CovidTracker Sprint 1

By

Andre Ibrahim - 40132881

Dan Raiu - 40108722

Daren Kafafian - 40100511

Domenic Seccareccia - 40063021

Ejazali Rezayi - 40101892

Jason Gerard - 40079266

Khagik Chris Astor - 40099665

Lucas Blanchard - 40060670

Rafi Stepanians - 40108731

A report submitted in partial fulfillment of the requirements of SOEN 390

Concordia University

February 1, 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

The purpose of this document is to give an overview of the problem, proposed solution, statement of scope, project description, product requirements, summary of work packages completed in sprint 1, a sprint 2 release plan, software architecture, risk assessment and management plan, user interface design, testing plan, defect tracking, and quality measurements. This document is targeted at all stakeholders of the system: product owners and development team..

1.1 Positioning

1.1.1 Problem Statement

The problem of	Lack of an easy to use tool to track, manage and coordinate the onset of positive COVID-19 patients on both a micro and macro level.
Affects	Patients, Doctors, Health Officials, Immigration Officers
The impact of which is	The inability for the government to properly handle and manage COVID-19 variants, hospital capacity and perform a safe reopening plan backed by data and science.
A successful solution would be	- A way to assign quarantine restrictions to positive COVID-19 patients - A way to monitor the status and symptoms of confirmed and unconfirmed patients with COVID-19 - Conduct contact tracing notify the people with whom COVID-19 patients have been in contact - Allow patients to update their COVID-19 status and symptoms - An easy way to arrange appointments between doctors and patients

Table 1: Problem Statement

1.1.2 Product Position Statement

For	Patients, Doctors, Health Officials, Immigration Officers and Administrators
Who	Manage, monitor and respond to COVID-19 related events and situations
CovidTracker	Is a responsive web application
Unlike	Covid Alert, ArriveCan
Our product	Is designed to ease the management and monitoring of COVID-19 across the province by contact tracing and notifying patients positive with COVID-19, allowing doctors to follow patients symptoms and arrange appointments with positive patients, assign quarantine restrictions and allowing patients to daily update their status and symptoms.

Table 2 : Product Position Statement

2.0 PROJECT DESCRIPTION

The project has a couple of main lenses where all features are derived from such as patient management, status report management, contact tracing, a messaging system, a notification system, a QR code system, and a detailed authentication and authorization layer in front of the system to make sure sensitive info is not shown to the wrong user.

Agile is an interactive software development methodology allowing software teams to produce working software quickly, test it, get feedback on it, and then iterate in quick cycles. Agile is being used given its methodology and to ensure that the product owner's needs -progress and requirements - are being satisfied by the development team throughout the project development lifecycle.

Development is broken down into 5 total sprints and the schedule will be as follows:

Sprint	Date (mm/dd/yyyy)
1	1/12/2022 - 2/2/2022
2	2/3/2022 - 2/23/2022
3	2/24/2022 - 3/16/2022
4	3/17/2022 - 4/6/2022
5	4/7/2022 - 4/18/2022

Table 3: Sprint Schedule

2.1 Stakeholders

Project stakeholders consist of users, the development team and project owners. The various roles assigned to each user, development team member and project owners are as follows and subsequently described in the following section.

- Users
    - Patients
    - Doctors
    - Health Officials
    - Immigration Officers
    - Administrators
- Product Owner
    - Yann-Gaël Guéhéneuc
    - Minani Jean Baptiste
- Project Champion
    - Jason Gerard
- Organizational Management Team
    - Andre Ibrahim
    - Ejazali Rezayi
    - Dan Raiu
    - Daren Kafafian
    - Domenic Seccareccia
    - Jason Gerard
    - Khagik Chris Astor
    - Lucas Blanchard
    - Rafi Stepanians
- Analysts
    - Andre Ibrahim
    - Ejazali Rezayi
    - Dan Raiu
    - Daren Kafafian
    - Domenic Seccareccia
    - Jason Gerard
    - Khagik Chris Astor
    - Lucas Blanchard
    - Rafi Stepanians
- Designers
    - Domenic Seccareccia
- Developers (Front end & Back end)

  - Domenic Seccareccia
  - Dan Raiu
  - Daren Kafafian
  - Ejazali Rezayi
  - Khagik Chris Astor
  - Rafi Stepanians
  - Lucas Blanchard
  - Andre Ibrahim
  - Jason Gerard
- Testers
  - Andre Ibrahim
  - Ejazali Rezayi
  - Dan Raiu
  - Daren Kafafian
  - Domenic Seccareccia
  - Jason Gerard
  - Khagik Chris Astor
  - Lucas Blanchard
  - Rafi Stepanians

2.1.1 Stakeholder Roles

2.1.1.1 Users

Users refer to anyone that uses the software for the functionality that it provides. They have an interest in this project since they use it to accomplish some of their tasks. Users consist of Patients, Doctors, Health Officials, Immigration Officers and Administrators.

2.1.1.2 Product Owner

The product owner is accountable for maximizing the value of the software being developed. His interest is in the delivery of the project in a timely manner with all requirements completed.

2.1.1.3 Project Champion

The project champion is the main driving force of the project fielding all external inquiries and responses. As such, they have one of the largest and most direct stakes in the project.

2.1.1.4 Organization Management Team

The organizational management team organizes and plans the activities that achieve the company's established goals. They will do this by allocating time to build the project schedule from start to finish, allocate resources, and plan meetings to reach pre-established deadlines. They also have an interest in the project both financially and personally as they are also university students.

2.1.1.5 Development Team

The development team has the same interests as the organizational management team. However, their impact, contribution and stake in the project are different. The development team is primarily focused on executing the activities that result in the system being realized and in turn satisfying established goals.

2.1.1.6 Analysts

Analysts are responsible for accessing and researching market opportunities and gathering requirements. They translate requirements to specifications allowing designers to design a system around those needs, developers to satisfy those needs and testers to ensure all developed features work according to specifications. Their stake in the project revolves around how the data impacts the user.

2.1.1.7 Designers

The designers are responsible for the system design aspects such as architectural design, user interface (UI) design and user experience (UX) design. Their stake in the project revolves around system accessibility, maintainability, upgradability and usability.

2.1.1.8 Developers

The developers must develop a system that satisfies the specifications outlined by the analysts and follows the designs - architectural and interface - specified by the designers. Their stake revolves around the implementation of the system features.

2.1.1.9 Testers

The testers are responsible for quality assurance (QA) during the development and deployment phases. They ensure each developed feature within a sprint passes all associated tests and satisfies the specifications outlined by the analysts. Their stake in the project revolves around user and system QA and quality control (QC).

3.0 REQUIREMENTS

The following requirements elicited from the product owners were turned into user stories and subsequently approved by the product owner. Each user story is associated with a corresponding EPIC, has a list of subtasks and a definition of done defining what must be completed in order for said user story to be considered as complete. Furthermore, the description section of each user story is broken down with the following information:

- Definition of done
- Requirements
    - UI Prototype
    - Front end
    - API (Optional)
    - Specifications (Google document attached containing all specifications associated to the user story)
    - Personas (accessible by)
- Acceptance tests
- System tests

Once the user interface design mockups are complete, a prototype is created allowing assigned developers to interact with the designs to get a better sense of how the feature is expected to be interacted by users. Associated links and attachments are then added in the comments section of the associated user story.

3.1 User Stories

For Sprint 1, **3 user stories and 12 tasks** have been elicited for a total of **56 user story points**.

See next page to view user stories for sprint 1.

T	Key	Summary	Story point estimate	Assignee(s)	P	Risk	Parent	Start date	Due	Status	Description
☑	COV-1	Setup GitHub repository	1	Jason Gerard	︽	Medium	⚡ Project Setup	13/Jan/22	15/Jan/22	DONE	**Definition of done:** All the devs should have access to a GitHub repository that they can contribute code to. **Requirements:** • Create repo • Add branch protection policies • Write basic documentation ◦ README ◦ Contribution guidelines ◦ Add maintainers ◦ Add CODEOWNERS
☑	COV-3	Setup basic web server	5	Jason Gerard	︽	Medium	⚡ Project Setup	13/Jan/22	15/Jan/22	DONE	**Definition of done:** There should be a bare bones typescript web server that is runnable. **Requirements:** • Typescript project • Restify web server framework • Inversify IoC framework
☑	COV-4	Setup basic project for front end client	5	Andre Ibrahim, Dan, Domenic Seccareccia, Khachig Astor, rafistep98	︽	Medium	⚡ Project Setup	22/Jan/22	23/Jan/22	DONE	**Definition of done:** When there is a bare bones react project that compiles, is runnable, and is merged into the main branch on the repo. **Requirements:** • Basic react app • Basic dependencies installed • Compiles and runs • Merged on github
☑	COV-5	Setup Jira project	1	Domenic Seccareccia, Jason Gerard	︿	Low	⚡ Project Setup	15/Jan/22	18/Jan/22	DONE	**Definition of done:** There should be a working Jira project and SCRUM board that the project manager can start contributing to. **Requirements:** • SCRUM board for the project • All the features setup ◦ custom issue fields ◦ report generation • All members should have appropriate access
☑	COV-6	Configure CI/CD pipeline for web server	2	Jason Gerard	=	Low	⚡ Project Infrastructure	15/Jan/22	17/Jan/22	DONE	**Definition of done:** There should be a CI/CD pipeline running through GitHub actions on every pull request and commit to the main branch. **Requirements:** • Through GitHub actions • Setup pipeline only for the web server • Should have steps ◦ Compile ◦ Lint and format ◦ Test
☑	COV-7	Setup docker for web server	5	Domenic Seccareccia, Jason Gerard	=	Low	⚡ Project Infrastructure	17/Jan/22	18/Jan/22	DONE	**Definition of done:** The web server and related database should be able to be run on docker with appropriate documentation. **Requirements:** • Dockerfile for web server • docker-compose.yml file for project local development
☑	COV-30	Add testing framework to web server	2	Jason Gerard	=	Low	⚡ Project Infrastructure	14/Jan/22	15/Jan/22	DONE	**Definition of done:** The web server should have a framework for running automated unit and integration tests using the appropriate commands and frameworks. **Requirements:** • Testing framework(s) should be added to the `package.json` file • Framework(s) should cover ◦ Unit tests ◦ Integration tests ◦ System/Acceptance tests • Should include a code coverage / quality metrics analyser • Should add scripts to run tests and coverage to `package.json`
☑	COV-33	Update risk management plan and table	2	Dan, Khachig Astor	﹀	Low	⚡ Documentation	26/Jan/22	29/Jan/22	DONE	**Definition of done:** There should be an initial risk management plan document and table spreadsheet with the information related to this sprint filled out. **Requirements:** • Should have 2 documents ◦ a table that outlines the risks associated with various stories and tasks ◦ a text document that outlines the risks associated with the project and how we will manage them **Documentation folder link:** https://drive.google.com/drive/folders/1ocCRDHgzcdrVHsaVoYR3DWzO_Zd2ajwa?usp=sharing

<table>
<tr><th>T</th><th>Key</th><th>Summary</th><th>Story point estimate</th><th>Assignee(s)</th><th>P</th><th>Risk</th><th>Parent</th><th>Start date</th><th>Due</th><th>Status</th><th>Description</th></tr>
<tr><td>☑</td><td>COV-38</td><td>Create initial software architecture document</td><td>3</td><td>Andre Ibrahim</td><td>⌄</td><td>Low</td><td>⚡ Documentation</td><td>24/Jan/22</td><td>31/Jan/22</td><td>DONE</td><td><b>Definition of done:</b> There should be an initial Software Architecture Document / Architecture Description with the information from sprint 1 filled out.
<b>Requirements:</b>
• A basic architecture description
• Include all the elements of the template
• Fill out basic sections
<b>Documentation folder link:</b>
https://drive.google.com/drive/folders/1ocCRDHgzcdrVHsaVoYR3DWzO_Zd2ajwa?usp=sharing</td></tr>
<tr><td>🔖</td><td>COV-42</td><td>As a User, I want to be able to sign up, so that I can access the apps features</td><td>13</td><td>Andre Ibrahim, Domenic Seccareccia, Ejazali Rezayi, Jason Gerard</td><td>⌃</td><td>High</td><td>⚡ Authentication and Authorization</td><td>17/Jan/22</td><td>31/Jan/22</td><td>DONE</td><td><b>Definition of done:</b> A user should be able to create an account through the app/web and have the data persist in the database.
<b>Requirements:</b>
<b>UI Prototype:</b>
• Should have all required fields
• Fields should be validated by mockup spec
<b>Front end:</b>
• Should be accessible to all logged out users (does not adjust based on persona)
• Fields should be validated before moving to next step of sign up
<b>API:</b>
• User data should be accepted as JSON through a HTTP API
• Data should be validated
• Password should be hashed
• Data should be persisted in the database through the repository layer
<b>Specifications</b>
https://docs.google.com/document/d/1C3p04EvYwDrRzkEms1i93RnMfW-e7B2UBCcJcgx7aoQ/edit?usp=sharing
<b>Personas (accessible by):</b>
• User
◦ Patient
◦ Doctor
◦ Health Official
◦ Immigration Officer
◦ Administrator
<b>Acceptance tests:</b>
<table>
<tr><td><b>AT-1</b></td><td><b>COV-42 - As a user I was to be able to sign up so that I can access the apps features</b></td></tr>
<tr><td><b>Acceptance Criteria</b></td><td>GIVEN that I am on the sign up page
AND that I input all required fields
AND that I clicked the Sign Up button
THEN my account should be created
AND I should be logged in
AND I should be redirected to the main screen.</td></tr>
<tr><td><b>Result</b></td><td><b>PASS</b></td></tr>
</table>
<b>System tests:</b>
<table>
<tr><td><b>ST-1</b></td><td><b>COV-42 - As a user I was to be able to sign up so that I can access the apps features</b></td></tr>
<tr><td><b>Steps to reproduce</b></td><td><b>Expected output for each step</b></td></tr>
<tr><td>1. Navigate to the sign up page (relative url "/sign_up")
2. Fill all required fields with valid inputs
3. Click the Sign Up Button</td><td>1. You should see the sign up page
2. The form should not give any input errors
3. The form should not give any input errors, your account should be created, you should be signed in, and you should be redirected to the main screen</td></tr>
</table>
</td></tr>
</table>

T	Key	Summary	Story point estimate	Assignee(s)	P	Risk	Parent	Start date	Due	Status	Description
	COV-48	As a User, I want to be able to sign in, so that I can access my account	8	Daren, Domenic Seccareccia, Jason Gerard	=	High	Authentication and Authorization	24/Jan/22	30/Jan/22	DONE	**Definition of done:** A user with an account should be able to sign in through the app / web.

