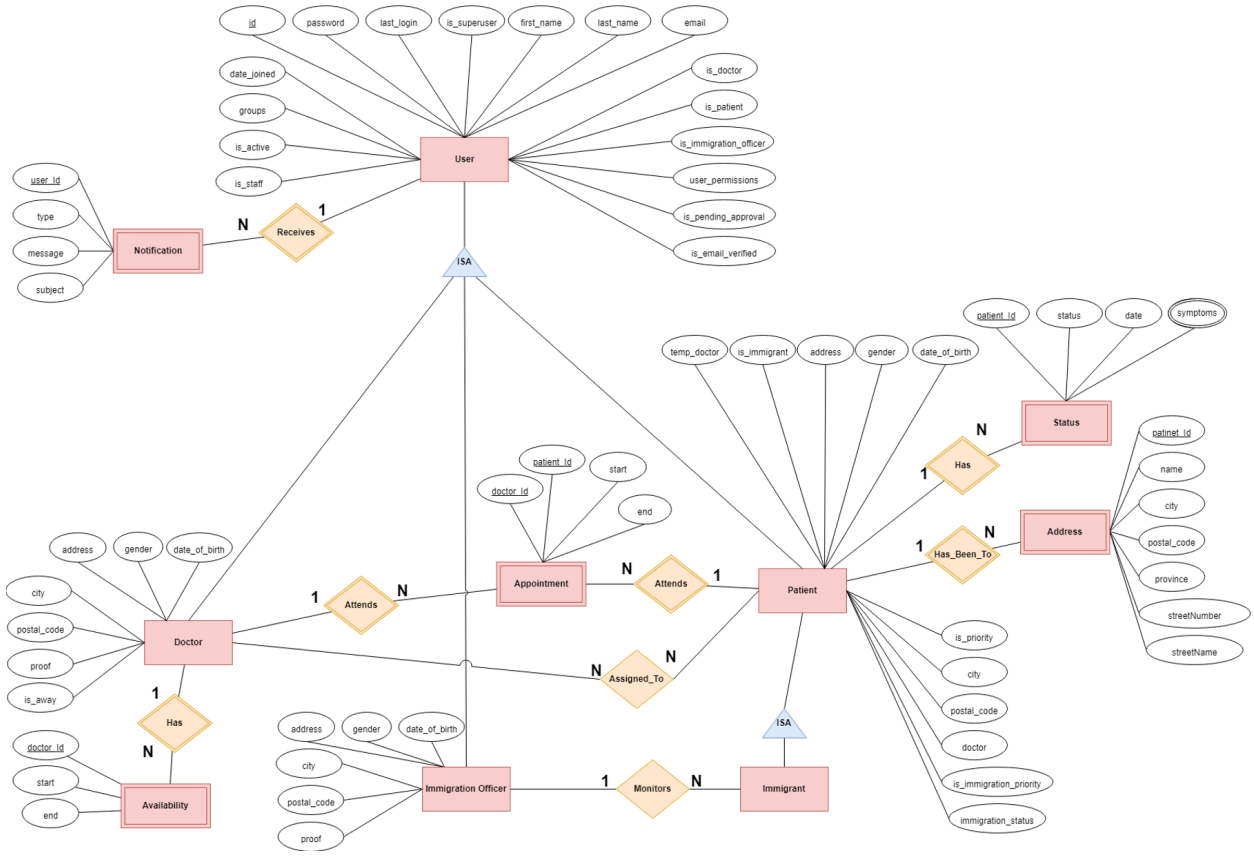


Figure 5.2.1: Entity-Relationship Diagram for Key User Relations/Attributes

Figure 5.2.1 represents an E.R diagram showing all major data members of our database as well as the relationship between Doctors/Immigration Officers and Patients. The User is the supertype of Doctor, Patient and Immigration Officer, containing numerous data members that are constant among both Doctors and Patients such as date joined, email, password, etc. The Doctors and Patients tables further add important data members that are not present in the Users table. The common primary key for all Users is their ID, however email is also designated as unique, so no two users can share an email. Patients have a foreign key *doctor* which points to a unique Doctor in the database.

The Assigned_To relationship represents the fact that 1 Doctor is assigned to N Patients, with each patient potentially having a replacement doctor in case of emergencies, hence N to N. There is partial participation on both sides for this relationship, as a Doctor need not have Patients and vice versa. Immigration officers and Immigrants have a similar relationship to Doctors and Patients, but with different privacy levels.

Finally, it must be noted that there is no explicit *Administrator* table: The Administrator is identified via the *is_superuser* boolean, allowing them to access all the admin functions of the site. This leads into the reason it was chosen for addresses, date of birth and gender to not be included in the

Users table: Administrators do not hold these kinds of data on their own, and thus they were left to the Doctors and Patients only.

For *Sprint 4*, more details related to *Patients* were added/modified: The *Patients* table now contains *is_priority* as a refactoring to better represent if that Patient has been marked as a priority. Furthermore, various fields related to immigration have been added: *is_immigrant* which shows that a Patient is an Immigrant, *is_immigration_priority* which simply denotes priority for an *Immigrant* and *immigration_status* which can be either 'Immigrant', 'Non-Permanent Resident' or 'None'. Finally, it is important to note that we decided to no longer include a reference to an Immigration Officer foreign key in *Immigrants*, as it was not deemed necessary for the *Immigrant* to know their specific officer. It is also worth noting that technically there is no *Immigrants* table officially, as the details needed to know if a *Patient* is an immigrant are provided in the fields of the *Patients* table. Thus it is here for illustration purposes but not a separate table.

Regarding additions related to Doctor emergencies, Patients now have a *temp_doctor* field and Doctors have an *is_away* field to denote they're away.

Weak entities such as Availability, Status, Address, Notification and Appointment, are reliant on one or more entities to provide a foreign key for them to use as their primary key. Each of the entities possessed enough functionalities and features to be distinct from the strong entity they rely on, but are still tied to them and cannot exist in a vacuum. As such every relationship that a weak entity possesses is an identifying relationship that explains how it ties to its dependent entity.

5.3 Activity Diagrams & Sequence Diagrams

5.3.1 Activity Diagrams

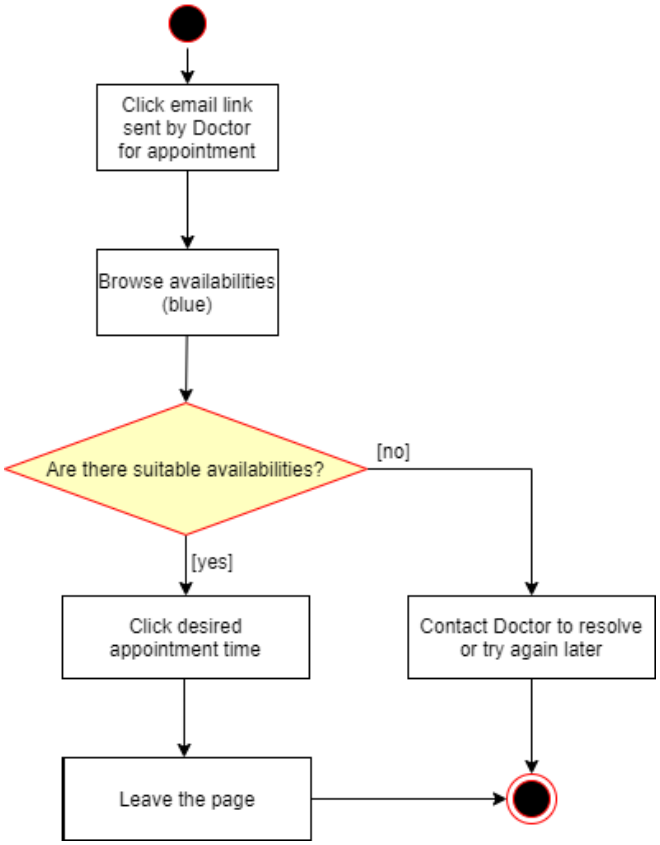


Figure 5.3.1.1: Patient Booking an Appointment with their Doctor

This diagram represents the process a Patient goes through when booking an appointment with their Doctor. It is assumed that the Doctor's availabilities are already set up and that the Patient has already made an account. The process is fairly simple and involves a nice UI, allowing Patients to pick a time based on the availability of their Doctor and if other Patients have booked appointments with that Doctor as well (availabilities denoted in blue, grey denotes unavailable). A precondition for this activity diagram is that the Doctor has sent the patient an appointment request via email, which redirects them to the site where they continue the rest of the process.

5.3.2 Sequence Diagrams

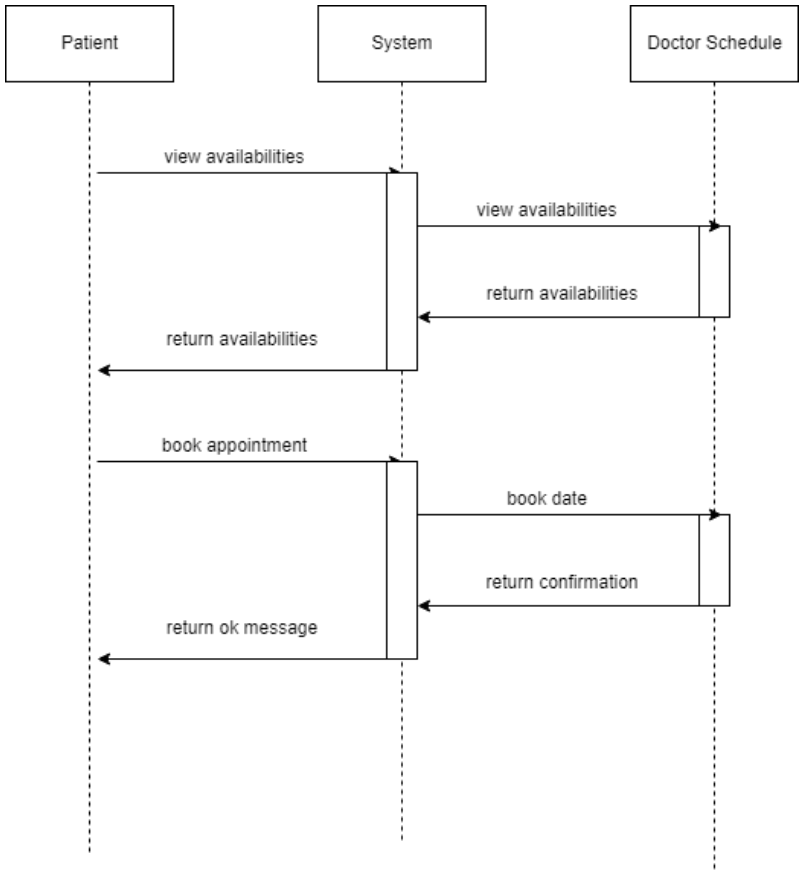


Figure 5.3.2.1: Patient Booking an Appointment with their Doctor

This diagram represents the process of a patient scheduling an appointment with their doctor. It retrieves the doctor's availability schedule after which the patient simply selects the required time for the appointment which updates the doctor's future availability. A confirmation message is returned to the user after the process is completed.

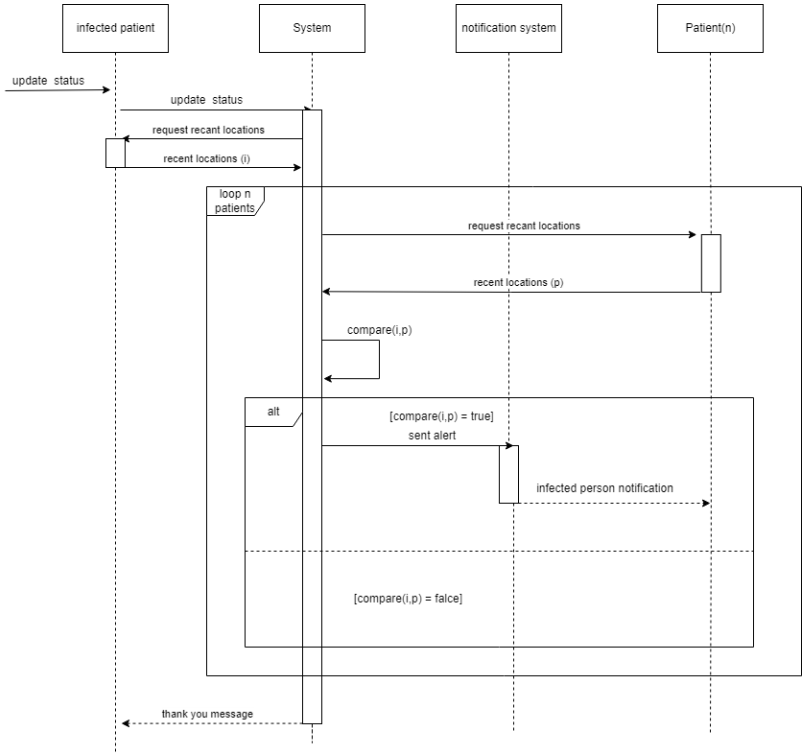


Figure 5.3.2.2: System Notifying Patients Once a new positive case is made

This diagram represents the process of background tracing implemented in the website. Once the user updates their status as infected it notifies the system, which then retrieves the infected patient's list of frequently accessed locations. Afterwhich, the system begins a loop for each other patient in the system retrieving the aforementioned patient's frequently visited location. Then both lists are compared and if any locations are shared the patient is notified of a potential chance of infection.

5.4 Domain Model

Below is the domain model for the application. This is the second version of the diagram, depicting the different conceptual classes that the application will contain. The diagram has been adjusted to show everything that either exists in our system or will be added in future sprints to our system. As this is still a work in progress, the domain model is always susceptible to change as the project moves on.

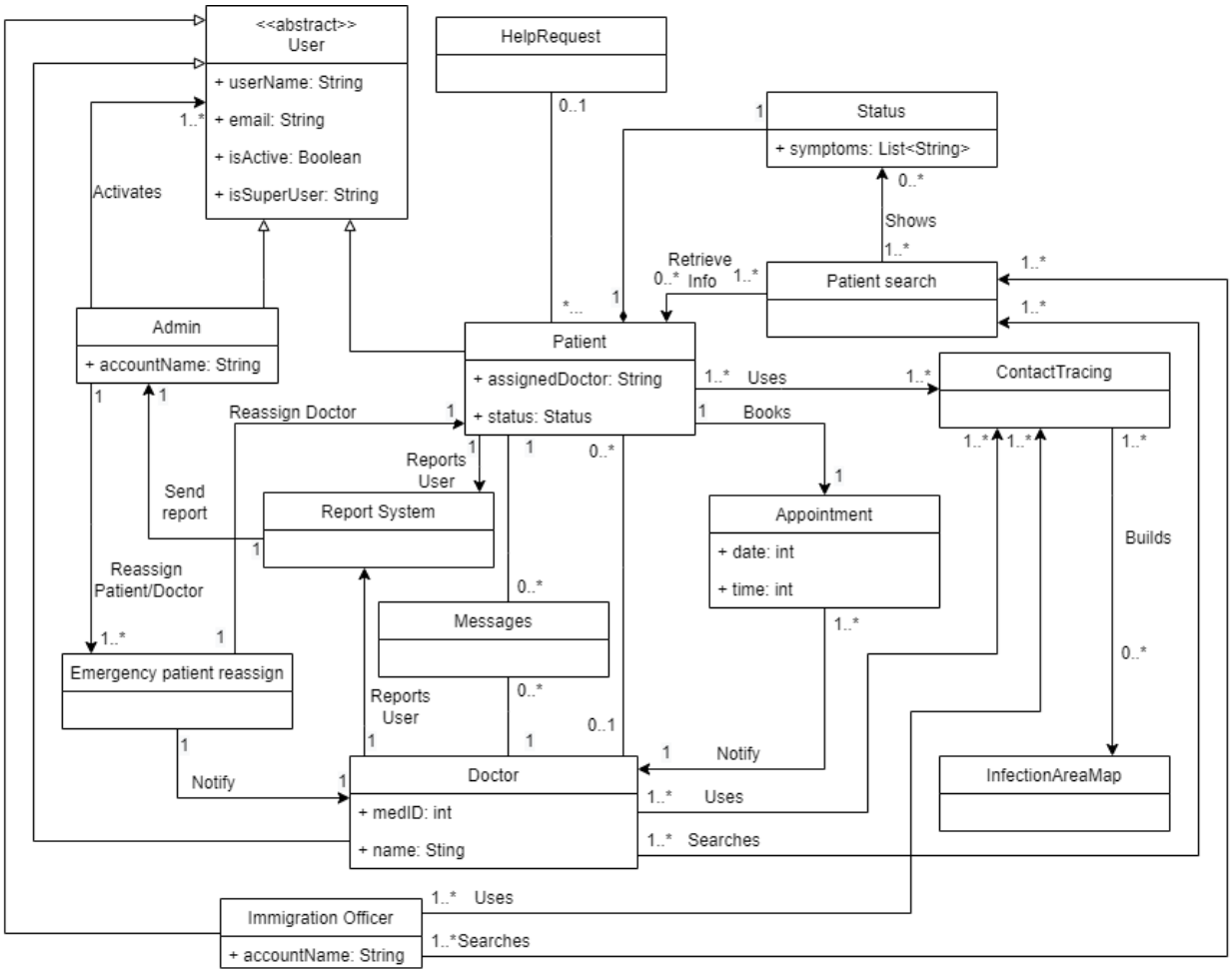


Figure 5.4.1 : Domain Model Diagram

5.5 Class Diagram

The class diagram below depicts the main class actors along with their interdependence, showing the end goal of what the system might look like. As can be seen, most of the main acting participants are inheriting the abstract class of User.

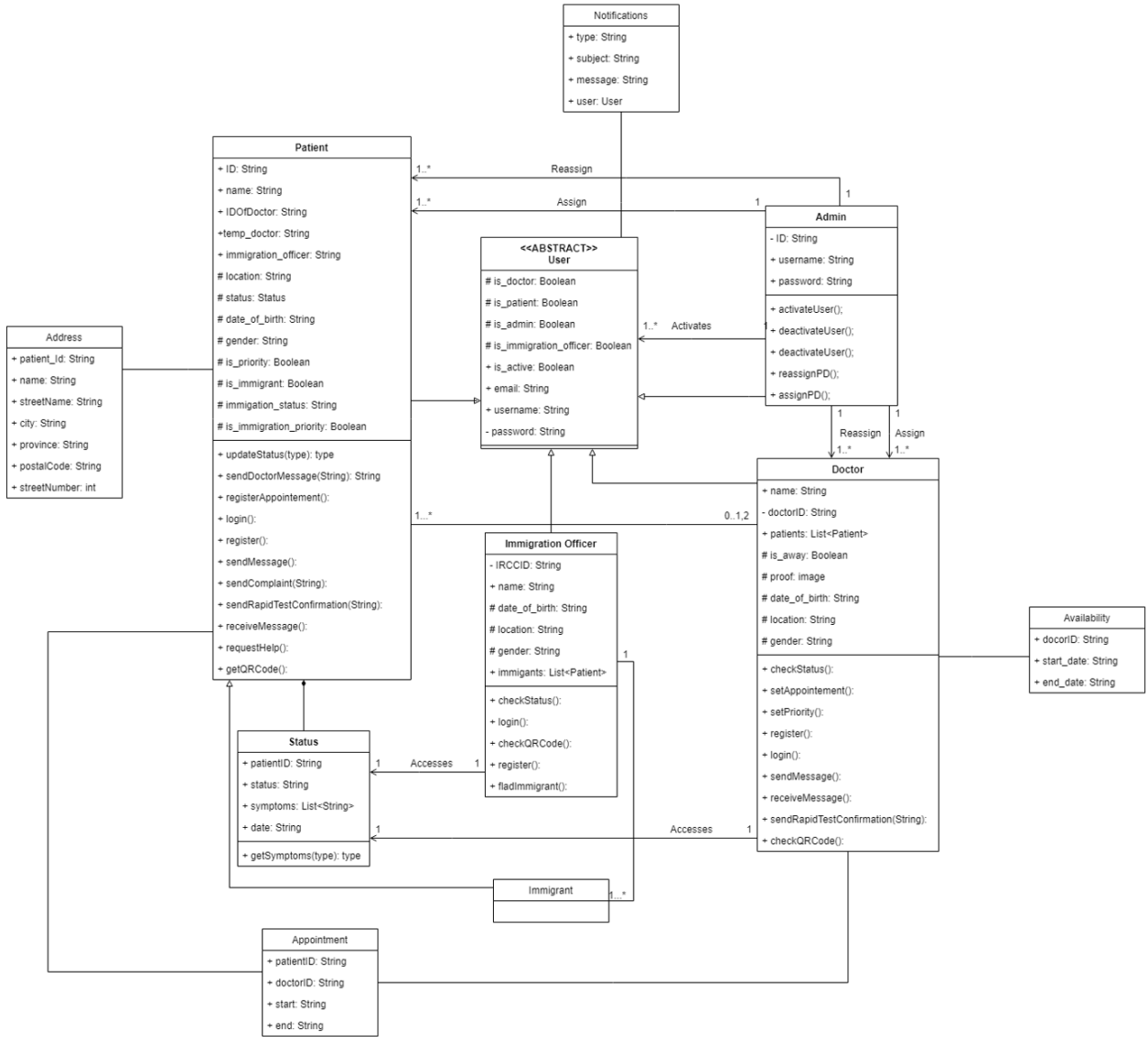


Figure 5.5.1: Class Diagram For Key Features

5.5.1 Modified Generated Class Diagram

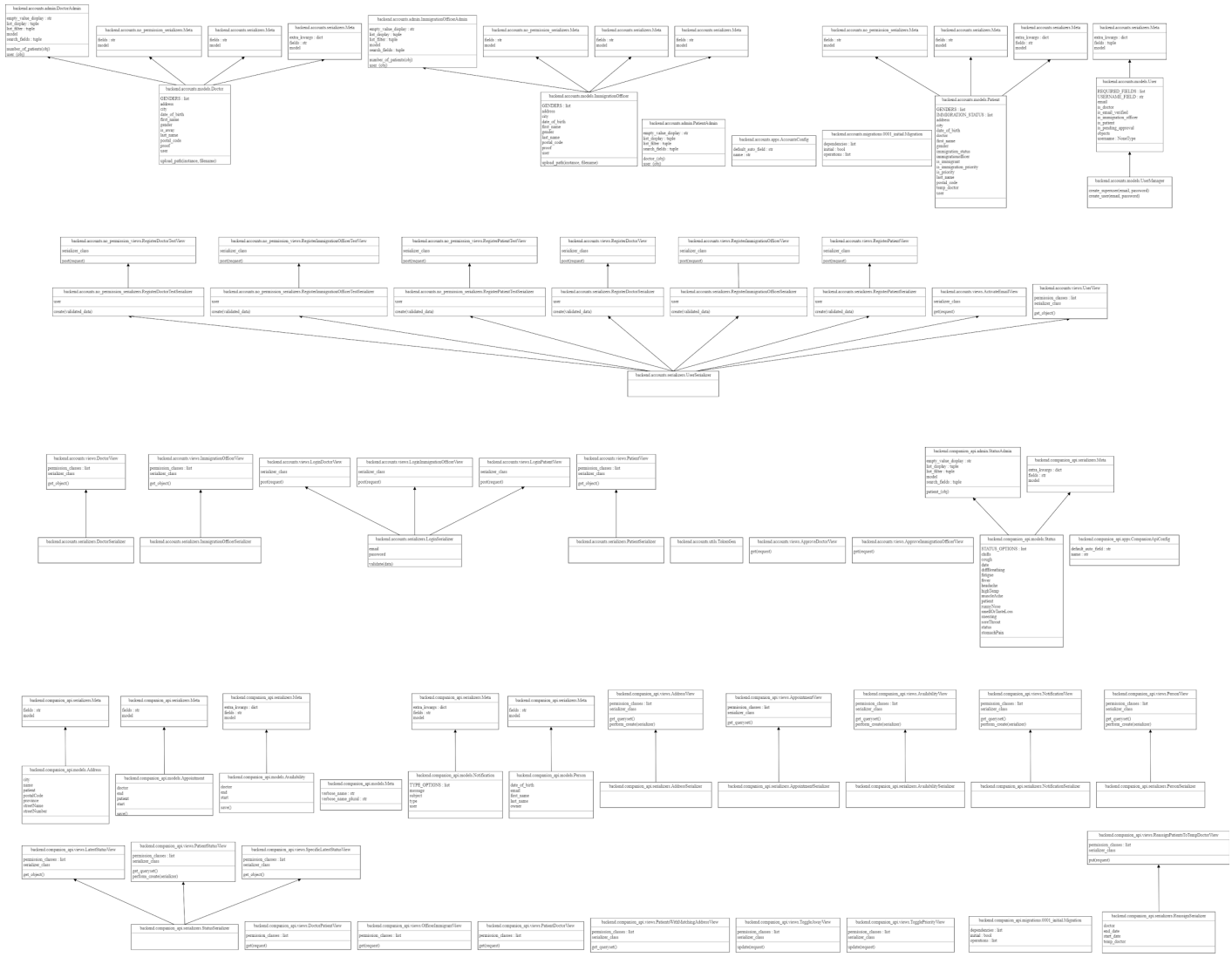


Figure 5.5.1.1: Generated Currently Implemented Class Diagram using pyreverse

Figure 5.5.1.1 was created using the tool *pyreverse*, which generates a class diagram based on the actual source code. This class diagram was modified slightly in order to better organize the classes and fit them on the page. Given that all classes are shown, it goes into far more detail in terms of actual implementation than *Figure 5.5.1*. This gives an overall idea of how the different views of Doctors and Patients interact with the serializers, which interact with the Doctors/Patients in the database. Note: the AdminDoctor and AdminPatient classes are specifically designed for the Administrators to view the appropriate information for Doctors and Patients. The tables from the Administrator page will view these versions of Doctors and Patients and give the Administrator useful tools such as searching for Patients/Doctors.

