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**](#_dk61dbxiw227) **4**

[**LIST OF FIGURES**](#_izuxrz68y5th) **5**

[**1.0 INTRODUCTION**](#_8gi4rorvg9rx) **7**

[1.1 Positioning](#_9nxirm0q0n9) 7

[1.1.1 Problem Statement](#_ey0y2hijvyx) 7

[1.1.2 Product Position Statement](#_vmzcns5bmt5) 8

[**2.0 PROJECT DESCRIPTION**](#_kaw8csxsu8zy) **9**

[2.1 Stakeholders](#_85dg39nqmnnp) 10

[2.1.1 Stakeholder Roles](#_45y2pgdy2yhg) 11

[**3.0 REQUIREMENTS**](#_koujzfmlost) **14**

[3.1 User Stories](#_gxpzqp76mguh) 14

[3.2 Backlog](#_zegdqdfmxu5f) 15

[**4.0 RELEASE PLANNING**](#_tpcklaugfjpr) **16**

[4.1. Sprint 1](#_q6o1p04164ej) 16

[4.1.1 Summary](#_vtdymtv9icoi) 16

[4.1.2 Retrospective](#_jk0w047ufca) 17

[4.2 Sprint 2](#_h7s3ui5qsdas) 18

[4.2.1 Planning](#_jpw8scq9ochi) 18

[**5.0 SOFTWARE ARCHITECTURE**](#_om9c86n2hjf2) **19**

[5.1 Stakeholder Concerns](#_mvlp55254g8v) 19

[5.2 Diagrams](#_o85op5stsdk3) 21

[5.2.1 Domain Model](#_v6ls467hm8nn) 21

[5.2.2 Component Diagram](#_im2xqjubdvs2) 22

[5.3 Tech Stack](#_3hlvao5rr475) 24

[5.3.1 Presentation Tier](#_k4ttptvvtrh3) 24

[5.3.2 Application Tier](#_yolcmn71l9oo) 25

[5.3.3 Database Tier](#_shfowowe04kk) 25

[5.4 External Libraries](#_k33ek925g2nd) 26

[5.4.1 Vuexy](#_rzmrhtm5crnk) 26

[**6.0 RISK ASSESSMENT AND MANAGEMENT PLAN**](#_7aqlnlz3p1cs) **28**

[**7.0 USER INTERFACE DESIGN**](#_nz2pw6ma7anf) **30**

[7.1 Personas](#_efd0y56fxuan) 30

[7.2 Supported Devices](#_i1vgfmyc2inh) 33

[7.3 UI Mockups and Prototypes](#_5bu6of2f05g6) 36

[7.3.1 Sign Up](#_tb7unk8zspcz) 37

[7.3.2 Sign In](#_84bxw14t4kud) 41

[7.3.3 Sign Out](#_fww6kepvil37) 42

[**8.0 TESTING PLAN AND REPORT**](#_j2jbja179p9) **44**

[8.1 Unit Tests](#_cpo584kbttl5) 44

[8.1.1 Client](#_2hodbtcc7aqw) 44

[8.1.2 Server](#_cq0exugs9fpb) 44

[8.2 Integration Tests](#_wv7zpjd3dq5g) 46

[8.2.1 Client](#_2offhb6g1t6b) 46

[8.2.2 Server](#_9vxppvx9asqd) 46

[8.3 Acceptance Tests](#_86yobbd3m4z7) 46

[8.4 System Tests](#_ok7u9dnkq3vk) 48

[8.5 Test Code Coverage](#_liql2nd33m9k) 49

[8.5.1 Client](#_o8br8qvuosgi) 49

[8.5.2 Server](#_y7aln7vsvspp) 49

[**9.0 DEFECT TRACKING AND REPORT**](#_l6f3h3a2ngvg) **51**

[**10.0 QUALITY MEASUREMENTS**](#_1qt0z8csqj8h) **52**

[10.1 Metrics Used](#_unw2pjaw1u0f) 52

[10.2 Cause of Results](#_7y78t7f9tfty) 52

[10.3 Improving the Results](#_oq9i7ygxop31) 53

[**APPENDIX A: TEAM COLLABORATION AND COMMUNICATION**](#_5h4nphm6kom) **54**

[A.1 Collaboration](#_iqwfeiqvltzt) 54

[A.2. Communication](#_hezdof2ynray) 54

[A.3 Tools](#_vfmp7j1cz67g) 55

[**APPENDIX B: GLOSSARY**](#_qgn0tjnvhr7l) **56**

[**REFERENCES**](#_gah3de2j7xxr) **57**

# LIST OF TABLES

| Table 1 Problem Statement | 6 |
| --- | --- |
| Table 2 Product Position Statement | 7 |
| Table 3 Sprints Schedule | 8 |
| Table 4 Supplementary Information | 18 |
| Table 5 Stakeholder Concern Traceability Matrix | 18 |
| Table 6 Risks Analysis Table | 27 |
| Table 7 Acceptance Test for COV-42 | 46 |
| Table 8 Acceptance Test for COV-48 | 46 |
| Table 9 Acceptance Test for COV-52 | 46 |