Requirements:

UI Prototype:

- Should contain email and password inputs

Front end:

- User should login with email and password
- User session should be created using a JWT
- Should store JWT in redux store and cookies / local storage
- Does not adjust based on persona

API:

- Should validate email and password
- Get user from db by email
- If hashed passwords match then encode a new JWT with the userId and expiration date (3600 seconds)
- Return encoded JWT to user

Specifications:

https://docs.google.com/document/d/1HLmlmzUciMa0j7RRtiJRvOaJCe14rchHjNB1BaspLlw/edit?usp=sharing

Personas (accessible by):

- User
    - Patient
    - Doctor
    - Health Official
    - Immigration Officer
    - Administrator

Acceptance Tests:

AT-2	**COV-48 - As a user I want to be able to sign in so that I can access my account**
Acceptance Criteria	GIVEN that I am on the sign in page AND that I input my valid email AND that I input my valid password AND that I clicked the Sign In button THEN I should be logged into the site AND my session should persist AND I should be redirected to the main screen.
Result	**PASS**

System Tests:

ST-2	**COV-48 - As a user I want to be able to sign in so that I can access my account**
Steps to reproduce	**Expected output for each step**
1. Navigate to the sign in page (relative url “/sign_in”) 2. Input your email and password 3. Click the Sign In Button	1. You should see the sign in page 2. The form should not give any input errors 3. The form should not give any input errors, you should be signed in, and you should be redirected to the main screen

<table>
<tr><th>T</th><th>Key</th><th>Summary</th><th>Story point estimate</th><th>Assignee(s)</th><th>P</th><th>Risk</th><th>Parent</th><th>Start date</th><th>Due</th><th>Status</th><th>Description</th></tr>
<tr><td>🔖</td><td>COV-52</td><td>As a User, I want to be able to sign out, so that I can delete my session</td><td>5</td><td>Domenic Seccareccia, Jason Gerard, Khachig Astor</td><td>=</td><td>High</td><td>⚡ Authentication and Authorization</td><td>28/Jan/22</td><td>30/Jan/22</td><td>DONE</td><td>**Definition of done:** A signed in user should be able to sign out which would clear the token from their cookies / localstorage.
Requirements:
UI Prototype:
• Should include desktop and mobile navbar
• Add sign out button to navbar
Front end:
• Implement navbar
• Implement sign out button when user is signed in
• When sign out button is clicked delete data from redux store and from local storage
• Does not adjust based on persona)
Specifications:
https://docs.google.com/document/d/1NIbO-MbMcLwhg7C3ubWRe9eY2HCvzBgXLNDEYFwDGE0/edit?usp=sharing
Personas (accessible by):
• User
◦ Patient
◦ Doctor
◦ Health Official
◦ Immigration Officer
◦ Administrator
Acceptance Tests:
<table>
<tr><td>**AT-3**</td><td>**COV-52 -As a user I want to be able to sign out so that I can delete my session**</td></tr>
<tr><td>**Acceptance Criteria**</td><td>GIVEN that I am on a page with a navbar
AND that I am signed in
AND that I clicked the Sign Out button
THEN I should be logged out of the site
AND my session should be deleted
AND I should be redirected to the sign in page.</td></tr>
<tr><td>**Result**</td><td>**PASS**</td></tr>
</table>
System Tests:
<table>
<tr><td>**ST-3**</td><td>**COV-52 -As a user I want to be able to sign out so that I can delete my session**</td></tr>
<tr><td>**Steps to reproduce**</td><td>**Expected output for each step**</td></tr>
<tr><td>1. Navigate to a page where the navbar can be seen
2. Click the Sign Out Button</td><td>1. You should see the sign out button in the navbar
2. You should be signed out, your session should be deleted, and you should be redirected to the Sign In page</td></tr>
</table>
Dev notes:
• JWTs are stateless and so the server cannot ‘invalidate’ one
• To logout with a JWT you simply clear the JWT stored in the users cookies / localstorage</td></tr>
<tr><td>☑</td><td>COV-76</td><td>Create architecture component diagram</td><td>2</td><td>Jason Gerard</td><td>=</td><td>Low</td><td>⚡ Documentation</td><td>24/Jan/22</td><td>28/Jan/22</td><td>DONE</td><td>**Definition of done:** There should be a component diagram defining the system-to-be architecture.
Requirements:
• Create `draw.io` file of architecture diagram
• Diagram should follow UML component diagram standards
• Should represent the layered architecture
◦ Controllers
◦ Services
◦ Repositories
Documentation folder link: https://drive.google.com/drive/folders/1ocCRDHgzcdrVHsaVoYR3DWzO_Zd2ajwa?usp=sharing
Diagram link: https://drive.google.com/file/d/12iKdkB5sOOqflkbyF5n1bY1xFr58-gT1/view?usp=sharing</td></tr>
<tr><td>☑</td><td>COV-84</td><td>Create domain model diagram</td><td>2</td><td>Andre Ibrahim, Jason Gerard</td><td>=</td><td>Low</td><td>⚡ Documentation</td><td>29/Jan/22</td><td>30/Jan/22</td><td>DONE</td><td>**Definition of done:** There should be an domain model diagram defining the entities and their relationships in our problem domain.
Requirements:
• Create `draw.io` file of a domain model diagram
• Diagram should follow UML class diagram diagram standards
• Should represent the important entities in our domain
◦ should not be overly specific (this shouldn’t have to be updated very often)
Documentation folder link: https://drive.google.com/drive/folders/1ocCRDHgzcdrVHsaVoYR3DWzO_Zd2ajwa?usp=sharing
Diagram link: https://app.diagrams.net/#G1dmbNVKywLPckfShuGJHM9J-3GodmuJS2</td></tr>
</table>

3.2 Backlog

See next page to view Backlog.

T	Key	Summary	Story point estimate	P	Risk	Parent
☑	COV-1	Setup GitHub repository	1	︽	Medium	⚡ Project Setup
☑	COV-3	Setup basic web server	5	︽	Medium	⚡ Project Setup
☑	COV-4	Setup basic project for front end client	5	︽	Medium	⚡ Project Setup
☑	COV-5	Setup Jira project	1	︿	Low	⚡ Project Setup
☑	COV-6	Configure CI/CD pipeline for web server	2	=	Low	⚡ Project Infrastructure
☑	COV-7	Setup docker for web server	5	=	Low	⚡ Project Infrastructure
🔖	COV-25	As a Patient, I want to update my status, so that I can keep my Doctor updated	13	=	High	⚡ Status Management
🔖	COV-26	As an Admin, I want to assign a Patient to a Doctor, so that I can manage the Patients	5	︿	Medium	⚡ Patient Management
🔖	COV-27	As an Admin, I want to view how many Patients a Doctor has, so that no Doctor has too many Patients	5	﹀	Low	⚡ Patient Management
☑	COV-30	Add testing framework to web server	2	=	Low	⚡ Project Infrastructure
☑	COV-33	Update risk management plan and table	2	﹀	Low	⚡ Documentation
☑	COV-38	Create initial software architecture document	3	﹀	Low	⚡ Documentation
🔖	COV-42	As a User, I want to be able to sign up, so that I can access the apps features	13	︿	High	⚡ Authentication and Authorization
🔖	COV-48	As a User, I want to be able to sign in, so that I can access my account	8	=	High	⚡ Authentication and Authorization
🔖	COV-52	As a User, I want to be able to sign out, so that I can delete my session	5	=	High	⚡ Authentication and Authorization
☑	COV-76	Create architecture component diagram	2	=	Low	⚡ Documentation
☑	COV-84	Create domain model diagram	2	=	Low	⚡ Documentation
🔖	COV-85	As an Admin, I want to add roles to users, so that I can manage access rights	13	︽	Medium	⚡ Authentication and Authorization
🔖	COV-90	As an Admin, I want the API to authorize users by role, so that access rights are managed	8	︽	High	⚡ Authentication and Authorization
🔖	COV-95	As a Doctor, I want to define the status report fields for my Patients, so I can properly track them	8	=	High	⚡ Status Management

T	Key	Summary	Story point estimate	P	Risk	Parent
🔖	COV-107	As a Patient, I want to input my COVID test results, so that I can report if I tested positive or negative	8	=	High	Patient Management
🔖	COV-108	As a Patient, I want to update my status for the day after already submitting, so that my Doctor stays up to date		=		Status Management
🔖	COV-111	As a Doctor or Health Official, I want to view all the line item statuses of a Patient, so that I can monitor their progress over time		=		Status Management
🔖	COV-112	As a Doctor or Health Official, I want to view the details of a single day of statuses of a Patient, so that I can view their progress at a point in time		=		Status Management
🔖	COV-113	As a Doctor or Health Official, I want to view a line item list of my patients with their most recent line item status update, so that I can keep track of any updates	5	=	Medium	Status Management
🔖	COV-114	As a Doctor, Health Official, or Immigration Officer, I want to flag certain patients, so that their updates are prioritized over others		=		Patient Management
🔖	COV-115	As a Doctor, I want to mark a Patients status update as "Reviewed", so that I can see which statuses I've already seen		=		Status Management
🔖	COV-116	As a Doctor, I want to book an appointment with a Patient, so that we can discuss their symptoms		=		Patient Management
☑	COV-117	Create generic websocket direct messaging infrastructure		=		Messaging System
🔖	COV-118	As a Doctor, I want to direct message my Patients, so that I can keep up to date with their questions		=		Messaging System
🔖	COV-119	As a Patient, I want to direct message my Doctor, so that I can ask them questions		=		Messaging System
🔖	COV-120	As a Patient, I want to mark my message with a priority level, so that my Doctor will view it quicker		=		Messaging System
🔖	COV-121	As a Doctor or Patient, I want to be able to generate a QR code for a status report, so that I can share it with others		=		QR Code System
🔖	COV-122	As a Doctor or Patient, I want to be able to generate a QR code for a lab test result, so that I can share it with others		=		QR Code System
🔖	COV-123	As a Doctor or Patient, I want to be able to view a line item list of my COVID test results, so that I'm aware of my diagnosis	5	⌄	Medium	Patient Management
🔖	COV-124	As a Doctor or Patient, I want to be able to view the details of a single COVID test result, so that I'm aware of my diagnosis		=		Patient Management
🔖	COV-125	As a Doctor or Health Official, I want to see a time series graph of my Patients symptoms, so that I can see any changes		=		Status Management
🔖	COV-126	As a Health Official, I want to contact trace who a Patient has been in contact with, so that I can manage who is at risk		=		Contact Tracing
🔖	COV-127	As a Health Official, I want to flag contact traced patients, so that I can monitor their status		=		Contact Tracing
🔖	COV-128	As a Health Official, I want to notify contact traced Patients, so that they are required to self quarantine		=		Notification System
☑	COV-129	Create generic email notification infrastructure		=		Notification System
☑	COV-130	Create generic SMS notification infrastructure		=		Notification System
🔖	COV-131	As a Doctor, I want to be notified when a Patient updates their status more than once in a day, so that I can look into their status		=		Notification System
🔖	COV-132	As a Doctor, I want to send a notification to a Patient, so that they are reminded of an upcoming appointment		=		Notification System

T	Key	Summary	Story point estimate	P	Risk	Parent
☑	COV-146	Setup a Dockerfile for the front end and integrate it with docker-compose	2	︾	Low	⚡ Project Infrastructure
☑	COV-147	Setup snapshot testing for the front end using Jest	1	⌄	Low	⚡ Project Infrastructure

4.0 RELEASE PLANNING

This section covers a summar and retrospective for sprint 1 and sprint 2 planning.

4.1. Sprint 1

4.1.1 Summary

Sprint 1 focused on delivering 3 user stories in the Authentication and Authorization epic. The user stories are for sign up, sign in, and sign out. We completed our sprint commitment and everything was delivered. There were not as many user stories this sprint as it focused on setting up the foundations of the system resulting in a heavy focus on various tasks that had to be completed around architecture and infrastructure. Our aim was to get the basics built and running properly so that we could start building key features without inducing immediate technical debt in the system. We set up the basic architecture for the web service, client, database and authentication system. The infrastructure for running all the different types of automated tests and a CI/CD pipeline integrated with our version control system were also added. Docker was integrated with docker-compose for the server and database.

Project velocity after Sprint 1: 56

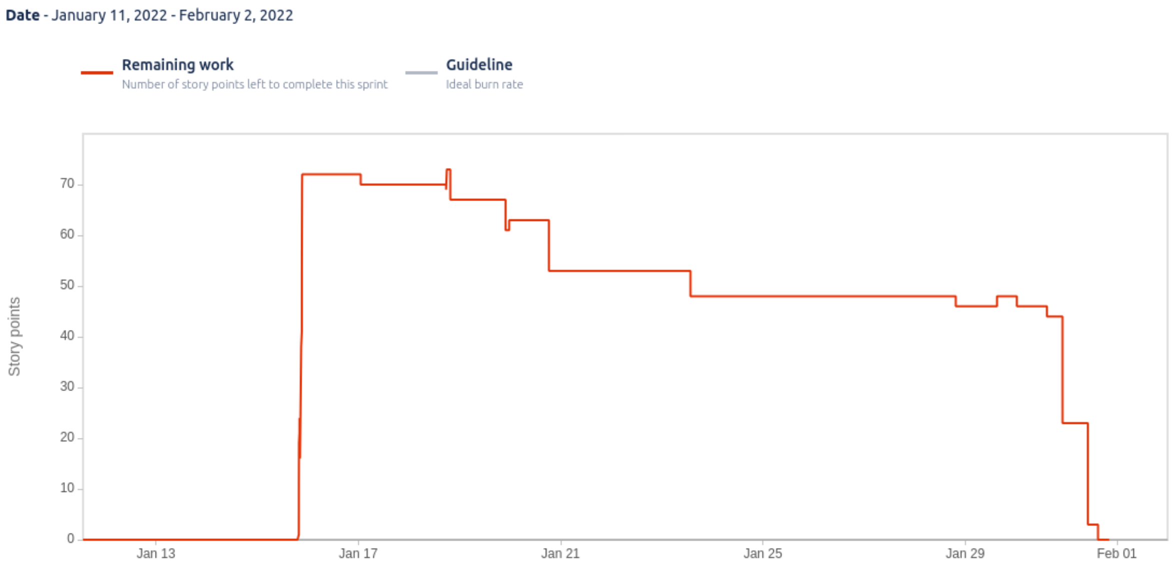
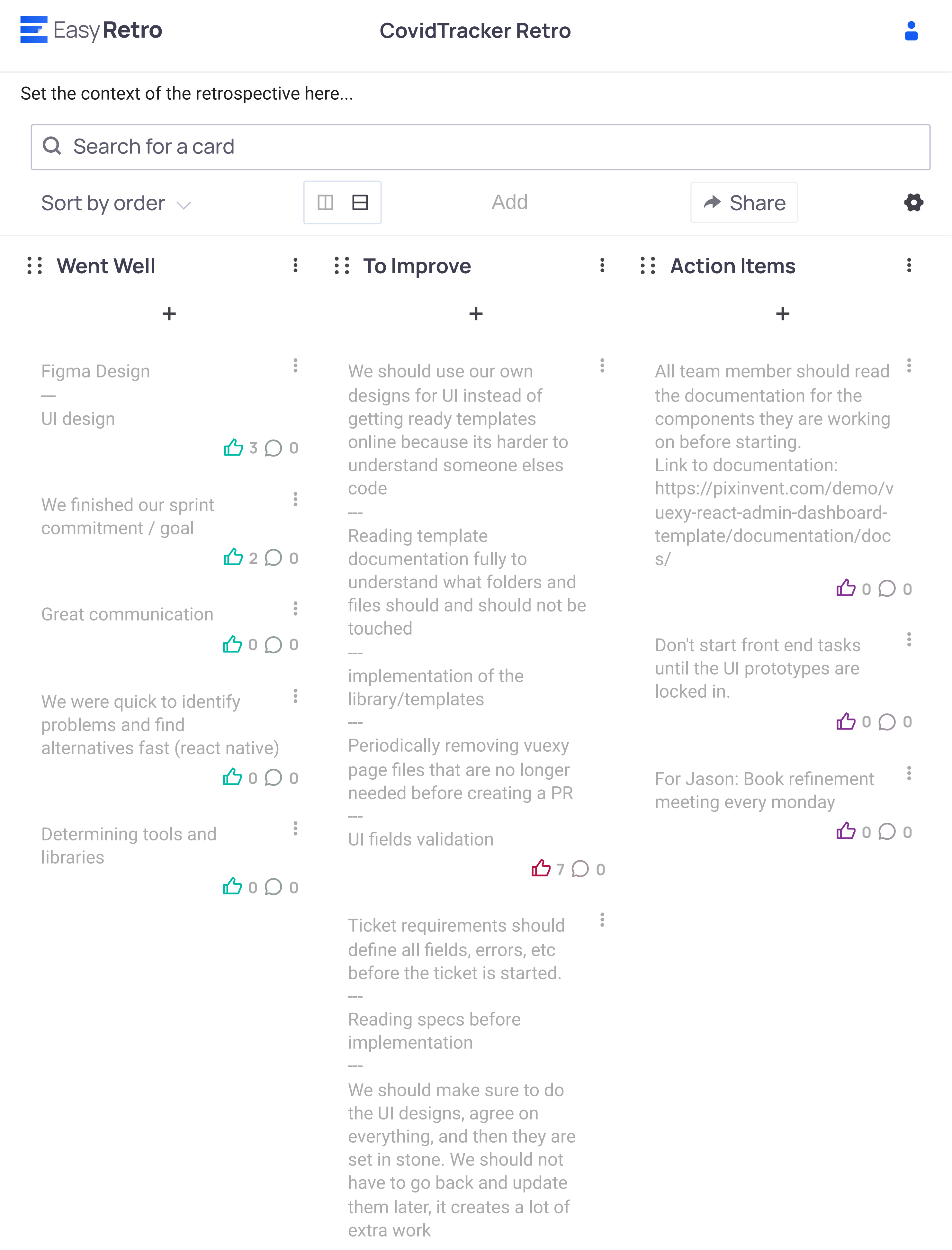


Figure 1: Burndown Chart

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Assignee(s)	Start date	Due
☑	COV-1	Setup GitHub repository	1	⌃⌃	Medium	⚡ Project Setup	DONE	Jason Gerard	13/Jan/22	15/Jan/22
☑	COV-3	Setup basic web server	5	⌃⌃	Medium	⚡ Project Setup	DONE	Jason Gerard	13/Jan/22	15/Jan/22
☑	COV-4	Setup basic project for front end client	5	⌃⌃	Medium	⚡ Project Setup	DONE	Andre Ibrahim, Dan, Domenic Seccareccia, Khachig Astor, rafistep98	22/Jan/22	23/Jan/22
☑	COV-5	Setup Jira project	1	⌃	Low	⚡ Project Setup	DONE	Domenic Seccareccia, Jason Gerard	15/Jan/22	18/Jan/22
☑	COV-6	Configure CI/CD pipeline for web server	2	=	Low	⚡ Project Infrastructure	DONE	Jason Gerard	15/Jan/22	17/Jan/22
☑	COV-7	Setup docker for web server	5	=	Low	⚡ Project Infrastructure	DONE	Domenic Seccareccia, Jason Gerard	17/Jan/22	18/Jan/22
☑	COV-30	Add testing framework to web server	2	=	Low	⚡ Project Infrastructure	DONE	Jason Gerard	14/Jan/22	15/Jan/22
☑	COV-33	Update risk management plan and table	2	⌄	Low	⚡ Documentation	DONE	Dan, Khachig Astor	26/Jan/22	29/Jan/22
☑	COV-38	Create initial software architecture document	3	⌄	Low	⚡ Documentation	DONE	Andre Ibrahim	24/Jan/22	31/Jan/22
🔖	COV-42	As a User, I want to be able to sign up, so that I can access the apps features	13	⌃	High	⚡ Authentication and Authorization	DONE	Andre Ibrahim, Domenic Seccareccia, Ejazali Rezayi, Jason Gerard	17/Jan/22	31/Jan/22
🔖	COV-48	As a User, I want to be able to sign in, so that I can access my account	8	=	High	⚡ Authentication and Authorization	DONE	Daren, Domenic Seccareccia, Jason Gerard	24/Jan/22	30/Jan/22
🔖	COV-52	As a User, I want to be able to sign out, so that I can delete my session	5	=	High	⚡ Authentication and Authorization	DONE	Domenic Seccareccia, Jason Gerard, Khachig Astor	28/Jan/22	30/Jan/22
☑	COV-76	Create architecture component diagram	2	=	Low	⚡ Documentation	DONE	Jason Gerard	24/Jan/22	28/Jan/22
☑	COV-84	Create domain model diagram	2	=	Low	⚡ Documentation	DONE	Andre Ibrahim, Jason Gerard	29/Jan/22	30/Jan/22

4.1.2 Retrospective

View the report of the sprint 1 retrospective meeting on the next page.



