THE UNIVERSITY OF
WINNIPEG

Dept. of Applied Computer Science

ACS-4901

Senior Systems Development Project

Prove IT – Project Completion Report

Team Members

Name	ID
Arshjot Ghuman	3114007
Liam Kristjanson	3116156
Usmaan Sahar	3065852
Vi Le	3119710
W A Shadman	3132520
Xiao Zhang	3131956

IS Director: Victor Balogun

Date: 22 March 2024

Contents

1. Project Overview

- Project Name: Prove IT
- Project Duration: September 6, 2023 – March 14, 2023
- Project Team:
    - Liam Kristjanson, Project Manager
    - Arshjot Ghuman, QA Engineer
    - Usmaan Sahar, Lead Designer
    - Vi Le, Lead Developer
    - W A Shadman, DevSecOps Engineer
    - Xiao Zhang, Technical Lead
- Product Owner: Liam Kristjanson
- Scrum Master: Liam Kristjanson

2. Executive Summary

2.1 Background

HomeTrumpeter Canada is a new company seeking to disrupt the property management industry. HomeTrumpeter has innovated a novel approach to property management through a centralized platform connecting tenants, homeowners, and service providers. To expand the reach of their product, they set out to enter the API economy. This would allow other developers to leverage the existing business logic of the HomeTrumpeter system, in exchange for a service fee. Our team was tasked to construct a new platform built on the foundations of the HomeTrumpeter API gateway, to "Prove IT" could be done.

2.2 Challenges

A key challenge of this project was the need to familiarize ourselves with HomeTrumpeter's API. We were tasked with building against this API as a "Black Box". This meant that we were given no access to the source code of the API, and no information about its internal functions or architecture. As our project was built entirely around HomeTrumpeter's API, it was important to understand the entities and relationships present in the API. HomeTrumpeter provided the development team with access to their Swagger UI to experiment with inputs and outputs of the API. The team used the Swagger UI throughout the project to experiment with and understand the API endpoints required to complete the requirements.

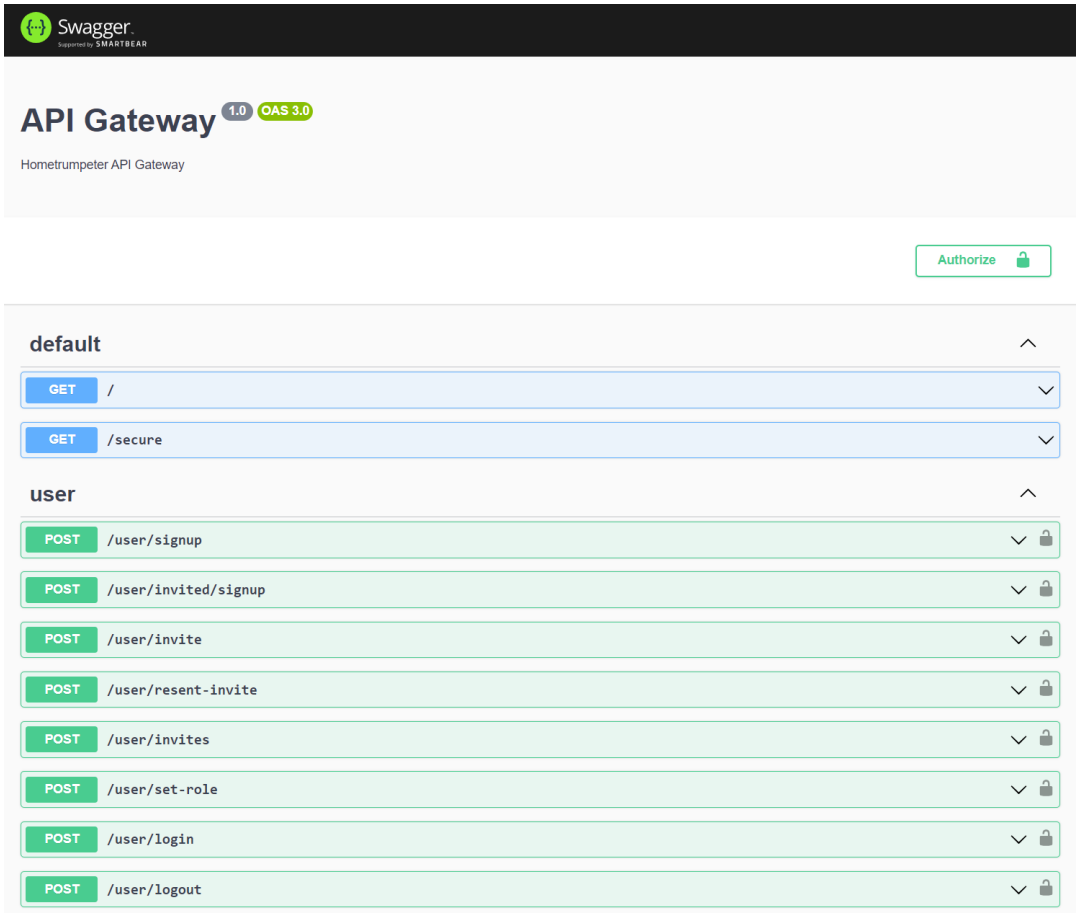


Figure 2.4.1 A screenshot of HomeTrumpeter's Swagger UI. This API gateway served as the foundation of the project. API endpoints are categorized based on usage and labelled as POST or GET requests.

Agile methodology insists upon early and consistent delivery of working software. As such, we were challenged to learn a wide range of unfamiliar technologies early in the project. Members of the team had to acquire a working knowledge of React, GitLab CI/CD, Docker, Kubernetes, and OpenShift to deliver a minimally viable product within the first sprint. We were able to overcome this challenge with

the help of HomeTrumpeter, who provided training materials on the required technologies. By dividing the training materials among team members, we quickly acquired the skills required for continued delivery of our product.

2.3 Achievements

Over 28 weeks, the development team was able to build a property management application that met all high-level requirements gathered at the project's inception. In doing so, we demonstrated the feasibility of HomeTrumpeter's API as a foundation for a tailored property management solution. Throughout the development process, we identified and shared issues as well as points of strength in HomeTrumpeter's API. Furthermore, the team was able to go beyond the initial requirements and develop an AI chatbot capable of creating service requests on behalf of users.

2.4 Outcomes

Overall, the project was a success for all parties involved. As students, the development team gained valuable experience with industry-standard technologies such as React, React Native, GitLab, Kubernetes, and OpenShift. The development team also benefitted from the mentorship of HomeTrumpeter stakeholders with industrial experience. Implementing the scrum framework for team organization allowed team members to experience the agile methodology, which is widely accepted in the industry today. HomeTrumpeter gained insights into the strengths and weaknesses of their API as a means of developing a standalone application. In addition, the Prove IT platform could be used as a proof-of-concept demonstrating the versatility of the HomeTrumpeter API to potential business partners.

3. Project Objectives

3.1 Objectives

The primary objective of the project was to evaluate the feasibility of the API created by HomeTrumpeter. Our objective was not to replace the existing HomeTrumpeter web app. By working closely with the team at HomeTrumpeter, our team strived to identify points of strength, or areas of improvement in the existing API. Based on our findings, HomeTrumpeter will be able to better tailor their API to the requirements of an independent platform.

The API economy has become an important sector of the tech industry in recent years. Companies such as Amazon, Meta, Google, and Microsoft all take advantage of this economy by offering their services through an API gateway. This economy can offer a newfound stream of business for HomeTrumpeter, who have already developed the required services for a property management system through the creation of their flagship product. By allowing other systems to leverage their services, HomeTrumpeter will be able to generate a new stream of income from a system they have already developed.

Our primary focus was the construction of a property management web application leveraging the existing HomeTrumpeter API. Later in the project, we expanded our scope to include the development of a mobile application, and an integrated AI chatbot.

3.2 Deliverables

3.2.1 Medium Fidelity Figma Prototype

The team developed a medium fidelity prototype to conceptualize and communicate our vision for the system. Such a prototype allowed our team to demonstrate the core functionalities of the system and solicit feedback from sponsors. The medium level of fidelity allowed us the flexibility to easily change UI deigns based on stakeholder feedback. The team used this prototype as a guide for interface design throughout the development process.