| Table 10 System Test for COV-42 | 47 |
| Table 11 System Test for COV-48 | 47 |
| Table 12 System Test for COV-52 | 48 |
| Table 13 Test Coverage for Each Sprint | 52 |

# 

# LIST OF FIGURES

| Figure 1 Burndown Chart | 15 |
| --- | --- |
| Figure 2 UML Domain Model Class Diagram of CovidTracker | 20 |
| Figure 3 Component Diagram of CovidTracker | 22 |
| Figure 4 Patient Persona | 29 |
| Figure 5 Doctor Persona | 30 |
| Figure 6 Health Official Persona | 30 |
| Figure 7 Immigration Officer Persona | 31 |
| Figure 8 Administrator Persona | 31 |
| Figure 9 Safari Web Browser Interface Elements | 32 |
| Figure 10 Google Chrome Web Browser Interface Elements | 33 |
| Figure 11 Apple iPhone 8 Buttons and Safari Web Browser Interface Elements | 34 |
| Figure 12 Apple iPhone 11 Buttons and Safari Web Browser Interface Elements | 35 |
| Figure 13 Sign Up Desktop & Tablet UI Mockup | 38 |
| Figure 14 Sign Up Mobile UI Mockup | 39 |
| Figure 15 Sign In Desktop & Tablet UI Mockup | 40 |
| Figure 16 Sign In Mobile UI Mockup | 41 |
| Figure 17 Sign Out Desktop & Mobile UI Mockup | 42 |
| Figure 18 Sign Out Mobile UI Mockup | 42 |
| Figure 19 Test Suite Report | 44 |
| Figure 20 Code Coverage Report of Server Side Code | 49 |

# 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.

## 3.2 Backlog

See next page to view Backlog.

# 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 |

### 4.1.2 Retrospective

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

## 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.

# 

# 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](https://drive.google.com/file/d/1dmbNVKywLPckfShuGJHM9J-3GodmuJS2/view?usp=sharing)

|  |
| --- |
| 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](https://drive.google.com/file/d/12iKdkB5sOOqflkbyF5n1bY1xFr58-gT1/view?usp=sharing)

|  |
| --- |
| 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 using GPS tracking |  |  | Medium | Severe |
| **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.

|  |
| --- |
| Figure 4: Patient Persona |

|  |
| --- |
| Figure 5: Doctor Persona |

|  |
| --- |
| Figure 6: Health Official Persona |

|  |
| --- |
| Figure 7: Immigration Officer Persona |

|  |
| --- |
| 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.

|  |
| --- |
| Figure 9: Safari Web Browser Interface Elements |

|  |
| --- |
| Figure 10: Google Chrome Web Browser Interface Elements |

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

|  |
| --- |
| 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](https://drive.google.com/file/d/1A7hJo-o5oS6mhmVmXt3k4YArJYgHFemo/view?usp=sharing)
* [UI and User Flow Mockup - Sign Up / Mobile](https://drive.google.com/file/d/1xjxnhr8e_YFsnyKksgtgLBl6HEjVPNoZ/view?usp=sharing)
* [▶ Prototype - Sign Up / Desktop & Tablet](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=291%3A4813&show-proto-sidebar=1&starting-point-node-id=291%3A4813)
* [▶ Prototype - Sign Up / Mobile](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=291%3A4700&show-proto-sidebar=1&starting-point-node-id=291%3A4700)