CovidTracker Retro

Set the context of the retrospective here...

Went Well

+

Figma Design

UI design

👍 3 💬 0

We finished our sprint commitment / goal

👍 2 💬 0

Great communication

👍 0 💬 0

We were quick to identify problems and find alternatives fast (react native)

👍 0 💬 0

Determining tools and libraries

👍 0 💬 0

To Improve

+

We should use our own designs for UI instead of getting ready templates online because its harder to understand someone elses code

---

Reading template documentation fully to understand what folders and files should and should not be touched

---

implementation of the library/templates

Periodically removing vuexy page files that are no longer needed before creating a PR

UI fields validation

👍 7 💬 0

Ticket requirements should define all fields, errors, etc before the ticket is started.

Reading specs before implementation

We should make sure to do the UI designs, agree on everything, and then they are set in stone. We should not have to go back and update them later, it creates a lot of extra work

Action Items

+

All team member should read the documentation for the components they are working on before starting.
Link to documentation: https://pixinvent.com/demo/vuexy-react-admin-dashboard-template/documentation/docs/

👍 0 💬 0

Don't start front end tasks until the UI prototypes are locked in.

👍 0 💬 0

For Jason: Book refinement meeting every monday

👍 0 💬 0

Following UI mockup elements 1 for 1

✕ 👍 6 💬 0

update the jira board when doing task ⋮

Taking initiative when Jira tickets are available

👍 5 💬 0

Ask for help to others when stuck on a part ⋮

---

if takes takes longer than it should we should update the team on it

✕ 👍 5 💬 0

we should have meetings more often ⋮

Communicating any changes to UI not present in the mockups

✕ 👍 4 💬 0

Starting documentation right away ⋮

👍 1 💬 0

4.2 Sprint 2

4.2.1 Planning

Sprint 2 will focus on delivering user stories in the Patient Management and Status Management epics. These stories focus on integrating the doctor, patient, and administrator roles and actions into the system. The features span the client, server, and database and so integration between all the different parts is always tricky. Our goal is to have frequent meetings and open channels of communication, as discussed in our retrospective meeting, in order to make integrating between developers easier. We have a couple UI bugs that we are also looking to clean up from the first sprint that we didn’t get a chance to tackle by the end of sprint 1. Finally we have 2 tasks for front end infrastructure work that needs to be finished. First is to set up the client application in a docker container and configure it with the docker-compose file. The second is to configure the snapshot testing environment for the front end components so we can start unit testing them.

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🟩	COV-25	As a Patient, I want to update my status, so that I can keep my Doctor updated	13	=	High	⚡ Status Management	TO DO	**Definition of done:** A patient should be able to submit their custom doctor defined status report once a day. **Requirements:** **UI Prototype:** • “Submit status report page” ◦ Input fields for all details should be defined ◦ Submit button ◦ Should be added to navbar for patients **Front end:** • “Submit status report page” ◦ When the page loads query the API to get a key value pair of which fields the patients have to fill out (true) and filter out the fields they are exempt from (false) ▪ {fieldName: boolean} ◦ Submit the form with all fields (including exempt), fields that the user is exempt from should be null ◦ Page should show an error if the form is submitted more than once in the same day by the user ▪ This error will come from the back end, just display it ◦ Page should only be available to the PATIENT role ◦ Should be added to navbar for patients **API:** • Endpoint to query status report fields for a patient ◦ GET to `/patient/:patientId/status/fields` ◦ Body: none ◦ Add route verification for “patientId in route is patientId in JWT” ◦ Query `status_fields` table to get fields for the user ◦ Returns status 200 and key value pairs ▪ {fieldName: boolean} • Endpoint to submit status report for a patient ◦ POST to `/patient/:patientId/status` ◦ Body: {fieldName: value} ◦ Add route verification for “patientId in route is patientId in JWT” ◦ Insert into `statuses` ◦ Returns status 201 **Database:** • New table called `statuses` which has a column for user_id, created_on, and a column for each possible field ◦ Primary key is a composite of user_id and created_on ◦ Fields should be nullable and each field not submitted by the user should be set to null **Personas (accessible by)** • Patient

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-26	As an Admin, I want to assign a Patient to a Doctor, so that I can manage the Patients	5	︿	Medium	⚡ Patient Management	TO DO	**Definition of done:** There should be a page for an admin to assign a patient by user Id to a doctor by user Id. **Requirements:** **UI Prototype:** • Input field for patient userId • Input field for doctor userId • Submit button • Should be added to navbar for admins **Front end:** • Page should on be accessible to admins • Should be added to navbar for admins **API:** • POST to `/patient/:patientId/doctor` • Body: doctorId • Add route verification for admin role only • Returns status 201 **Database:** • Add new column `assigned_doctor_id` to `patients` table that is a foreign key to a `doctor_id` in the `doctors` table ◦ This column will hold the passed value of doctor id for the patient **Personas (accessible by)** • Administrator
🔖	COV-27	As an Admin, I want to view how many Patients a Doctor has, so that no Doctor has too many Patients	5	﹀	Low	⚡ Patient Management	TO DO	**Definition of done:** There should be a page for admins to view were they can see a list of doctors and the number of patients each has. **Requirements:** **UI Prototype:** • List view page with a column of doctor names, doctor user ids, and the number of patients • Should be added to navbar for admins **Front end:** • Page should only be accessible to admin role • Should be added to navbar for admins **API** • GET to `/doctor/patient_count` • Add route verification for admin role only • Returns status 200 ◦ List of key value pair {doctorName, doctorId, patientCount} **Personas (accessible by)** • Administrator

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-85	As an Admin, I want to add roles to users, so that I can manage access rights	13	⏫	Medium	⚡ Authentication and Authorization	TO DO	**Definition of done:** An Admin user should be able to add roles to a user by user id through the “add roles” page. **Requirements:** **UI Prototype:** • Page only accessible by admin role • Input field for userId • Check boxes (multi select) for roles to be added • Should be added to navbar for admins **API:** • New endpoint to add roles to a user ◦ POST to `/user/:userId/role` ◦ Body: list of roles to be added ◦ Should create row in corresponding tables for each role passed with the userId ◦ Add route verification for admin role only ◦ Returns status 201 ◦ Should be added to navbar for admins • Update sign up flow to add default role of USER to all users accounts created **Database:** • Add new tables for all role types ◦ types ▪ patients ▪ doctors ▪ health_officials ▪ immigration_officers ▪ admins ◦ These should only have a primary key {role_singular}_id that is also a foreign key to the users table ▪ Example: patient_id, doctor_id, admin_id, etc **Personas (accessible by):** • Adminstrator

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-90	As an Admin, I want the API to authorize users by role, so that access rights are managed	8	⏫	High	⚡ Authentication and Authorization	TO DO	**Definition of done:** Routes on the front end and back end should have easy to use helper function to manage access rights. **Requirements:** **Front end:** • Should store roles in redux and cookies/local storage • Should have a helper function that can be applied to routes to limit access to certain roles by the values in redux and if the user is logged in or not ◦ isAuthorizedForRole([“ADMIN”]) ◦ isLoggedIn() **API:** • Sign In and Sign Up API should return ◦ JWT which should contain ▪ userId ◦ list of the user roles ◦ userId ◦ first and last name • Should have a helper function that can be applied to API endpoints to limit access to certain roles by the value passed in the authorization header and if the user is logged in or not ◦ {{Authorization: Bearer Unknown macro: {token} }} ◦ isAuthorizedForRole([“ADMIN”]) ◦ isLoggedIn() **Personas (accessible by)** • Administrator

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-95	As a Doctor, I want to define the status report fields for my Patients, so I can properly track them	8	=	High	⚡ Status Management	TO DO	**Definition of done:** A doctor should be able to define the list of fields their patients must fill out for the status report. **Requirements:** **UI Prototype:** • “Define status report for patient page” ◦ input field for patient user id ◦ Check list (multi select) from master list of acceptable fields ▪ temperature ▪ weight ▪ list of symptoms ▪ **TBD, needs to be refined** ◦ submit button ◦ Should be added to navbar for doctors **Front end:** • “Define status report for patient page” ◦ On submission should send a payload to the back end of a list of key value pairs for which fields are active (true) and which fields are disabled (false) for the user ▪ {fieldName: boolean} ◦ Page should only be available to the DOCTOR role ◦ Should be added to navbar for doctors **API:** • Endpoint to submit status report fields for a patient ◦ POST to `/patient/:patientId/status/fields` ◦ Body: {fieldName: boolean} ◦ Add route verification for “patient is assigned to doctor” only ◦ Returns status 201 **Database:** • Create a table `status_fields` that has a column for a patient_id and columns for each possible status report field ◦ The status report field columns will have a type of boolean NOT NULL ▪ true if its assigned to the patient ▪ false if its not **Personas (accessible by)** • Doctor

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-107	As a Patient, I want to input my COVID test results, so that I can report if I tested positive or negative	8	=	High	⚡ Patient Management	TO DO	**Definition of done:** A patient should be able to submit a form that details the results of an externally taken Covid test to persist the data in the system. **Requirements:** **UI Prototype:** • Should contain form fields ◦ Test result ◦ Type of test ◦ Location of test ◦ Date of test ◦ **To be refined** • submit button **Front end:** • Add page to navbar “submit test result” • Implement all form fields and data validation • Should only be accessible by patients **API:** • POST to `/patient/:patiendId/test` • Body: Fields of form with input • Should check patient id in jwt is patient id in path param • Store result in new db table • Return status 201 **Database:** • Create new db table `test_results` ◦ Columns for ▪ testId primary key ▪ patientId ▪ all other form fields

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-113	As a Doctor or Health Official, I want to view a line item list of my patients with their most recent line item status update, so that I can keep track of any updates	5	=	Medium	⚡ Status Management	TO DO	**Definition of done:** A doctor should be able to view a page with a line item list of their patients most recent status updates and a health official should be able to view a page with a line item list of all the patients most recent status updates. **Requirements:** **UI prototype:** • The design for both doctors and health officials should be the same • Should be a table list view • should only show the most important fields of the status not a detailed view **Front end:** • Both doctors and health officials should use the same page the only difference will be the data provided by the back end. • The list items should not be clickable, adding a clickable detailed view for each patient is added in another story **Back end:** • Endpoint for a doctors patients status will include a doctorId query parameter ◦ GET to ▪ {{/patient/status?doctorId= Unknown macro: {doctorId} &latest=true}} ▪ `/patient/status?latest=true` ◦ Body: None ◦ Add route verification for “is health official or does doctor id in query param match id in JWT” ◦ Return the latest status for each patient (filtered by doctor id when doctor id passed) ▪ Each status should contain all necessary info to populate the front end table ◦ Returns status 200 ▪ List of patient statuses

T	Key	Summary	Story point estimate	P	Risk	Parent	Status	Description
🔖	COV-123	As a Doctor or Patient, I want to be able to view a line item list of my COVID test results, so that I'm aware of my diagnosis	5	⌄	Medium	⚡ Patient Management	TO DO	**Definition of done:** A patient should be able to access a page with a line item list of their covid test results. **Requirements:** **UI Prototype:** • Should be a table with the important info about a test result ◦ postive / negative ◦ date of test ◦ type of test **Front end:** • Should implement UI design • Should only be accessible to patient role **API:** • GET to `/patient/:patientId/test` • Body: None • Check if patient id is the same id as in the JWT • Return status 200 ◦ List of covid test results ordered by date desc (newest at the top)
☑	COV-146	Setup a Dockerfile for the front end and integrate it with docker-compose	2	︾	Low	⚡ Project Infrastructure	TO DO	**Requirements:** We need to create a Dockerfile for the client side React application so that configuring the development environment is easier. This should also be adding to the docker-compose file where it depends on the server. This way we can start all our docker containers i.e. our local development environment with a single command.
☑	COV-147	Setup snapshot testing for the front end using Jest	1	⌄	Low	⚡ Project Infrastructure	TO DO	**Requirements:** We want to be able to generate snapshot stubs of our UI components and test against them. We already have the Jest testing framework installed and running. We just need to setup the snapshot testing component and add a npm script in the package.json file to run them.

5.0 SOFTWARE ARCHITECTURE

This section provides an overview of the system to be built using both a domain model and a component diagram depicting and describing the chosen design decisions of the system.

Date Issued	January 11, 2022
Status	Sprint 1 completed
Authors	Jason Gerard, Andre Ibrahim, Domenic Seccareccia
Reviewers	Domenic Seccareccia, Jason Gerard
Scope	The domain model covers the domain of the application while the component diagram covers the entire system in development.
Context	This is the first sprint for the web application “CovidTracker”. Diagrams will be expanded and improved over each sprint iteration.

Table 4: Supplementary Information

5.1 Stakeholder Concerns

Stakeholder concerns associated with CovidTracker are depicted in the following Stakeholder Concern Traceability Matrix. Only stakeholders that have a concern impacted by the systems architecture are present in this table.