3.2.2 GitLab CI/CD Pipeline

To facilitate continuous delivery of software throughout the agile process, the team strived to develop a CI/CD pipeline using GitLab. The continuous integration portion of the pipeline allowed our team to automate unit tests, functional testing, static code analysis, and dependency scanning. The continuous deployment portion of the pipeline automated the process of building our application, containerizing it through Docker, and deploying it to HomeTrumpeter's private cloud infrastructure.

3.2.3 Front-End Web Application

The primary deliverable of the project is a React web application leveraging HomeTrumpeter's API. This application connects homeowners, tenants, and service providers to streamline access

to property services. Homeowners can contact service providers for repairs or maintenance and track the progress of maintenance requests made by tenants.

3.2.4 Back-End Proxy

The back-end proxy mediates communication between the web application and HomeTrumpeter's API. Security is ensured by implementing rate limiting, exposing only the necessary HomeTrumpeter endpoints, and storing the API key in the cloud environment.

3.2.5 React Native Mobile App

To expand the availability of our platform, the team developed a React Native mobile application for both iPhone and Android. The mobile application offers a subset of the functionality of the web application, allowing access for tenants, homeowners and service providers on the go.

3.2.5 Integrated AI Chatbot

The team developed an AI chatbot to enhance user interaction and streamline service requests on the platform. Designed to handle natural language inputs, the chatbot simplifies the process of navigating HomeTrumpeter's vast service options. It autonomously identifies relevant services based on user descriptions, improving efficiency and accessibility for tenants. Built with scalability in mind, its architecture allows for future enhancements and integration of new features, making it a key component in enriching the platform's ecosystem and user experience.

4. Sprint Overview

4.1 Sprint 1

a. Duration: September 6 – October 12

b. Key Deliverables: Project Proposal, Project Plan

c. Achievements:

   i. Defined overall objectives and timeline for the project

ii. Solicited high-level requirements from HomeTrumpeter stakeholders

iii. Selected scrum methodology, assigned roles, and defined the project framework.

d. Challenges

i. Forming a new team, getting to know each other.

ii. Finding time for regular Scrum meetings that work with everyone's different schedules.

iii. Learning the goals and challenges faced by a new, unfamiliar company stakeholders.

4.2 Sprint 2

a. Duration: October 12 – October 26

b. Key Deliverables: Conceptual Prototypes, Basic CI/CD Pipeline, Create Account and Login Functionality

c. Achievements:

i. Developed Prototypes for Create Account, Login, Invite Tenant, Add Property, Invite Service Provider, and Create Service Request functionalities.

ii. Delivered a Basic GitLab CI/CD Pipeline capable of deploying a web application to HomeTrumpeter's OpenShift architecture.

iii. Selected technologies to build the project based on HomeTrumpeter's existing architecture, and their vision for the project.

d. Challenges:

i. Learning the integrations between GitLab, Kubernetes, and OpenShift to successfully develop our CI/CD pipeline

ii. Learning new technologies in our stack such as React, OpenShift, Kubernetes, and GitLab.

4.3 Sprint 3

a. Duration: October 26 – November 9

b. Key Deliverables: Threat Modelling Diagram, Promotion of Environments and Security Scanning functionality in CI/CD pipeline

c. Achievements:

    i. Configured our CI/CD pipeline to deploy to Development, Staging, and Production environments.

    ii. Identified key assets, threat agents, and countermeasures through threat modelling.

d. Challenges:

    i. Deploying to multiple environments required an increase in the complexity of our deployment architecture.

    ii. It was discovered that the construction of a backend proxy would be required to not expose HomeTrumpeter's API key to all users of the system.

4.4 Sprint 4

a. Duration: November 9 – November 23

b. Key Deliverables: Backend API Proxy, CI/CD Infrastructure to deploy the proxy to all environments.

c. Achievements:

    i. Developed a backend proxy to allow the web app to securely communicate with HomeTrumpeter's API

    ii. Successfully adapted the project timelines to deliver the backend proxy on time.

d. Challenges:

    i. Adapting the product backlog to include the backend system which was not part of the initial requirements.

ii. Changes to the configuration of frontend deployments to ensure they communicate with the correct backend proxy according to their environment.

4.5 Sprint 5

a. Duration: November 23 – December 7

b. Key Deliverables: Functionality for homeowners to add and validate properties.

c. Achievements:

i. Successfully delivered functionality to add a property to the system and validate its existence, despite the challenges presented by this feature.

d. Challenges:

i. Property creation and validation required numerous calls to HomeTrumpeter's API to fetch Cities and ZIP Codes and validate the street address. This led to the feature taking longer than expected to develop.

4.6 Sprint 6

a. Duration: December 7 – December 21

b. Key Deliverables:

i. Functionality for homeowners to invite tenants

ii. Functionality for tenants to accept an invitation, create an account, and log in.

iii. Improvements to CI/CD pipeline to deploy to an additional HomeTrumpeter data center.

c. Achievements

i. Improved CI/CD pipeline to deploy to both of HomeTrumpeter's US data centers to increase product uptime.

ii. Expanded the product to another user group, allowing tenants to onboard into the system.

d. Challenges

i. HomeTrumpeter's existing email invitation architecture could not be leveraged for our own product.

4.7 Sprint 7

a. Duration: December 21 – January 4

b. Key Deliverables:

i. Functionality allowing homeowners to invite service providers

ii. Functionality allowing service providers to accept an invitation, create an account, and log into the system.

iii. Functionality allowing service providers and tenants to receive their invitation to the system by email.

c. Achievements:

i. Expanded our product to our third and final user group, allowing service providers to onboard into the system.

ii. Developed an infrastructure using NodeMailer allowing email invitations to be sent to tenants and service providers.

4.8 Sprint 8

a. Duration: January 4 – January 18

b. Key Deliverables:

i. Functionality allowing service providers to post their available services.

ii. Functionality allowing tenants to initiate a service request to their homeowner.

c. Achievements:

    i. Developed the foundational requirements for service request functionality

d. Challenges:

    i. Balancing continuous delivery of product improvements with the requirements of the Detailed Design Review.

4.9 Sprint 9

a. Duration: January 18 – February 1

b. Key Deliverables:

    i. Functionality allowing homeowners to request a quote for service from their own service providers and approve a quote once one is received.

    ii. Functionality allowing service providers to send quotes for service to homeowners.

c. Achievements:

    i. Completed development of the core functionality of our product, allowing service requests to be created by the tenant, received by the homeowner, before being quoted and completed by the service provider.

d. Challenges:

    i. An issue was discovered between our backend proxy and HomeTrumpeter's API that was causing service requests to disappear. HomeTrumpeter's developers added error handling to their API to resolve the issue.

4.10 Sprint 10

a. Duration: February 1 – February 15

b. Key Deliverables:

    i. Functionality allowing a service provider to mark jobs as in progress or completed.

ii. Functionality allowing a service provider to apply for a background check to gain public availability in the system.

iii. Functionality allowing homeowners to search for verified public service providers to respond to their service requests.

iv. Functionality allowing a homeowner to request and review background check from their tenants.

c. Achievements:

i. Successfully integrated our product with background check provider Certn, allowing for background checks to be requested and processed for tenants and service providers.

d. Challenges:

i. Certn's testing environment did not allow us to manipulate the responses from a requested background check, making it difficult to test the functionality of the system under different background check results.

4.11 Sprint 11

a. Duration: February 15 – February 29

b. Key Deliverables:

i. Various quality improvements to user interfaces throughout the web application.

ii. AI Chatbot capable of parsing service requests from natural language.

iii. React Native mobile application with basic homeowner features ported from the web app.

c. Achievements:

i. Succeeded in developing a chatbot capable of creating service requests based on a natural language conversation. This will add a new type of interaction to our system that may make it more usable for less technically literate users.

ii. Developed a React Native mobile app from scratch, porting some of the homeowner features, and allowing for expansion of other features.

d. Challenges:

i. Selecting an appropriate set of models to achieve the desired behavior and functionality of the chatbot.

ii. The initial setup for the React Native app was time-consuming. All members of the team had to learn to set up a new development environment to contribute to the mobile app.

4.12 Sprint 12

a. Duration: February 29 – March 14

b. Key Deliverables:

i. UI overhaul to the dashboards of the web application

ii. Tenant and Service provider functionalities on the react native mobile app

c. Achievements:

i. Successfully created mobile app functionality for both tenants and service providers, expanding on previous functionality

ii. Updates to the web application dashboard offer better aesthetics for the upcoming presentation

d. Challenges:

i. Testing the large-scale UI changes to ensure no issues were created anywhere in the web application as a result.

ii. Balancing product improvement with preparation for the PCR.