Furthermore, we can see that the Login logic contains a view for Doctors logging in and Patients logging in, these send POST requests to the serializer which calls a validate method to ensure that the values the User passed in are valid, and are valid for their appropriate role (Doctor/Patient).

We can also see that the User Serializer is used for registering Doctors and Patients, as Doctors/Patients are associated/"are" Users. Thus the details of a User are filled in with their JSON setup and we need the User Serializer to help with this.

Finally, classes related to the notification system are shown as well, with the notification view and serializer helping in delivering notifications to users.

New logic for the Immigration Officer was added as well, including serializers and views required for integration with the rest of the system. While Immigration Officers are similar to Doctors in terms of the pages they can view, there are still differences among them that needed to be considered and thus the separate views were created.

5.6 Component/Architecture Diagram

The component diagram displays the different subsystems of the applications and how they interact with each other.

5.6.1 UI

The main framework chosen is React due to the familiarity of the developers with it. It is also very easy to make the application responsive with this framework which is necessary due to there only being a web app. React makes use of Components to be able to reuse code through the application like a navigation bar in all the pages. It also uses a Router system to easily and quickly navigate between different page urls.

As for the design and style of certain UI elements like tables and buttons, there were a few frameworks that were considered. Two main ones were discussed due to once again the familiarity of the developers with the frameworks: BootStrap and MUI. The one that ended up being chosen is the MUI plugin due to its integration with React. Being designed specifically to work with React, it is much easier to integrate than bootstrap. It also offers a wide variety of templates and themes that are more sophisticated which will save some time when adding the styling to certain elements.

5.6.2 Domain

For the Domain model, one of the core classes is User as it has involvement with most classes in our project. Users will have access to some technical services such as Login, Appointments, and the Report System. While the User has access to all this, the system in general also interacts with classes that will interact with the User. For example, other domain classes such as Notifications will always interact with Users after the user books an appointment. It is also important to note that Contact Tracing also plays a big role in the domain model as every type of User interacts with it.

5.6.3 Technical Services

For the backend of the application, the Django REST Framework has been decided to be used. It is a powerful framework for building web applications due to its REST APIs which provides many tools for dealing with authentication, JSON deserializing and many others. It is a framework that is familiar to the developers and that is relevant to the application being built since it is a Web app. It will work with an Authentication tool that will take care of the security aspect of the application. It will make the Login tool more secure for the users.

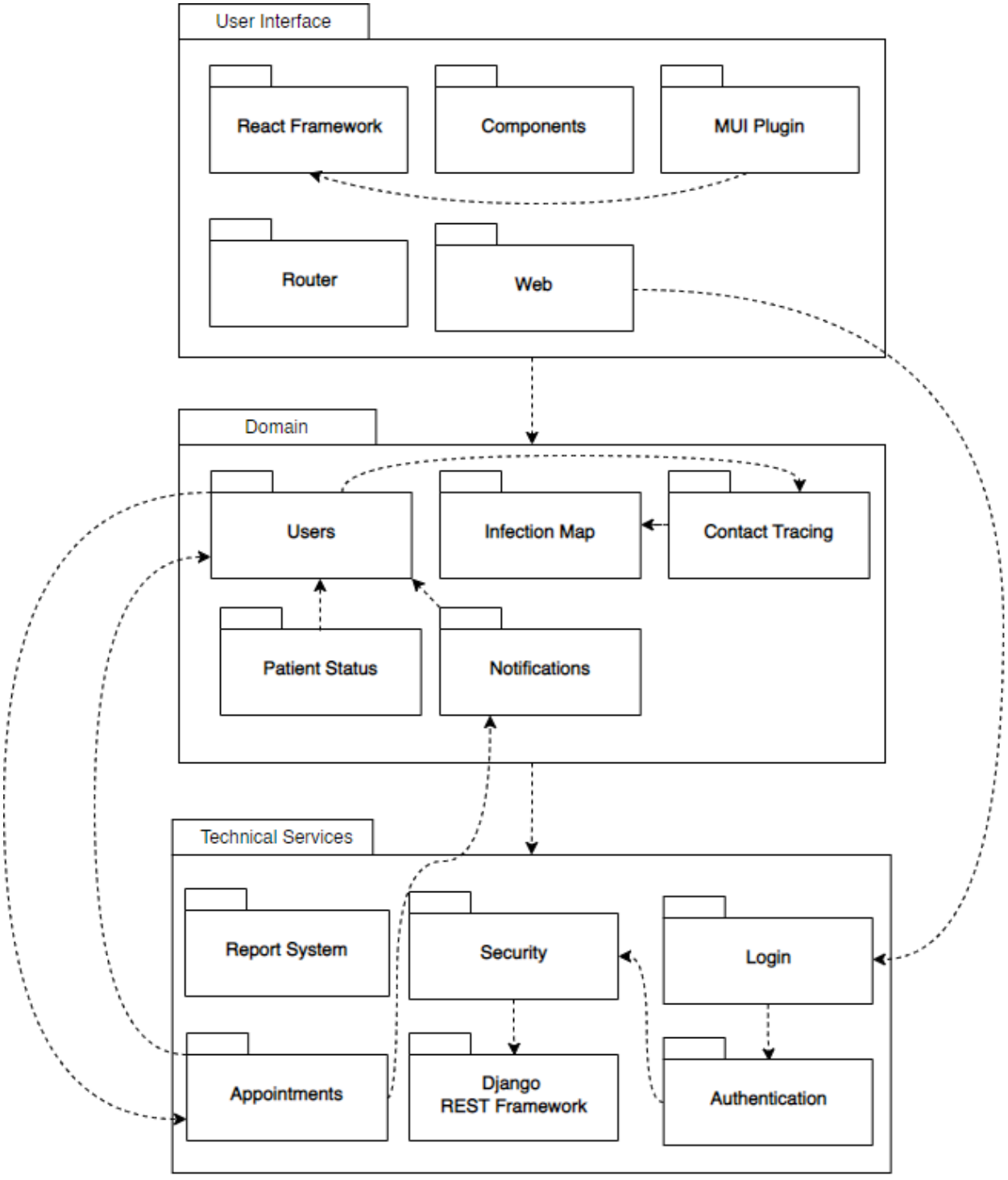


Figure 5.6.1 : Package Diagram

6. Risk Management

6.1 Probability Matrix

Probability						
	High					
	Medium			R-3 R-7 R-12 R-17	R-4 R-6 R-10 R-11 R-19	R-8
	Low	R-13	R-14 R-18 R-20	R-1 R-16	R-2 R-5 R-9 R-15	
		Minimal	Minor	Moderate	Significant	Severe
		Impact				

6.2 Risk Analysis

Risk ID	Description	Probability	Impact	Resolved in Sprint	Strategy and Effectiveness
R-1	Lack of skills (ie. web development)	Low	Moderate	1	Each member will work to their strengths on different parts of the project. Identifying our needs early on to make better decisions and strategies.
R-2	Unclear list of requirements	Low	Significant	1	Reviewing the project plan document and requesting clarification and information from the project owner.
R-3	Unmet goals in specified timeframe	Medium	Moderate	2	When tasks start falling behind schedule, contact team members responsible to discuss reasons why. Modify effort values for

					each task to reflect real difficulty if need be. Insist on a policy of asking for help if you're falling behind. Discuss with PO if some items are not feasible for the sprint after internal discussion.
R-4	Poor productivity	Medium	Significant	2	Have frequent team meetings in order to get everyone on track. Have teammates encourage others to get their work done on time, team lead reminds everyone of deadlines.
R-5	Poor code quality	Low	Significant	2	Use linting tools in order to get an automated response early if code is of poor quality. Never commit code directly to master and have at least 1-2 reviewers required for every pull request in order to audit work done.
R-6	Loss of data due to external factors (ie. computer crashes, corrupted data)	Medium	Significant	1	Saving all the necessary documents on Google drive, and Github will allow easy access on any computer. Saving a backup of the data on our local drive in case of no internet access.
R-7	Late changes in requirements	Medium	Moderate	2	Design code to be adaptable and minimize dependencies to a specific implementation. Have constant discussions with PO and revise all new documents provided in order to minimize the chances of being surprised.
R-8	Broken Authentication	Medium	Severe	2	Follow best practices in regards to setting up authentication in Django.
R-9	Developing the wrong software functionality	Low	Significant	2	Consult the PO or professor whenever a requirement is not clear. Ensure all team members are familiar with the story descriptions and audit work done by different sub-teams to ensure the right functionalities

					are being developed.
R-10	Developing a poor UI	Medium	Significant	2	Use an existing library (MUI) to build a responsive UI. Discuss layout with PO and teammates to ensure a clear, functional and visually appealing UI is developed. Prototype the UI in order to get feedback before full commitment.
R-11	Lack of login monitoring	Medium	Significant	3	Add a field that shows when a User of any form last logged in, allowing for greater visibility from the Administrator's POV.
R-12	System components breaking due to adding new features	Medium	Moderate	3	Add more unit/integration tests to the testing suite for frontend and backend. Perform system testing for each PR.
R-13	System not sending user real-time but late updates and notifications	Low	Minimal	3	Run timed tests to ensure notifications are being added on time as expected.
R-14	QR code not being recognized/read by device/camera	Low	Minor	4	QR Code is dynamically sized such that it is large enough even on small screens. Resolution of the QR Code itself is low enough such that it operates on low resolution monitors without complications.
R-15	Patient misplacement (ie. health care workers and immigrations officers are not assigned the correct patient)	Low	Significant	3	Ensure proper patient visibility on the Administrators side in order to quickly diagnose issues, allow users to send reports to the Administrator if they feel something is wrong (Doctor noticing an incorrect Patient, Immigration Officer noticing non-immigrant in his/her view, etc.)
R-16	Inability to contact the product owner in case of uncertain or unspecified	Low	Moderate	2	Commit to a policy of asking questions as early as possible in order to avoid running into a scenario where the PO is

	requirements				indisposed near the due date. Come up with backup plans in case PO is unavailable at a critical time (contact the professor for the question, contact another TA).
R-17	Features not passing testing requirements	Medium	Moderate	3	Discuss with the team how to improve the feature or if testing requirements were too ambitious/out of the scope. Discuss with PO if issue persists.
R-18	System not properly respecting the appropriate patient-to-doctor ratio assignment	Low	Minor	3	Educate Administrators on the number of Patients they should assign to a Doctor, allow Doctors to complain to Administrators if they feel their Patient count is too high. Be prepared to implement automated Patient balancing tool if it becomes too difficult for Administrators to manage
R-19	Deployed system exhibiting poor performance.	Medium	Significant	4	More CPU cores and memory can be allocated dynamically through Amazon EC2 Portal.
R-20	External COVID API goes out of service or is no longer available	Medium	Minor	4	Design the system for retrieving data from the API calls to be as modular as possible, allowing API calls to be swapped out fairly simply while maintaining the same design from the client's perspective.

7. User Interface Design

Our user interface will be designed to have a dynamic design depending on the type of user that is using it (for instance, a patient will have a different dashboard than a health official). However, for the login, every user (patient, health official, medical doctor, immigration officer, and admin) will have the same requirements to login into the web application. Moreover, on the sign up page, every persona will have to enter the same information to create a new account except the administrators. Administrators will be people specifically selected that will have their account immediately created in the database in order to limit the risk of someone impersonating an administrator.

Examples of Personas:

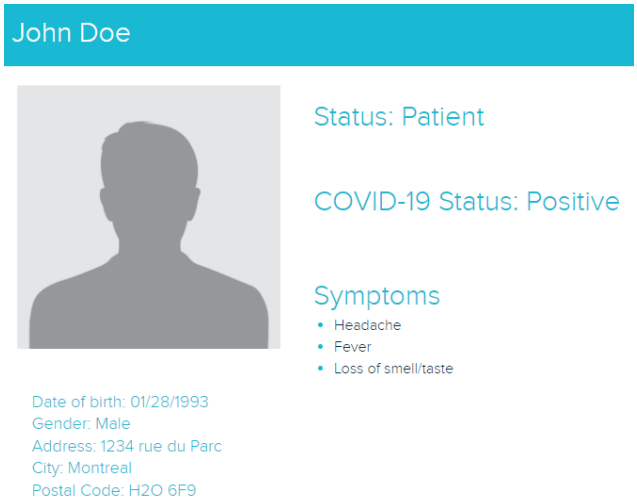


Figure 7.1: Example of Patient

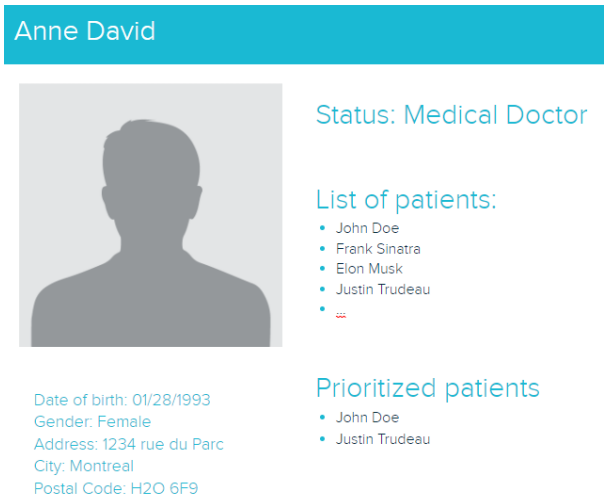


Figure 7.2: Example of Medical Doctor

7.1 Sprint 1

In the first sprint, one of the objectives was to design and implement the interface of the Sign In page. To do so, the MUI plugin was used to produce a clean and aesthetic form. When the user opens the web application, he is prompted to the Sign In page to verify his credentials. Both the email address and the password are required for the user to be successfully logged in, which is denoted by the asterisk sign. The user can then choose to check the checkbox “Remember me” to facilitate the next time he logs in. The user can then sign in by clicking the “SIGN IN” button, which will redirect him to the home page if successfully signed in. Under the Sign In button, the user can click on the “Forgot password?” link, to reset his password, or on the “Don’t have an account? Sign Up” link to register as a new user. Regarding the error tolerance, the password is hidden to ensure privacy, and the email address field is verified to make sure it has a valid email structure.

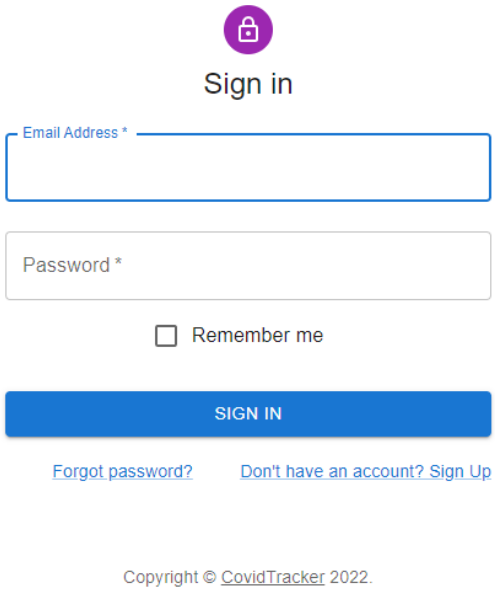


Figure 7.1.1: Login Page

Regarding the Sign Up page, it looks relatively the same as the login page except there are more fields in order to gather all the information needed to create a user profile. For the date of birth, the user can decide to click on the calendar icon to have a calendar pop-up which will allow him to select the date of birth directly on a calendar. Also, the gender section is a dropdown field that allows the user to only select between male, female and other to keep it simple in the database. The status type field is where the user can select whether he is a patient, a health official, a medical doctor, or an immigration officer. As explained earlier, the admin role will be attributed by the developers directly in the database to avoid any security leak. Once the user has entered all the information, he can click on the “SIGN UP” button which will ensure that every field is validated and send the new account request to the backend. The user can also go back to the login page by clicking the blue hyperlink under the sign up button.

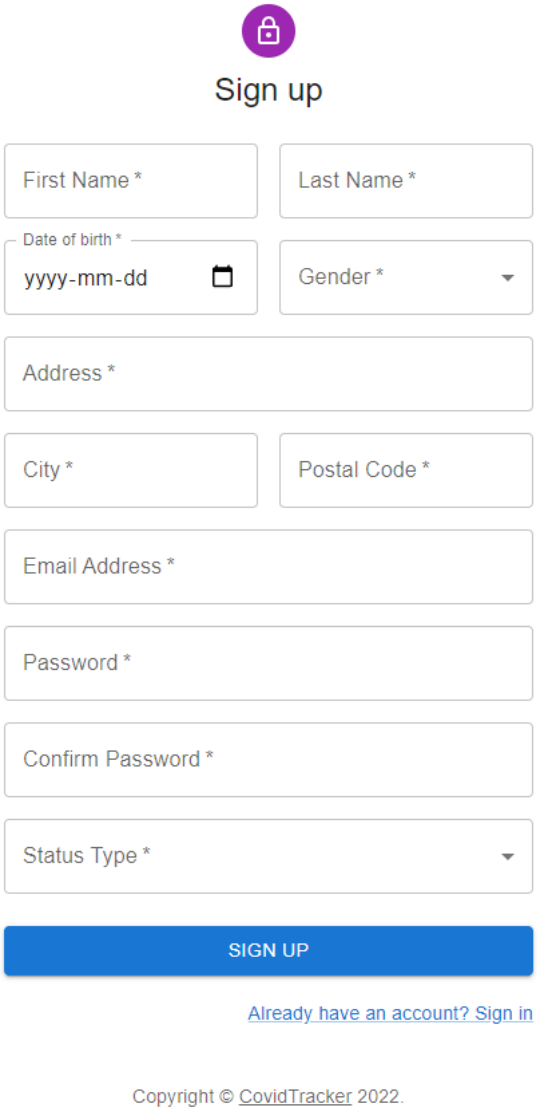


Figure 7.1.2: Sign Up Page

7.2 Sprint 2

One of the important features that will be added to our website next sprint is a priority patient list as shown in *Figure 7.2.1*. Doctors should be able to look at their patients and flag them as a priority patient so they can keep track of them and remind themselves that they need more attention. This will also help as any status update on a patient will be listed as a more important notification to the doctor. The doctor will be presented with a list of patients and their details, and to the right, they can flag whoever they want to.