	Developer	**Project Champion**	**Testers**	**Product Owner**	**User**
System failure	⊗	⊗			

security breach	⊗	⊗			
unscalable architecture	⊗	⊗			
tightly coupled layers	⊗	⊗			
System complexity	⊗	⊗	⊗		
longer development time		⊗		⊗	⊗

Table 5: Stakeholder Concern Traceability Matrix

5.2 Diagrams

5.2.1 Domain Model

The UML domain model for CovidTracker describing all system entities, relationships and associations is represented by the UML domain model diagram seen in Figure 2. The diagram can be viewed in draw.io through this link:

- UML Domain Model Class Diagram of CovidTracker

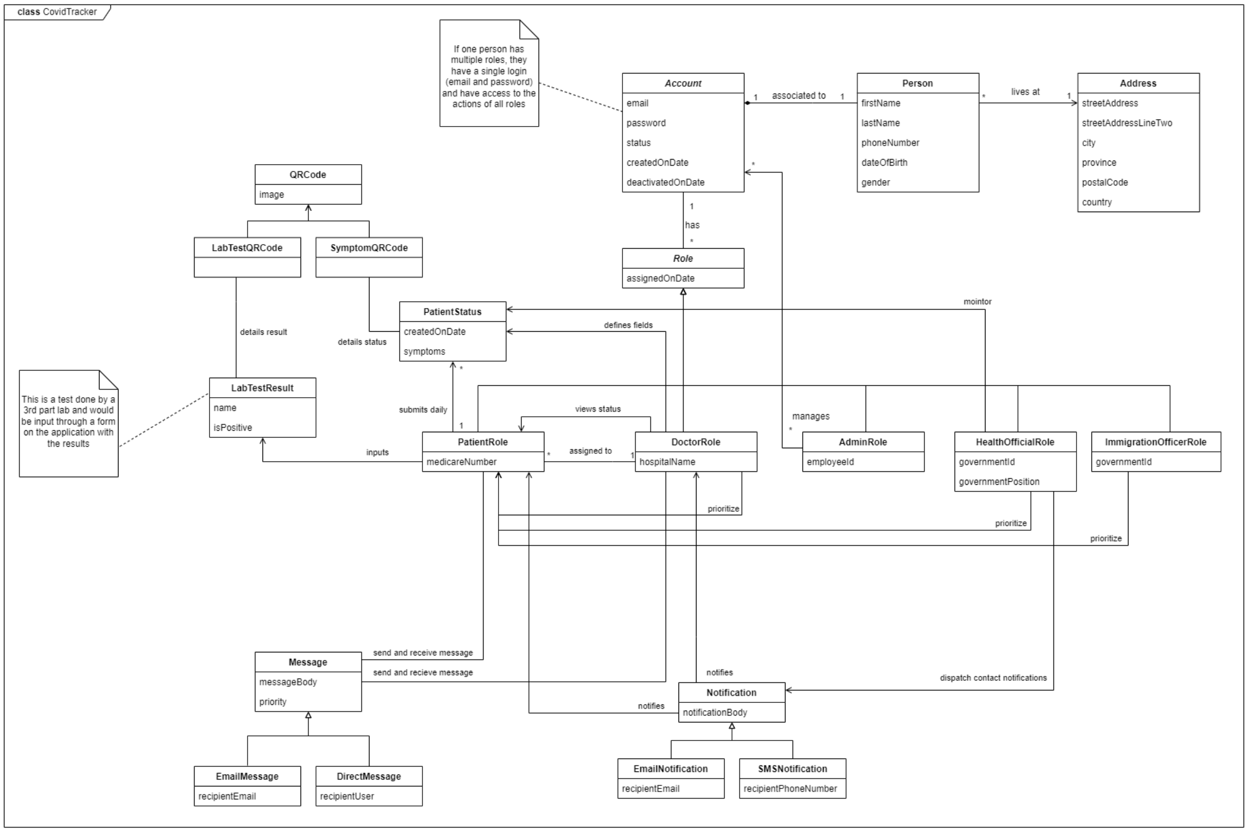


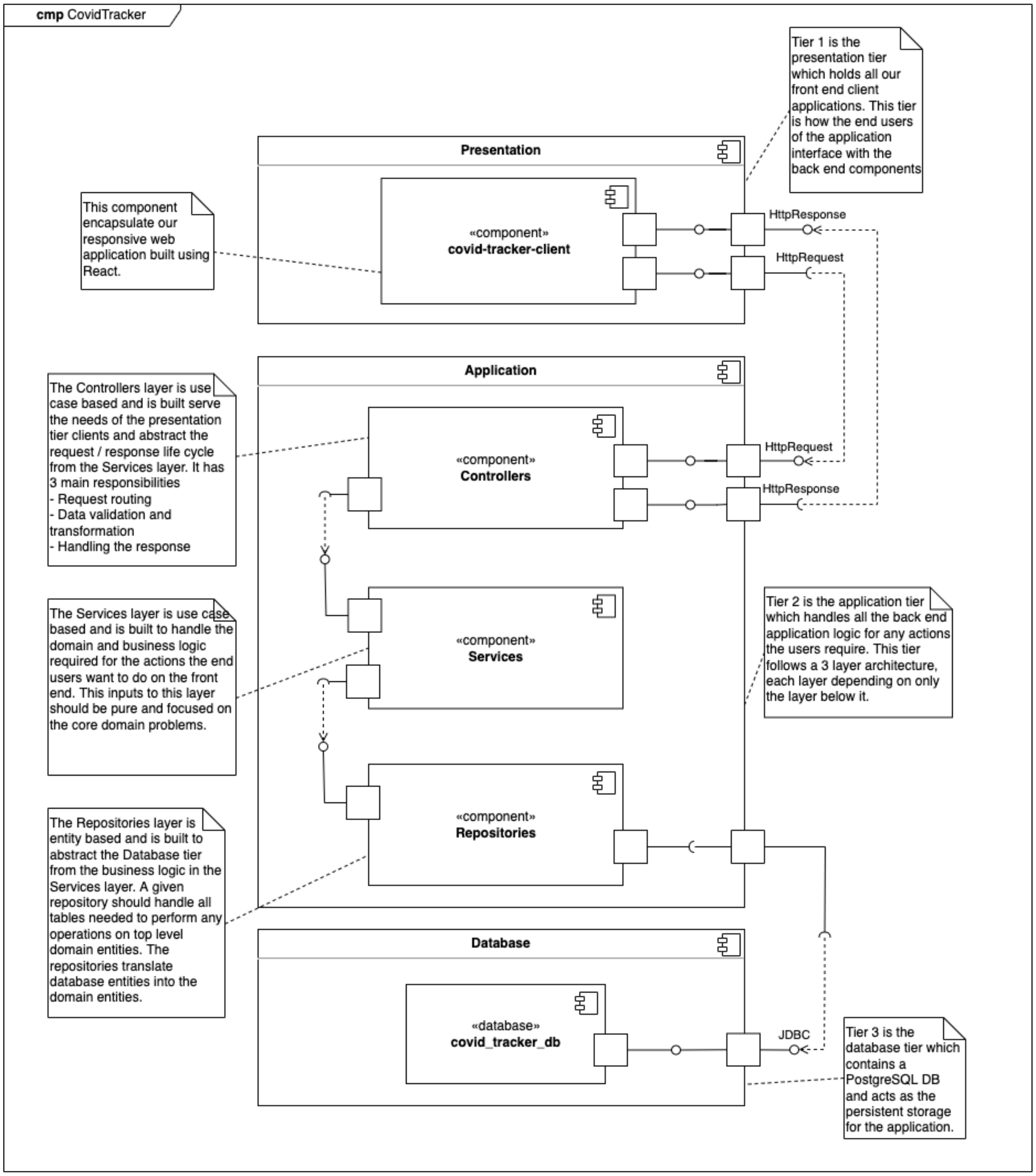
Figure 2: UML Domain Model Class Diagram of CovidTracker

A few key decisions were made during the creation of the Domain Model. The user entity is split into 2 separate person and account entities. A user has a single account that then can have many specialized roles through inheritance (i.e. DoctorRole, AdminRole, PatientRole etc.). This allows the scenario whereby a user can be/have both an administrator and patient account if such a scenario were to arise. Therefore, encapsulating the core attributes of a role and keeping it modular from the account itself gives us a lot of flexibility when designing the authentication and authorization system.

5.2.2 Component Diagram

The UML component diagram for CovidTracker (see Figure 3) describes the layers of the systems as well as the components within each layer and their relationships. The diagram can be viewed in draw.io through this link:

- Component Diagram of CovidTracker



Tier 1 is the presentation tier which holds all our front end client applications. This tier is how the end users of the application interface with the back end components

This component encapsulate our responsive web application built using React.

The Controllers layer is use case based and is built serve the needs of the presentation tier clients and abstract the request / response life cycle from the Services layer. It has 3 main responsibilities
- Request routing
- Data validation and transformation
- Handling the response

The Services layer is use case based and is built to handle the domain and business logic required for the actions the end users want to do on the front end. This inputs to this layer should be pure and focused on the core domain problems.

Tier 2 is the application tier which handles all the back end application logic for any actions the users require. This tier follows a 3 layer architecture, each layer depending on only the layer below it.

The Repositories layer is entity based and is built to abstract the Database tier from the business logic in the Services layer. A given repository should handle all tables needed to perform any operations on top level domain entities. The repositories translate database entities into the domain entities.

Tier 3 is the database tier which contains a PostgreSQL DB and acts as the persistent storage for the application.

Figure 3: Component Diagram of CovidTracker

The architecture of the system follows a 3-Tier architecture with the middle application tier using a layered architecture. The 3 tiers include the presentation tier, application tier and database tier. The presentation tier is the front end of the application. The application tier contains our web server which handles all the business logic of our application. Lastly, the database tier is the persistent storage layer for the application. The application tier has 3 layers which include: controller, service, and repository layers that allows the system to have low coupling and high cohesion. We utilize dependency injection to further separate the concerns between our layers. The standard flow of data starts at the presentation tier (the frontend) where a HTTP request is made to the application tier. The controller layer handles all routing, passing the data to the service layer, then to the repository layer to convert to schema form and make the JDBC connection with the database tier to persist the data.

5.3 Tech Stack

5.3.1 Presentation Tier

The client is encapsulated by the presentation tier and currently contains a single front end service, built using JavaScript and React, which is the primary way a client can interact with the system.

React was chosen because it is one of the largest and most tested front end frameworks on the open source market. Other competitors include Vue.JS and Angular. Angular is outdated and leans towards a thick client approach which is mostly considered an anti-pattern with front end web applications as they can become slow and hard to maintain overtime. While Vue.JS is a great framework, React ended up being chosen simply because more people on the team had experience with it.

The front end also uses SCSS which is a CSS preprocessing language that allows us to extend what can normally be done with CSS such as adding variables and mixins.

The client will be running in its own docker container which will allow it to be easily productionalized and deployed while also being compatible on all our developers' local machines.

5.3.2 Application Tier

The application tier consists of a single web server accepting HTTP requests from the front end client. This monolithic service is built using TypeScript and runs inside a docker container. TypeScript was chosen due to its clean syntax and advanced type system. As a result of TypeScript being transpiled in JavaScript, all JavaScript libraries work with TypeScript as well. This means we can also take advantage of the vast open source libraries built over the years by the web service oriented JavaScript community. Since we wanted to use a statically typed language to make it easier to model our domain in code and build clean type hierarchies, Python and JavaScript were out. Java was also considered however even with recent advancements aiming to improve the language it is still verbose and extremely opinionated in its packages with Java Spring being a perfect example. There was simply a desire to choose a language that was more flexible.

The web framework the service uses is Restify paired with the dependency injection package called Inversify. Restify was chosen because it is a simple framework that is not opinionated and allows us to only use and build what is needed. Thus abstracting away a lot of the boiler plate while still providing the ability to customize the service as needed. Other alternatives include express which has great 3rd party library support due to its popularity.

The web service uses an NPM package called pg which is a non-blocking client for PostgreSQL that provides the ability to write native SQL statements and abstract away the database connection pooling and type coercion. The main alternative for database interaction would be to use an object relational mapper (ORM), a popular one being TypeORM, that abstract away the entire concept of SQL statements and schema and leave a simple API for the application developer to work with. In our team's experience ORMs only make it easy to do the simple stuff which is easy to do anyways. When it comes to the complex parts you have to write the SQL by hand anyways so it's best to have full control from the start.

5.3.3 Database Tier

The system is currently using a single PostgreSQL database to store all the needed relational data. PostgreSQL was chosen as it is the most modern and maintained open source relational database. There is a large community that creates many libraries for all different types of languages and so many people on the team were already familiar with its standard SQL syntax. Other alternatives include MySQL and

Oracle Database but these database systems lack some of the advanced features and polish PostgreSQL has. The PostgreSQL instance our application uses runs inside its own docker container to make setup quick and easy for the development team.

5.4 External Libraries

5.4.1 Vuexy

Vuexy is a user interface templating library used for building user interface components and layouts in CovidTracker. Vuexy provides a fast and easy approach for building responsive web applications as it is built with a mobile first mindset. As such, this ensures that any element, component or feature built with the corresponding set of tools provided results in a responsive application from the onset. Without such a library, more painstaking time would be required to ensure that all user interface components function as expected regardless of platform. As such, developers can spend more time focusing on device compatibility and fine tuning the overall user experience.

Vuexy's underlying architecture is built on various external React libraries and Bootstrap 5, thus ensuring vast compatibility and various ways to build elements. Developers are free to decide whether or not using a pre-built Vuexy component is a better option instead of either using a basic Bootstrap 5 component or building it from scratch.

Unfortunately, Vuexy does have some major disadvantages. The main one is the sheer complexity of the library. While one would assume it is simple in nature whereby you only need to search and find the relevant components, attempting to find, decipher and then import the associated code is rather complicated. There is an extreme amount of interconnectivity between the provided files that result in a less than ideal amount of time being spent understanding what is necessary for a single component to work. Secondly, due to the interconnectivity of files, it is a rather tedious process to delete any non relevant files given the complex references between all files and displayed errors if not removed properly. Lastly, while the documentation is rather informative and thorough, there is more to be desired in regards to further explanation of certain components and elements.

While other user interface templating libraries were researched - CoreUI, Fuse and Isomorphic -, Vuexy was decided as the go to for two reasons. The first one was due to the variety of pre-built elements and components provided. For example, with the inclusion of layouts, forms, authentication, localization, charts, graphs and interactive data table components, developers do not have to spend many hours building these complicated UI components once a level of understanding is achieved on how to use them. Secondly, Figma UI design files are included, a rarity for these types of libraries. The inclusion of these files are extremely helpful during UI mockup design as the designers do not have to recreate all the elements from the template from scratch. Thus, allowing designers to spend more time focusing on the overall user interface and experience. This also helps ensure both the UI mockups and developed interfaces are a 1 for 1 match.

6.0 RISK ASSESSMENT AND MANAGEMENT PLAN

The risk assessment and management plan for CovidTracker is depicted and described in the following table. No risks were added or turned out to not be risks in sprint 1.

Risk ID	Description	Resolved in Sprint	Strategy and Effectiveness	Probability	Impact
R-1	Computers can crash causing us to lose our work	1	We decided to store our work in the cloud. This strategy has proven successful thus far.	Low	Severe
R-2	Database crashing			Low	Severe
R-3	Database leak and all the patient's medical record is stolen			Low	Severe
R-4	Not having the same versions of software in our systems	1	We decided to use docker to ensure every member is on the same version while working on the project.	Low	Minor
R-5	Changes in the law preventing us from			Medium	Severe

	using GPS tracking				
R-6	Code being pushed to the main without validation	1	A CI/CD pipeline is used in Github and at least one code approval is needed before any code code is merged in the main branch. This ensures the author of the code cannot just merge the code without approval.	Low	Moderate
R-7	Timeline estimates unrealistic	1	Playing poker was used. Every member took a vote on each task for a realistic timeline and we took the average length as the final length for a task.	Medium	Moderate
R-8	Project team availability	1	Every member dedicated a certain amount of time for this class from the beginning of the semester.	Low	Moderate
R-9	Weak user participation (if no one decides to use the website)			Medium	Moderate

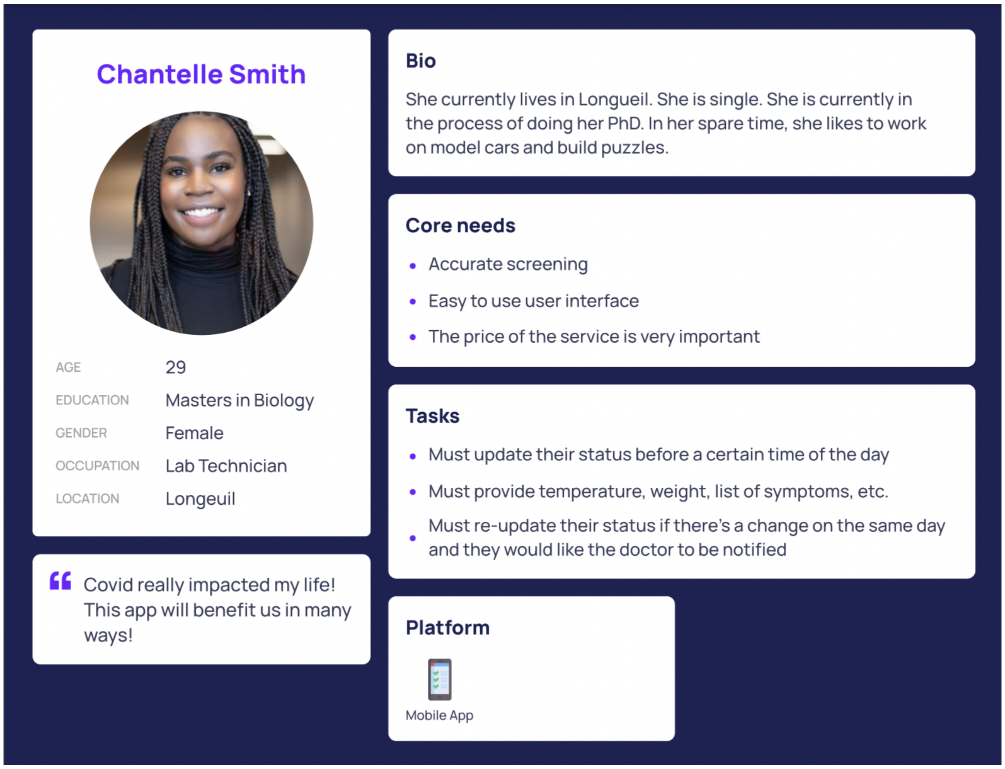
Table 6: Risk Analysis Table

7.0 USER INTERFACE DESIGN

All personas, supported devices, UI and user flow mockups, and interactive prototypes are depicted and described in the following sections.

7.1 Personas

CovidTracker is accessible by the following five user personas: Patients (see Figure 4), Doctors (see Figure 5), Health Officials (see Figure 6), Immigration Officers (see Figure 7) and Administrators (see Figure 8). Each persona is considered to be representative of a certain archetype within the general demographic.