5. Project Metrics

5.1 Story Points

The key metric used throughout the project to measure project progress, completion velocity, and work effort was story point estimation. Story points are a comparative measure of the estimated effort required to complete a single work item. Each item in the product backlog was assigned story points, with more difficult items being assigned more. These estimates would be used during sprint planning to help ensure that work was divided as equally as possible among team members. The team also strived to ensure that a similar number of story points were completed each sprint, keeping in line with the Agile principle of sustainable progress.

5.2 Burndown Charts

During the completion of each sprint, the team would review a burndown chart, to assess the progress being made towards the sprint's goals. The burndown chart plots the number of story points currently completed against the number of story points expected to be completed at each date. If the number of completed story points lagged significantly behind the expectation at the halfway point of the sprint, this would indicate to the team that some action should be taken to ensure the on-time completion of the sprint. Such actions could include additional person-hours being allocated to the project, or the re-prioritization of goals initially set for the sprint. One of these burndown charts is pictured in figure 5.2.1.

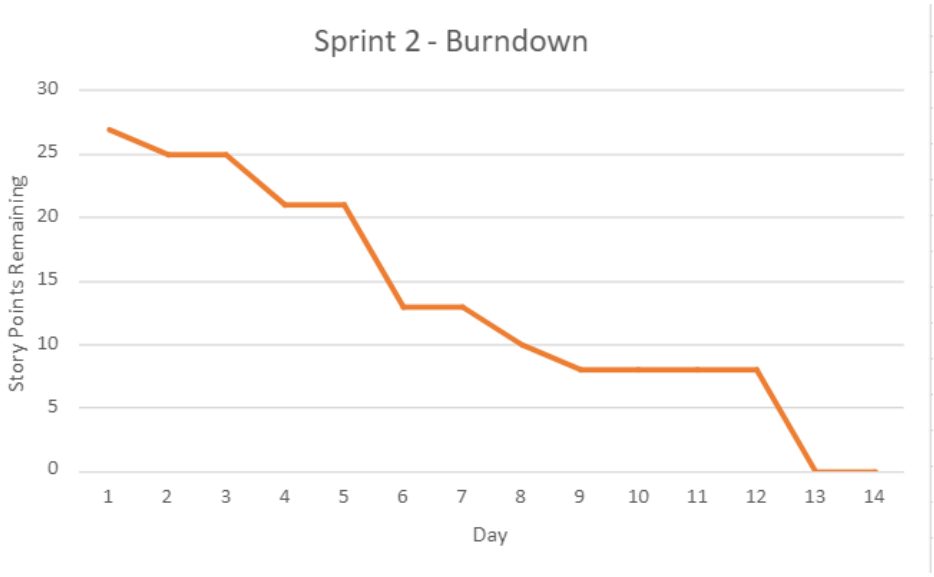


Figure 5.2.1 - A burndown chart for Sprint 2 of Project Prove IT. This chart plots the number of story points completed in a given sprint against the number of days remaining in the sprint.

5.3 Final Project Metrics

The team completed a total of 382 story points over 12 sprints, for an average of 31.8 story points per sprint. Story points completed per sprint had a standard deviation of 8.4 story points, giving a coefficient of variation of 26%. These metrics indicate that the team was able to make consistent, significant progress, measured in working software throughout the project. The variation in story points completed per sprint reflect expected fluctuations in output based on changing requirements, team members availability to contribute to the project, as well as over – and underestimation of required effort. The rate at which story points were completed each sprint increased throughout the project. This is reflective of the continued learning of the team members, allowing us to complete work more efficiently as we gained experience with the employed technologies.

6. Product Backlog Evolution

6.1 Product Backlog Creation

Our product backlog began with a list of 6 high-level requirements HomeTrumpeter wished to see implemented in a new application leveraging their existing API. These requirements were as follows:

1. Login using the API gateway

2. Tenant onboarding
3. Service provider onboarding
4. Service request creation
5. Background check functionality
6. AI service provider verification using public information

Given the project timeline of 24 weeks, the required work needed to be split into 12 two-week sprints. Each of the requirements were allocated 2 sprints to be completed. Each high-level requirement was then broken down into individual work items, and story-point estimations for the required effort to complete each item were assigned. Work items were refined, added, or removed as necessary at the beginning of each sprint based on stakeholder feedback and learnings from the previous sprint.

4.0	**Sprint - 1 - Login with API Portal**			**7**			
4.1	Sprint 1 - Planning Meeting	ALL		0	05 Oct 2023 6:00PM	05 Oct 2023 7:00PM	Not Started
4.2	Develop CI/CD Pipeline	Unassigned		3			Not Started
4.3	Login Gateway Wireframe	Unassigned		2			Not Started
4.4	Add Property Wireframe	Unassigned		2			Not Started
4.5	Sprint 1 - Retrospective	Project Team		0			Not Started
5.0	**Sprint - 2 - Login with API Portal**			**14**			
5.1	Sprint 2 - Planning / Sprint 1 Demo	ALL		0	12 Oct 2023 6:00PM	12 Oct 2023 7:00PM	Not Started
5.2	Homeowner/Manager Create Account	Unassigned		8			Not Started
5.3	Homeowner/Manager Login	Unassigned		5			Not Started
5.4	Homeowner/Manager Logout	Unassigned		1			Not Started
6.0	**Sprint - 3 - Login with API Portal**			**16**			
6.1	Sprint 3 - Planning / Sprint 2 Demo	ALL		0	26 Oct 2023 6:00PM	26 Oct 2023 7:00PM	Not Started
6.2	Add Property	Unassigned		8			Not Started
6.3	View Property Detail	Unassigned		3			Not Started
6.4	Login API - Testing	Unassigned		5			Not Started
7.0	**Sprint - 4 - Tenant Onboarding**			**20**			
7.1	Sprint 4 - Planning / Sprint 3 Demo	ALL		0	09 Nov 2023 6:00PM	09 Nov 2023 7:00PM	Not Started
7.2	Tenant Invitation Wireframe	Unassigned		2			Not Started
7.3	Send Tenant Invitation	Unassigned		5			Not Started
7.4	System Study Review	Project Team		13		13 Nov 2023 11:59PM	Not Started
8.0	**Sprint - 5 - Tenant Onboarding**			**16**			
8.1	Sprint 5 - Planning / Sprint 4 Demo	ALL		0	23 Nov 2023 6:00PM	23 Nov 2023 7:00PM	Not Started
8.2	Tenant Accept Invitation	Unassigned		3			Not Started
8.3	Tenant Create Account	Unassigned		5			Not Started
8.4	Tenant Login	Unassigned		3			Not Started
8.5	Tenant Onboarding - Testing	Unassigned		5			Not Started
9.0	**Sprint - 6 - Service Provider Onboarding**			**13**			
9.1	Sprint 6 - Planning / Sprint 5 Demo	ALL		0	07 Dec 2023 6:00PM	07 Dec 2023 7:00PM	Not Started
9.2	Invite Service Provider Wireframe	Unassigned		2			Not Started
9.3	Send Service Provider Invite	Unassigned		3			Not Started
9.4	Create Service Provider Account	Unassigned		5			Not Started
9.5	Service Provider Login	Unassigned		3			Not Started
10.0	**Sprint - 7 - Service Provider Onboarding**			**11**			
10.1	Sprint 7 - Planning / Sprint 6 Demo	ALL		0	21 Dec 2023 6:00PM	21 Dec 2023 7:00PM	Not Started
10.2	View My Services	Unassigned		3			Not Started
10.3	Add a Service	Unassigned		3			Not Started
10.4	Service Provider Onboarding - Testing	Unassigned		5			Not Started
11.0	**Sprint - 8 - Service Request Creation**			**26**			
11.1	Sprint 8 - Planning / Sprint 7 Demo	ALL		0	04 Jan 2024 6:00PM	05 Jan 2024 7:00PM	Not Started
11.2	Create Service Request Wireframe	Unassigned		3			Not Started
11.3	Create Service Request	Unassigned		3			Not Started
11.4	Request Quote	Unassigned		2			Not Started
11.5	View Service Requests	Unassigned		3			Not Started
11.6	View Service Request Details	Unassigned		2			Not Started
11.7	Detailed Design Review	Unassigned		13		08 Jan 2024 11:59PM	Not Started
12.0	**Sprint - 9 - Service Request Creation**			**21**			
12.1	Sprint 9 - Planning / Sprint 8 Demo	ALL		0	18 Jan 2024 6:00PM	18 Jan 2024 7:00PM	Not Started
12.2	Send Proposal	Unassigned		2			Not Started
12.3	View Proposed Quotes	Unassigned		3			Not Started
12.4	View Proposed Quote Details	Unassigned		2			Not Started
12.5	Approve Proposed Quote	Unassigned		2			Not Started
12.6	Proposal Notification	Unassigned		2			Not Started
12.7	Proposal Approval Notification	Unassigned		2			Not Started
12.8	Service Request - Testing	Unassigned		8			Not Started
13.0	**Sprint - 10 - Background Check**			**13**			
13.1	Sprint 10 - Planning / Sprint 9 Demo	ALL		0	01 Feb 2024 6:00PM	01 Feb 2024 7:00PM	Not Started
13.2	Background Check Wireframe	Unassigned		2			Not Started
13.3	Apply for Public Service Provider	Unassigned		3			Not Started
13.4	Send Certn Background Check Request	Unassigned		5			Not Started
13.5	View Status of Background Check	Unassigned		3			Not Started
14.0	**Sprint - 11 - Background Check**			**16**			
14.1	Sprint 11 - Planning / Sprint 10 Demo	ALL		0	15 Feb 2024 6:00PM	15 Feb 2024 7:00PM	Not Started
14.2	Process Completed Background Check from Certn	Unassigned		8			Not Started
14.3	Grant Public Service Provider Status	Unassigned		3			Not Started
14.4	Background Check - Testing	Unassigned		5			Not Started
15.0	**Sprint - 12 - Service Provider ML Model**			**19**			
15.1	Sprint 12 - Planning / Sprint 11 Demo	ALL		0	29 Feb 2024 6:00PM	29 Feb 2024 7:00PM	Not Started
15.2	Research for ML Model	Unassigned		3			Not Started
15.3	Development of ML Model	Unassigned		8			Not Started
15.4	Integration of ML Model into System	Unassigned		8			Not Started