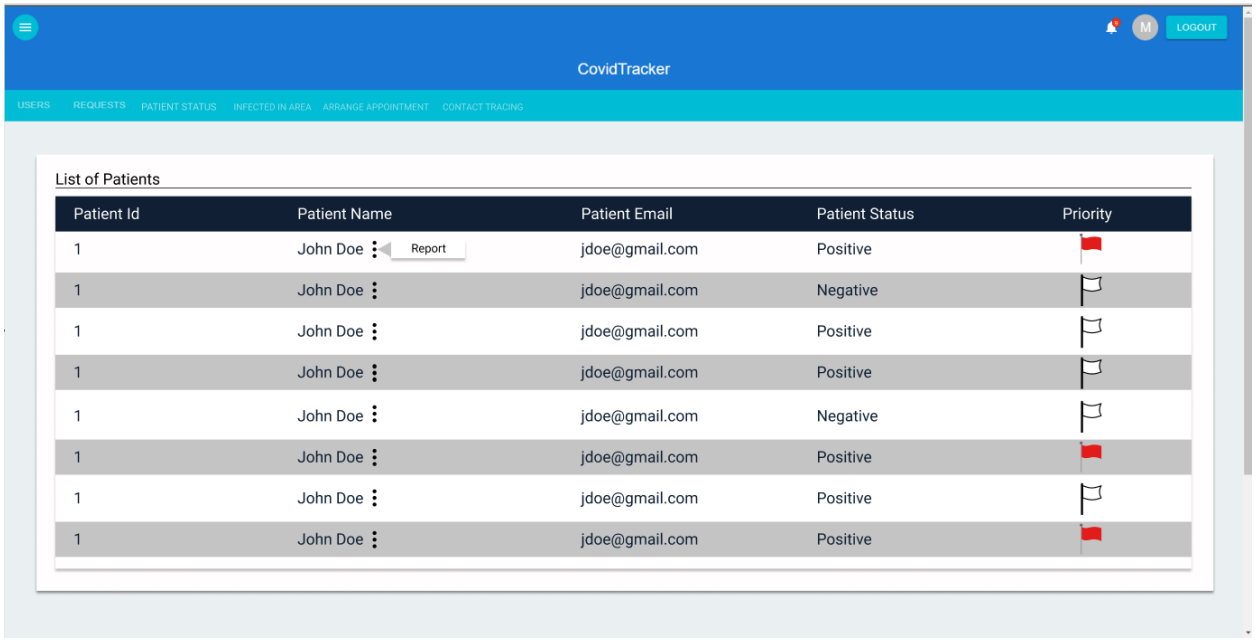


Figure 7.2.1: List of patients with priority

Every user should also have the option to report someone else, whether it is a patient reporting a doctor, or vice versa. This feature is necessary in case either the doctor or patient do not cooperate properly. A doctor can report their patient as shown in *Figure 7.2.1* or a patient can report their doctor, and they will have a popup to write their reason for the report as shown in *Figure 7.2.2*. This will go to admins so they can take action accordingly

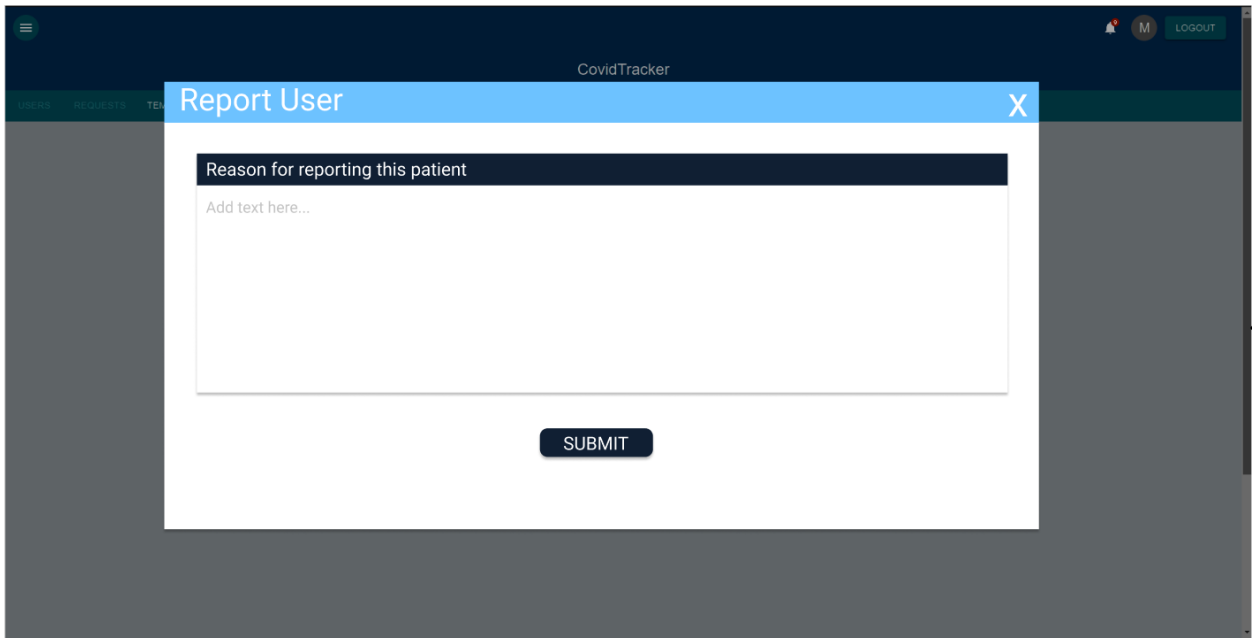


Figure 7.2.2: Report User Pop Up

Users will be able to see any important notification that may concern them next to their profile icon, where the bell is, where the number on the bell represents notifications that are unread. Clicking it will show a small box of notifications like in *Figure 7.2.3*. Clicking an individual notification will show it in more detail and will mark the notification as read as shown in *Figure 7.2.4*.

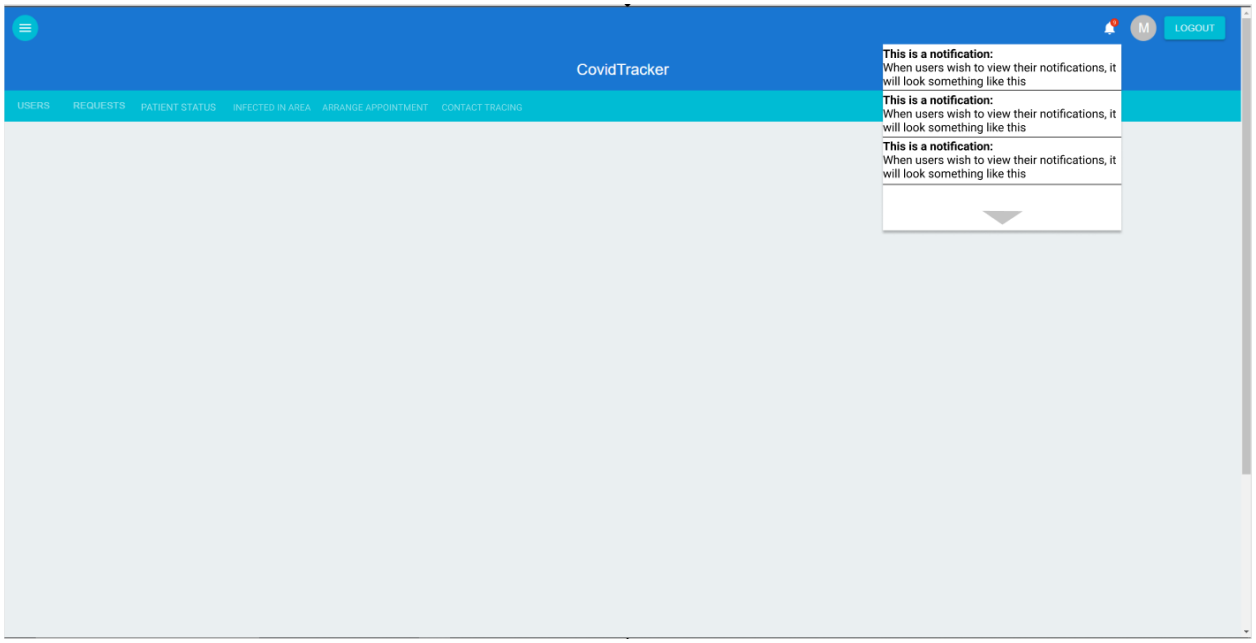


Figure 7.2.3: Notification List

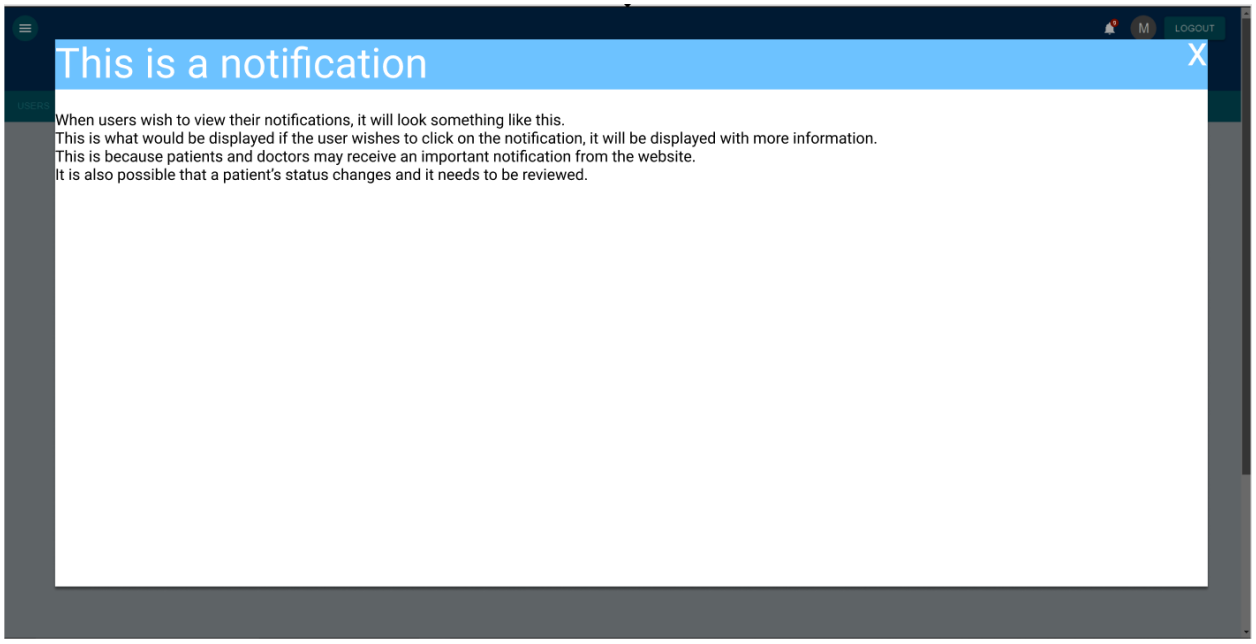


Figure 7.2.4: Notification Pop Up

Users will be able to check areas around them where people have visited that tested positive. This will allow users to judge if a certain area is worth visiting. On this tab, areas will be displayed with a clickable address that leads to Google Maps, showing the number of recent infections and the risk level, as shown in *Figure 7.2.5*.

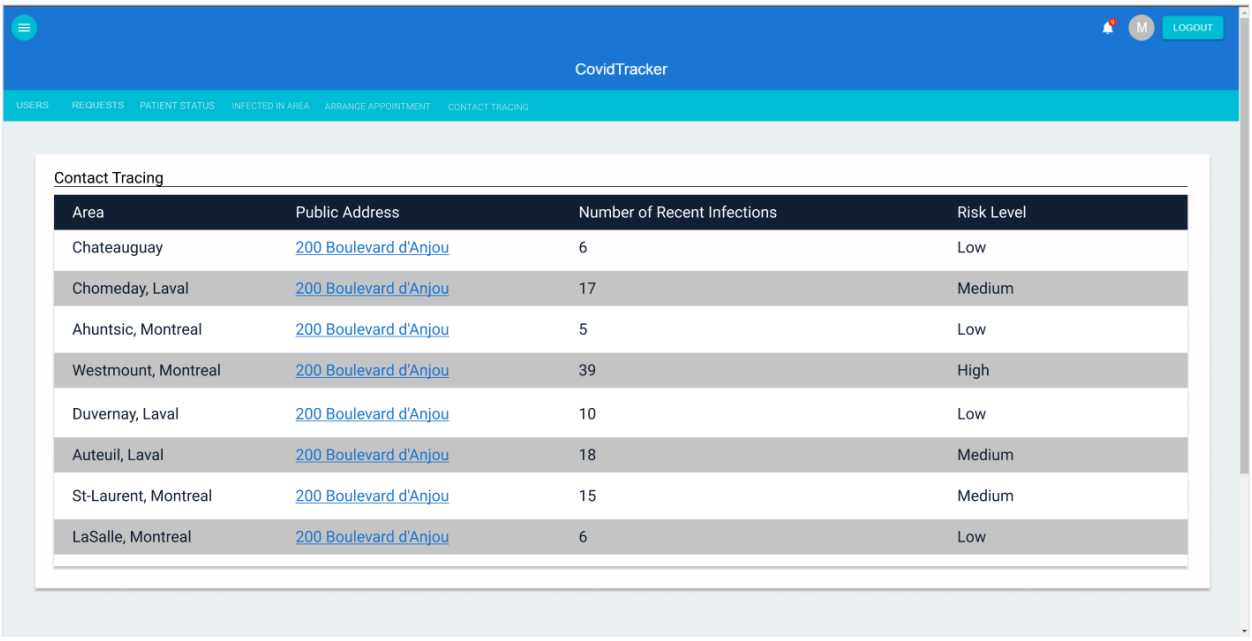


Figure 7.2.5: Contact Tracing Page

This feature is similar to the contact tracing feature, but instead of checking general areas, it will check the number of positive cases around the user. This will allow users to judge if visiting the area is safe or not. In *Figure 7.2.6*, the map is displayed so users can see their area, and a number of positive people is given within their area, along with an indicator if the area is generally safe or not.

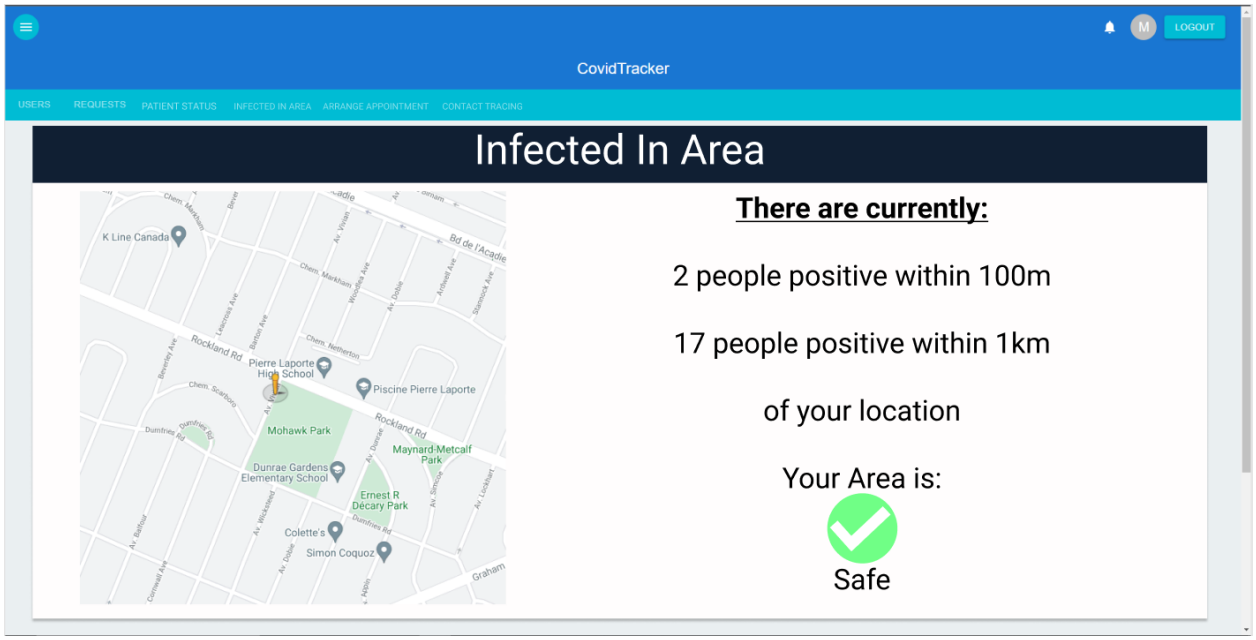
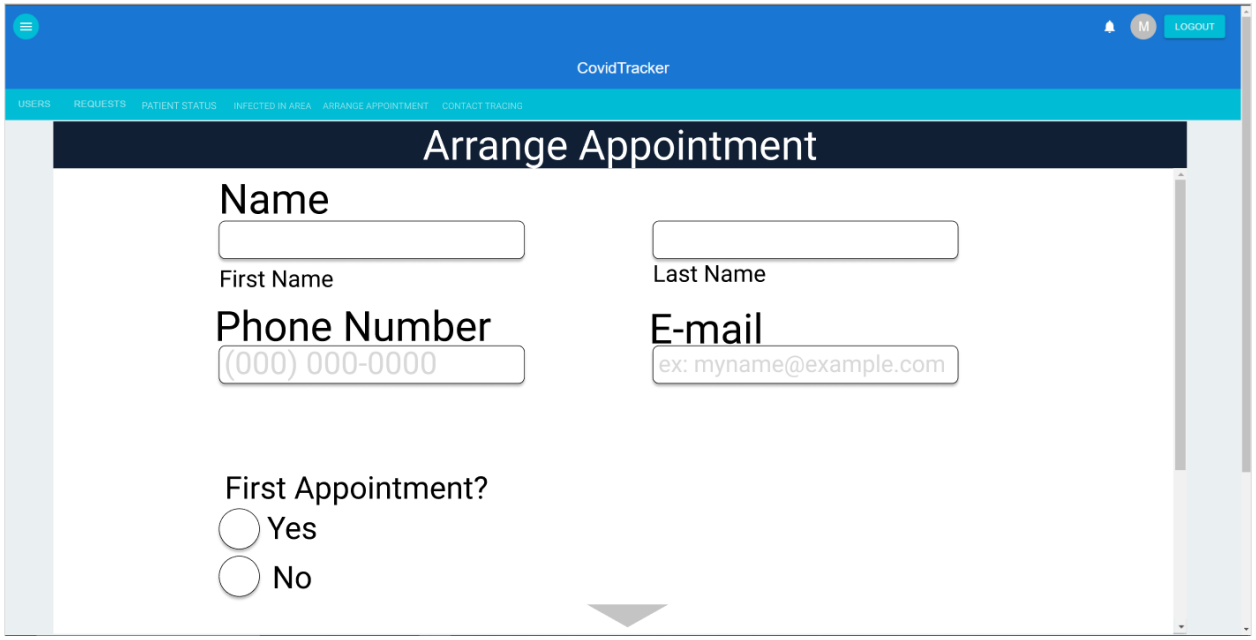


Figure 7.2.6: Infected In Area Page

The final feature that will be added next sprint is an appointment arrangement page. Patients will be able to book appointments with their doctors with the click of a button, and will be able to fill in data for their appointment as shown in Figure *7.2.7*. Once they submit, the associated doctor will be notified and both the patient and doctor will be marked for an appointment at that time and day.



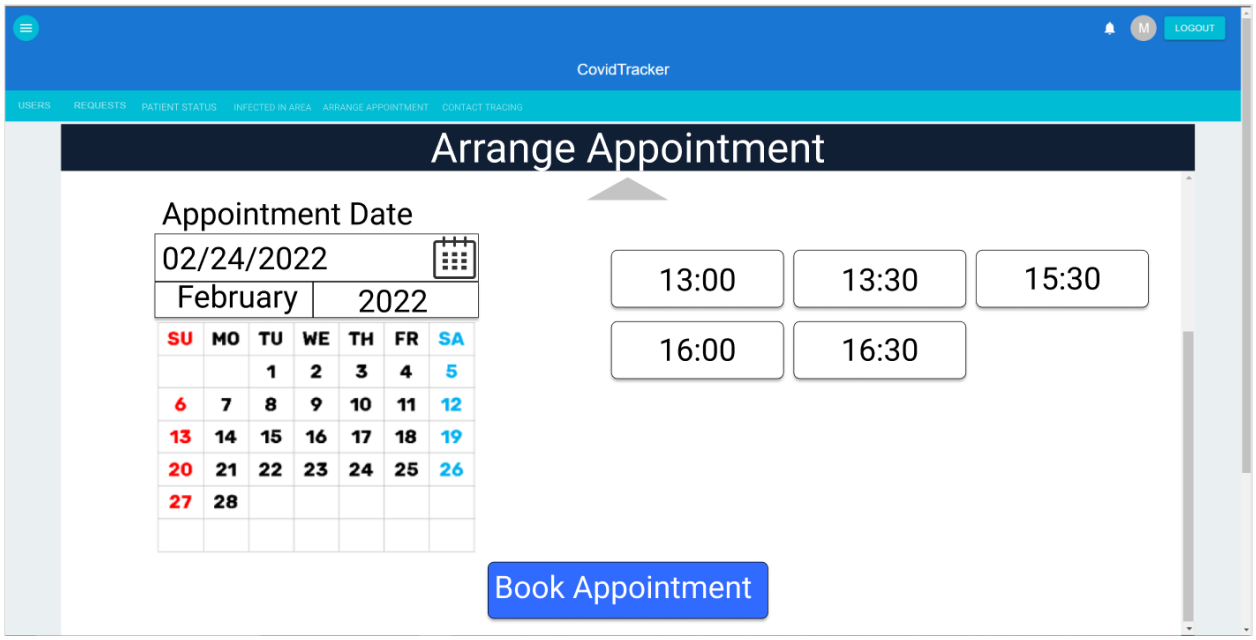


Figure 7.2.7: Appointment Page

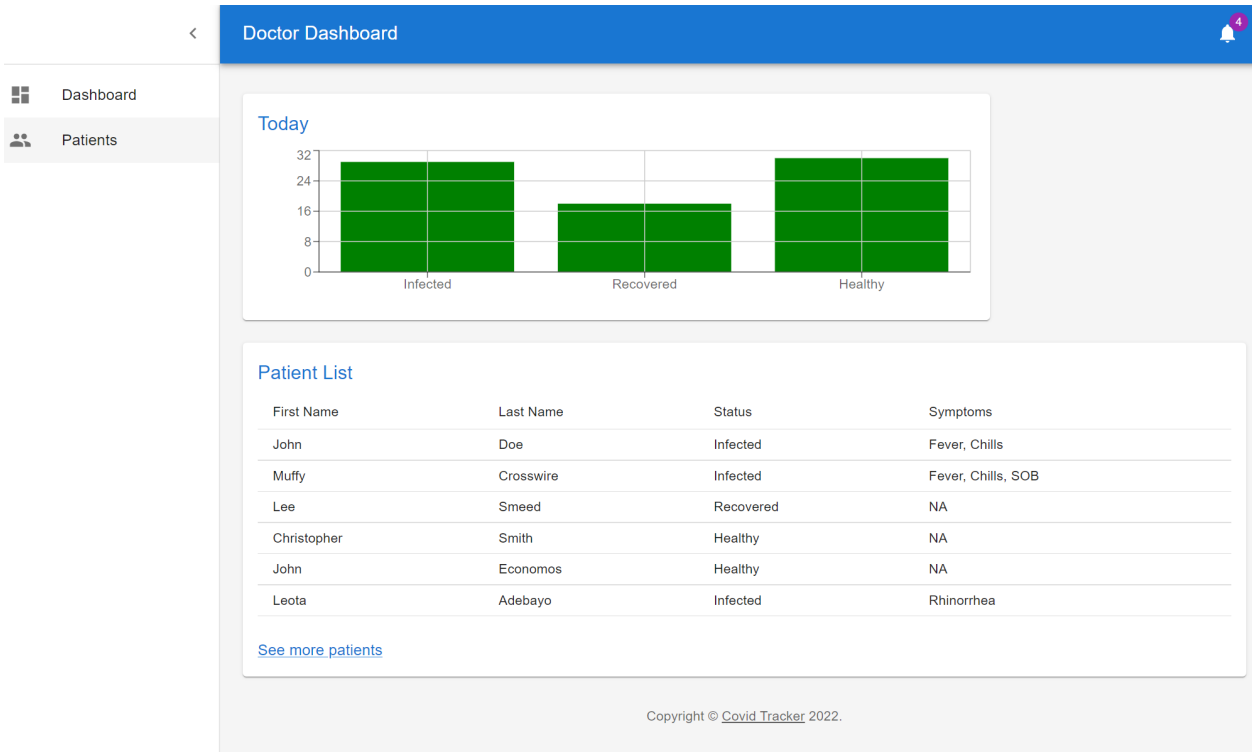


Figure 7.2.8: Doctor's Patient Dashboard Page

The above depicts a prototype of a doctor's patient page on their dashboard. The layout of the dashboard (ie. color scheme and format structure) will be consistent with the others as seen in the previous figures for the future sprint. The doctors will be able to access a bar chart that will display the current number of patients of each status (infected, recovered, and healthy). In addition, a personalized list of patients assigned to them is shown along with their current symptoms.