Chantelle Smith

AGE	29
EDUCATION	Masters in Biology
GENDER	Female
OCCUPATION	Lab Technician
LOCATION	Longeuil

“ Covid really impacted my life! This app will benefit us in many ways!

Bio

She currently lives in Longueil. She is single. She is currently in the process of doing her PhD. In her spare time, she likes to work on model cars and build puzzles.

Core needs

- Accurate screening
- Easy to use user interface
- The price of the service is very important

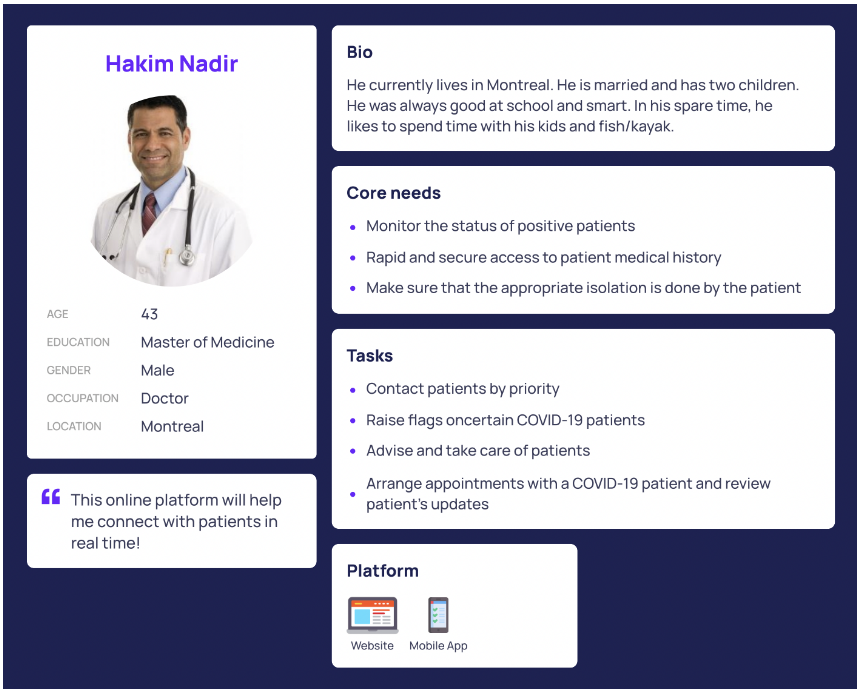
Tasks

- Must update their status before a certain time of the day
- Must provide temperature, weight, list of symptoms, etc.
- Must re-update their status if there's a change on the same day and they would like the doctor to be notified

Platform

Mobile App

Figure 4: Patient Persona



Hakim Nadir

AGE 43

EDUCATION Master of Medicine

GENDER Male

OCCUPATION Doctor

LOCATION Montreal

“This online platform will help me connect with patients in real time!

Bio

He currently lives in Montreal. He is married and has two children. He was always good at school and smart. In his spare time, he likes to spend time with his kids and fish/kayak.

Core needs

- Monitor the status of positive patients
- Rapid and secure access to patient medical history
- Make sure that the appropriate isolation is done by the patient

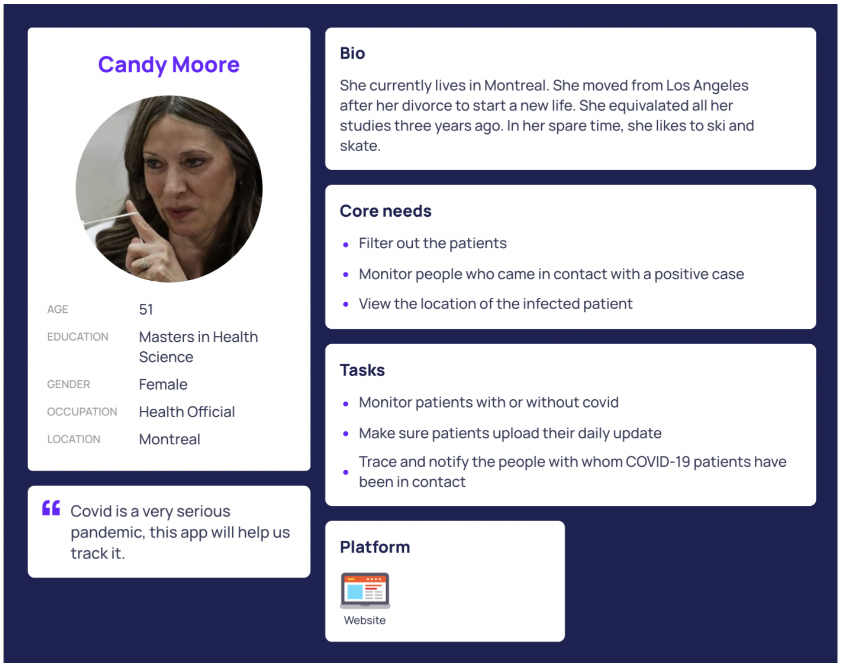
Tasks

- Contact patients by priority
- Raise flags oncertain COVID-19 patients
- Advise and take care of patients
- Arrange appointments with a COVID-19 patient and review patient's updates

Platform

Website Mobile App

Figure 5: Doctor Persona



Candy Moore

AGE 51

EDUCATION Masters in Health Science

GENDER Female

OCCUPATION Health Official

LOCATION Montreal

“Covid is a very serious pandemic, this app will help us track it.

Bio

She currently lives in Montreal. She moved from Los Angeles after her divorce to start a new life. She equivalated all her studies three years ago. In her spare time, she likes to ski and skate.

Core needs

- Filter out the patients
- Monitor people who came in contact with a positive case
- View the location of the infected patient

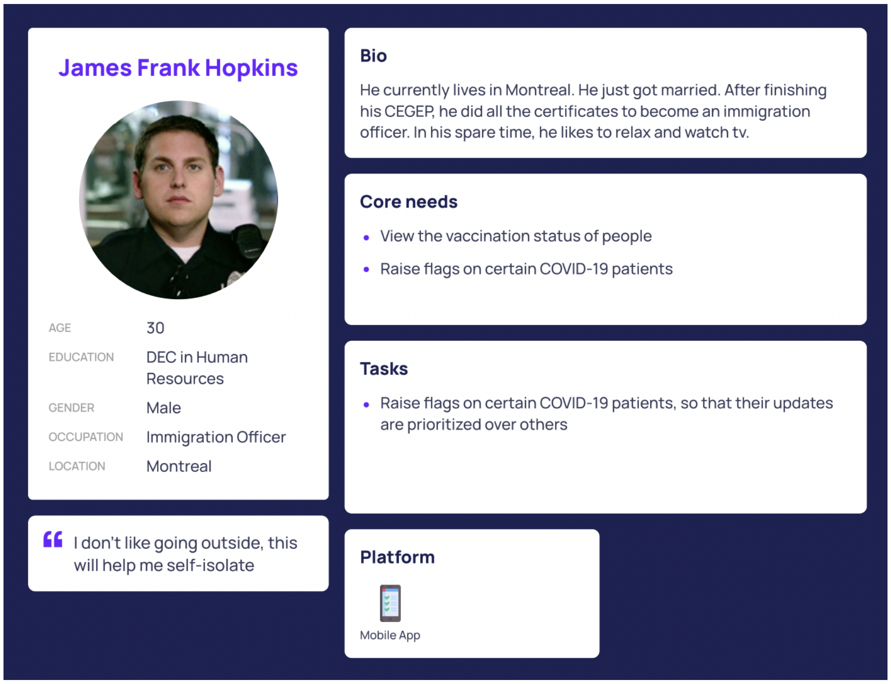
Tasks

- Monitor patients with or without covid
- Make sure patients upload their daily update
- Trace and notify the people with whom COVID-19 patients have been in contact

Platform

Website

Figure 6: Health Official Persona



James Frank Hopkins

AGE	30
EDUCATION	DEC in Human Resources
GENDER	Male
OCCUPATION	Immigration Officer
LOCATION	Montreal

> I don't like going outside, this will help me self-isolate

Bio

He currently lives in Montreal. He just got married. After finishing his CEGEP, he did all the certificates to become an immigration officer. In his spare time, he likes to relax and watch tv.

Core needs

- View the vaccination status of people
- Raise flags on certain COVID-19 patients

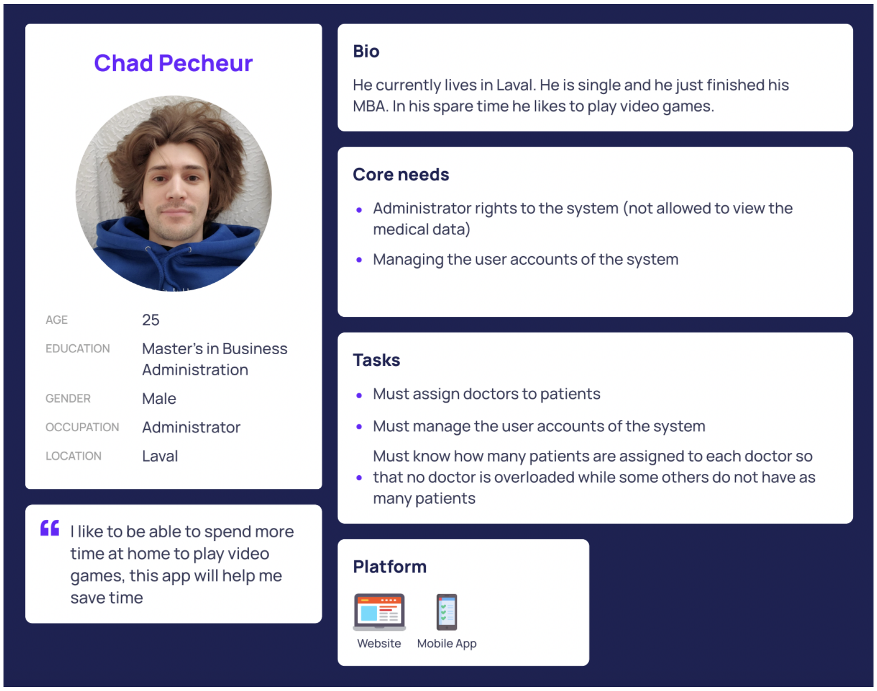
Tasks

- Raise flags on certain COVID-19 patients, so that their updates are prioritized over others

Platform

Mobile App

Figure 7: Immigration Officer Persona



Chad Pecheur

AGE	25
EDUCATION	Master's in Business Administration
GENDER	Male
OCCUPATION	Administrator
LOCATION	Laval

> I like to be able to spend more time at home to play video games, this app will help me save time

Bio

He currently lives in Laval. He is single and he just finished his MBA. In his spare time he likes to play video games.

Core needs

- Administrator rights to the system (not allowed to view the medical data)
- Managing the user accounts of the system

Tasks

- Must assign doctors to patients
- Must manage the user accounts of the system
- Must know how many patients are assigned to each doctor so that no doctor is overloaded while some others do not have as many patients

Platform

Website Mobile App

Figure 8: Administrator Persona

7.2 Supported Devices

CovidTracker currently supports desktop and mobile platforms. More specifically, regardless of desktop device, all desktop based web browsers are supported. Likewise, regardless of mobile device, all mobile based web browsers are supported. Figures have been provided below describing the various physical and virtual interface elements present on some of the supported devices.

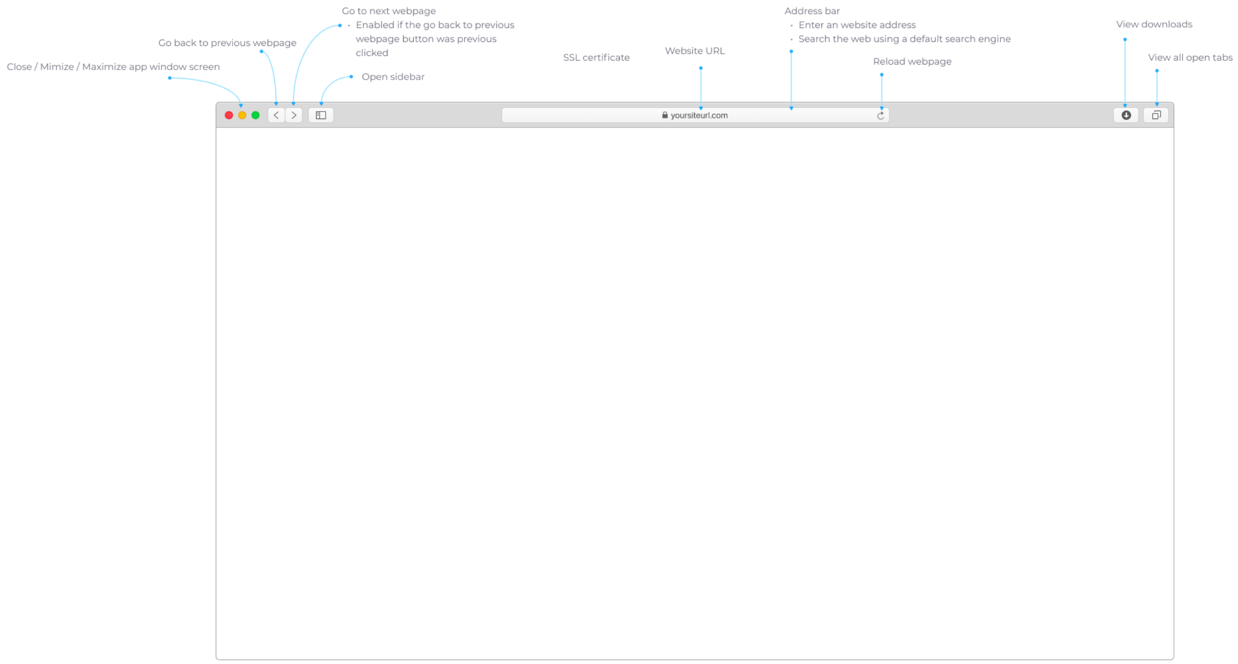

Go to next webpage
• Enabled if the go back to previous webpage button was previous clicked
Go back to previous webpage
Close / Mimize / Maximize app window screen
Open sidebar
SSL certificate
Website URL
Address bar
• Enter an website address
• Search the web using a default search engine
Reload webpage
View downloads
View all open tabs
yoursiteurl.com

Figure 9: Safari Web Browser Interface Elements

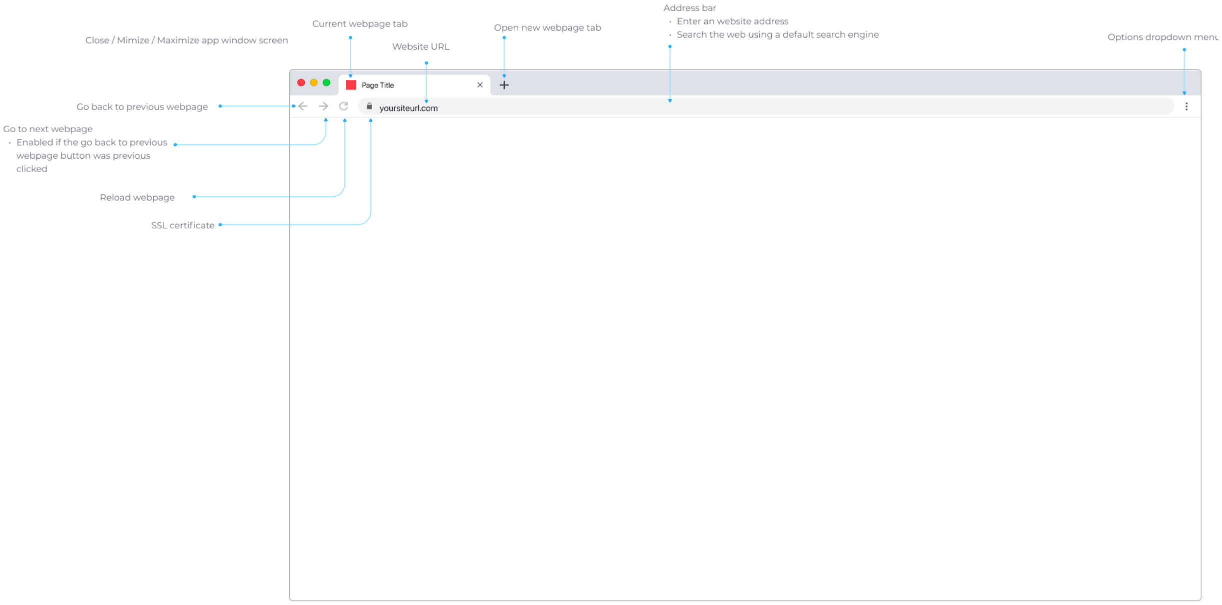

Address bar
• Enter an website address
• Search the web using a default search engine
Current webpage tab
Open new webpage tab
Close / Mimize / Maximize app window screen
Website URL
Options dropdown menu
Page Title
yoursiteurl.com
Go back to previous webpage
Go to next webpage
• Enabled if the go back to previous webpage button was previous clicked
Reload webpage
SSL certificate