Figure 6.1.1 The Product Backlog, as created at the beginning of the project. The requirements assigned to each sprint are left at a high level, to be more extensively planned in that sprint's planning meeting.

6.2 Addition of Backend Proxy

At the end of Sprint 2 the team had developed basic login and account creation functionalities in a simple frontend-only application. During the sprint demo meeting, HomeTrumpeter stakeholders expressed concern that their API key would be exposed to threat agents with this simple architecture. It was suggested that a backend proxy be developed to act as an intermediary between the Prove IT application and HomeTrumpeter's API. This requirement was not initially planned for and required changes to the product backlog to accommodate. With the flexibility offered by the agile framework, the team was able to move work items initially planned for Sprint 3 into later sprints. The team was

able to successfully adapt to this changing requirement and produce the backend system on-time as requested. A diagram of this proxy architecture is given in Figure 6.2.1

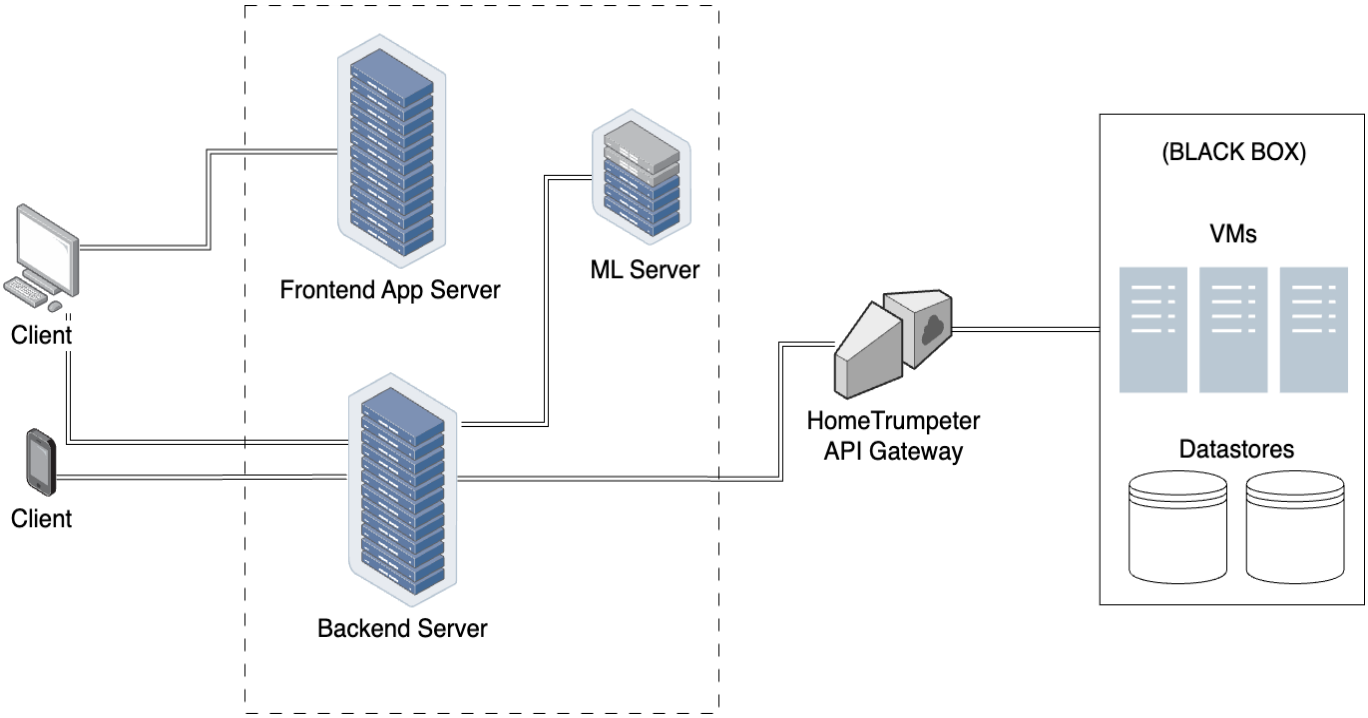


Figure 6.2.1 A diagram of Project Prove IT's proxy architecture.

6.3 React Native Mobile Development

At the beginning of the project, HomeTrumpeter stakeholders expressed interest in developing a React Native mobile application alongside our web app. The team decided to focus on development of the web application first, and to develop the mobile app only if time allowed. All high-level requirements for the web application were completed ahead of schedule, allowing the team to develop a React Native spinoff application near the end of the project.

6.4 Changes to AI Component

During initial requirements gathering, HomeTrumpeter stakeholders expressed interest in developing an AI-leveraging feature as part of the project. HomeTrumpeter's initial concept for this feature was a sentiment analysis model to help with the verification of service providers. This model would work in compliment with the background check, using sentiment analysis on publicly available information about service providers in the system to evaluate their trustworthiness. HomeTrumpeter gave discretion to the team regarding this requirement and were open to other ideas for an AI feature.

It was decided at the midpoint of the project that developing an AI model to evaluate service providers would not be feasible given the budget, personnel and timeline of the project. The team devised a new plan, to develop an AI chatbot to help users create service requests. HomeTrumpeter was very receptive to this idea. Development of the AI component was already slated towards the end of the project, so a change in concept was not disruptive to project timelines. The flexibility to change requirements offered by agile saved a large amount of effort that would have been wasted extensively planning an unfeasible feature.

7. System Architecture

7.1 Overview

Our web application utilizes the typical architecture of a full-stack web application. Two layers are hosted on the Prove IT server, consisting of the front-end and back-end. The popular framework ReactJS provides a foundation for building the front-end web application. ExpressJS and NodeJS are used to create the back-end proxy. As we are building against the HomeTrumpeter API, a database is not required.

The React Native mobile application accesses the same backend infrastructure as the web application. However, unlike the web application, the mobile app is not publicly available, as public deployment to the App Store or Google Play requires a lengthy review and approval process.

Our AI chatbot, "Homie" is deployed as an independent service comprised of two components. The RASA chatbot processes the natural language input and produces natural language responses. If a service request is detected by the chatbot, the RASA actions server creates a service request on behalf of the user. This service request will then be passed off to the Prove IT server for processing.