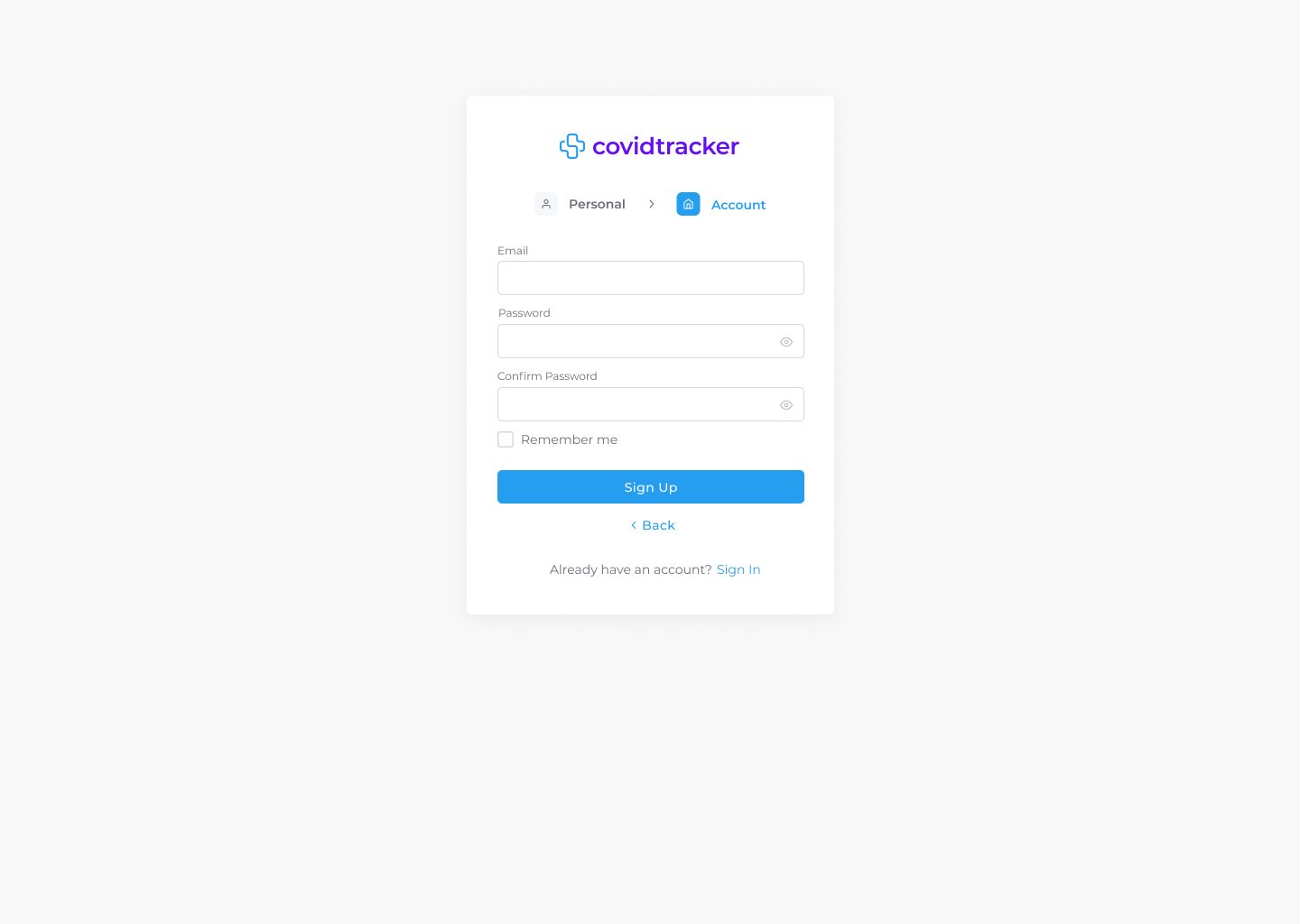