Figure 10: Google Chrome Web Browser Interface Elements

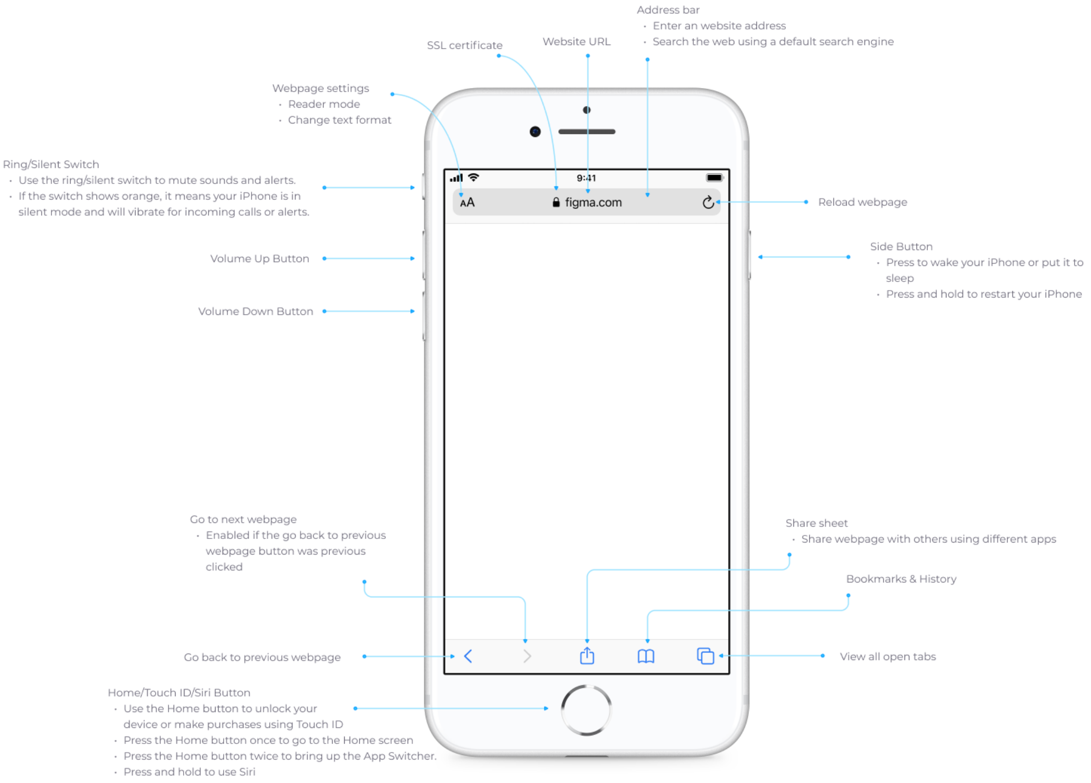

Address bar
· Enter an website address
· Search the web using a default search engine
SSL certificate
Website URL
Webpage settings
· Reader mode
· Change text format
Ring/Silent Switch
· Use the ring/silent switch to mute sounds and alerts.
· If the switch shows orange, it means your iPhone is in silent mode and will vibrate for incoming calls or alerts.
9:41
AA
figma.com
Reload webpage
Side Button
· Press to wake your iPhone or put it to sleep
· Press and hold to restart your iPhone
Volume Up Button
Volume Down Button
Go to next webpage
· Enabled if the go back to previous webpage button was previous clicked
Share sheet
· Share webpage with others using different apps
Bookmarks & History
Go back to previous webpage
View all open tabs
Home/Touch ID/Siri Button
· Use the Home button to unlock your device or make purchases using Touch ID
· Press the Home button once to go to the Home screen
· Press the Home button twice to bring up the App Switcher.
· Press and hold to use Siri

Figure 11: Apple iPhone 8 Buttons and Safari Web Browser Interface Elements

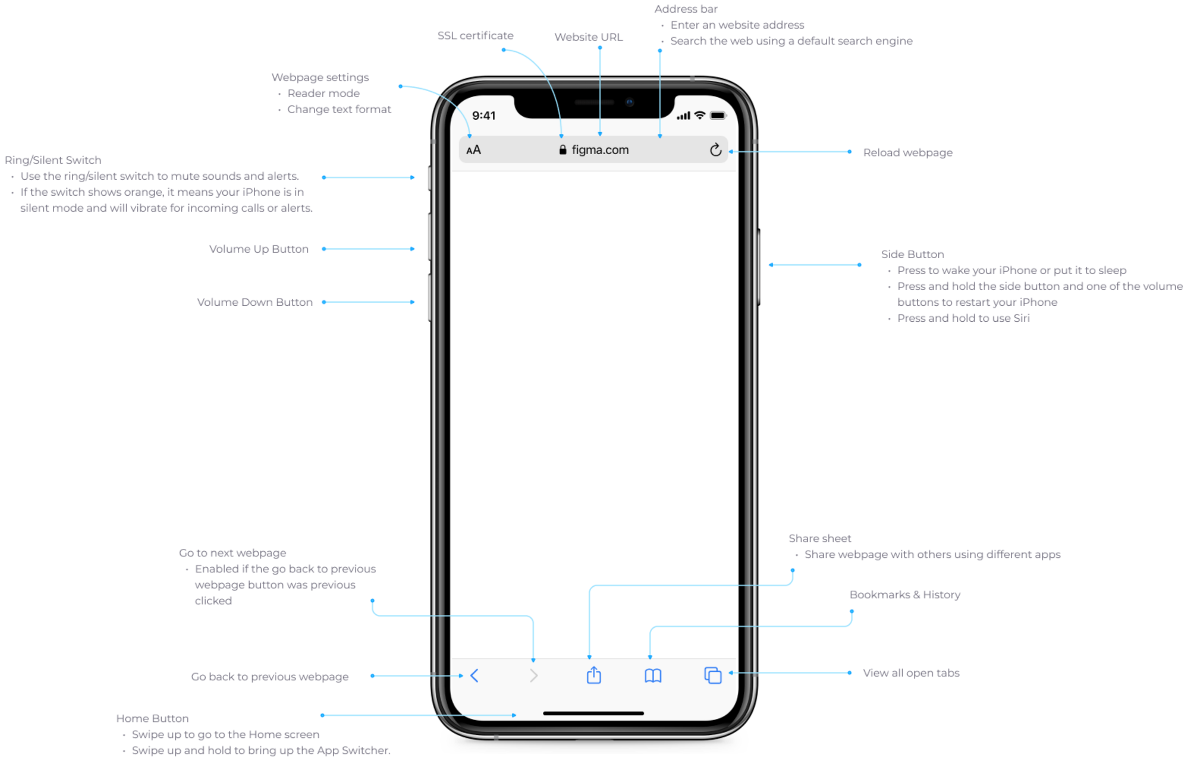

Address bar
· Enter an website address
· Search the web using a default search engine
SSL certificate
Website URL
Webpage settings
· Reader mode
· Change text format
9:41
AA
figma.com
Reload webpage
Ring/Silent Switch
· Use the ring/silent switch to mute sounds and alerts.
· If the switch shows orange, it means your iPhone is in silent mode and will vibrate for incoming calls or alerts.
Volume Up Button
Volume Down Button
Side Button
· Press to wake your iPhone or put it to sleep
· Press and hold the side button and one of the volume buttons to restart your iPhone
· Press and hold to use Siri
Share sheet
· Share webpage with others using different apps
Go to next webpage
· Enabled if the go back to previous webpage button was previous clicked
Bookmarks & History
Go back to previous webpage
View all open tabs
Home Button
· Swipe up to go to the Home screen
· Swipe up and hold to bring up the App Switcher.

Figure 12: Apple iPhone 11 Buttons and Safari Web Browser Interface Elements

While the all user interface mockups and prototypes were created with tablet support in mind (as seen in 5.3 UI Mockups and Prototypes), this platform has not been properly tested as of yet. As such, tablet devices and their respective web browsers are currently not officially supported. Such devices will be added in future sprints.

7.3 UI Mockups and Prototypes

All UI mockups and associated interactive prototypes are created in Figma. The Figma is organized with the following pages: Components, Personas, Supported Devices, Research, Drafts and UI. The Components page contains all reusable UI elements - logo, form elements, buttons, etc. - which designers might need to use when designing the various mockups. The Personas page contains all the personas information, as discussed in section 5.1 Personas. The Supported Devices page contains information about the various supported devices the application currently supports, as discussed in section 5.2 Supported Devices. The Research page is where various website links, ideas and snippets that one might have come across reside for possible future reference. The Drafts page contains UI mockups or elements that were either discarded or partially worked on. The UI page contains the finalized UI mockups and their associated interactive prototype.

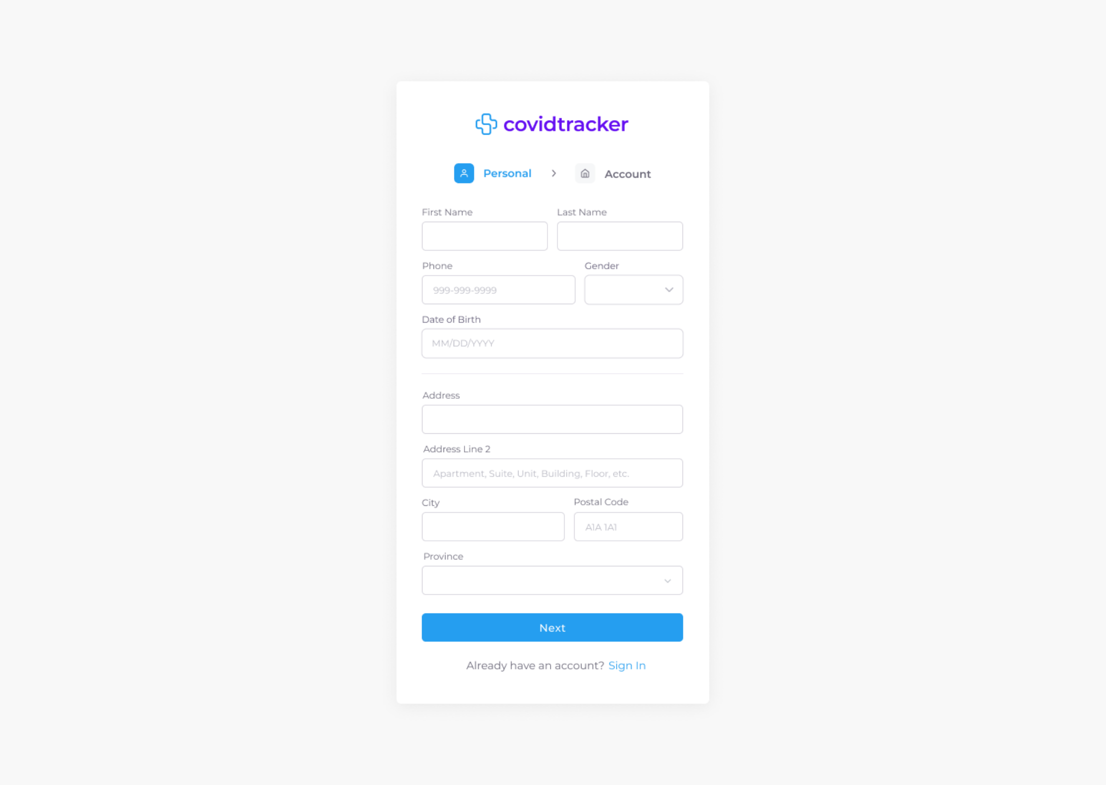
A set of user interface (UI) mockups and interactive prototypes are created for each corresponding user story. The mockups are broken down into groups based on the platform they represent - desktop, tablet and mobile - resulting in a platform specific accessible and ease of use user experience. Each set of mockups are organized in the following manner: the first row describes the user flow steps for the associated user story and each subsequent row below represents various states a particular interface in any given column can have. User flow steps proceed from left to right (start to finish) while each child mockup in a given column can depict one of the following states: active, filled or error. Subsequently, once all the mockups are completed, an interactive prototype is created.

7.3.1 Sign Up

COV-42 - As a User, I want to be able to sign up, so that I can access the apps features

An active user account is required to interact with all features in CovidTracker. As such, if the user does not have an account, they must first sign up to create an account. The sign up page is reached by a clear visible link at the bottom of the sign in page. There are two separate steps that must be completed during the sign up process: a user must fill in their personal information and secondly account information. Context awareness is provided to the user by way of a wizard at the top of the form highlighting the associated icon and text corresponding to the step the user is currently in during the sign up process. Such an element also helps users have a clear visible guideline regarding how many steps are required to be completed. The UI does not adjust based on persona. A selection of UI mockups for desktop, tablet, and mobile can be seen in Figures 13 and 14. All UI mockups, user flow and associated interactive prototypes for desktop, tablet and mobile platforms are accessible at the following links:

- UI and User Flow Mockup - Sign Up / Desktop & Tablet
- UI and User Flow Mockup - Sign Up / Mobile
- ▶ Prototype - Sign Up / Desktop & Tablet
- ▶ Prototype - Sign Up / Mobile