7.2 Detailed Design

The frontend, backend and chatbot servers are hosted as separate microservices on HomeTrumpeter's OpenShift Kubernetes platform. The front-end and backend servers are publicly accessible, whereas the chatbot server and HomeTrumpeter's API are protected behind the rate limiting and validation implemented by the backend server. This design reduces the potential for Prove IT to be used as an attack surface against HomeTrumpeter's API. Email invitations are handled by the Node Mailer

component of the backend server. All data flowing into Node Mailer invitations are sanitized to prevent cross-site scripting attacks. Figure 7.2.1 shows our system architecture.

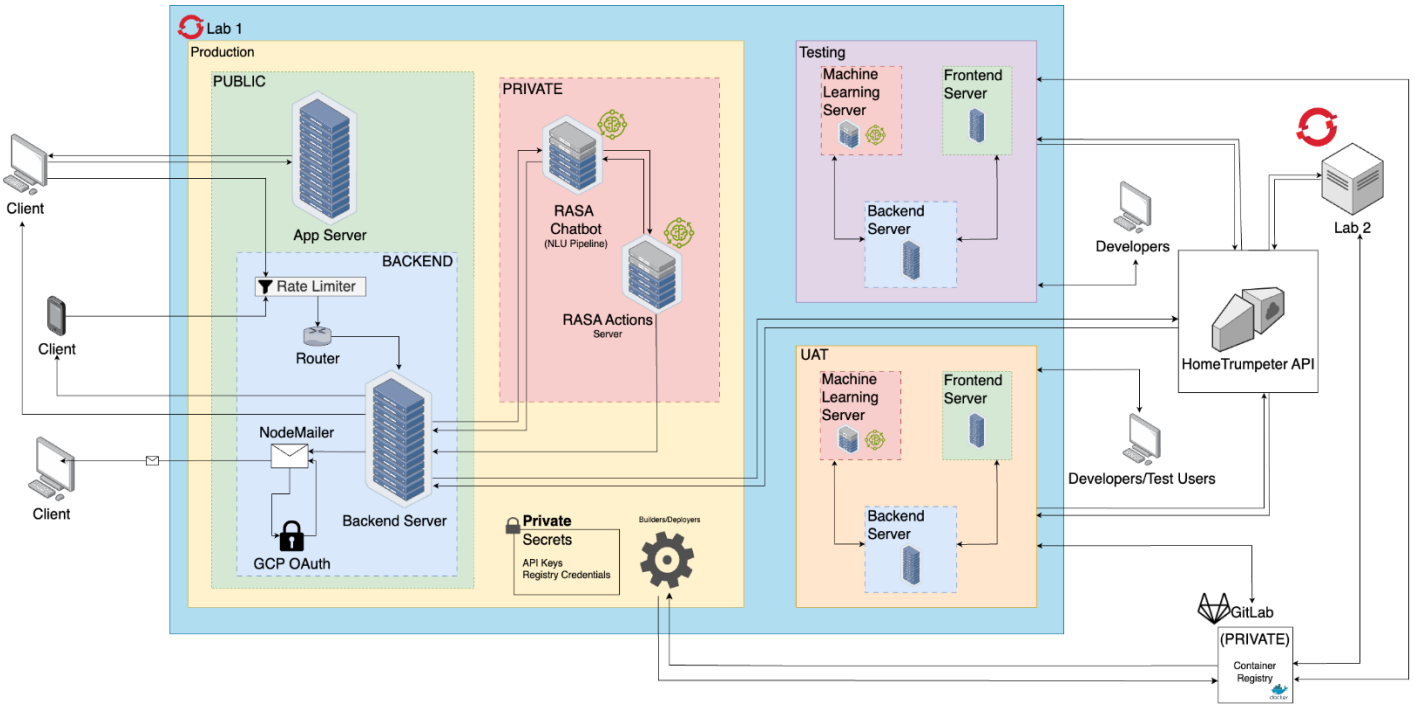


Figure 7.2.1 Architecture diagram for Prove IT. The front-end web server, back-end proxy server and chatbot server are hosted on the same cluster within the OpenShift private cloud. The system is deployed to HomeTrumpeter clusters in two locations, Lab 1 and Lab 2.

The separation into separate microservices allows each system to grow/shrink depending on resource usage. System administrators can scale the system to fit their needs.

7.3 JWT Session Management

Authentication and session management are implemented using the JWT tokens provided by the HomeTrumpeter API. The JWT tokens are stored in the local storage of the web browser and retrieved for POST and GET requests. Requests to HomeTrumpeter's API are required to contain both a JWT token held on the client side, as well as the API key, which is securely stored in the OpenShift environment. These multiple layers of security help ensure that unauthorized requests are not granted access to data stored in HomeTrumpeter's architecture.

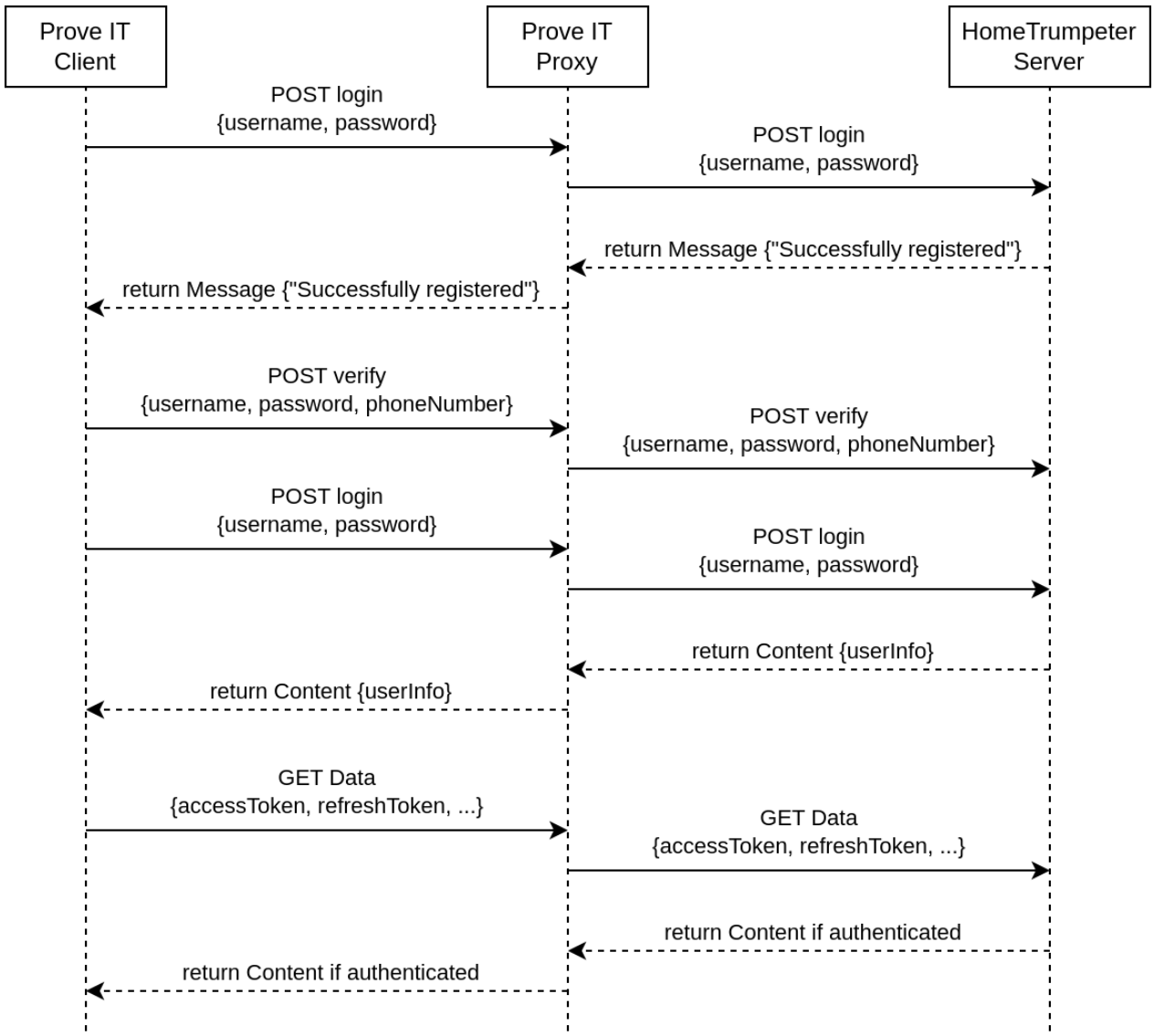


Figure 7.3.1 System sequence diagram for account creation using JWT session management. User login information is delivered as a POST request, which returns an access token and refresh token to be stored in Local Storage. Data will be accessed through GET requests which will include access tokens for authentication. Access tokens that expire are renewed with refresh tokens.