7.3 Sprint 3

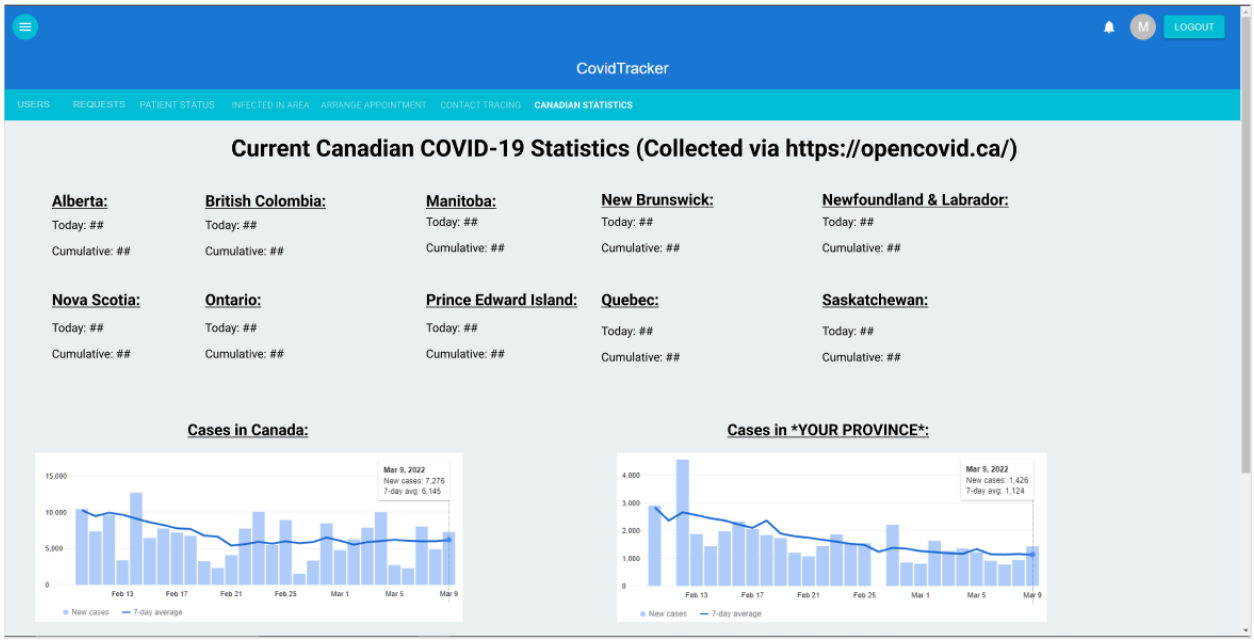


Figure 7.3.1: API to Other COVID-19 Applications

This figure represents the planned outline for the API connection to opencovid.ca, which is an open source database providing useful, Canada-wide COVID data. This prototype shows a nicely laid out version of the data present in the database, showing cases per province today and cumulative since the start of the pandemic. Furthermore, graphs will be present to show the cases in your province as well as Canada wide. The specific sizes of the graphs and how many months they show is still up for debate and being experimented with.

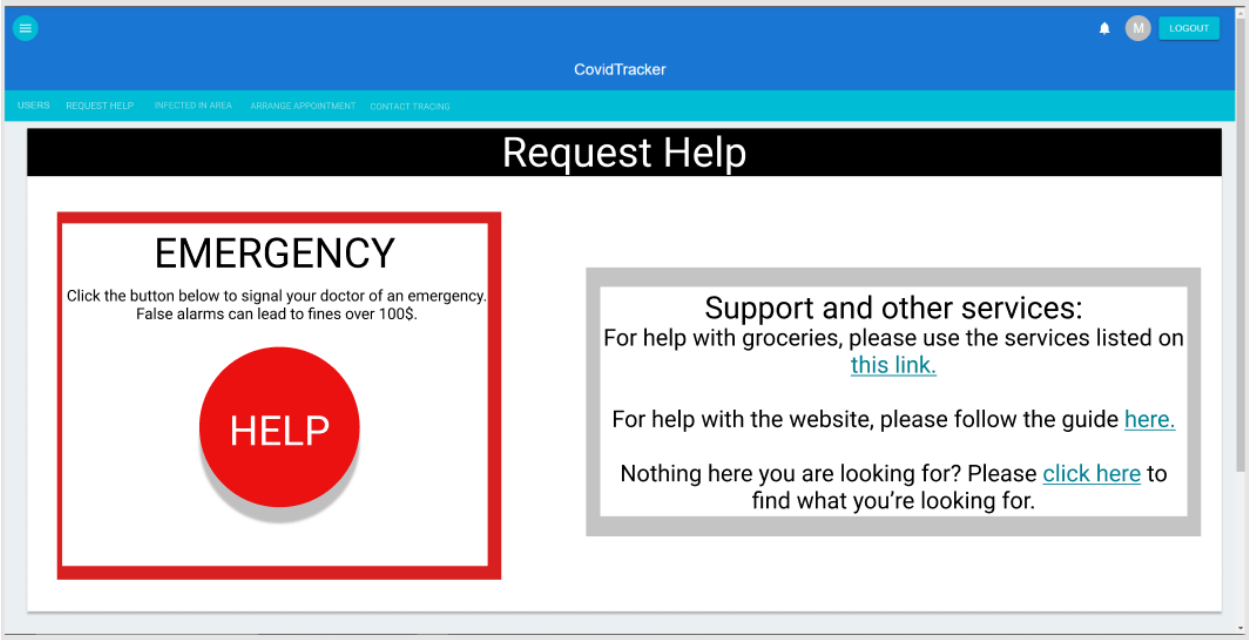
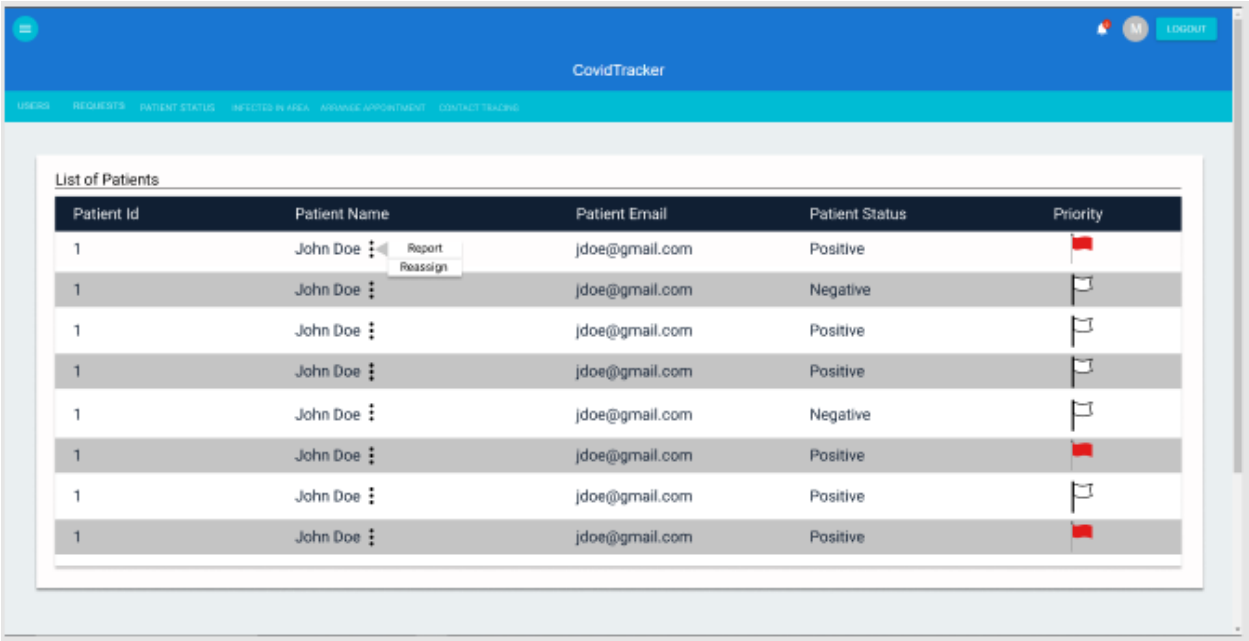


Figure 7.3.2: Request Help

The prototype above will be available for patients who have tested positive for COVID-19. This prototype will have a big red button handy for patients who are in need of an emergency. Pressing the button will let the assigned doctor know that the patient is in immediate help and needs attention immediately, which is similar to dialing 9-1-1, but with the click of a button. Patients with COVID will also be able to have access to other links which offer services to help them with things like groceries and shopping.



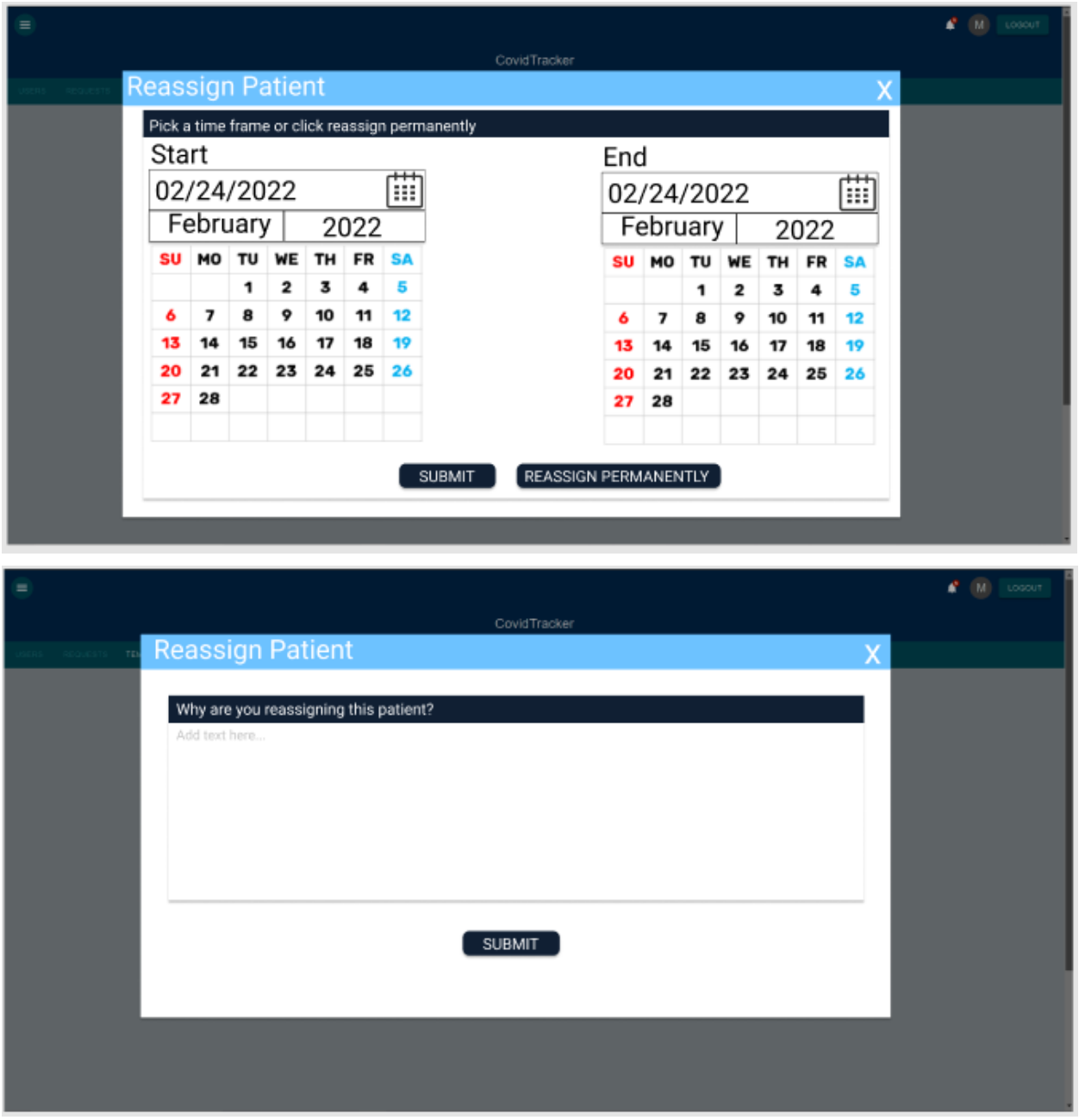


Figure 7.3.3: Emergency Reassign Patient

The feature above will be used by doctors that need to reassign a patient due to some circumstance. Doctors will have the option while looking at the patient dashboard to select a patient and reassign them to some other doctor either for a duration or forever. The doctor must also provide a reason as to why the patient has to be reassigned as it will give admins some understanding as to why this action was taken. Once the patient reassignment is successful, the doctors will receive a notification to let them know that the action was successful and the patient will receive a notification that their doctor has been switched.

Figure 7.3.4: QR Code for Individual Patient Records

The next prototype as shown in *Figure 7.3.4* is a QR code that patients will have access to whenever they want. This feature will allow patients to generate a QR code of their medical records, which will include their current status, some detail on their conditions, history of status changes, and their current and past doctors. This can also be downloaded as a PDF to avoid having to constantly generate and save the patient's current data on their phone or computer.

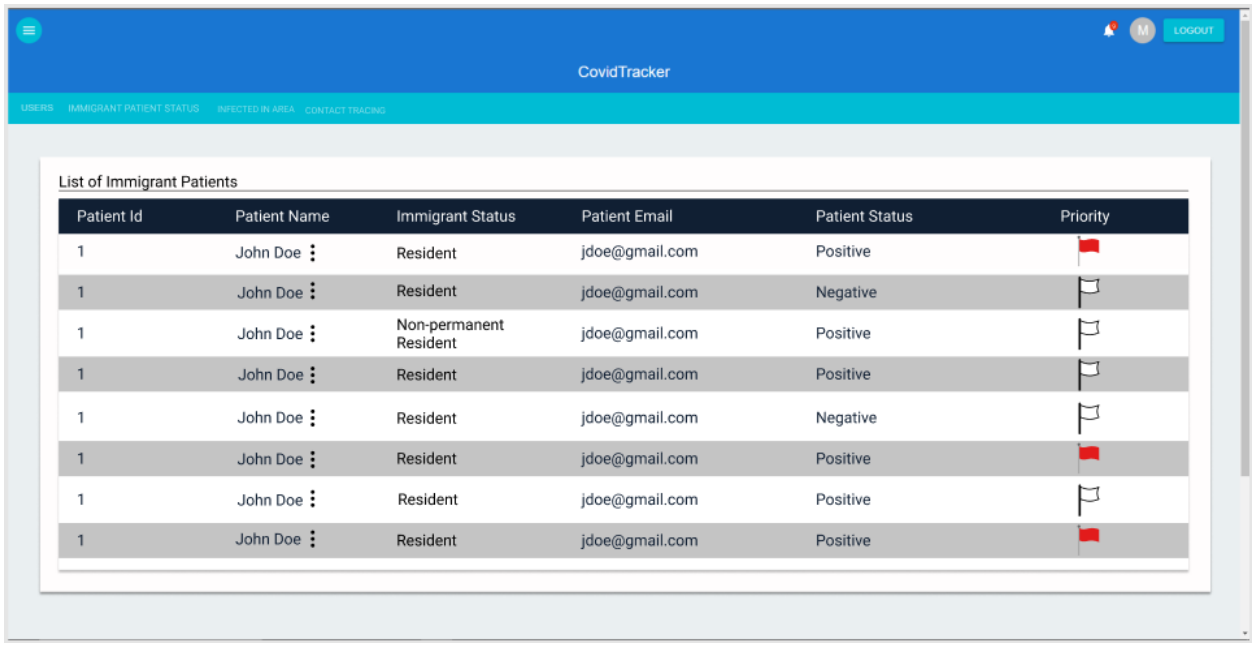


Figure 7.3.5: Immigrant Patient Dashboard

Similarly to doctors, immigration officers will have the ability to look at a patient dashboard to track patients and their statuses. In this case, immigration officers will have access to a dashboard of immigrant patients, which will show their COVID status along with their immigrant status as shown in *Figure 7.3.5*. The immigration officer will also have the ability to flag these patients to help them keep track.

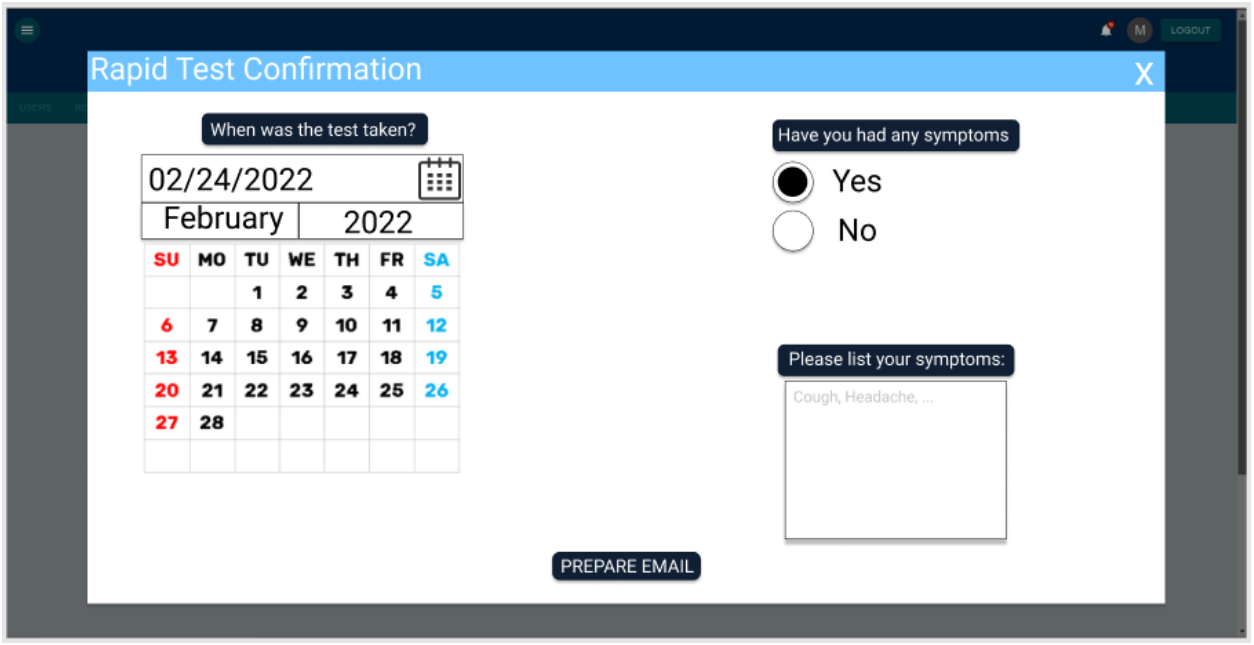


Figure 7.3.6: Rapid Test Confirmation

The final feature shown in the figure above is a prototype that prepares rapid test confirmation email for patients that test positive via rapid test. The patient has to fill a few points, which are the day the rapid test was taken and any symptoms the patient may be feeling. Once the patient has filled out the questions, they may click the button below to automatically generate an email with their details. The patient may review the details and must include a clear picture of their test. Once all this is done, they can send the email to their doctor.

7.4 Sprint 4

Given that all features were completed this sprint, there are no new UI prototypes to demonstrate for *Sprint 5*. This is because *Sprint 5* is designated as a wrapup sprint, where we will be taking care of any tech debt, refactoring and increasing test coverage.

8. Testing Plan and Report

8.1 Testing Tool Chosen

8.1.1 Backend: unittest

The tool chosen for unit testing for this project is unittest:

unittest was initially based on JUnit and is part of the Python standard library. While it is simple in its output, Django's unit tests use it to run their tests, thus it is a good fit for our project. Furthermore, in tandem with the *coverage* tool from Python, it can produce detailed coverage statistics. Most importantly, it is supported by GitHub Actions, and thus we have set up a YAML file to automatically run all unit tests when pushing to a branch.

Figure 8.1.1.1 is an example of the test report given in a text file by unittest. It includes important test setup details and lists the actual tests run, along with the commented description of that test case (see the last 2 lines before the vertical dashed line).

```
Found 1 test(s).
Creating test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...

Operations to perform:
  Synchronize unmigrated apps: corsheaders, messages, rest_framework, staticfiles
  Apply all migrations: admin, auth, companion_api, contenttypes, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying companion_api.0001_initial... OK
  Applying sessions.0001_initial... OK
System check identified no issues (0 silenced).
test_person_attribute (tests.test_models.PersonTestCase)
Check that the people are properly stored in the database ... ok

----------------------------------------------------------------------
Ran 1 test in 0.004s

OK
Destroying test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...
```

Figure 8.1.1.1: Test Report Output of unittest Django Library

```
Name                                          Stmts   Miss Branch BrPart  Cover   Missing
-----------------------------------------------------------------------------------------
companion_api\__init__.py                         0      0      0      0   100%
companion_api\admin.py                            1      0      0      0   100%
companion_api\apps.py                             4      0      2      0   100%
companion_api\migrations\0001_initial.py          5      0      2      0   100%
companion_api\migrations\__init__.py              0      0      0      0   100%
companion_api\models.py                           8      0      2      0   100%
companion_api\serializers.py                      6      0      4      0   100%
companion_api\urls.py                             7      0      0      0   100%
companion_api\views.py                            9      0      4      0   100%
core\__init__.py                                  0      0      0      0   100%
core\asgi.py                                      4      4      0      0     0%   10-16
core\settings.py                                 19      0      0      0   100%
core\urls.py                                      3      0      0      0   100%
core\wsgi.py                                      4      4      0      0     0%   10-16
manage.py                                        12      2      2      1    79%   12-13, 21->exit
-----------------------------------------------------------------------------------------
TOTAL                                            82     10     16      1    89%
```

Figure 8.1.1.2: Test Coverage Output Using Python Coverage Tool

Coverage for **manage.py**: 79%

12 statements | 10 run | 2 missing | 0 excluded | 1 partial

```
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks."""
3  import os
4  import sys
5
6
7  def main():
8      """Run administrative tasks."""
9      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
10     try:
11         from django.core.management import execute_from_command_line
12     except ImportError as exc:
13         raise ImportError(
14             "Couldn't import Django. Are you sure it's installed and "
15             "available on your PYTHONPATH environment variable? Did you "
16             "forget to activate a virtual environment?"
17         ) from exc
18     execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
```

Figure 8.1.1.3: HTML Coverage Report Output Example

Figure 8.1.1.2 demonstrates the test coverage using the Coverage Python tool that works in conjunction with unittest. It shows various important coverage statistics such as total number of statements vs missed statements, branches vs partial branch coverage, etc. Thus it is a very useful and concise tool to show test coverage. Furthermore, *Figure 8.1.1.3* shows an example of the HTML coverage

report that *Coverage* can generate as well, giving the same output as *Figure 8.1.1.2* but also allowing you to click on each Class covered and view exactly the coverage that occurred.