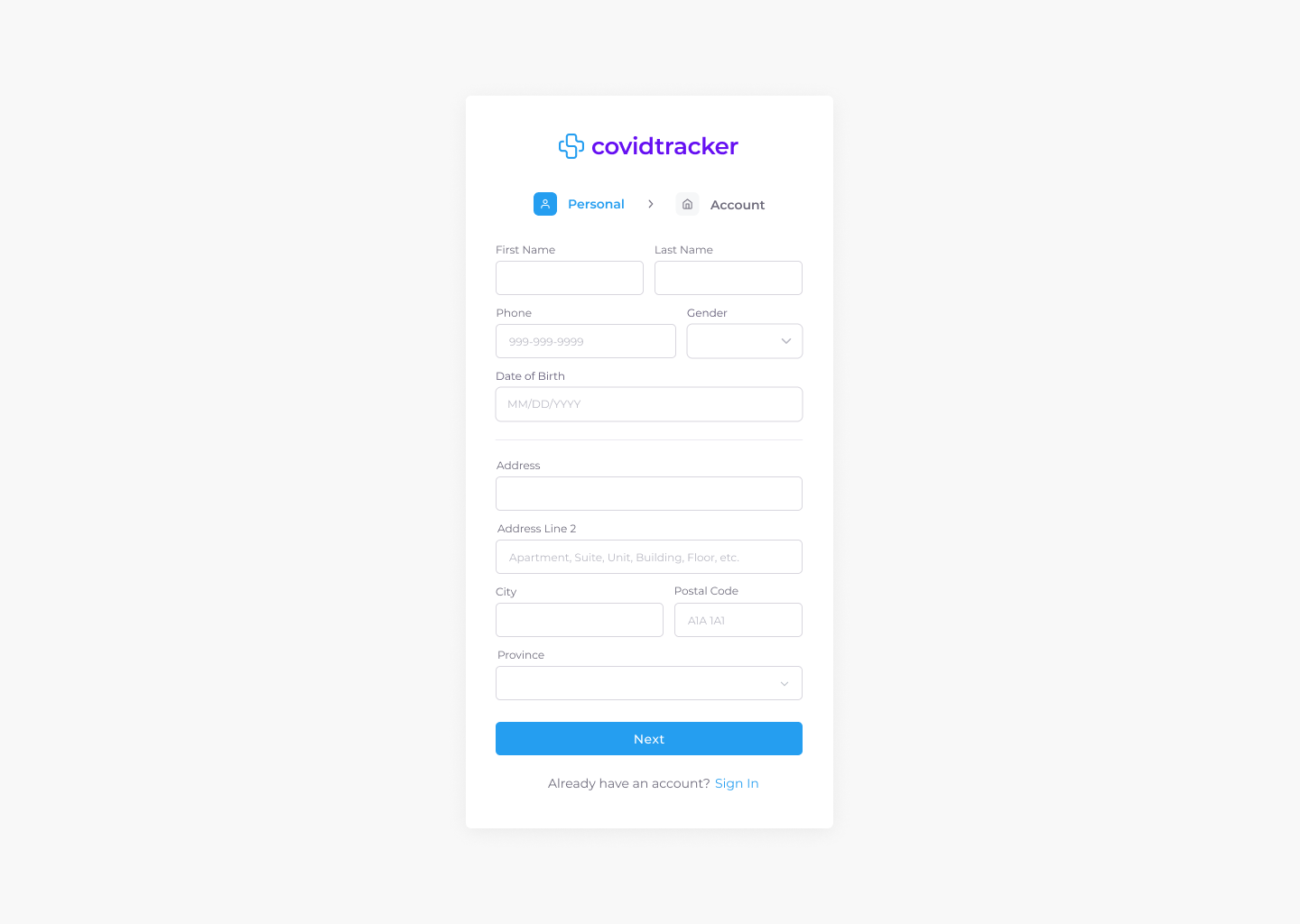