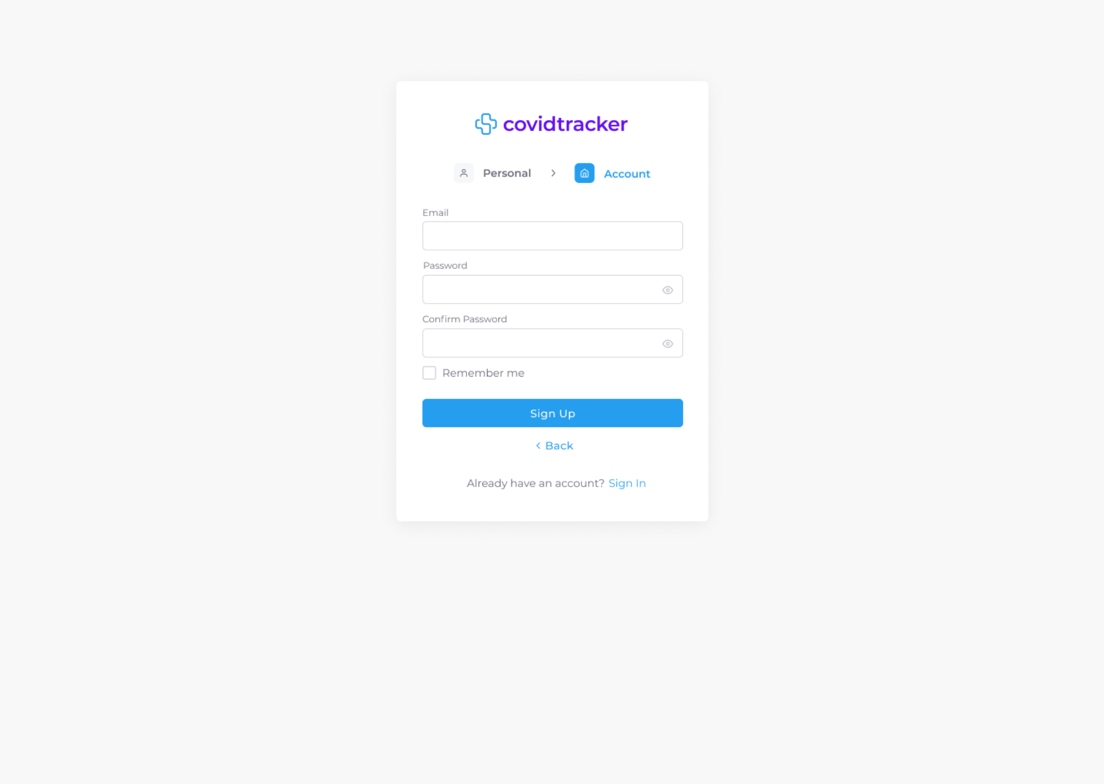


Figure 13: Sign Up Desktop & Tablet UI Mockup

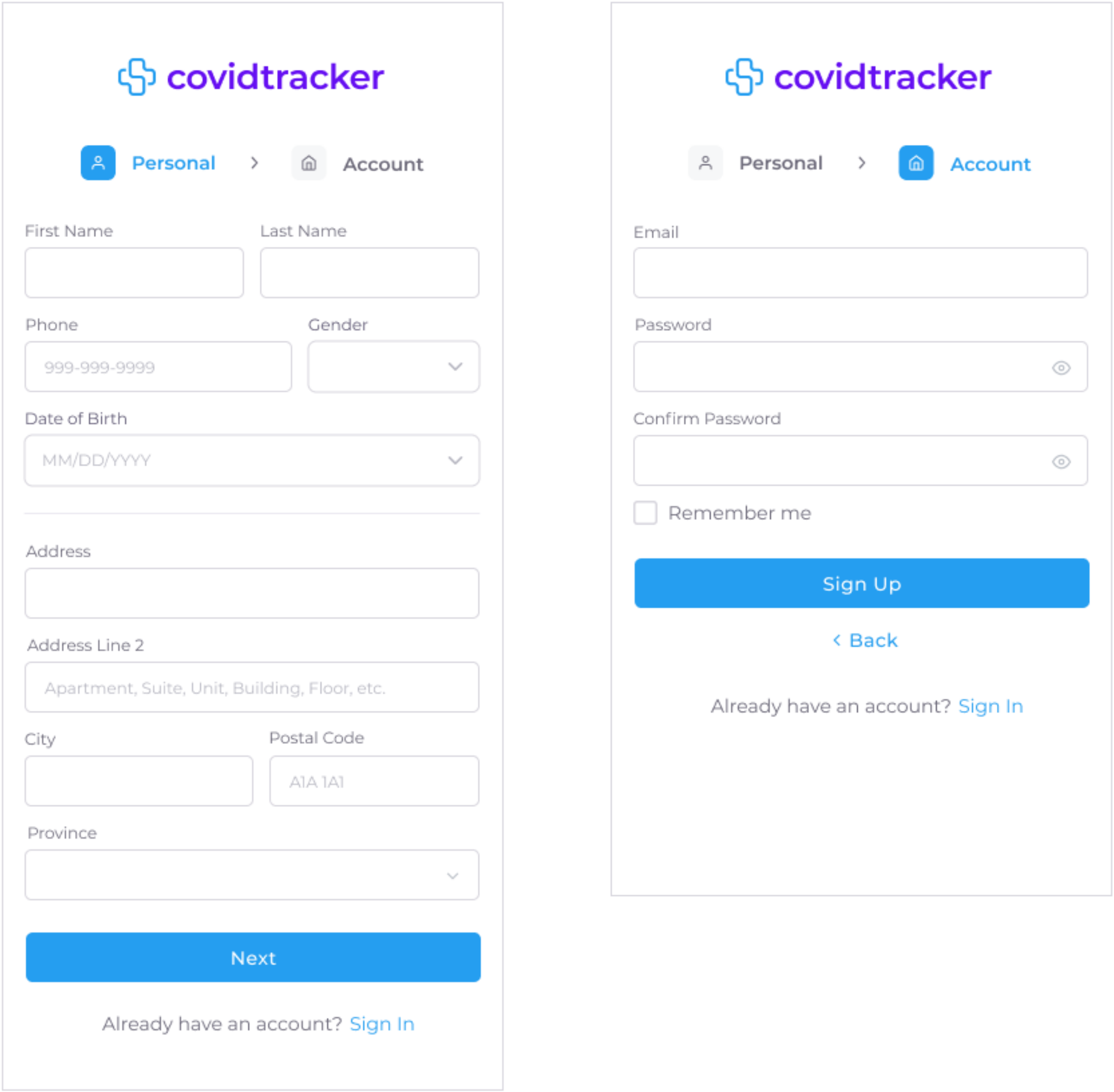


Figure 14: Sign Up Mobile UI Mockup

7.3.2 Sign In

COV-48 - As a User, I want to be able to sign in, so that I can access my account

A non logged in user is automatically redirected to the sign in page when trying to access the CovidTracker website. The UI does not adjust based on persona. A selection of UI mockups for desktop, tablet, and mobile can be seen in Figures 15 and 16. All UI mockups, user flow and associated interactive prototypes for desktop, tablet and mobile platforms are accessible at the following links:

- UI and User Flow Mockup - Sign In / Desktop & Tablet
- UI and User Flow Mockup - Sign In / Mobile
- ▸ Prototype - Sign In / Desktop & Tablet
- ▸ Prototype - Sign In / Mobile

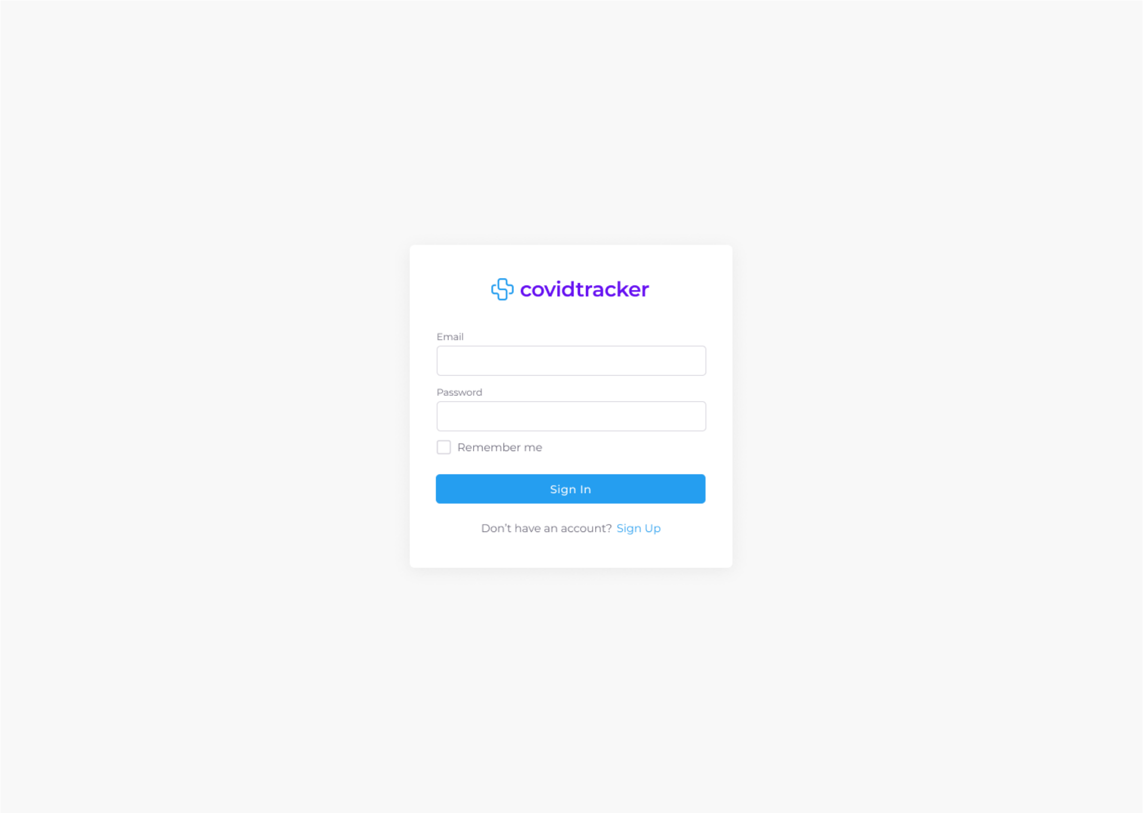


Figure 15: Sign In Desktop & Tablet UI Mockup

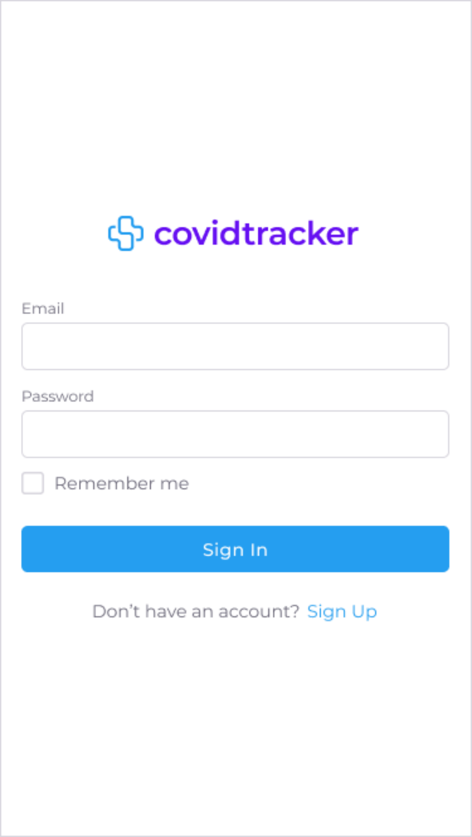


Figure 16: Sign In Mobile UI Mockup

7.3.3 Sign Out

COV-52 - As a User, I want to be able to sign out, so that I can delete my session

A user is only able to sign out of their account if they are currently signed in. The UI does not adjust based on persona. A selection of UI mockups for desktop, tablet, and mobile can be seen in Figures 17 and 18. All UI mockups, user flows and associated interactive prototypes for desktop, tablet and mobile platforms are accessible at the following links:

- UI and User Flow Mockup - Sign Out / Desktop & Tablet
- UI and User Flow Mockup - Sign Out / Mobile
- ▸ Prototype - Sign Out / Desktop & Tablet
- ▸ Prototype - Sign Out / Mobile

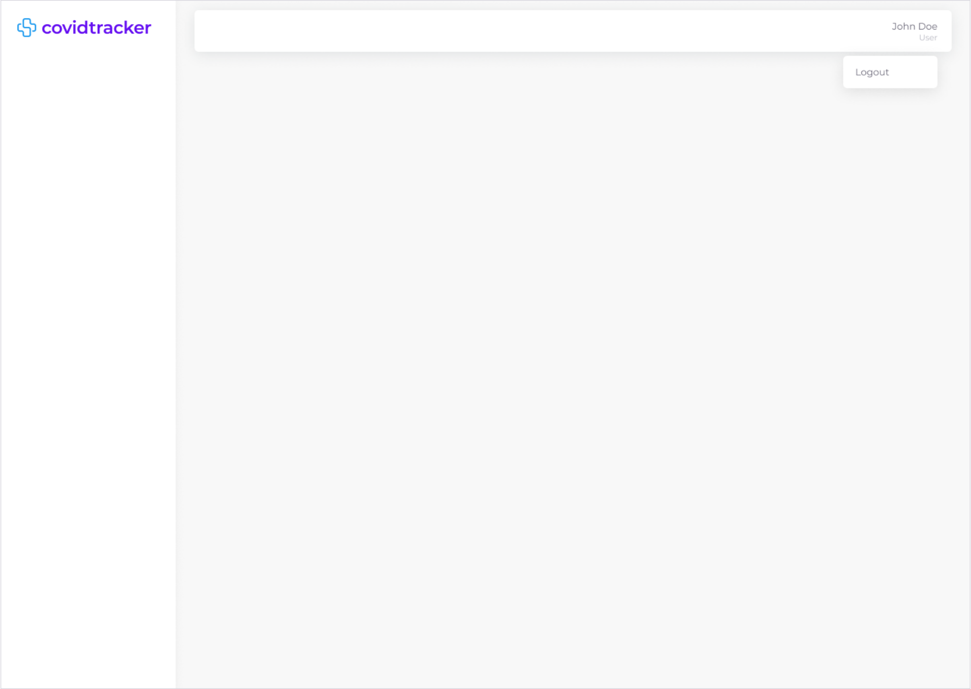


Figure 17: Sign Out Desktop & Mobile UI Mockup

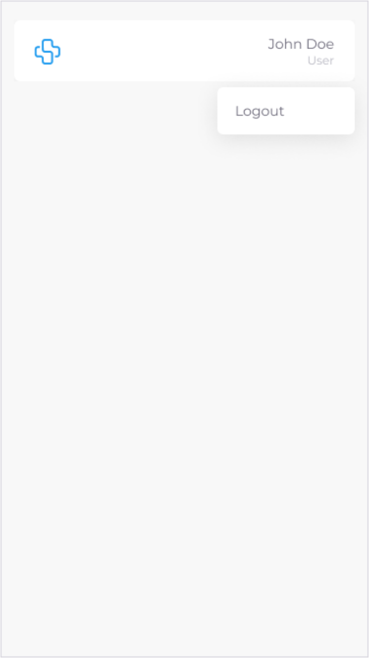


Figure 18: Sign Out Mobile UI Mockup

8.0 TESTING PLAN AND REPORT

8.1 Unit Tests

8.1.1 Client

Unit tests for the client are automated tests that are run through the CI/CD pipeline on every pull request and commit on the main branch. These tests can be run using the command `npm run test`. We will also be using snapshot testing which will render our front end javascript into HTML and save it in a file. We can then compare it later to make sure no unintended changes were made to the rendered HTML.

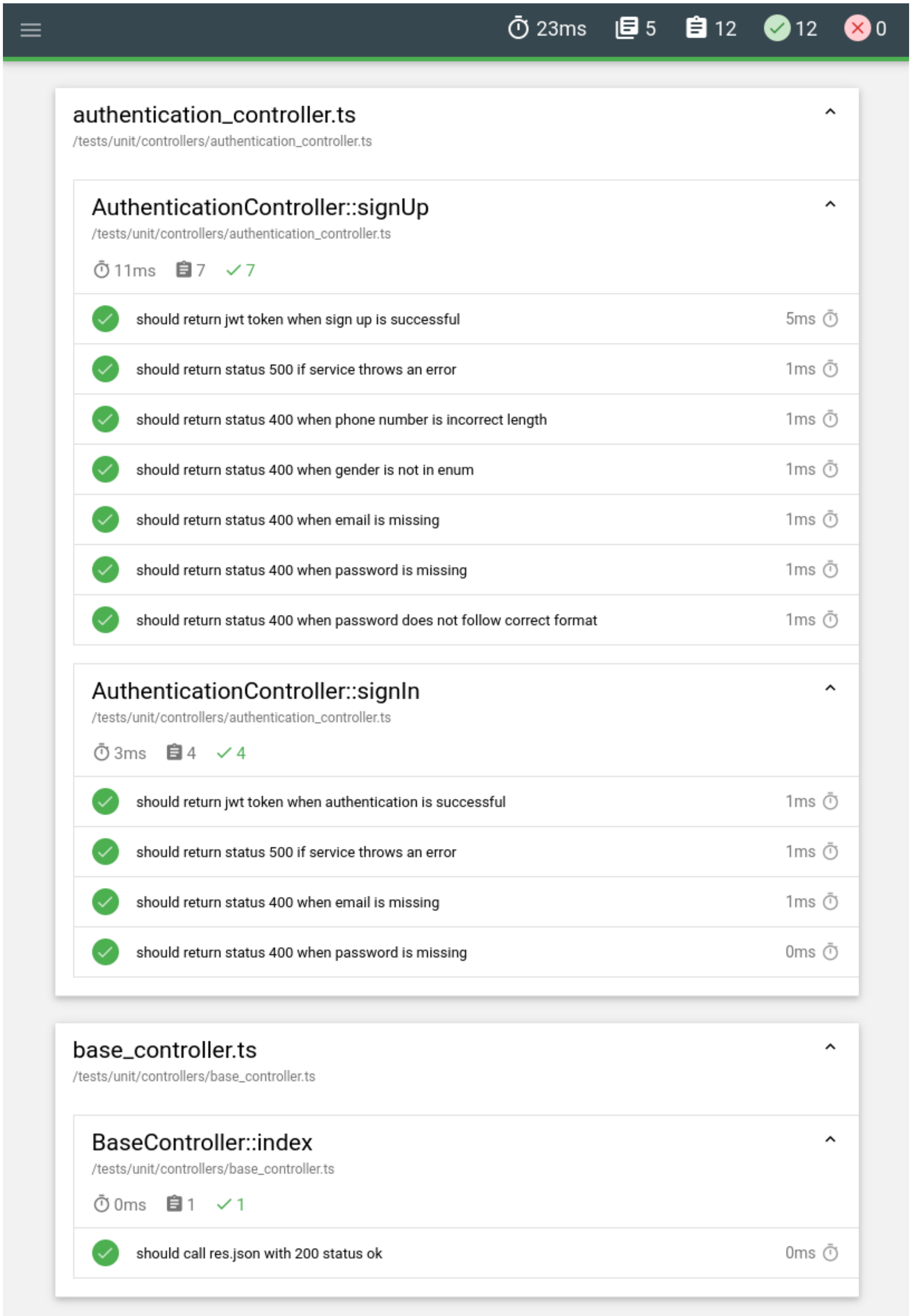
All unit tests, including snapshot tests, for the front end will be using the Jest testing framework. We chose this framework because it has the best support for snapshot testing which is the primary way we will be unit testing front end components.

8.1.2 Server

Unit tests for the server are automated tests that are run through the CI/CD pipeline on every pull request and commit on the main branch. These tests can be run using the command `npm run test:unit`.

All unit tests for the server will be using the mocha testing framework and the sinon library to generate spies, mocks, fakes, and stubs. We chose mocha because it has the best support for TypeScript testing suites and integrates well with chai - our assertion library - and sinon which allows us to creates spies, mocks, fakes, and stubs extremely easy so there is little to no boilerplate required when writing unit tests.