Advantages:

- Simple to use
- Django's unit testing library uses it
- Works with *coverage*, which can generate an HTML report showing skipped lines and partially executed branches.
- Provides automatic mocking of databases
- Works with GitHub actions, allowing all unit tests to automatically run on **every commit.**

Disadvantages:

- Command line based execution only
- No graph in coverage

8.1.2 Frontend: React-scripts

React-scripts is a build suite designed to configure React projects. Built into it is a unit testing tool which, without significant configuration, runs tests for the project's frontend using Jest, a JavaScript Testing Framework. As React ships with this build suite by default, it is both the most efficient and least conflicting suite to test our project with, and it is for this reason which we do so. *Figure 8.4.1* depicts a sample test output, which indicates the number of tests and test suites passed, along with the total number of tests, and the time it took to complete the tests.

Advantages:

- Simple to set up (npm install react-scripts)
- Easy to run (npm test is redirected to react-scripts test, and can be run automatically on build)
- Indicates where the test(s) failed

Disadvantages:

- Tests are not immediately obvious to set up
- Rendering pages to test for elements can be counter-intuitive
- Number of failed tests is not shown
- Requires command line execution

8.2 Alternative Tools Researched

8.2.1 Pytest

Pytest was another tool that was researched. It provides similar functionality to unittest but requires less imports and special classes. Furthermore, it can speed up testing by running multiple tests in parallel. This is unlike unittest, which can only run tests sequentially. The major downside with Pytest is that it is more of a hassle to set up with Django, which natively is designed with unittest testing in mind. For this reason, as we are using Django in our project, the decision was made not to go with Pytest.

Advantages:

- Requires even less code to work than unittest
- Simple to use
- Support for its own coverage tool “pytest-cov”
- Runs tests in parallel

Disadvantages:

- More work to set up for Django
- Requires command line execution

8.3 Unit Testing

As mentioned in *Section 8.1*, the tool chosen to do unit testing is *unittest*. There are two ways in which unit tests are run for this project: The first is via the batch file TestRunner.bat, which a developer can use to run all the relevant tests and generate an HTML report with test coverage. Furthermore, with the help of GitHub Actions, every time a commit is made to a branch, all the unit tests are automatically run and their pass/fail status is reflected on GitHub (See *Section 8.3.2* for special details).

*Note, while many of these tests deal with a database, we considered them as unit tests instead of integration tests since it is deeply linked with the database and they do not appear as separate modules. Please see *Section 8.3.1* for specifics on definitions for Integration Testing.

Test Class	Test Details:	Result	Sprint Added
backend.tests.test_companion_api.test_doctor_patient_views.TestDoctorPatientViews	Doctor_Patient: Doctor can get their temp patients when there is one	**PASS**	**5**
backend.tests.test_companion_api.test_models.ModelTestCases	Test superuser can access ReassignPatientsToTempDoctorView	**PASS**	**5**
backend.tests.test_companion_api.test_queryset.TestQuerySet	query_set: Testing query set of PatientsWithMatchingAddressView is successfully called when patient is authenticated	**PASS**	**5**
backend.tests.test_accounts.test_approval.TestApprovalViews	Approval: Testing activate email view to ensure a patient's email can be verified	**PASS**	**5**
backend.tests.test_accounts.test_approval.TestApprovalViews	Approval: Testing approve immigration officer view	**PASS**	**5**

backend.tests.test_accounts.test_approval.TestApprovalViews	Approval: Testing approve doctor view	**PASS**	**5**
backend.tests.test_companion_api.test_doctor_patient_views.TestDoctorPatientViews	Doctor_Patient: Patient can get their doctor	**PASS**	**5**
backend.test_companion_api.test_toggle_views.TestToggleViews	Toggle Doctor Is Away: Doctor can toggle if they are away	**PASS**	**5**
backend.test_companion_api.test_toggle_views.TestToggleViews	Toggle Immigration Priority: Immigration Officer can toggle their immigrant priority	**PASS**	**5**
backend.test_companion_api.test_toggle_views.TestToggleViews	Toggle Patient Priority: Doctor can toggle their patient priority	**PASS**	**5**
backend.test_companion_api.test_notification_views.TestNotificationViews	Notification: Patient can get all their availabilities when there is n	**PASS**	**5**
backend.test_companion_api.test_notification_views.TestNotificationViews	Notification: Patient can get all their notifications when there is one	**PASS**	**5**
backend.test_companion_api.test_notification_views.TestNotificationViews	Notification: Patient cannot get their notifications when there are none	**PASS**	**5**
backend.test_companion_api.test_immigrationOffficer_patient__views.TestImmigrationOfficerImmigrantViews	ImmigrationOfficer_Immigrant: Immigration Officer can get their immigrants when there are n	**PASS**	**5**
backend.test_companion_api.test_immigrationOffficer_patient__views.TestImmigrationOfficerImmigrantViews	ImmigrationOfficer_Immigrant: Immigration Officer can get their immigrants when there is one	**PASS**	**5**
backend.test_companion_api.test_immigrationOffficer_patient__views.TestImmigrationOfficerImmigrantViews	mmigrationOfficer_Immigrant: Immigration Officer can get their immigrants when there are none	**PASS**	**5**
backend.test_companion_api.test_availaibility_views.TestAvailabilityViews	Availability: Doctor can get all their availabilities when there is n	**PASS**	**5**
backend.test_companion_api.test_availaibility_views.TestAvailabilityViews	Availability: Doctor can get all their availabilities when there is one	**PASS**	**5**
backend.test_companion_api.test_availaibility_views.TestAvailabilityViews	Availability: Doctor cannot get their availabilities when there are none	**PASS**	**5**

backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Patient can get all their appointments when there is n	**PASS**	**5**
backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Doctor can get all their appointments when there is n	**PASS**	**5**
backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Patient can get all their appointments when there is one	**PASS**	**5**
backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Doctor can get all their appointments when there is one	**PASS**	**5**
backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Patient cannot get their appointments when there are none	**PASS**	**5**
backend.test_companion_api.test_appointment_views.TestAppointmentViews	Appointment: Doctor cannot get their appointments when there are none	**PASS**	**5**
backend.test_companion_api.test_address_views.TestAddressViews	Address: Patient can get all their availabilities when there is n	**PASS**	**5**
backend.test_companion_api.test_address_views.TestAddressViews	Address: Patient can get all their addresses when there is one	**PASS**	**5**
backend.test_companion_api.test_address_views.TestAddressViews	Address: Patient cannot get their addresses when there are none	**PASS**	**5**
backend.tests.test_companion_api.test_models.ModelTestCases	Admin: Ensure superuser (Admin) can properly be created with valid email and a password	**PASS**	**3**
backend.tests.test_companion_api.test_models.ModelTestCases	Admin: Ensure superuser (Admin) cannot be created when is_staff is set to false	**PASS**	**3**
backend.tests.test_companion_api.test_models.ModelTestCases	Admin: Ensure superuser (Admin) cannot be created when is_superuser is set to false	**PASS**	**3**
backend.tests.test_companion_api.test_models.ModelTestCases	Admin: Ensure superuser (Admin) cannot be created without an email	**PASS**	**3**
backend.tests.test_companion_api.test_doctor_patient_views.TestDoctorPatientViews	Doctor_Patient: Doctor can get their patients when there are n	**PASS**	**3**

backend.tests.test_companion_api.test_doctor_patient_views.TestDoctorPatientViews	Doctor_Patient: Doctor can get their patients when there are none	**PASS**	**3**
backend.tests.test_companion_api.test_doctor_patient_views.TestDoctorPatientViews	Doctor_Patient: Doctor can get their patients when there is one	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Doctor can get their Patient latest status when there is n	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Doctor can get their Patient latest status when there is none	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Doctor can get their Patient latest status when there is one	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Patient can get all their status when there is n	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Patient can get their status when there are none	**PASS**	**3**
backend.tests.test_companion_api.test_status_views.TestStatusViews	Status: Patient can get all their status when there is one	**PASS**	**3**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with no data	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor can register with correct data	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with no User (every Doctor contains a User)	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with no email attribute	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with no password attribute	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with an invalid email format	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Registration: Doctor cannot register with an existing email (emails are unique)	**PASS**	**2**

backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Login: Doctor can login once registered	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Login: Doctor cannot login if not registered	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	User View: Return a valid token for the Doctor when authorized	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	User View: If no token for the Doctor, we are unauthorized	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	User View: Invalid token means Doctor is unauthorized	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Logout View: Doctor can logout when they have a valid token	**PASS**	**2**
backend.tests.test_accounts.test_doctor_views.TestDoctorViews	Logout View: Doctor cannot logout when they do not have a valid token	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with no data	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient can register with correct data	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with no User (every Patient contains a User)	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with no email attribute	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with no password attribute	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with an invalid email format	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Registration: Patient cannot register with an existing email (emails are unique)	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Login: Patient can login once registered	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Login: Patient cannot login if not registered	**PASS**	**2**

tests.test_accounts.test_patient_views.TestPatientViews	User View: Return a valid token for the Patient when authorized	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	User View: If no token for the Patient, we are unauthorized	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	User View: Invalid token means Patient is unauthorized	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Logout View: Patient can logout when they have a valid token	**PASS**	**2**
tests.test_accounts.test_patient_views.TestPatientViews	Logout View: Patient cannot logout when they do not have a valid token	**PASS**	**2**
tests.test_companion_api.test_models.ModelTestCases	Check that the people are properly stored in the database	**PASS**	**1**
Summary		**70/70 tests passed**	

Tests added Sprint 2: **+28 (2800% increase in tests since previous sprint)**

Tests added Sprint 3: **+13 (45% increase in tests since previous sprint)**

Tests added Sprint 4: **0**

Tests added Sprint 5: **+28 (67% increase in tests since sprint 3/4)**

8.3.1 Important Note on Integration Testing

An important point that needs to be specifically addressed is that we do not have any explicit “integration tests” in our test suite. While this may seem strange, this is related to how the Django framework deals with testing: The library being used is a Django modified unittest library, so all the tests written look as if they are unit tests and we deal with them via very simple method calls, but in fact all these calls are intrinsically linked with the serializers, database and other implicit calls that occur in the runtime of the system.

```
def test_superuser_can_be_created(self):
    """Admin: Ensure superuser (Admin) can properly be created with valid email and a password"""
    email = "admin@email.com"
    superuser = User.objects.create_superuser(email,self.fake.password())
    #Ensure that the superuser was created with the right email
    self.assertEqual(superuser.email,email)
```

Figure 8.3.1.1 Example of “Unit Test”

Take for instance the simple test illustrated in *Figure 8.3.1.1*, it is simply ensuring that a super user can be created which is something that is absolutely in the domain of a unit test in a language like *Java* for example, however in Django the simple command to create a superuser interfaces with the serializer, does various implicit checks in the framework and interfaces with a special database, temporarily created whenever a unit test suite runs. Considering these aspects, this should be defined as an *Integration Test*, but as is visible from the figure, the test itself is extremely simple and only tests a single unit.

It is for this reason that we decided to stick with calling our tests “unit tests”, because in respect to Django’s testing setup, they are the simplest units we can test. At the same time they do cover all the aspects of an integration test, and thus we are not too concerned to create separate integration tests.

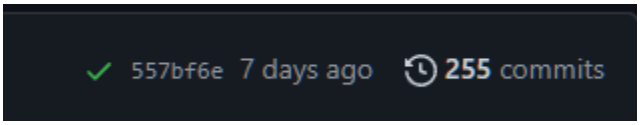
Another critical detail to note is that these unit tests run on every commit and PR in our GitHub repository via GitHub Actions. So we are constantly doing integration testing by testing every merge of our system and the effect of every PR.

8.3.2 GitHub Actions for Integration Testing

As mentioned prior, we are using GitHub Actions for our efforts of testing the integration of new components of our system and running all of our unit tests. In short, GitHub Actions are a sort of cloud hosted VM that perform a set of instructions based on details provided in a special YAML file. GitHub Actions are incredibly useful and help provide CI functionality natively and for free on GitHub repos.

```yaml
name: Django CI

on:
  push:
  pull_request:

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      max-parallel: 4
      matrix:
        python-version: [3.9]

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install Dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r backend/requirements.txt
    - name: Run Tests
      run: |
        cd backend
        coverage run --branch --source='.' manage.py test tests -v 2
        coverage report -m  --omit="*/test*,*/core*"

```

Figure 8.3.2.1: YAML file for GitHub Action



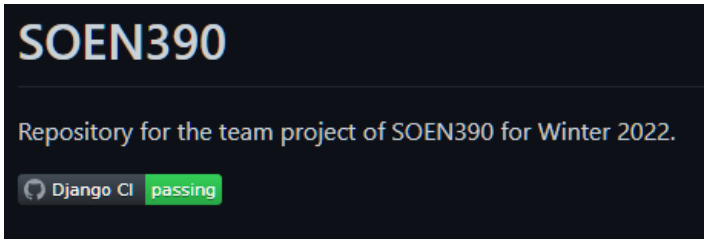


Figure 8.3.2.2: Status of Tests Passing/Failing on Commit in GitHub

Figure 8.3.2.2 illustrates some of the ways that we have visibility on whether the tests passed on integration from a PR or from a commit in a separate branch. As the tests are running, we get a yellow dot, a pass from the Action gives a green checkmark (and a passing flag in the README) and a failure gives a red X.

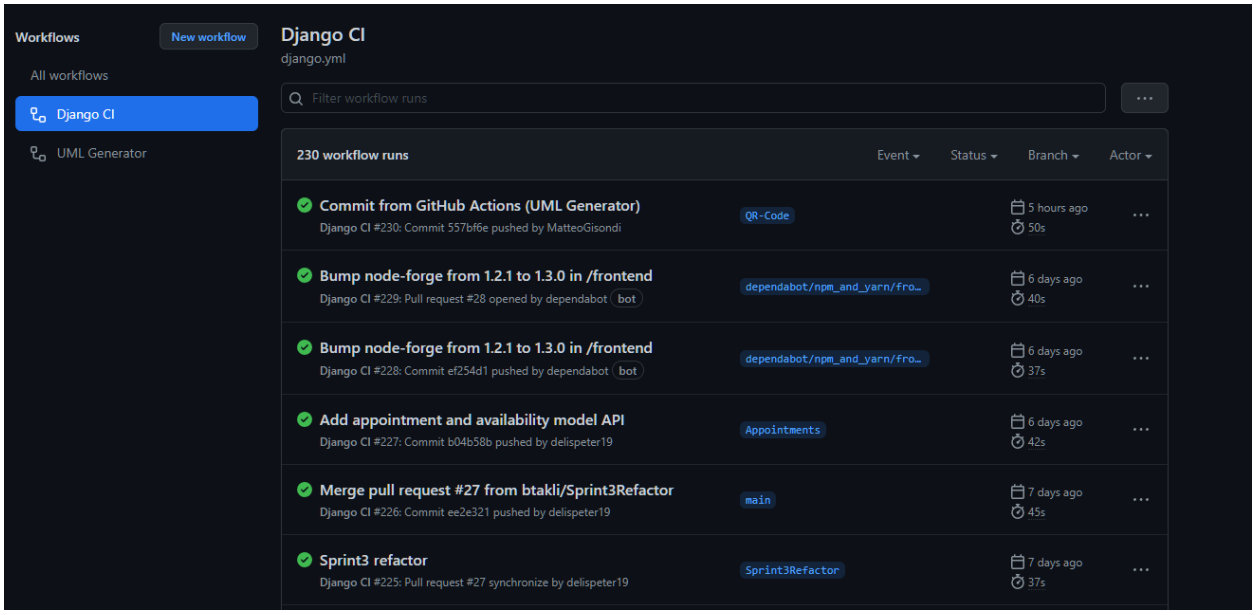


Figure 8.3.2.3: GitHub Actions History

You may also view a history of past runs on GitHub as well, providing excellent visibility for the team and allowing for details as to why tests failed (See *Figure 8.3.2.4* for visibility of logs of execution).

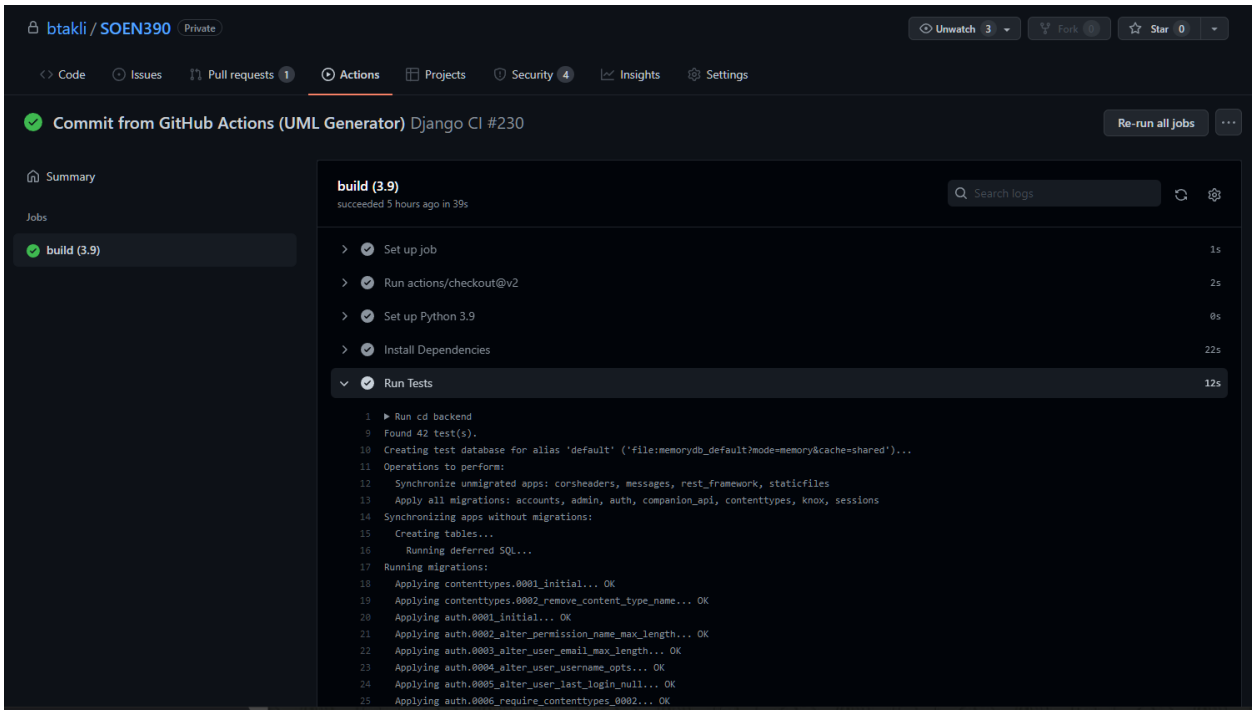


Figure 8.3.2.4: Logs and Views of Steps for GitHub Actions Run

8.4 Test code coverage

8.4.1 Backend Coverage

Sprint 2 Notes:

The backend coverage report is automatically generated in the logs of the GitHub action on every commit and pull request. In order to generate the nicely formatted HTML report as shown below, TestRunner.bat must be used. With 29 tests so far, test coverage is high at 85%. While a coverage decrease of 2% seems rather strange for 28 additional tests, it is very important to note that the files themselves grew a great deal during this sprint. Thus we would have been happy to maintain our previous coverage of 87%, but this near-stability is excellent news and a good sign that our tests are effectively covering the desired behavior.

A lot of work has been done to cover both “happy paths” and “unhappy paths” for the tests. The core functionality is all covered in unit tests and thus increases our confidence in the reliability of the features. Much of the missed elements are related to exceptions being thrown or if statements that should never be evaluated as false. Many of the classes with missed statements are all simple and related to basic Django setup, thus there is no issue with missed branches/classes and we are not concerned about this. However, we do wish to increase coverage for accounts\admin.py and accounts\models.py as these are core classes.

Sprint 3 Notes:

Sprint 3 we managed to get a test coverage of 88% on the backend, which is a decent improvement from last sprint. We are still happy that we managed to maintain approximately the same coverage despite adding over 100 statements to the backend since last time. We experienced more pressure than expected this time around and plan to look into developing more complete testing for next sprint, but all the core functionality is tested and so we are not too concerned.

Sprint 4 Notes:

New efforts were made to increase testing coverage however there was a lack of time to update them properly in this report. The proper values will be displayed in the presentation along with data on new tests. This issue in the documentation will be resolved in *Sprint 5*.

Sprint 5 Notes:

A massive amount of tests were added this sprint, the single largest amount of tests since *Sprint 1*. Paired with the refactoring, our coverage remained more or less the same as *Sprint 4*, but we do not consider this to be an issue as most core logic is covered properly and many of the missing statements are not critical to test.