Figure 13: Sign Up Desktop & Tablet UI Mockup

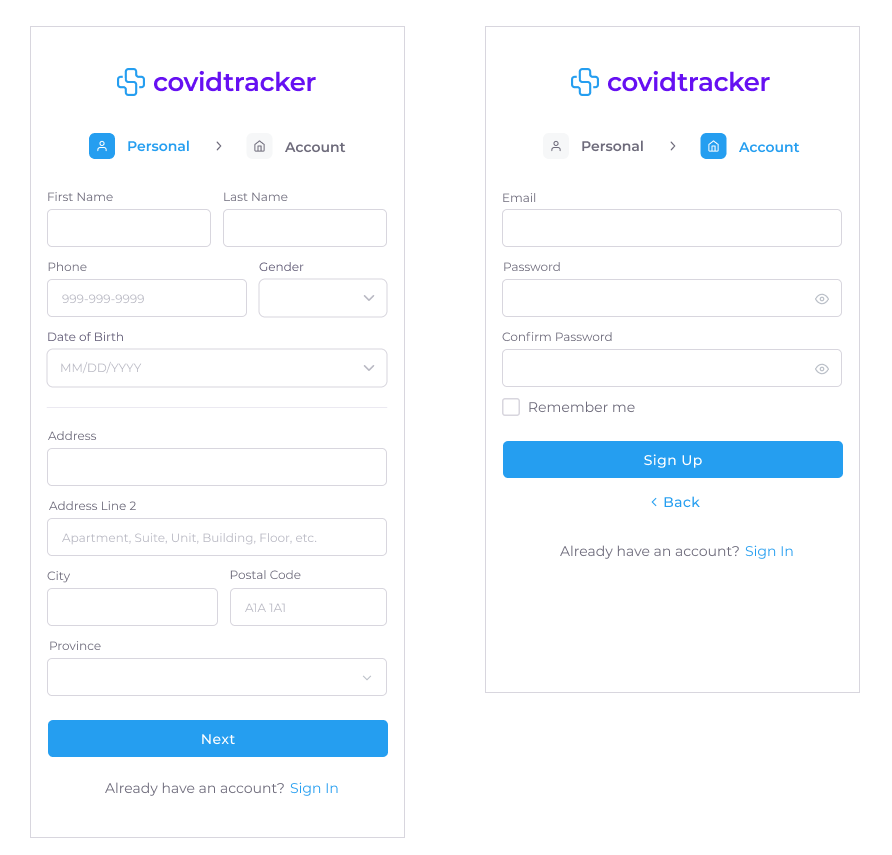
****

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](https://drive.google.com/file/d/1ygeYPusKKwFrbDOo8KGS_eEk7PQi0tLp/view?usp=sharing)
* [UI and User Flow Mockup - Sign In / Mobile](https://drive.google.com/file/d/1mE2_2KfZcaT-ROMQlXTF2Zmf-LsV_rQG/view?usp=sharing)
* [▶ Prototype - Sign In / Desktop & Tablet](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=27%3A75&show-proto-sidebar=1&starting-point-node-id=27%3A75)
* [▶ Prototype - Sign In / Mobile](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=45%3A5546&show-proto-sidebar=1&starting-point-node-id=45%3A5546)