8. Database Design

Our objective for this project was to build a stateless system, leveraging HomeTrumpeter's existing architecture for data persistence. As we were building against their API as a 'black box', we did not have access to any formal documentation regarding the database schemas or architecture. The team was able to devise an understanding of the entities and relationships present in HomeTrumpeter's model through experimentation with the API and their existing product. We produced the following

domain class diagram as a result, covering the entities and relationships relevant to the scope of our system.

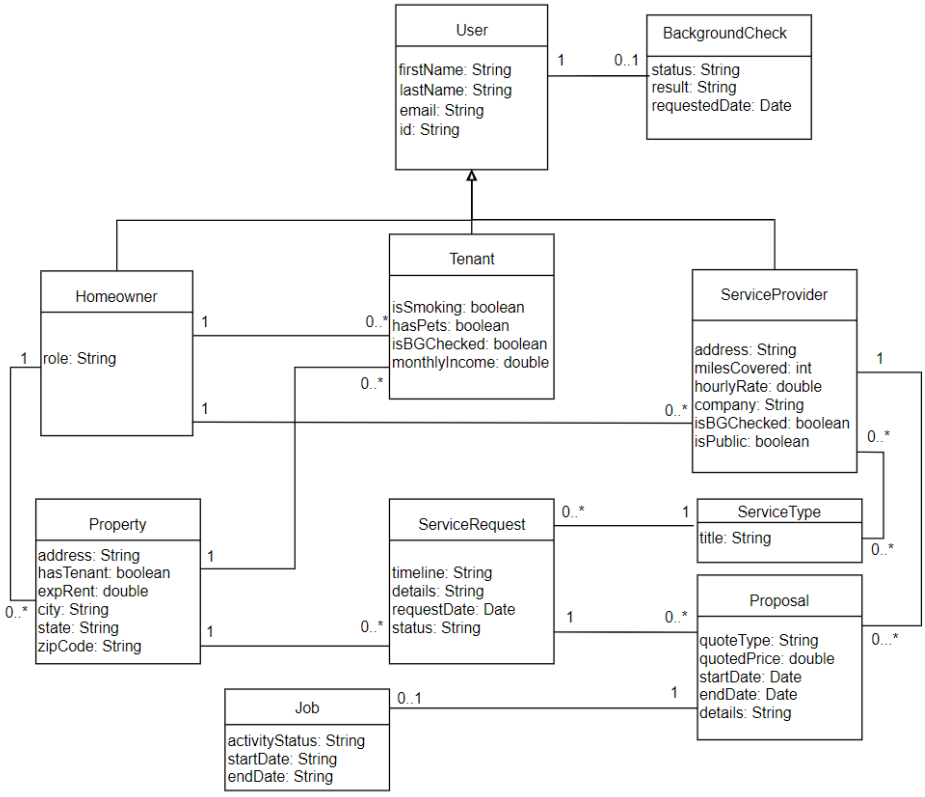


Figure 8.1 A domain class diagram for Project Prove IT. The domain of HomeTrumpeter's system is larger than shown here. This DCD covers only entities relevant to Prove IT.

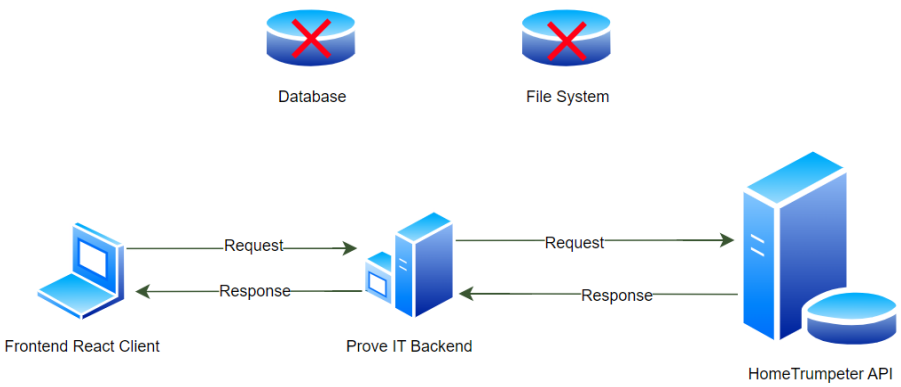


Figure 8.2 Data storage and access within the Prove IT back-end system. The persistence layer does not include a database nor file storage system. All data will be retrieved from the HomeTrumpeter database and returned to the frontend application.

9. User Interface Design

9.1 Overview

In accordance with the agile methodology, an iterative approach was taken to user interface design. The interaction design process served as the guide for our UI design. At the beginning of the project, user stories were identified in collaboration with HomeTrumpeter stakeholders. These user stories would form the basis for our prototyping process. Prototypes were drafted first on paper, allowing for a wide range of potential designs to be considered. The paper prototypes also allowed us to consider other methods of design and what features would be placed on what page. Following paper prototypes, horizontal prototyping in Figma allowed stakeholders to provide feedback on a breadth of functionalities. It also provided an opportunity to do minimal testing on each feature being developed to determine whether it required any additional consideration. Finally, vertical prototyping integrated all functional requirements into the horizontal prototype after all feedback was considered.

User Group	User Story
Homeowner	As a homeowner, I would like to sign up for the services, so that I can add/save/edit my information.
	As a homeowner, I would like to add properties, so that I can keep track of them.
	As a homeowner, I would like to invite tenants, so that I can rent my properties to them.
	As a homeowner, I would like to approve tenants, so that I can ensure my tenants are all up to certain standards.
	As a homeowner, I would like to invite private service providers, so that I can assign them to fix issues with my properties.
	As a homeowner, I would like to browse service providers so that I can select the best provider for a required service.
	As a homeowner, I would like to request a quote from a service provider, so that they can provide a price for their services.
	As a homeowner, I would like to review and approve quotes from service providers, so that we can agree on a price for a given service.
	As a homeowner, I would like to track the status of a job, so that I can know when the service is in progress, and when it has been completed.
Service Provider	As a service provider, I would like to accept invitations from homeowners, so that I can provide service to them.
	As a service provider, I would like to sign up, so that I can add/save/edit my information.
	As a service provider, I would like to review service requests and provide a quote so that I can offer my business to homeowners.
	As a service provider, I would like to update the status of a ticket, so that I can notify the homeowner and tenant of the service progress.
	As a service provider, I would like to be able to mark my job as completed, so that I can finalize the work and get paid.
Tenant	As a tenant, I would like to accept invitations from homeowners, so that I can become a tenant at the respective property.
	As a tenant, I would like to have access to my homeowner's contact information, so that I can contact them in the case of any issues.
	As a tenant, I would like to sign up, so that I can add/save/edit my personal information.
	As a tenant, I would like to submit maintenance tickets, so that I can request services.
	As a tenant, I would like to track the status of my tickets, so that I know the progress of my service request.

Figure 9.1.1 User stories gathered as part of the design process. These user stories served as the basis for our interaction design process.