Coverage report: 87%

Module	statements	missing	excluded	branches	partial	coverage
companion_api__init__.py	0	0	0	0	0	100%
companion_api\admin.py	3	1	0	2	1	60%
companion_api\apps.py	4	0	0	2	0	100%
companion_api\migrations\0001_initial.py	5	0	0	2	0	100%
companion_api\migrations__init__.py	0	0	0	0	0	100%
companion_api\models.py	6	0	0	2	0	100%
companion_api\serializers.py	6	0	0	4	0	100%
companion_api\urls.py	7	0	0	0	0	100%
companion_api\views.py	8	0	0	4	0	100%
core__init__.py	0	0	0	0	0	100%
core\asgi.py	4	4	0	0	0	0%
core\settings.py	19	0	0	0	0	100%
core\urls.py	3	0	0	0	0	100%
core\wsgi.py	4	4	0	0	0	0%
manage.py	12	2	0	2	1	79%
Total	**81**	**11**	**0**	**18**	**2**	**87%**

Figure 8.4.1.1 Backend Coverage Report Sprint 1

Coverage report: 85%

Module	statements	missing	excluded	branches	partial	coverage
__init__.py	0	0	0	0	0	100%
accounts__init__.py	0	0	0	0	0	100%
accounts\admin.py	37	9	0	6	0	74%
accounts\apps.py	4	0	0	2	0	100%
accounts\migrations\0001_initial.py	8	0	0	2	0	100%
accounts\migrations\0002_user_is_pending.py	4	0	0	2	0	100%
accounts\migrations\0003_rename_is_pending_user_is_pending_approval_and_more.py	4	0	0	2	0	100%
accounts\migrations\0004_doctor_address_doctor_city_doctor_date_of_birth_and_more.py	5	0	0	2	0	100%
accounts\migrations\0005_patient_address_patient_city_patient_date_of_birth_and_more.py	4	0	0	2	0	100%
accounts\migrations\0006_alter_doctor_first_name_alter_patient_first_name.py	4	0	0	2	0	100%
accounts\migrations__init__.py	0	0	0	0	0	100%
accounts\models.py	60	12	0	14	1	77%
accounts\serializers.py	66	10	0	30	1	89%
accounts\templatetags\frontend_site_url.py	6	2	0	0	0	67%
accounts\urls.py	10	0	0	0	0	100%
accounts\utils.py	5	1	0	2	0	86%
accounts\views.py	91	22	0	22	0	79%
companion_api__init__.py	0	0	0	0	0	100%
companion_api\admin.py	5	1	0	2	1	71%
companion_api\apps.py	4	0	0	2	0	100%
companion_api\migrations\0001_initial.py	7	0	0	2	0	100%
companion_api\migrations__init__.py	0	0	0	0	0	100%
companion_api\models.py	8	0	0	2	0	100%
companion_api\serializers.py	6	0	0	4	0	100%
companion_api\urls.py	7	0	0	0	0	100%
companion_api\views.py	10	2	0	2	0	83%
manage.py	12	2	0	2	1	79%
Total	**367**	**61**	**0**	**104**	**4**	**85%**

Figure 8.4.1.2 Backend Coverage Report Sprint 2

Coverage report: 88%

Module	statements	missing	excluded	branches	partial	coverage
__init__.py	0	0	0	0	0	100%
accounts__init__.py	0	0	0	0	0	100%
accounts\admin.py	38	9	0	6	0	75%
accounts\apps.py	4	0	0	2	0	100%
accounts\migrations\0001_initial.py	9	0	0	2	0	100%
accounts\migrations__init__.py	0	0	0	0	0	100%
accounts\models.py	60	4	0	14	0	95%
accounts\serializers.py	66	10	0	30	1	89%
accounts\templatetags\frontend_site_url.py	6	2	0	0	0	67%
accounts\urls.py	6	0	0	0	0	100%
accounts\utils.py	5	1	0	2	0	86%
accounts\views.py	92	22	0	20	0	79%
companion_api__init__.py	0	0	0	0	0	100%
companion_api\admin.py	16	4	0	4	0	70%
companion_api\apps.py	4	0	0	2	0	100%
companion_api\migrations\0001_initial.py	8	0	0	2	0	100%
companion_api\migrations\0002_status_caugh_status_chills_status_diffbreathing_and_more.py	5	0	0	2	0	100%
companion_api\migrations\0003_remove_status_caugh_alter_status_date.py	5	0	0	2	0	100%
companion_api\migrations\0004_alter_status_options_notification.py	6	0	0	2	0	100%
companion_api\migrations\0005_alter_notification_user.py	6	0	0	2	0	100%
companion_api\migrations\0006_alter_notification_user.py	6	0	0	2	0	100%
companion_api\migrations__init__.py	0	0	0	0	0	100%
companion_api\models.py	38	1	0	8	0	98%
companion_api\serializers.py	17	0	0	12	0	100%
companion_api\urls.py	11	0	0	0	0	100%
companion_api\views.py	53	8	0	14	0	88%
manage.py	12	2	0	2	1	79%
Total	**473**	**63**	**0**	**130**	**2**	**88%**

Figure 8.4.1.3 Backend Coverage Report Sprint 3

Coverage report: 85%

Module	statements	missing	excluded	branches	partial	coverage
__init__.py	0	0	0	0	0	100%
accounts__init__.py	0	0	0	0	0	100%
accounts\admin.py	38	9	0	6	0	75%
accounts\apps.py	4	0	0	2	0	100%
accounts\migrations\0001_initial.py	9	0	0	2	0	100%
accounts\migrations\0002_patient_is_priority.py	4	0	0	2	0	100%
accounts\migrations__init__.py	0	0	0	0	0	100%
accounts\models.py	61	4	0	14	0	95%
accounts\serializers.py	66	10	0	30	1	89%
accounts\templatetags\frontend_site_url.py	6	2	0	0	0	67%
accounts\urls.py	6	0	0	0	0	100%
accounts\utils.py	5	1	0	2	0	86%
accounts\views.py	92	22	0	20	0	79%
companion_api__init__.py	0	0	0	0	0	100%
companion_api\admin.py	16	4	0	4	0	70%
companion_api\apps.py	4	0	0	2	0	100%
companion_api\migrations\0001_initial.py	8	0	0	2	0	100%
companion_api\migrations__init__.py	0	0	0	0	0	100%
companion_api\models.py	46	1	0	10	0	98%
companion_api\serializers.py	21	0	0	16	0	100%
companion_api\urls.py	12	0	0	0	0	100%
companion_api\views.py	85	26	0	26	0	71%
manage.py	12	2	0	2	1	79%
Total	**495**	**81**	**0**	**140**	**2**	**85%**

Figure 8.4.1.3 Backend Coverage Report Sprint 4

Coverage report: 86%

Module	statements	missing	excluded	branches	partial	coverage
__init__.py	0	0	0	0	0	100%
accounts__init__.py	0	0	0	0	0	100%
accounts\admin.py	49	12	0	8	0	75%
accounts\apps.py	4	0	0	2	0	100%
accounts\migrations\0001_initial.py	9	0	0	2	0	100%
accounts\migrations__init__.py	0	0	0	0	0	100%
accounts\models.py	85	5	0	16	0	95%
accounts\no_permission_serializers.py	42	0	0	12	0	100%
accounts\no_permission_views.py	33	0	0	6	0	100%
accounts\serializers.py	83	32	0	36	0	70%
accounts\templatetags\frontend_site_url.py	6	2	0	0	0	67%
accounts\urls.py	7	0	0	0	0	100%
accounts\utils.py	23	10	0	2	0	60%
accounts\views.py	124	28	0	34	4	80%
companion_api__init__.py	0	0	0	0	0	100%
companion_api\admin.py	17	4	0	4	0	71%
companion_api\apps.py	4	0	0	2	0	100%
companion_api\migrations\0001_initial.py	8	0	0	2	0	100%
companion_api\migrations\0002_rename_date_appointment_start_appointment_end.py	4	0	0	2	0	100%
companion_api\migrations__init__.py	0	0	0	0	0	100%
companion_api\models.py	72	5	0	22	4	90%
companion_api\serializers.py	34	0	0	26	0	100%
companion_api\urls.py	14	0	0	0	0	100%
companion_api\views.py	182	20	0	52	5	88%
manage.py	12	2	0	2	1	79%
Total	**812**	**120**	**0**	**230**	**14**	**86%**

Figure 8.4.1.4 Backend Coverage Report Sprint 5

8.4.2 Frontend Coverage

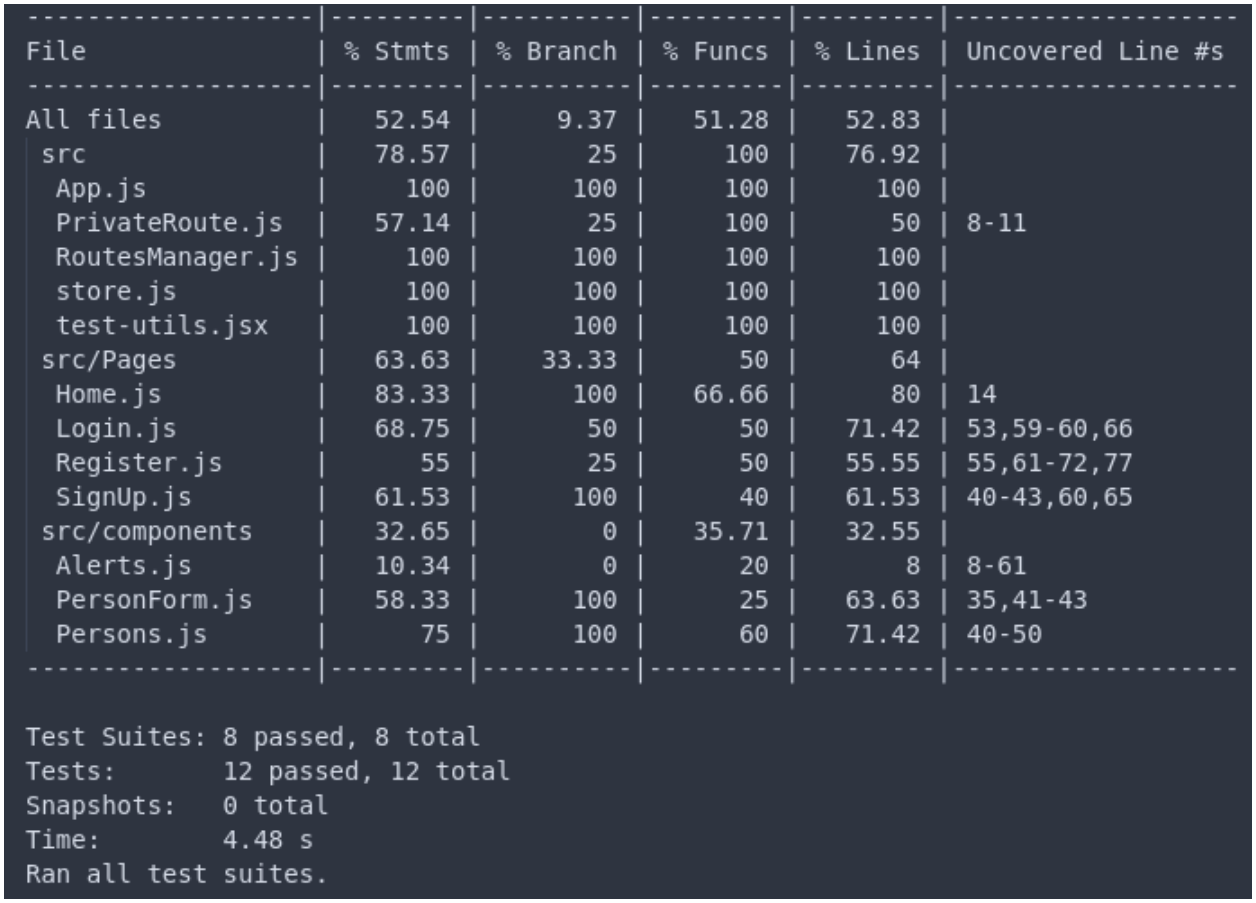
Code coverage is automatically generated with the --coverage flag modifier in react-scripts. Contained in the *Figure 8.4.1* report are various fields: % Stmts (% Statements covered), % Branch (% Branches covered), % funcs (% functions covered), % Lines (% Lines of Code covered), and Uncovered Line #s (lines which are not covered by any existing tests).

Considerations:

- May not achieve full coverage without a large number of tests
- Coverage report is very minimal and not-interactive

```
 PASS  src/App.test.js
  ✓ renders without crashing (42ms)

----------|----------|----------|----------|----------|-------------------|
File      |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
----------|----------|----------|----------|----------|-------------------|
All files |        0 |        0 |        0 |        0 |                   |
----------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.242s
```

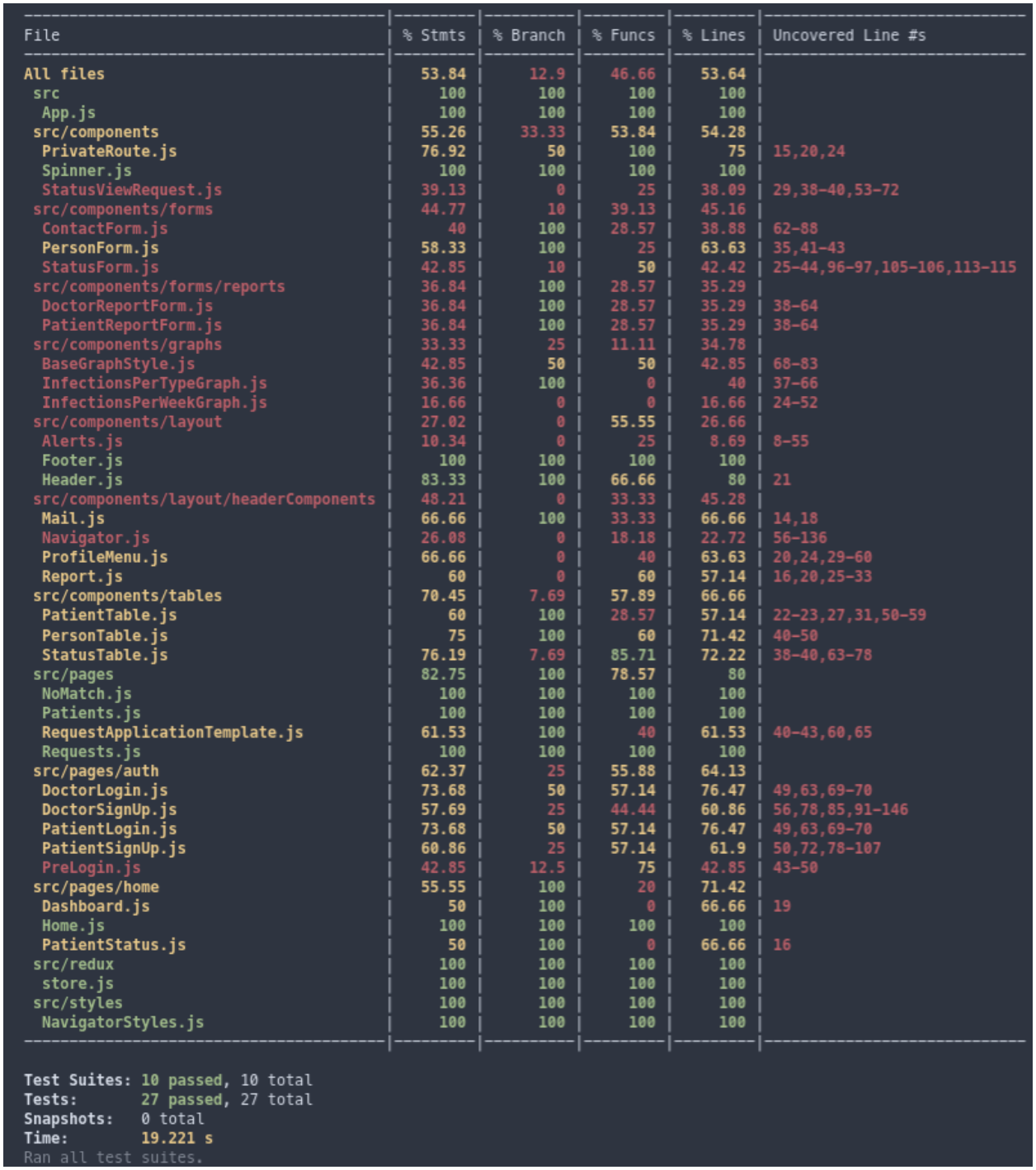
Figure 8.4.2.1: Frontend Coverage Report Sprint 1



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	52.54	9.37	51.28	52.83	
src	78.57	25	100	76.92	
App.js	100	100	100	100	
PrivateRoute.js	57.14	25	100	50	8-11
RoutesManager.js	100	100	100	100	
store.js	100	100	100	100	
test-utils.jsx	100	100	100	100	
src/Pages	63.63	33.33	50	64	
Home.js	83.33	100	66.66	80	14
Login.js	68.75	50	50	71.42	53,59-60,66
Register.js	55	25	50	55.55	55,61-72,77
SignUp.js	61.53	100	40	61.53	40-43,60,65
src/components	32.65	0	35.71	32.55	
Alerts.js	10.34	0	20	8	8-61
PersonForm.js	58.33	100	25	63.63	35,41-43
Persons.js	75	100	60	71.42	40-50

Test Suites: 8 passed, 8 total
Tests: 12 passed, 12 total
Snapshots: 0 total
Time: 4.48 s
Ran all test suites.

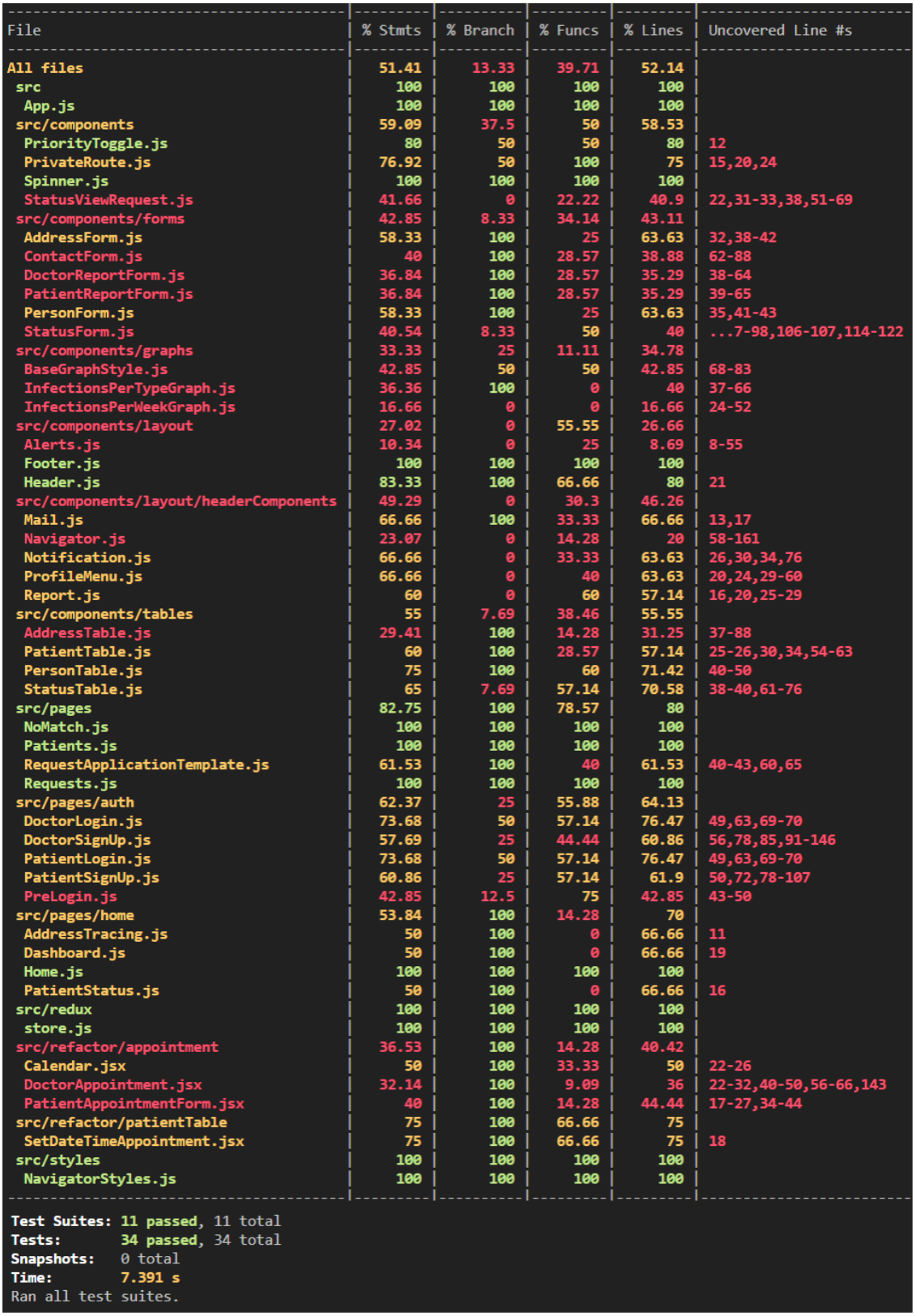
Figure 8.4.2.2: Frontend Coverage Report Sprint 2



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	53.84	12.9	46.66	53.64	
src	100	100	100	100	
App.js	100	100	100	100	
src/components	55.26	33.33	53.84	54.28	
PrivateRoute.js	76.92	50	100	75	15,20,24
Spinner.js	100	100	100	100	
StatusViewRequest.js	39.13	0	25	38.09	29,38-40,53-72
src/components/forms	44.77	10	39.13	45.16	
ContactForm.js	40	100	28.57	38.88	62-88
PersonForm.js	58.33	100	25	63.63	35,41-43
StatusForm.js	42.85	10	50	42.42	25-44,96-97,105-106,113-115
src/components/forms/reports	36.84	100	28.57	35.29	
DoctorReportForm.js	36.84	100	28.57	35.29	38-64
PatientReportForm.js	36.84	100	28.57	35.29	38-64
src/components/graphs	33.33	25	11.11	34.78	
BaseGraphStyle.js	42.85	50	50	42.85	68-83
InfectionsPerTypeGraph.js	36.36	100	0	40	37-66
InfectionsPerWeekGraph.js	16.66	0	0	16.66	24-52
src/components/layout	27.02	0	55.55	26.66	
Alerts.js	10.34	0	25	8.69	8-55
Footer.js	100	100	100	100	
Header.js	83.33	100	66.66	80	21
src/components/layout/headerComponents	48.21	0	33.33	45.28	
Mail.js	66.66	100	33.33	66.66	14,18
Navigator.js	26.08	0	18.18	22.72	56-136
ProfileMenu.js	66.66	0	40	63.63	20,24,29-60
Report.js	60	0	60	57.14	16,20,25-33
src/components/tables	70.45	7.69	57.89	66.66	
PatientTable.js	60	100	28.57	57.14	22-23,27,31,50-59
PersonTable.js	75	100	60	71.42	40-50
StatusTable.js	76.19	7.69	85.71	72.22	38-40,63-78
src/pages	82.75	100	78.57	80	
NoMatch.js	100	100	100	100	
Patients.js	100	100	100	100	
RequestApplicationTemplate.js	61.53	100	40	61.53	40-43,60,65
Requests.js	100	100	100	100	
src/pages/auth	62.37	25	55.88	64.13	
DoctorLogin.js	73.68	50	57.14	76.47	49,63,69-70
DoctorSignUp.js	57.69	25	44.44	60.86	56,78,85,91-146
PatientLogin.js	73.68	50	57.14	76.47	49,63,69-70
PatientSignUp.js	60.86	25	57.14	61.9	50,72,78-107
PreLogin.js	42.85	12.5	75	42.85	43-50
src/pages/home	55.55	100	20	71.42	
Dashboard.js	50	100	0	66.66	19
Home.js	100	100	100	100	
PatientStatus.js	50	100	0	66.66	16
src/redux	100	100	100	100	
store.js	100	100	100	100	
src/styles	100	100	100	100	
NavigatorStyles.js	100	100	100	100	

Test Suites: 10 passed, 10 total
Tests: 27 passed, 27 total
Snapshots: 0 total
Time: 19.221 s
Ran all test suites.