A generated unit test report of the system is depicted in the following figure.



authentication_controller.ts
/tests/unit/controllers/authentication_controller.ts

AuthenticationController::signUp
/tests/unit/controllers/authentication_controller.ts
11ms 7 ✓ 7

✓ should return jwt token when sign up is successful 5ms
✓ should return status 500 if service throws an error 1ms
✓ should return status 400 when phone number is incorrect length 1ms
✓ should return status 400 when gender is not in enum 1ms
✓ should return status 400 when email is missing 1ms
✓ should return status 400 when password is missing 1ms
✓ should return status 400 when password does not follow correct format 1ms

AuthenticationController::signIn
/tests/unit/controllers/authentication_controller.ts
3ms 4 ✓ 4

✓ should return jwt token when authentication is successful 1ms
✓ should return status 500 if service throws an error 1ms
✓ should return status 400 when email is missing 1ms
✓ should return status 400 when password is missing 0ms

base_controller.ts
/tests/unit/controllers/base_controller.ts

BaseController::index
/tests/unit/controllers/base_controller.ts
0ms 1 ✓ 1

✓ should call res.json with 200 status ok 0ms

Figure 19: Test Suite Report

8.2 Integration Tests

8.2.1 Client

Integration tests for the client are automated tests that are run through the CI/CD pipeline on every pull request and commit on the main branch. These tests can be run using the command `npm run test`. All integration tests for the front end will be using the Jest testing framework. We chose Jest for the same reasons mentioned in the above section.

These tests use a mock API that returns mock server responses to test the integration between all the client side code and the server API.

8.2.2 Server

Integration tests for the server are automated tests that are run through the CI/CD pipeline on every pull request and commit on the main branch. These tests can be run using the command `npm run test:integration`.

All integration tests for the server will be using the mocha testing framework and supertest in order to create a callable instance of our web server. We chose to use supertest because it provides the easiest integration with our web framework library.

These tests use a database to test the integration between all the server side code and the database implementation.

8.3 Acceptance Tests

Acceptance tests will be documented and run manually to show an entire flow of the application. These tests will use the client to interface with the server which will persist the data in the database. All these tests will be based on the user stories to ensure that all user flows work as specified by the requirements.

These tests can be automated using a tool like Selenium in order to mock a real user interacting with the full system. The acceptance tests are also written in Gherkin Syntax which is a behavioral driven development syntax that allows us to define our tests in terms of user state and behavior.

Below are the current acceptance tests for the system. These can also be found attached to each user story description.

AT-1	**COV-42 - As a user I was to be able to sign up so that I can access the apps features**
Acceptance Criteria	GIVEN that I am on the sign up page AND that I input all required fields AND that I clicked the Sign Up button THEN my account should be created AND I should be logged in AND I should be redirected to the main screen.
Result	**PASS**

Table 7: Acceptance Test for COV-42

AT-2	**COV-48 - As a user I want to be able to sign in so that I can access my account**
Acceptance Criteria	GIVEN that I am on the sign in page AND that I input my valid email AND that I input my valid password AND that I clicked the Sign In button THEN I should be logged into the site AND my session should persist AND I should be redirected to the main screen.
Result	**PASS**

Table 8: Acceptance Test for COV-48

AT-3	**COV-52 -As a user I want to be able to sign out so that I can delete my session**
Acceptance Criteria	GIVEN that I am on a page with a navbar AND that I am signed in AND that I clicked the Sign Out button THEN I should be logged out of the site AND my session should be deleted AND I should be redirected to the sign in page.

Result	PASS

Table 9: Acceptance Test for COV-52

8.4 System Tests

System tests will be documented and run manually to show an entire flow of the application. These tests will use the client to interface with the server which will persist the data in the database. All these tests will be based on the user stories to ensure that all user flows work as specified by the requirements

These tests can be automated using a tool like Selenium in order to mock a real user interacting with the full system.

Below are the current system tests for the system. These can also be found attached to each user story description.

ST-1	**COV-42 - As a user I was to be able to sign up so that I can access the apps features**
Steps to reproduce	**Expected output for each step**
1. Navigate to the sign up page (relative url "/sign_up") 2. Fill all required fields with valid inputs 3. Click the Sign Up Button	1. You should see the sign up page 2. The form should not give any input errors 3. The form should not give any input errors, your account should be created, you should be signed in, and you should be redirected to the main screen

Table 10: System Test for COV-42

ST-2	**COV-48 - As a user I want to be able to sign in so that I can access my account**
Steps to reproduce	**Expected output for each step**

1. Navigate to the sign in page (relative url "/sign_in") 2. Input your email and password 3. Click the Sign In Button	1. You should see the sign in page 2. The form should not give any input errors 3. The form should not give any input errors, you should be signed in, and you should be redirected to the main screen

Table 11: System Test for COV-48

ST-3	**COV-52 -As a user I want to be able to sign out so that I can delete my session**
Steps to reproduce	**Expected output for each step**
1. Navigate to a page where the navbar can be seen 2. Click the Sign Out Button	1. You should see the sign out button in the navbar 2. You should be signed out, your session should be deleted, and you should be redirected to the Sign In page

Table 12: System Test for COV-52

8.5 Test Code Coverage

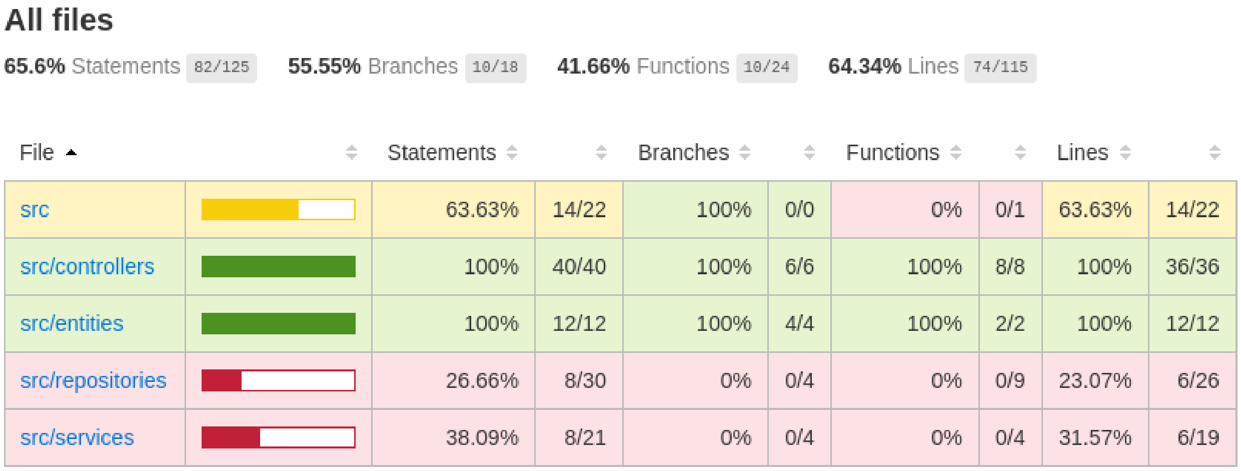
8.5.1 Client

A report of the code coverage can be generated by running the command `npm run test --coverage` this will produce a coverage report of the client side code.

8.5.2 Server

A report of the code coverage can be generated by running the command `npm run coverage` this will produce a coverage report of the server side code.

A code coverage report of the server side code is depicted in the following figure. NYC/Istanbul was used to compute the code coverage. It reports coverage by folder, you can then click into the folder and view the other folders coverage or individual file coverage. Then you can open a specific file to view line by line coverage reports.



All files

65.6% Statements 82/125 **55.55%** Branches 10/18 **41.66%** Functions 10/24 **64.34%** Lines 74/115

File ▲		Statements		Branches		Functions		Lines	
src	🟨	63.63%	14/22	100%	0/0	0%	0/1	63.63%	14/22
src/controllers	🟩	100%	40/40	100%	6/6	100%	8/8	100%	36/36
src/entities	🟩	100%	12/12	100%	4/4	100%	2/2	100%	12/12
src/repositories	🟥	26.66%	8/30	0%	0/4	0%	0/9	23.07%	6/26
src/services	🟥	38.09%	8/21	0%	0/4	0%	0/4	31.57%	6/19

Figure 20: Code Coverage Report of Server Side Code

9.0 DEFECT TRACKING AND REPORT

Sprint 1 had very few bugs, as the majority of which were taken care of either before a pull request merge or after a follow up pull request related to the associated user story. There were however, a few larger and unforeseen bugs that we went ahead and created specific bug tickets in Jira. All bugs found in sprint 1 were all related to minor UI adjustments and classified as low priority which will be handled in sprint 2. There were no bugs associated with the backend implementation.

The reason for pushing these bugs is to ensure the implemented UI and UI mockups are a 1 to 1 match. Over time these differences can start to add up and become unmanageable so it's best to take care of them quickly as they appear. See next page to view the generated report.

T	Key	Summary	Story point estimate	P	Risk	Parent	Description	Sprint	Status
🟥	COV-144	The navbar shows a placeholder name instead of the logged in users name	1	⌄	Low	⚡ Authentication and Authorization	Currently when a user logs in the navbar will show a placeholder first and last name. This should show the logged in users first and last name. After the sign in and sign up requests start returning the users name in the payload and saving it in the redux store we can update the navbar to fetch this data from the store and display it.	COV Sprint 2	TO DO
🟥	COV-145	The drop down inputs in the sign in form are pre-selected when they should not have a default value	1	︾	Low	⚡ Authentication and Authorization	Currently the sign in form has some drop down input for selecting the gender and province. These drop downs have pre-selected values so that no validation needs to take place. This should be fixed so that there is no value as default and be required to have a user input the selection.	COV Sprint 2	TO DO

10.0 QUALITY MEASUREMENTS

This section depicts and describes the various metrics being used, the cause of the results and how the results can be improved.

10.1 Metrics Used

The following metrics are being used:

- **Statement coverage**: Checks to see if each statement in the program has been executed while running the test suite.
- **Branch coverage:** Checks to see if all conditional branches (if statement and terinaries) are covered while running the test suite.
- **Function coverage:** Checks to see if every function in the source code was called at least once while running the test suite.
- **Line coverage:** Checks if each physical line in the source code has been executed at least once while running the test suite. This is mostly covered by statement coverage, which is generally superior because it ignores coding styles better, but we are including it for completeness of all generated metrics.
- **Linting errors:** Errors we receive if we run our automated linting package, ESLint. The types of errors and severity are defined by our .eslintrc file. This includes checks against many things, primarily language standards.
- **Formatting errors:** Errors we receive if we run our automated formatting package, prettier. This package has defined an opinionated formatting standard that can automatically be applied to most code, but some must still be manually formatted to follow the standard.

10.2 Cause of Results

There aren't any formatting and linting errors due to the project being started with the packages enabled and having added them into our CI/CD pipeline for the server side code. As a result, this ensures code cannot be merged in if any errors are present.

Test coverage is lower than expected in sprint 1 as there was not enough time to implement all necessary unit tests for each module. Therefore, only the main interactions were covered. There will be an increased focus on test coverage in the following sprints.

10.3 Improving the Results

The test coverage can definitely be improved by simply requiring all pull requests to have full test coverage. Through this process we will incrementally increase test coverage to 90%-100% overtime without having to dedicate a larger block of time.

	Sprint 1	**Sprint 2**	**Sprint 3**	**Sprint 4**	**Sprint 5**
Statement Coverage	65.6%				
Branch Coverage	55.55%				
Function Coverage	41.66%				
Line Coverage	64.34%				
Linting Errors	0				
Formatting Errors	0				

Table 13 : Test Coverage for Each Sprint

APPENDIX A: TEAM COLLABORATION AND COMMUNICATION

Stakeholders use a set of tools to collaborate and communicate throughout the project lifecycle.

A.1 Collaboration

- **Google Suite (Docs, Drive, Sheets)**:
  G Suite is a collection of business, productivity, collaboration, and education software developed and powered by Google. The primary G Suite tools include Gmail, Drive, Docs, Sheets, Slides, Forms, Calendar, Google+, Sites, Hangouts, and Keep. [2] Google Suite is used for documentation since it is widely accessible and available to all development team members.
- **GitHub:** GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. We use github to be able to work on different sections of the code at the same time and have a version control. [3]

A.2. Communication

- **Discord:** Discord is a free voice, video, and text chat app that's used by tens of millions of people ages 13+ to talk and hang out with their communities and friends. [6] Discord is used for communication and meetings among development team members. Voice and text channels are named according to the different development team groups (i.e. back end, front end and UI design).
- **Slack:** Slack is a messaging app for business that connects people to the information they need. By bringing people together to work as one unified team, Slack transforms the way organizations communicate. [5] Slack is used to communicate with the product owners when clarification is needed or to schedule meetings.

- **Zoom:** Zoom is a cloud-based video conferencing platform that can be used for video conferencing meetings, audio conferencing, webinars, meeting recordings, and live chat. [1] Zoom is used for meetings with the product owners.

A.3 Tools

- **Issue and project tracking tool:** Jira
  https://www.atlassian.com/software/jira.
  Jira is a software application used for issue tracking and project management. The tool has become widely used by agile development teams to track bugs, stories, epics, and other tasks. [4]
- **Diagram modeling tool:** Draw.io
  https://app.diagrams.net/
  Draw.io is an online diagram editor that enables you to create flowcharts, UML, entity relation, network diagrams, mockups and more.
- **User interface design and prototyping tool:** Figma
  https://www.figma.com/
  Figma is a UI and UX design application, with excellent design, prototyping, and code-generation tools. It's arguably the industry's leading interface design tool, with robust features which support teams working on every phase of the design process.

APPENDIX B: GLOSSARY

- **Application Programming Interface (API):** An application programming interface (API) is a computing interface which defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. [7]
- **Logical Layered Architecture:** Layered architecture is an architecture pattern that promotes high cohesion and low coupling through separation of concerns by layers. Each layer depends on the layer below it.
- **UML Domain Model:** A conceptual view of the domain represented through UML classes and relationships. [8]
- **Risk Management:** Practice of identifying, evaluating, and preventing or mitigating risks to a project that have the potential to impact the desired outcomes.
- **Database:** Databases store aggregations of data records or files that contain information, such as sales transactions, customer data, financials and product information. [9]
- **UI prototype:** User interface prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. [10]
- **UI/UX mockup :** A mockup is a static wireframe that includes more stylistic and visual UI details to present a realistic model of what the final page or application will look like. [11]
- **CI/CD pipeline:** Series of steps that must be performed in order to deliver a new version of software. Continuous integration/continuous delivery (CI/CD) pipelines are a practice focused on improving software delivery using either a DevOps or site reliability engineering (SRE) approach. [12]

REFERENCES

1. Barron, Sophia. “Everything You Need to Know about Using Zoom.” *Owl Labs Blog*, https://resources.owllabs.com/blog/zoom#:~:text=Zoom%20is%20a%20cloud%2Dbased,meeting%20recordings%2C%20and%20live%20chat.
2. Decker, Allie. “The Ultimate Guide to G Suite.” *HubSpot Blog*, 31 Mar. 2019, https://blog.hubspot.com/marketing/google-suite#:~:text=G%20Suite%20is%20a%20collection,Google%20Apps%20for%20Your%20Domain%E2%80%9D.
3. “Hello World.” *GitHub Docs*, https://docs.github.com/en/get-started/quickstart/hello-world.
4. “Jira.” *ProductPlan*, 9 Feb. 2021, https://www.productplan.com/glossary/jira/.
5. Slack. “What Is Slack?” *Slack Help Center*, https://slack.com/help/articles/115004071768-What-is-Slack-#:~:text=Slack%20is%20a%20messaging%20app,transforms%20the%20way%20organizations%20communicate.
6. “What Is Discord: A Guide for Parents and Educators.” *Discord*, https://discord.com/safety/360044149331-What-is-Discord#:~:text=Discord%20is%20a%20free%20voice,with%20their%20communities%20and%20friends.&text=The%20vast%20majority%20of%20servers,touch%20and%20spend%20time%20together.
7. Business Standard. “What Is API, API Definition, API News.” *Business Standard*, https://www.business-standard.com/about/what-is-api.
8. *UML Class Diagrams as a Conceptual Models*, http://www.cs.sjsu.edu/~pearce/modules/lectures/ooa/domain/domainModels.htm
9. Lutkevich, Ben, and Adam Hughes. “What Is a Database? Definition from Searchdatamanagement.” *SearchDataManagement*, TechTarget, 27 Sept. 2021,

https://searchdatamanagement.techtarget.com/definition/database#:~:text=Computer%20databases%20typically%20store%20aggregations,data%2C%20financials%20and%20product%20information.&text=They%20collect%20information%20on%20people,can%20be%20observed%20and%20analyzed.

10. *User Interface (UI) Prototypes: An Agile Introduction*, http://agilemodeling.com/artifacts/uiPrototype.htm.
11. “Wireframes vs Mockups: Determining the Right Level of Fidelity for Your Project.” *Wireframes vs Mockups Explained* | *Lucidchart Blog*, 27 Feb. 2020, https://www.lucidchart.com/blog/wireframes-vs-mockups.
12. “What Is a CI/CD Pipeline?” *Red Hat - We Make Open Source Technologies for the Enterprise*, https://www.redhat.com/en/topics/devops/what-cicd-pipeline.