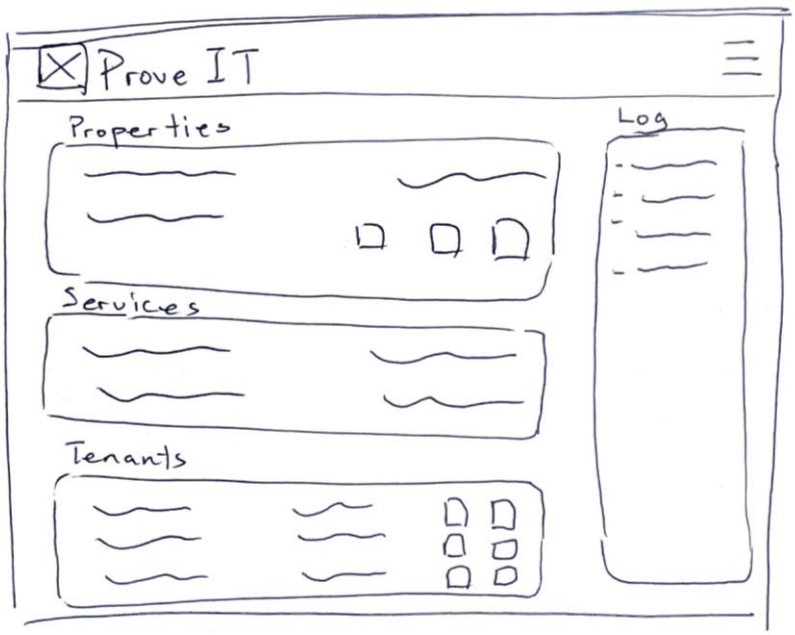


Figure 9.1.2 An early paper prototype for the Homeowner dashboard of Prove IT.

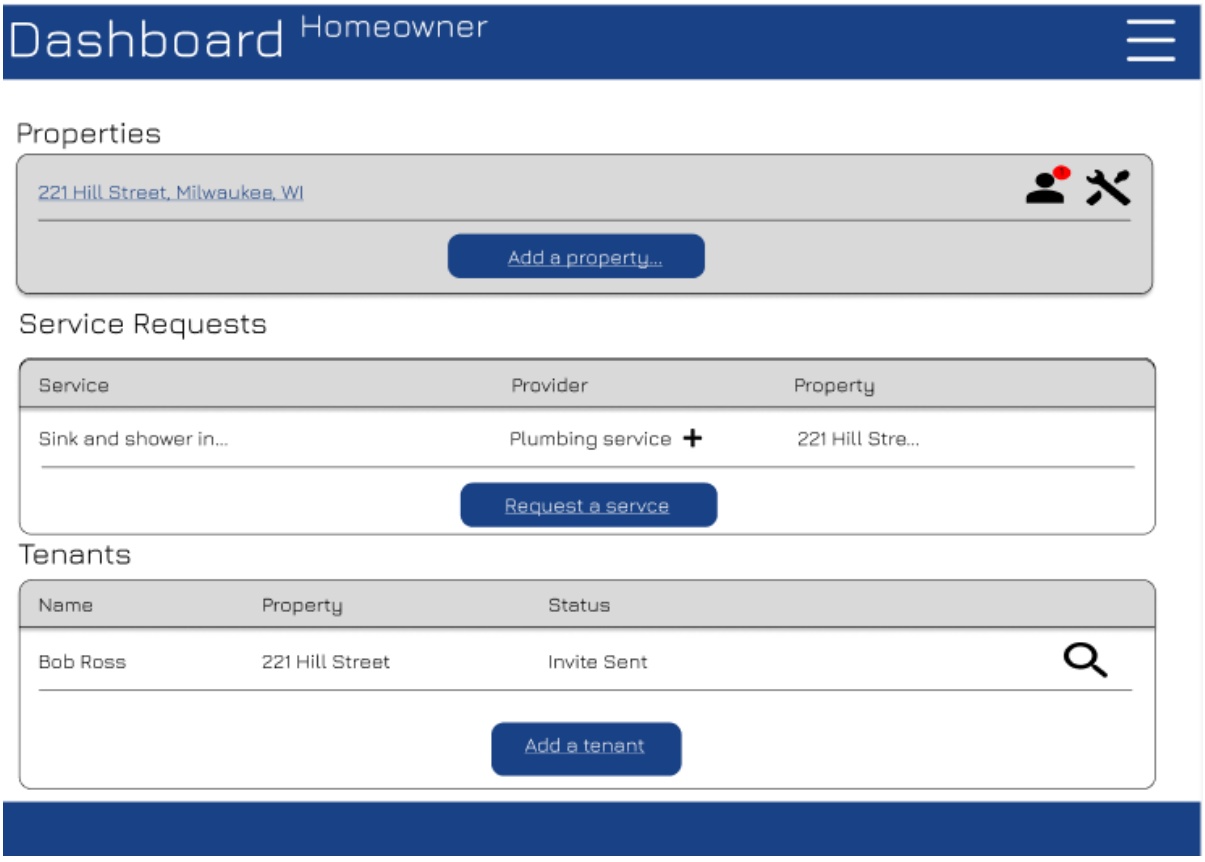


Figure 9.1.3 The homeowner dashboard, as implemented in our medium-fidelity Figma prototype.

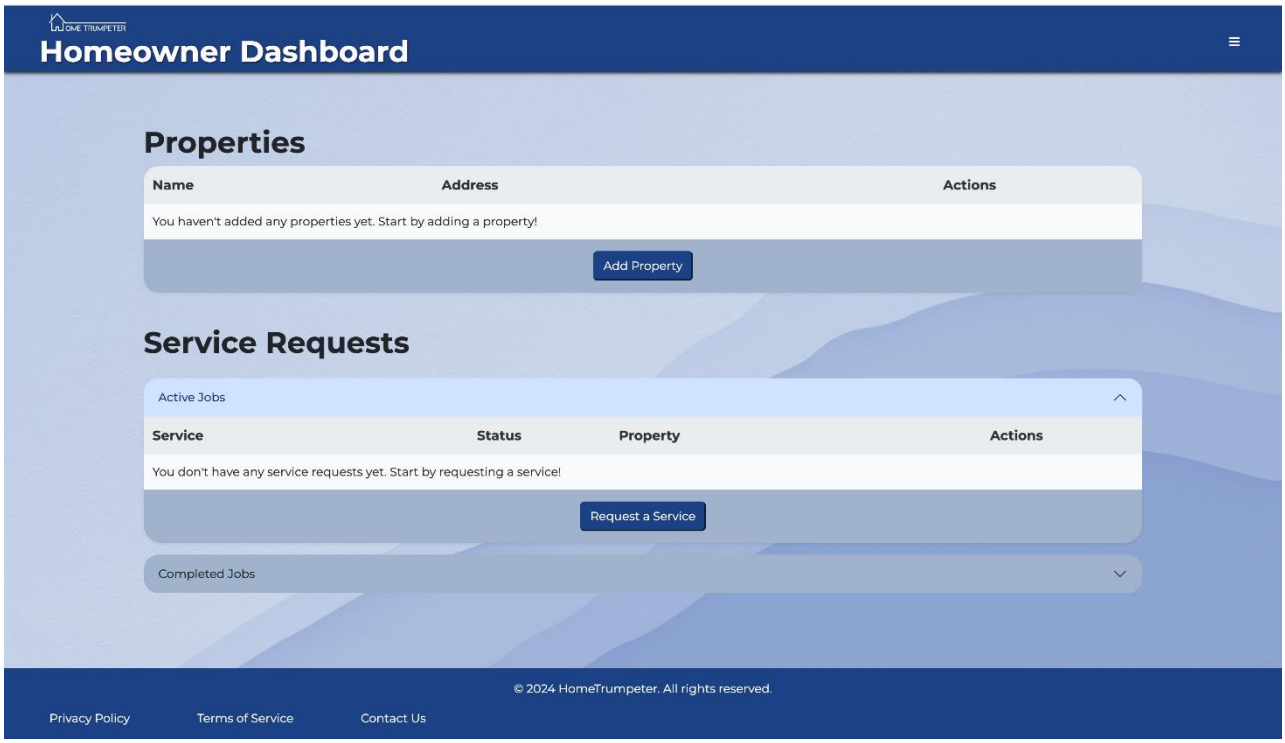


Figure 9.1.4 The final delivered design for the homeowner dashboard.

9.2 Usability Goals, Design Principles and Heuristics

Human Computer Interaction (HCI) design principles were considered during the prototyping and development processes. Consistency is established by maintaining a brand kit, which provides recognizable colour schemes, logos, icons, and fonts. Feedback is provided by hover effects and loading icons when action is taken. Flexbox and grid layouts in CSS are used to create logical mappings. Affordance was created by adding outset borders to all clickable buttons, to give them the appearance of a real three-dimensional button. Constraints were implemented by de-activating buttons while the system was in a loading state. This constraint prevents users from repeatedly pressing the same button, making the same API call that is already in progress.