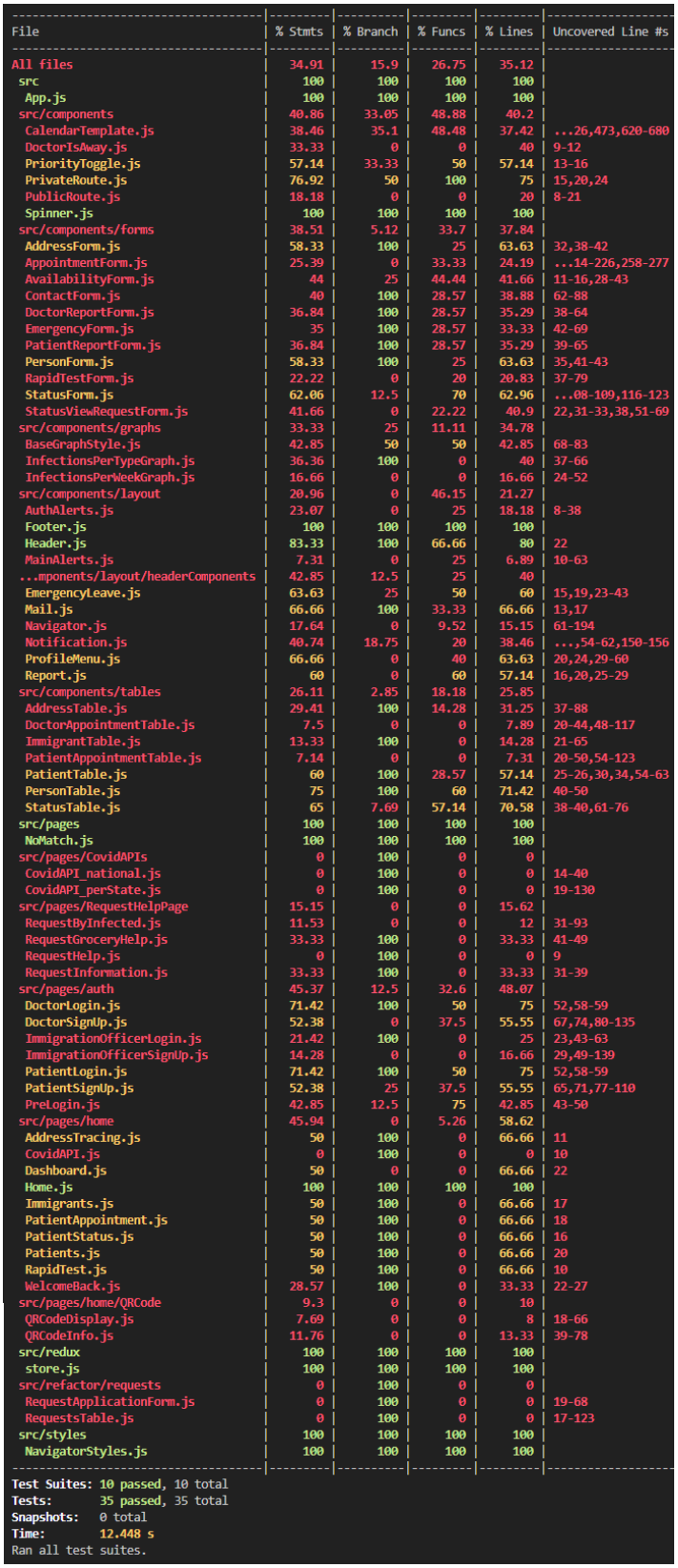
Figure 8.4.2.3: Frontend Coverage Report Sprint 3



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	51.41	13.33	39.71	52.14	
src	100	100	100	100	
App.js	100	100	100	100	
src/components	59.09	37.5	50	58.53	
PriorityToggle.js	80	50	50	80	12
PrivateRoute.js	76.92	50	100	75	15,20,24
Spinner.js	100	100	100	100	
StatusViewRequest.js	41.66	0	22.22	40.9	22,31-33,38,51-69
src/components/forms	42.85	8.33	34.14	43.11	
AddressForm.js	58.33	100	25	63.63	32,38-42
ContactForm.js	40	100	28.57	38.88	62-88
DoctorReportForm.js	36.84	100	28.57	35.29	38-64
PatientReportForm.js	36.84	100	28.57	35.29	39-65
PersonForm.js	58.33	100	25	63.63	35,41-43
StatusForm.js	40.54	8.33	50	40	...7-98,106-107,114-122
src/components/graphs	33.33	25	11.11	34.78	
BaseGraphStyle.js	42.85	50	50	42.85	68-83
InfectionsPerTypeGraph.js	36.36	100	0	40	37-66
InfectionsPerWeekGraph.js	16.66	0	0	16.66	24-52
src/components/layout	27.02	0	55.55	26.66	
Alerts.js	10.34	0	25	8.69	8-55
Footer.js	100	100	100	100	
Header.js	83.33	100	66.66	80	21
src/components/layout/headerComponents	49.29	0	30.3	46.26	
Mail.js	66.66	100	33.33	66.66	13,17
Navigator.js	23.07	0	14.28	20	58-161
Notification.js	66.66	0	33.33	63.63	26,30,34,76
ProfileMenu.js	66.66	0	40	63.63	20,24,29-60
Report.js	60	0	60	57.14	16,20,25-29
src/components/tables	55	7.69	38.46	55.55	
AddressTable.js	29.41	100	14.28	31.25	37-88
PatientTable.js	60	100	28.57	57.14	25-26,30,34,54-63
PersonTable.js	75	100	60	71.42	40-50
StatusTable.js	65	7.69	57.14	70.58	38-40,61-76
src/pages	82.75	100	78.57	80	
NoMatch.js	100	100	100	100	
Patients.js	100	100	100	100	
RequestApplicationTemplate.js	61.53	100	40	61.53	40-43,60,65
Requests.js	100	100	100	100	
src/pages/auth	62.37	25	55.88	64.13	
DoctorLogin.js	73.68	50	57.14	76.47	49,63,69-70
DoctorSignUp.js	57.69	25	44.44	60.86	56,78,85,91-146
PatientLogin.js	73.68	50	57.14	76.47	49,63,69-70
PatientSignUp.js	60.86	25	57.14	61.9	50,72,78-107
PreLogin.js	42.85	12.5	75	42.85	43-50
src/pages/home	53.84	100	14.28	70	
AddressTracing.js	50	100	0	66.66	11
Dashboard.js	50	100	0	66.66	19
Home.js	100	100	100	100	
PatientStatus.js	50	100	0	66.66	16
src/redux	100	100	100	100	
store.js	100	100	100	100	
src/refactor/appointment	36.53	100	14.28	40.42	
Calendar.jsx	50	100	33.33	50	22-26
DoctorAppointment.jsx	32.14	100	9.09	36	22-32,40-50,56-66,143
PatientAppointmentForm.jsx	40	100	14.28	44.44	17-27,34-44
src/refactor/patientTable	75	100	66.66	75	
SetDateTimeAppointment.jsx	75	100	66.66	75	18
src/styles	100	100	100	100	
NavigatorStyles.js	100	100	100	100	

Test Suites: 11 passed, 11 total
Tests: 34 passed, 34 total
Snapshots: 0 total
Time: 7.391 s
Ran all test suites.

Figure 8.4.2.4: Frontend Coverage Report Sprint 4



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	34.91	15.9	26.75	35.12	
src	100	100	100	100	
App.js	100	100	100	100	
src/components	40.86	33.05	48.88	40.2	
CalendarTemplate.js	38.46	35.1	48.48	37.42	...26,473,620-680
DoctorIsAway.js	33.33	0	0	40	9-12
PriorityToggle.js	57.14	33.33	50	57.14	13-16
PrivateRoute.js	76.92	50	100	75	15,20,24
PublicRoute.js	18.18	0	0	20	8-21
Spinner.js	100	100	100	100	
src/components/forms	38.51	5.12	33.7	37.84	
AddressForm.js	58.33	100	25	63.63	32,38-42
AppointmentForm.js	25.39	0	33.33	24.19	...14-226,258-277
AvailabilityForm.js	44	25	44.44	41.66	11-16,28-43
ContactForm.js	40	100	28.57	38.88	62-88
DoctorReportForm.js	36.84	100	28.57	35.29	38-64
EmergencyForm.js	35	100	28.57	33.33	42-69
PatientReportForm.js	36.84	100	28.57	35.29	39-65
PersonForm.js	58.33	100	25	63.63	35,41-43
RapidTestForm.js	22.22	0	20	20.83	37-79
StatusForm.js	62.06	12.5	70	62.96	...08-109,116-123
StatusViewRequestForm.js	41.66	0	22.22	40.9	22,31-33,38,51-69
src/components/graphs	33.33	25	11.11	34.78	
BaseGraphStyle.js	42.85	50	50	42.85	68-83
InfectionsPerTypeGraph.js	36.36	100	0	40	37-66
InfectionsPerWeekGraph.js	16.66	0	0	16.66	24-52
src/components/layout	20.96	0	46.15	21.27	
AuthAlerts.js	23.07	0	25	18.18	8-38
Footer.js	100	100	100	100	
Header.js	83.33	100	66.66	80	22
MainAlerts.js	7.31	0	25	6.89	10-63
...mponents/layout/headerComponents	42.85	12.5	25	40	
EmergencyLeave.js	63.63	25	50	60	15,19,23-43
Mail.js	66.66	100	33.33	66.66	13,17
Navigator.js	17.64	0	9.52	15.15	61-194
Notification.js	40.74	18.75	20	38.46	...,54-62,150-156
ProfileMenu.js	66.66	0	40	63.63	20,24,29-60
Report.js	60	0	60	57.14	16,20,25-29
src/components/tables	26.11	2.85	18.18	25.85	
AddressTable.js	29.41	100	14.28	31.25	37-88
DoctorAppointmentTable.js	7.5	0	0	7.89	20-44,48-117
ImmigrantTable.js	13.33	100	0	14.28	21-65
PatientAppointmentTable.js	7.14	0	0	7.31	20-50,54-123
PatientTable.js	60	100	28.57	57.14	25-26,30,34,54-63
PersonTable.js	75	100	60	71.42	40-50
StatusTable.js	65	7.69	57.14	70.58	38-40,61-76
src/pages	100	100	100	100	
NoMatch.js	100	100	100	100	
src/pages/CovidAPIs	0	100	0	0	
CovidAPI_national.js	0	100	0	0	14-40
CovidAPI_perState.js	0	100	0	0	19-130
src/pages/RequestHelpPage	15.15	0	0	15.62	
RequestByInfected.js	11.53	0	0	12	31-93
RequestGroceryHelp.js	33.33	100	0	33.33	41-49
RequestHelp.js	0	100	0	0	9
RequestInformation.js	33.33	100	0	33.33	31-39
src/pages/auth	45.37	12.5	32.6	48.07	
DoctorLogin.js	71.42	100	50	75	52,58-59
DoctorSignUp.js	52.38	0	37.5	55.55	67,74,80-135
ImmigrationOfficerLogin.js	21.42	100	0	25	23,43-63
ImmigrationOfficerSignUp.js	14.28	0	0	16.66	29,49-139
PatientLogin.js	71.42	100	50	75	52,58-59
PatientSignUp.js	52.38	25	37.5	55.55	65,71,77-110
PreLogin.js	42.85	12.5	75	42.85	43-50
src/pages/home	45.94	0	5.26	58.62	
AddressTracing.js	50	100	0	66.66	11
CovidAPI.js	0	100	0	0	10
Dashboard.js	50	0	0	66.66	22
Home.js	100	100	100	100	
Immigrants.js	50	100	0	66.66	17
PatientAppointment.js	50	100	0	66.66	18
PatientStatus.js	50	100	0	66.66	16
Patients.js	50	100	0	66.66	20
RapidTest.js	50	100	0	66.66	10
WelcomeBack.js	28.57	100	0	33.33	22-27
src/pages/home/QRCode	9.3	0	0	10	
QRCodeDisplay.js	7.69	0	0	8	18-66
QRCodeInfo.js	11.76	0	0	13.33	39-78
src/redux	100	100	100	100	
store.js	100	100	100	100	
src/refactor/requests	0	100	0	0	
RequestApplicationForm.js	0	100	0	0	19-68
RequestsTable.js	0	100	0	0	17-123
src/styles	100	100	100	100	
NavigatorStyles.js	100	100	100	100	

Test Suites: 10 passed, 10 total
Tests: 35 passed, 35 total
Snapshots: 0 total
Time: 12.448 s
Ran all test suites.

Figure 8.4.2.5: Frontend Coverage Report Sprint 5

8.4.3 Sprint Recap

8.4.3.1 Sprint 2 Recap

As the frontend has undergone significant overhaul, both the number of components and the number of tests has significantly increased. As frontend testing had not been established in Sprint 1, the baseline for Sprint 2's testing coverage has not properly been established. The 8 Test Suites composed of 12 collective tests produce a coverage of 52.54%. This value will increase in future sprints as Redux boilerplate functionality is currently not being extensively tested. This is the reason for which certain lines (i.e. lines 53,59-60,66 in Login.js) are marked as uncovered.

8.4.3.2 Sprint 3 Recap

This sprint, we added a large amount of features of high complexity such as contact tracing and arranging appointments between Doctors and Patients. We underestimated the difficulty of these tasks and as a result did not accomplish all that we had planned to for this sprint. We wanted to deploy our website for our demo, but unfortunately were unable to do so in time as we were focused on having the features we promised work. Despite this, overall the sprint went well and we hope to do better in *Sprint 4*. Furthermore, we more or less maintained the same testing coverage for the backend. We wished to increase our test coverage a bit more, but we are still happy with the results and as our stories to complete get less difficult and numerous, we can divert more effort to increasing our test coverage.

The testing coverage for the frontend has increased significantly, with tests established for all new components of the frontend. Coverage *technically* remains low, however this is largely due to only rendering being tested. Since there was a fairly involved overhaul of the pre-existing file structure, tests had to be migrated and adjusted accordingly, which resulted in some generic tests. Statement coverage is at 53.84%, which may seem low, however this is considered acceptable as the logic is tested in the backend, while this ~50% accounts for all pages being correctly rendered.

8.4.3.3 Sprint 4 Recap

This sprint, all promised features were successfully implemented. While we had some more involved features such as the QR code *Patient* records and *Doctor* emergencies, much of the logic related to *Immigration Officers* was a relatively simple conversion from the existing *Doctor* logic. This sprint marked the implementation of all features promised at the beginning of development. We would have liked to do more testing this sprint, but unfortunately time constraints made this difficult. For *Sprint 5* we plan to increase test coverage and to refactor code changes before we finish this project. Furthermore, the documentation will be thoroughly reviewed to ensure all information is adequate and a closing retrospective of the whole project will be added as well.

Frontend testing did not see any improvement this sprint, but as before we are not too concerned given that all the important logic is tested via our unit tests and thus does not need to be repeated on the frontend side. Furthermore, much of the missed lines, as before, are static lines that need not be tested.

8.4.3.4 Sprint 5 Recap

This sprint focused entirely on refactoring code, improving test coverage and finalizing the documentation. Test coverage improvements were our biggest concern this time around due to the lack of improvement in *Sprint 4* and we added a large amount of tests this time around. Efforts were also made to ensure everyone is up to speed on the documentation and the code so that we can all answer basic questions in all domains. This sprint was short in length but offered a good opportunity to fix any technical debt accumulated throughout the previous sprints, thus helping us refine our project to something we can be proud of.

Given the large amount of refactors in the frontend codebase, many changes had to be made to tests. As such, only 1 test was added, however since no functionality changed and test results are at a 100% pass rate, this is not a concern. Frontend test coverage is at 51%.

8.5 Acceptance Testing

AT-1	**AT-1 As a User, I want to be able to login, so that I can access my personal information.**	**Input**
Acceptance Criteria	Given that I am on the login page, and that I input my user name, and that I input my password, and that I clicked LOGIN, then I should reach my home page.	Username Password
		Expected Output
		Reach home page.
Result	**PASS**	
Comments		

AT-2	**AT-2 As a User, I want to be able to create an account and specify my role in the application so that I can make use of the application.**	**Input**
Acceptance Criteria	Given that I am on the sign-up page, And I select my role (Doctor, Patient,..) and that I input my personal information (full name, DOB, medicare #, home address, Doctor ID (if role = doctor))	Role Personal Information Password

	and that I input & confirm a new password, and that I clicked sign-up, then I should be redirected to the login page and login using my information, provided my account has been approved (Doctors need approval from an Administrator).	**Expected Output**
		Reach login page given approved account.
Result	**PASS**	
Comments		

AT-3	**AT-3 As an Administrator, I want to be able to see, accept, and reject account creation requests for non-"Patient" roles so that each User may use the application according to their role.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and there are pending account creation requests, then I should see them and be able to verify & accept/reject them based on the information provided.	
		Expected Output
		Ability to view, accept, or reject account creation requests.
Result	**PASS**	
Comments		

AT-4	**AT-4 As an Administrator, I want to be able to assign a Doctor to a Patient, so that communication and information sharing may be established between the two.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and there are Doctors & Patients signed up to the service, then I should see a list of Patients without a Doctor, and I should see see a list of available Doctors,	
		Expected Output
		View all unassigned Patients and available

	then I should be able to assign a Doctor to a Patient, then the Patient should be notified that a Doctor has been assigned to them.	Doctors. Ability to assign Doctors to Patients. Patient is notified.
	PASS	
Comments		

AT-5	**AT-5 As a Patient, I want to be able to update my status within a predefined time frame, so that I can inform my Doctor of my condition.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have a Doctor, then I should be able to update my status based on the list of requested details, and my Doctor should be able to see my new status.	Status Update
		Expected Output
		Accepted status update. Ability for Doctor to view status.
Result	**PASS**	
Comments		

AT-6	**AT-6 As a Doctor/Health Official, I want to be able to monitor the status and symptoms of confirmed and unconfirmed COVID-19 patients, so that I can take appropriate measures to ensure their safety and the safety of others.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have patients under my supervision, and that my Patients have updated their status, then I should be able to view their current status and symptoms in order to take appropriate action (if any).	
		Expected Output
		Ability to view Patients' statuses.

Result	**PASS**	
Comments		

AT-7	**AT-7 As a Doctor, I want to be able to see an easily digestible visual summary of the status of my patients, so that I can quickly grasp the situation the patient is in.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have patients under my supervision, and that my Patients have updated their status, then I should be able to see easily digestible graphs with summaries for details for all my patients (# infected, # recovered, etc.).	
		Expected Output
		Ability to view graphs of Patients.
Result	**PASS**	
Comments		

AT-8	**AT-8 As a Patient, I want to be able to contact my Doctor within the application, so that I can communicate with them about my condition/other relevant topics.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have a Doctor, then I should be able to contact my Doctor via a button in the app, redirecting me to my email client in order to send the specific question.	Select the button to Contact Doctor.
		Expected Output
		Redirected to email client.
Result	**PASS**	

Comments		

AT-9	**AT-9 As a User, I want to be able to see notifications pertaining to me so that I may act upon them.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have an unread notification, then I should be able to see relevant notifications (Admin messages, Patient update (as a Doctor), Doctor's requests (as a Patient)).	
		Expected Output
		Ability to view relevant notifications.
Result	**PASS**	
Comments		

AT-10	**AT-10 As a Doctor, I want to be able to prioritize any of my Patients so that I can see their updates more efficiently.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and that I have one or more Patients assigned to me, then I should be able to flag them by clicking a flag icon next to their name on my Patient table, putting their notifications in a special priority list.	Select flag icon next to Patients
		Expected Output
		All notifications from flagged Patients are visible in a special priority list.
Result	**PASS**	
Comments		

AT-11	AT-11 As a User, I want to be informed of any contact I might have had with someone who is infected. Implemented in a log of frequently visited locations. This database can be used to inform others if someone else who visits those locations has tested positive recently.	Input
Acceptance Criteria	Given that I am logged in to the system, and that I have inputted my frequented locations, and another User who frequents the same locations sets their status as infected, then I (and my Doctor if applicable) should be informed of this fact in order to take the proper precautions.	Frequented locations
		Expected Output
		Receive notification if another User frequents my locations.
Result	PASS	
Comments		

AT-12	AT-12 As an Administrator, I want to resolve all conflicts between users and ensure that all Users are respecting the Terms and Services.	Input
Acceptance Criteria	Given that I am logged in to the system, and that there are reports from Users, then I should be able to see the reports as well as the reasons for why the User reported another User in order to resolve the situation and contact them if need be.	
		Expected Output
		Ability to view User reports and contact the Users.
Result	PASS	
Comments		

AT-13	AT-13 As a Doctor, I would like to be able to schedule a time with my patient to speak to them.	Input
Acceptance Criteria	Given that I am logged in to the system,	Availabilities

	and that I have a Patient, then I should be able to ask the Patient to book an appointment based on my availability, where he/she can then select from a calendar GUI and confirm the appointment, registering it for both of us.	**Expected Output**
		Patient can select an availability. When availability is chosen, the appointment is registered.
Result	**PASS**	
Comments		

AT-14	**AT-14 As a User, I want to check if the area that I am currently in contains many people who tested positive or not to ensure my safety.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, then I should be able to see a map listing the number of infected people living within a general radius of my area.	
		Expected Output
		Ability to view map with the number of nearby infected.
Result	**PASS**	
Comments		

AT-15	**AT-15 As a Patient, I would like to confirm with my Doctor how likely it is that I have COVID-19 after doing the rapid test.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and I have taken a rapid test, and I have entered the results of my rapid test into the system,	Rapid test results
		Expected Output

	then my Doctor can see the results of the test and inform me on the appropriate steps to take.	Ability for my Doctor to view the results and email me.
Result	**PASS**	
Comments		

AT-16	**AT-16 As a Patient who tested positive for COVID-19, I would like the ability to notify my Doctor of potential emergencies and I would like some help getting food since I cannot go shopping myself.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and I have tested positive for COVID-19, and I have updated my status in the system, then I should be able to contact my Doctor in the event of a medical emergency, and I should be able to access relevant resources providing me with tips on what to do as well as potential suggestions of delivery services I can use to order food.	Status Contact Doctor
		Expected Output
		Ability to easily access relevant resources.
Result	**PASS**	
Comments		

AT-17	**AT-17 As an Administrator, I want to be able to see if a Doctor has declared an emergency, so that I can reassign their Patients temporarily to another Doctor.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, and a Doctor has declared an emergency/requested a Patient be reassigned, then I can reassign the Patient to a new Doctor, storing the previous Doctor's information for easy reversal.	Temporarily reassign Patients
		Expected Output
		Patients now temporarily report to new Doctors.