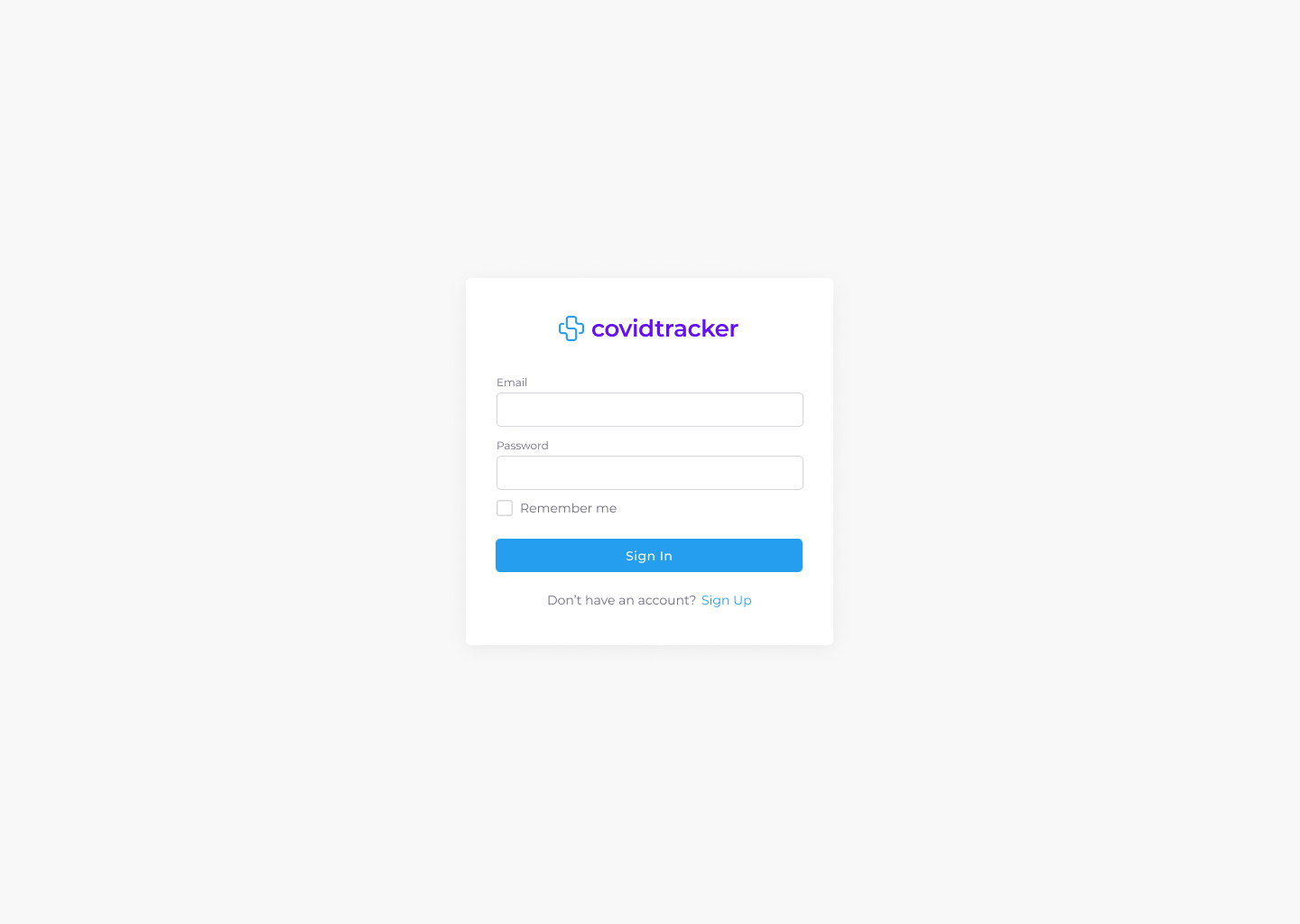


Figure 15: Sign In Desktop & Tablet UI Mockup

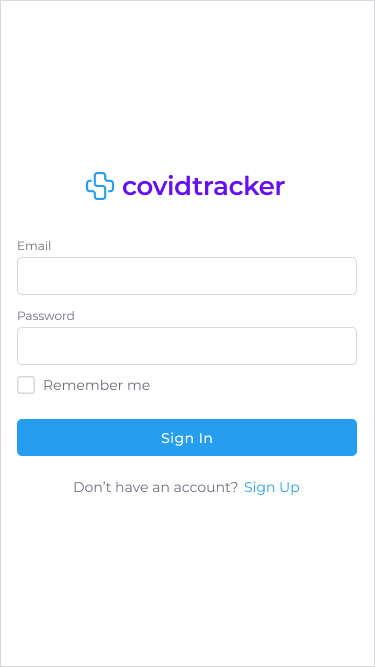


Figure 16: Sign In Mobile UI Mockup

### 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](https://drive.google.com/file/d/1aBEessNnjKTMYejaxLRgl3ICZSnfnvnA/view?usp=sharing)
* [UI and User Flow Mockup - Sign Out / Mobile](https://drive.google.com/file/d/1Xu3q0fpfmBsl3HSH66UVZhbvRqKUUnMx/view?usp=sharing)
* [▶ Prototype - Sign Out / Desktop & Tablet](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=183%3A1196&show-proto-sidebar=1&starting-point-node-id=183%3A1196)
* [▶ Prototype - Sign Out / Mobile](https://www.figma.com/proto/It4nC0bENajBMuQdTviwQG/covidtrack?node-id=206%3A4022&show-proto-sidebar=1&starting-point-node-id=206%3A4022)

# 

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.

## 

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.

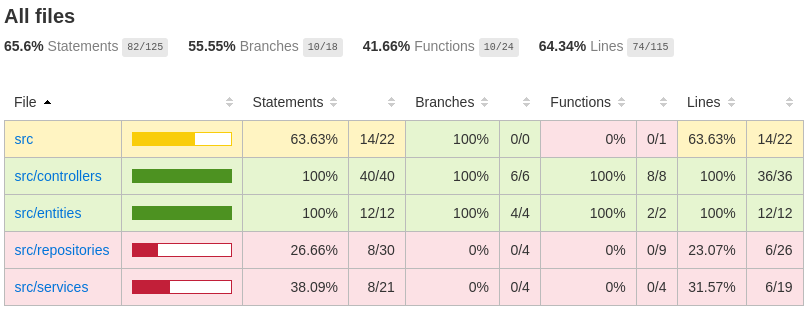


Figure 20: Code Coverage Report of Server Side Code

# 

# 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.

# 

# 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>.