		Patients return to original Doctors upon their return.
Result	**PASS**	
Comments		

AT-18	**AT-18 As a User, I want to have an interface to other COVID-19 applications, so that I can have more useful data and resources at my disposal.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, then I should be able to see a page with relevant information (infections, recoveries, etc.) obtained via an API to an existing COVID application, with citations as to where the data comes from.	
		Expected Output
		Ability to access other COVID applications.
Result	**PASS**	
Comments		

AT-19	**AT-19 As a Patient, I want to be able to generate a QR code of my record containing relevant information, so that relevant officials can easily see my information at a glance without revealing all personal details.**	**Input**
Acceptance Criteria	Given that I am logged in to the system, Then I should be able to generate a QR code containing relevant information (infection status, symptoms, etc.) to only Doctors/Immigration Officers.	
		Expected Output
		Receive QR code.
Result	**PASS**	
Comments		

AT-20	AT-20 As an Immigration Officer, I want to be able to monitor Immigrant Patients under my jurisdiction, so that I can take appropriate measures based on their status.	Input
Acceptance Criteria	Given that I am logged in to the system, and that I have immigrant Patients under my supervision, and that my Patients have updated their status, then I should be able to view their current status and symptoms in order to take appropriate action (if any).	
		Expected Output
		Ability to view Patients' statuses.
Result	PASS	
Comments		

AT-21	AT-21 As an Immigration Officer, I want to be able to flag certain Immigrant Patients, so that I can monitor them more closely.	Input
Acceptance Criteria	Given that I am logged in to the system, and that I have one or more immigrant Patients assigned to me, then I should be able to flag them by clicking a flag icon next to their name on my Patient table, putting their notifications in a special priority list.	Select flag icon next to Patients
		Expected Output
		All notifications from flagged Patients are visible in a special priority list.
Result	PASS	
Comments		

8.6 System Tests

Multiple components of the system are tested at the same time to monitor and analyze the behaviour of its output. The main focus is on the interaction between the components that rely on each other including external services such as APIs of different web sources. Integration and system testing is best applied for code regressions as the system becomes more complex.

Testify is one of the tools that is used for integration and system testing and can be written in Python. The tool was made to substitute unittest. It's a package that is readily available in Github and is compatible with Django.

Advantages:

- Tests written in unittest will run with the testify tool with minor adjustments
- Provides better error catching and messages on unexpected calls/parameters, argument types, and better stack tracing
- Package comes with useful testing utilities such as its own code coverage profiling and integration
- Allows additional plugins to include more functionality and reporting for testing
- Github-based

Disadvantages:

- The use of mocks is dependent on directory calls (ie. -dir) if testing is done manually
- Lack of proper and useful documentation. The user will require to do some research to find resources
- Not good for parallel testing

In addition, PyTest is a tool that can be used as mentioned in *Section 8.2.1*. It can also accommodate complex applications and functionalities. Thus, it is suitable for integration and system testing as well. Compared to Testify, PyTest can run in parallel which helps decrease the test running time.

For this particular test, it can be done manually, or it can be automated. If chosen manually, testing the components of the system is preferably done for its first initial run to understand the behavior of the system to determine its intended behavior. It will give insights about the interaction between the components and to see if adjustments are needed. After which, the integration tests can be done automatically by an automated tool if the testing is chosen to be automated. PyTest offers automated integrated testing, which is useful to cut down testing time.

Jenkins is an open source automated server that allows continuous building, integration, testing, and deployment. If automating integration and system tests is chosen, using PyTest and Jenkins will be helpful as the system becomes complex by the addition of new features. Automated testing of features will reduce testing time and catch errors that may arise during development.

Step-by-step test(s) based on user stories:

Note: ST stands for System Test and -X stands for the ID number of the user story.

Below is a test case for the login feature.

ST-1	**ST-1 -** As a User, I want to be able to login, so that I can access my personal information
Acceptance Criteria	Given that I am on the Login page, the UI Login page is appropriately displayed. As I input the correct username and password, and that I clicked LOGIN, then the inputs are retrieved from the UI, sent to be processed by the system, and fetched in the database. After the data is fetched, then the system processes that data and returns the UI displaying my personal information and application features.
Result	**ACCEPTED**
Comments	

ST-2	**ST-2** As a User, I want to be able to create an account and specify my role in the application so that I can make use of the application.
Acceptance Criteria	Given that I am on the sign-up page, the ability to select a role (Doctor, Patient,...) is present. As I enter my email, its format is checked to make sure that it is a valid email form. As I enter and confirm my password, the system checks that they match. As I click sign-up after entering valid information, then I should be redirected to the login page and login using my information provided my account has been

	approved (Doctors need approval from an Administrator). The account should now be present in the database with all the correct fields. The password should be encrypted thus an Administrator will not be able to see your password.
Result	**ACCEPTED**
Comments	

ST-3	**ST-3** As an Administrator, I want to be able to see, accept, and reject account creation requests for non-“Patient” roles so that each User may use the application according to their role.
Acceptance Criteria	Given I am logged in as an AdministratorI should see a Table with Doctors who are not yet approved and be able to approve them by pressing the check box present on the UI based on the information provided during sign-up. Their status should be successfully changed in the database.
Result	**ACCEPTED**
Comments	

ST-4	**ST-4** As an Administrator, I want to be able to assign a Doctor to a Patient, so that communication and information sharing may be established between the two.
Acceptance Criteria	Given that I am logged into the system as an Administrator, I should be able to assign a Doctor to a Patient via a drop down menu on the Patient’s table entry on my UI. The updated link to a Doctor should be present in the database.

Result	**ACCEPTED**
Comments	

ST-5	**ST-5** As a Patient, I want to be able to update my status within a predefined time frame, so that I can inform my Doctor of my condition.
Acceptance Criteria	Given that I am logged in as a Patient and am currently assigned a Doctor, then I should be able to update my status based on the list of requested details, and my Doctor should be able to see my new status. The database should reflect the fact that the status has changed and properly push that to the Doctor's view for the appropriate Patient.
Result	**ACCEPTED**
Comments	

ST-6	**ST-6** As a Doctor/Health Official, I want to be able to monitor the status and symptoms of confirmed and unconfirmed COVID-19 patients, so that I can take appropriate measures to ensure their safety and the safety of others.
Acceptance Criteria	Given that I am logged in to the system, and that I have patients under my supervision, and that my Patients have updated their status, then I should be able to view their current status and symptoms in order to take appropriate action (if any).

Result	**ACCEPTED**
Comments	

ST-7	**ST-7** As a Doctor, I want to be able to see an easily digestible visual summary of the status of my patients, so that I can quickly grasp the situation the patient is in.
Acceptance Criteria	Given that I am logged in to the system as a Doctor and have > 0 Patients, and that my Patients have updated their status, then I should be able to see easily digestible graphs with summaries for details for all my patients (# infected, # recovered, etc.). The data should be accurate based on the data in the database.
Result	**ACCEPTED**
Comments	

ST-8	**ST-8** As a Patient, I want to be able to contact my Doctor within the application, so that I can communicate with them about my condition/other relevant topics.
Acceptance Criteria	Given that I am logged in as a Patient, and that I have a Doctor, then I should be able to contact my Doctor via a button in the app, redirecting me to my email client in order to send the specific question. The client should take care of redirecting to the devices' email client, auto filling in the relevant fields.

Result	**ACCEPTED**
Comments	

ST-9	**ST-9** As a User, I want to be able to see notifications pertaining to me so that I may act upon them.
Acceptance Criteria	Given that I am logged in as any User, and that I have an unread notification stored for my account on the backend, then I should be able to see these notifications on the frontend, and clicking/viewing them should set their status to “read” on the backend, removing the notification bubble.
Result	**ACCEPTED**
Comments	

ST-10	**ST-10** As a Doctor, I want to be able to prioritize any of my Patients so that I can see their updates more efficiently.
Acceptance Criteria	Given that I am logged in to the system as a Doctor with >= 1 Patients assigned to me, then I should be able to flag them by clicking a flag icon next to their name on my Patient table, updating their flagged status on the backend and propagating between logins on the frontend, and their notifications should be present in a prioritized list.
Result	**ACCEPTED**
Comments	

ST-11	**ST-11** As a User, I want to be informed of any contact I might have had with someone who is infected. Implemented in a log of frequently visited locations. This database can be used to inform others if someone else who visits those locations has tested positive recently.
Acceptance Criteria	Given that I am logged in to the system, and that I have inputted my frequented locations, which will be stored on the backend for my user account, and another User who frequents the same locations sets their status as infected, then I (and my Doctor if applicable) should be informed of this fact in order to take the proper precautions. The backend will check against the frequented locations of all other Users and if one of them is infected and there is a match, the notification will be pushed to the frontend.
Result	**ACCEPTED**
Comments	

ST-12	**ST-12** As an Administrator, I want to resolve all conflicts between users and ensure that all Users are respecting the Terms and Services.
Acceptance Criteria	Given that I am logged in to the system as an Administrator, and that there are reports from Users which have been sent through the report form, then I should be able to see the reports via my Administrator email address with the reasons for why the User reported another User in order to resolve the situation and contact them via email if need be.

Result	**ACCEPTED**
Comments	

ST-13	**ST-13** As a Doctor, I would like to be able to schedule a time with my patient to speak to them.
Acceptance Criteria	Given that I am logged in to the system as a Doctor, and that I have a Patient registered as under my care on the backend, then I should be able to ask the Patient to book an appointment based on my availability which is stored on the backend. The Patient should then select from a calendar GUI and confirm the appointment, registering it for both of us, blocking off other Patients from selecting that timeslot.
Result	**ACCEPTED**
Comments	

ST-14	**ST-14** As a User, I want to check if the area that I am currently in contains many people who tested positive or not to ensure my safety.
Acceptance Criteria	Given that I am logged in to the system, then I should be able to see a map listing the number of infected people living within a general radius of my area. The data is fetched based on the home addresses assigned to each User as well as their status as infected/not infected.
Result	**ACCEPTED**

Comments	

ST-15	**ST-15** As a Patient, I would like to confirm with my Doctor how likely it is that I have COVID-19 after doing the rapid test.
Acceptance Criteria	Given that I am logged in to the system, and I have taken a rapid test, and I have entered the results of my rapid test into the system, Then the results should be stored on the backend and my data updated on the database to include the rapid test result. When the Doctor checks my result, the backend should fetch the results from the database and display it for the Doctor to see.
Result	**ACCEPTED**
Comments	

ST-16	**ST-16** As a Patient who tested positive for COVID-19, I would like the ability to notify my Doctor of potential emergencies and I would like some help getting food since I cannot go shopping myself.
Acceptance Criteria	Given that I am logged in to the system, and I have tested positive for COVID-19, and I have updated my status in the system using the existing methodology as mentioned in *US*/*ST-5*, then I should be able to contact my Doctor in the event of a medical emergency using the existing contact system which uses an email client, and I should be able to access relevant resources providing me with tips on what to do as well as potential suggestions of delivery services I can use to order food. These resources can be modified by the developers as need be and as the situation changes.

Result	**ACCEPTED**
Comments	

ST-17	**ST-17** As an Administrator, I want to be able to see if a Doctor has declared an emergency, so that I can reassign their Patients temporarily to another Doctor.
Acceptance Criteria	Given that I am logged in to the system, and a Doctor has declared an emergency/requested a Patient be reassigned, then this request should be stored on the backend, which will then propagate to the frontend as I log in or check my email. I can then reassign the Patient to a new Doctor, storing the previous Doctor's information in the database for easy reversal when the emergency/declared reassignment period ends.
Result	**ACCEPTED**
Comments	

ST-18	**ST-18** As a User, I want to have an interface to other COVID-19 applications, so that I can have more useful data and resources at my disposal.
Acceptance Criteria	Given that I am logged in to the system, then I should be able to see a page with relevant information (infections, recoveries, etc.) obtained via an API to an existing COVID application, with citations as to where the data comes from. The API should be referenced using the documented API calls and be updated appropriately as the external system updates the information pushed out to the API calls.
Result	**ACCEPTED**
Comments	

ST-19	**ST-19** As a Patient, I want to be able to generate a QR code of my record containing relevant information, so that relevant officials can easily see my information at a glance without revealing all personal details.
Acceptance Criteria	Given that I am logged in to the system, Then I should be able to generate a QR code containing relevant information (infection status, symptoms, etc.) to only Doctors/Immigration Officers. The QR code should route to a specific link for each Patient that includes a hashed key for their Patient ID. A Patient/outsider should not be able to access information from a QR code even if they have the link. It should also not be possible to increment the hashed key to find another Patient's QR code.
Result	**ACCEPTED**
Comments	

ST-20	**ST-20** As an Immigration Officer, I want to be able to monitor Immigrant Patients under my jurisdiction, so that I can take appropriate measures based on their status.
Acceptance Criteria	Given that I am logged in to the system, and that I have immigrant Patients under my supervision, and that my Patients have updated their status, then I should be able to view their current status and symptoms in order to take appropriate action (if any). This functions identically to *ST-6* but should only allow the status of immigrant Patients to be viewed.
Result	**ACCEPTED**
Comments	

ST-21	**ST-21** As an Immigration Officer, I want to be able to flag certain Immigrant Patients, so that I can monitor them more closely.
Acceptance Criteria	Given that I am logged in to the system, and that I have one or more immigrant Patients assigned to me, then I should be able to flag them by clicking a flag icon next to their name on my Patient table, putting their notifications in a special priority list. This functions identically to *ST-10* but should only allow visibility of immigrant Patients.
Result	**ACCEPTED**
Comments	

9. Defect Tracking and Report

Defect ID	Description	Discovered	Resolved	Status
D-1	Front-end tests are not fully set up. Will be finalized in sprint 2.	Sprint 1	Sprint 2	**CLOSED**
D-2	Did not have time to deploy the site to be publicly hosted	Sprint 3	Sprint 4	**CLOSED**
D-3	Did not have time to properly update test statistics in report	Sprint 4	Sprint 5	**CLOSED**

10. Quality Measurements

10.1 Metrics

	Sprint 1
Summary	** Total ** LOC: 111 LLOC: 54 SLOC: 60 Comments: 3 Single comments: 15 Multi: 3 Blank: 33 - Comment Stats (C % L): 3% (C % S): 5% (C + M % L): 5%

	Sprint 2
Summary	** Total ** LOC: 780 LLOC: 324 SLOC: 492 Comments: 74 Single comments: 84 Multi: 25 Blank: 179 - Comment Stats (C % L): 9% (C % S): 15% (C + M % L): 13%

	Sprint 3
Summary	** Total ** LOC: 1451 LLOC: 562 SLOC: 970 Comments: 159 Single comments: 167 Multi: 25 Blank: 289 - Comment Stats (C % L): 11% (C % S): 16% (C + M % L): 13%

	Sprint 4
Summary	** Total ** LOC: 1621 LLOC: 636 SLOC: 1029 Comments: 175 Single comments: 190 Multi: 25 Blank: 305 - Comment Stats (C % L): 11% (C % S): 16% (C + M % L): 13%

	Sprint 5
Summary	** Total ** LOC: 2021 LLOC: 935 SLOC: 1457 Comments: 125 Single comments: 141 Multi: 25 Blank: 398

	- Comment Stats (C % L): 6% (C % S): 9% (C + M % L): 7%

10.2 Comparison

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Delta (from sprint 4)
LOC: 111 LLOC: 54 SLOC: 60 Comments: 3 Single comments: 15 Multi: 3 Blank: 33	LOC: 780 LLOC: 324 SLOC: 492 Comments: 74 Single comments: 84 Multi: 25 Blank: 179	LOC: 1451 LLOC: 562 SLOC: 970 Comments: 159 Single comments: 167 Multi: 25 Blank: 289	LOC: 1621 LLOC: 636 SLOC: 1029 Comments: 175 Single comments: 190 Multi: 25 Blank: 305	LOC: 2021 LLOC: 935 SLOC: 1457 Comments: 125 Single comments: 141 Multi: 25 Blank: 398	LOC: 400 LLOC: 300 SLOC: 400 Comments: -50 Single comments: -50 Multi: 0 Blank: 100

Sprint 5 has seen a large increase in LLOC, since much of the refactoring rid the codebase of unnecessary commented lines. Additionally, since a lot of work went into code formatting, there was an increase in blank lines.

10.3 Metric Description

Defined in *section 10.1* are several chosen metrics to assess the code quality of the project's backend. The metrics chosen are Cyclomatic Complexity, Maintainability Index, Logical Lines of Code, and Comment Stats. Provided is a brief description of each one:

Cyclomatic Complexity: Gives insight into how *difficult* it may be to test, maintain and update existing modules. A high CC indicates a poor code quality, and means that the number of decision trees in any given module is high. Radon (the quality tool of choice for this project), produces an integer (1 being the least complex).

Maintainability Index: This is another metric which attempts to calculate code *complexity*. It factors in Lines Of Code, Cyclomatic Complexity and Halstead volume to produce a number to indicate how complex a module is. In the case of radon, a number from 0-100 (and corresponding letter grade) is generated. The formula used by Radon is as follows:

$$MI = \max\left[0, 100\,\frac{171 - 5.2\ln V - 0.23G - 16.2\ln L + 50\sin(\sqrt{2.4C})}{171}\right]$$

Where:

- V is the Halstead Volume;
- G is the total Cyclomatic Complexity;
- L is the number of Source Lines of Code (SLOC);
- C is the percent of comment lines (important: converted to radians).

LLOC: The number of Logical Lines of Code is a good indication for how *bloated* a given module is. Sometimes if LLOC is high, it may be beneficial to create a separation of concerns by subdividing a module into sub-modules. As the project is still in early development, the LLOC is low for each module.

Comment Statistics: This provides insight into how well *documented* the project is. This is not factored in in cyclomatic complexity, nor Maintainability Index, nor LLOC, but it can provide important data on maintainability. Also indicated is C% L, C % S and C + M % L. These are described as follows:

- C % L: the ratio between number of comment lines and LOC, expressed as a percentage;
- C % S: the ratio between number of comment lines and SLOC, expressed as a percentage
- C + M % L: the ratio between number of comment and multiline strings lines and LOC, expressed as a percentage.

Further information is provided in the radon documentation.

11. Release Plan Table & Chart

11.1 Release Plan Table

PROJECT NAME	PROJECT MANAGER	START DATE	END DATE	OVERALL PROGRESS		PROJECT DELIVERABLE	Covid Tracker Website
CovidTracker	Brandon Takli	1/11/2022	4/18/2022	100%		SCOPE STATEMENT	The CovidTracker Web Application will provide a means of contact and communication between COVID-19 patients and doctors, while allowing Health Officials and Immigration Officers to monitor the situation.

AT RISK	TASK NAME	RESPONSIBLE	STORY POINTS	START	FINISH	DURATION (DAYS)	STATUS	COMMENTS
	Sprint 1	Dimitri Kirbizakis	3	1/11/22	2/2/22	22	Complete	
FALSE	Login	Peter Delis	3	1/21/22	1/25/22	4	Complete	
	Sprint 2	Liam Burchell-Reyes	28	2/3/22	2/23/22	20	Complete	
FALSE	Sign Up	Peter Delis	5	1/30/22	2/5/22	6	Complete	Started early near the end of Sprint 1
FALSE	Account Creation Request	Dimitri Kirbizakis	5	2/7/22	2/16/22	9	Complete	
FALSE	Assign Doctors to Patients	Daniel Macicasan	2	2/11/22	2/16/22	5	Complete	
FALSE	Patient Status Update	Andrei Grosuliac	3	2/10/22	2/14/22	4	Complete	
FALSE	Monitor Status & Symptoms of Patients	Ineke Nguyen	5	2/12/22	2/20/22	8	Complete	

FALSE	High Level Patient Dashboard	David Seleznev	3	2/13/22	2/17/22	4	Complete	
FALSE	Contact Doctor/Patient	Peter Delis	5	2/13/22	2/19/22	6	Complete	Note: Internal changes were made to implementation, accounting for discrepancies in JIRA tasks. Meets requirements regardless.
	Sprint 3	Brandon Takli	21	2/24/22	3/16/22	20	Complete	
FALSE	Receive Notifications	Dimitri Kirbizakis	5	2/26/22	3/5/22	7	Complete	
FALSE	Prioritize Patient	Ineke Nguyen	3	2/27/22	3/5/22	6	Complete	
TRUE	Contact Tracing	Andrei Grosuliac	5	2/26/22	3/6/22	8	Complete	Note: this feature is volatile and is prone to change should the customer reconsider the practicality of the planned implementation.
FALSE	Report System	Matteo Gisondi	2	2/28/22	3/5/22	5	Complete	
FALSE	Arrange Appointment	Ineke Nguyen	3	3/2/22	3/9/22	7	Complete	
FALSE	Infected in Area	David Seleznev	3	3/7/22	3/14/22	7	Complete	
	Sprint 4	Liam Burchell-Reyes	24	3/17/22	4/6/22	20	Complete	Dates are tentative
FALSE	Rapid Test Confirmation	Daniel Macicasan	2	3/19/22	3/24/22	5	Complete	
FALSE	Request Help	Ineke Nguyen	3	3/21/22	3/28/22	7	Complete	
FALSE	Doctor Emergency Reassign Patients	Andrei Grosuliac	3	3/22/22	3/27/22	5	Complete	
FALSE	API Interface to Other COVID-19 Applications	Ineke Nguyen	5	3/24/22	4/2/22	9	Complete	

FALSE	QR Code for Individual Patient Records	Matteo Gisondi	5	3/27/22	4/4/22	8	Complete	
FALSE	Immigration Officer Monitoring Immigrant Patients	Dimitri Kirbizakis	3	3/28/22	4/5/22	8	Complete	
FALSE	Immigration Officer Flagging Immigrant Patients	Dimitri Kirbizakis	3	3/29/22	4/5/22	7	Complete	
	Sprint 5	**Brandon Takli**	**0**	**4/7/22**	**4/18/22**	**11**	**Complete**	

11.2 Release Plan Chart