Consideration was also given to the Nielsen Norman Group's ten usability heuristics. Of particular importance was the visibility of the system status. Nearly all actions throughout the Prove IT system require some type of API call and response. During this time, the user should be made aware that something is happening "behind the scenes". We followed this heuristic by implementing loading spinners for every API call.

Another of Nielsen Norman Group's heuristics is the ability of users to recognize, diagnose, and recover from errors. This heuristic was implemented through error messages clearly indicated in red. To combat the effects of change blindness, these error messages would be displayed as close as possible to the button the user pressed to trigger the error message. Fields failing validation are highlighted in red to direct the user towards recovery from the error.

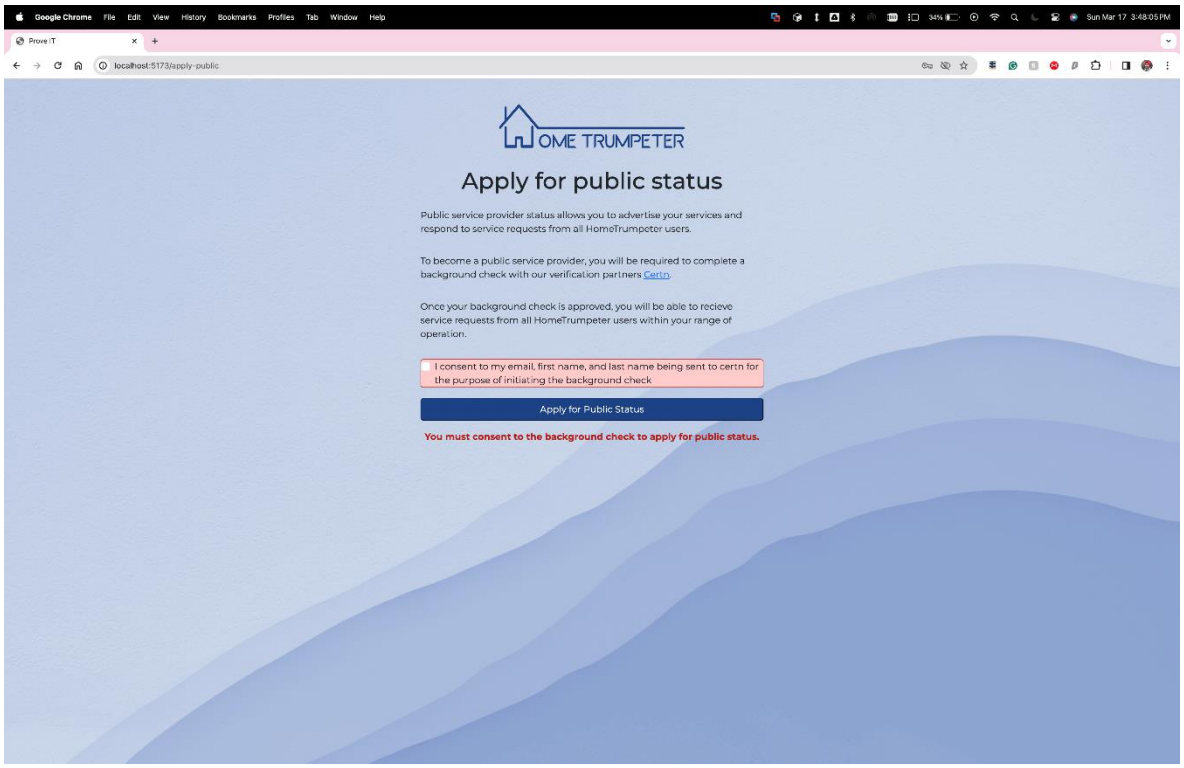


Figure 9.2.1 A clear error message, presented in plain language. The interface highlights the area in need of attention to help the user easily recover from the error.

Finally, the heuristic of recognition rather than recall is implemented when adding a property. Our system requires users to input a property's state, city, street address, and ZIP code for validation purposes. Most people have no problem remembering the city and state they live in, but the arbitrary ZIP code offers more difficulty. Our system addresses this by requesting the city and state first, before offering a list of ZIP codes applicable to that city. With this design, users can recognize their zip code among a small list, instead of being forced to recall it from memory.

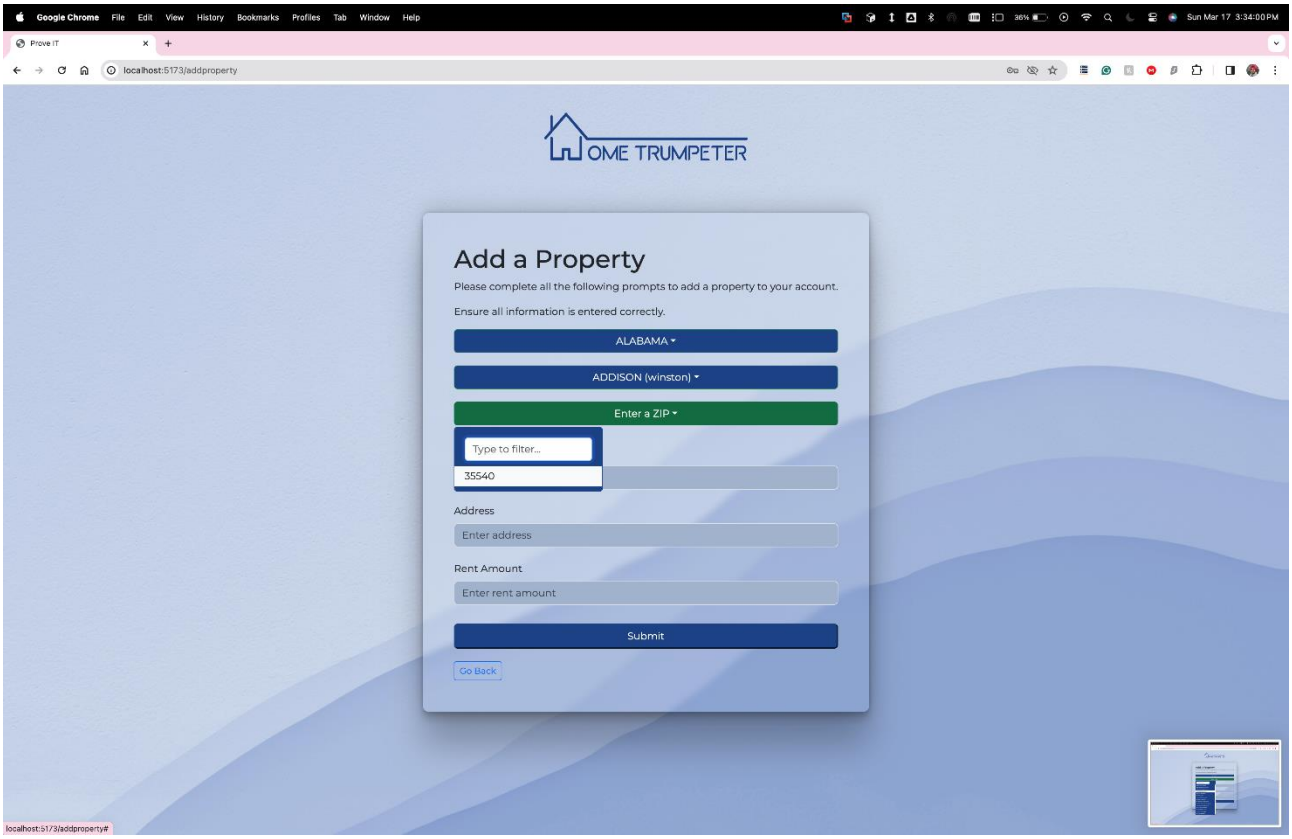


Figure 9.2.2 The ZIP code input interface. Prove IT automatically fetches valid ZIP codes for the selected city. Some cities have more than one zip code, in which case a list would be presented in the dropdown.

9.3 Adjustments Made

As we strived to deliver continuous improvement throughout the project, re-designing an existing interface would sometimes be necessary. One example of such a re-design took place with our request quote page. This page allows homeowners to browse available service providers and request a quote for a particular job. Initially, we employed the Card interface metaphor to implement this feature. However, in later discussions, it was recognized that the “scanability” of the card layout was not ideal. A homeowner will likely want to select a provider based on the lowest price, or shortest timeline available. To better facilitate such scanning, we arranged the service provider information into a simple table. The result was a much more scannable and space-efficient interface.

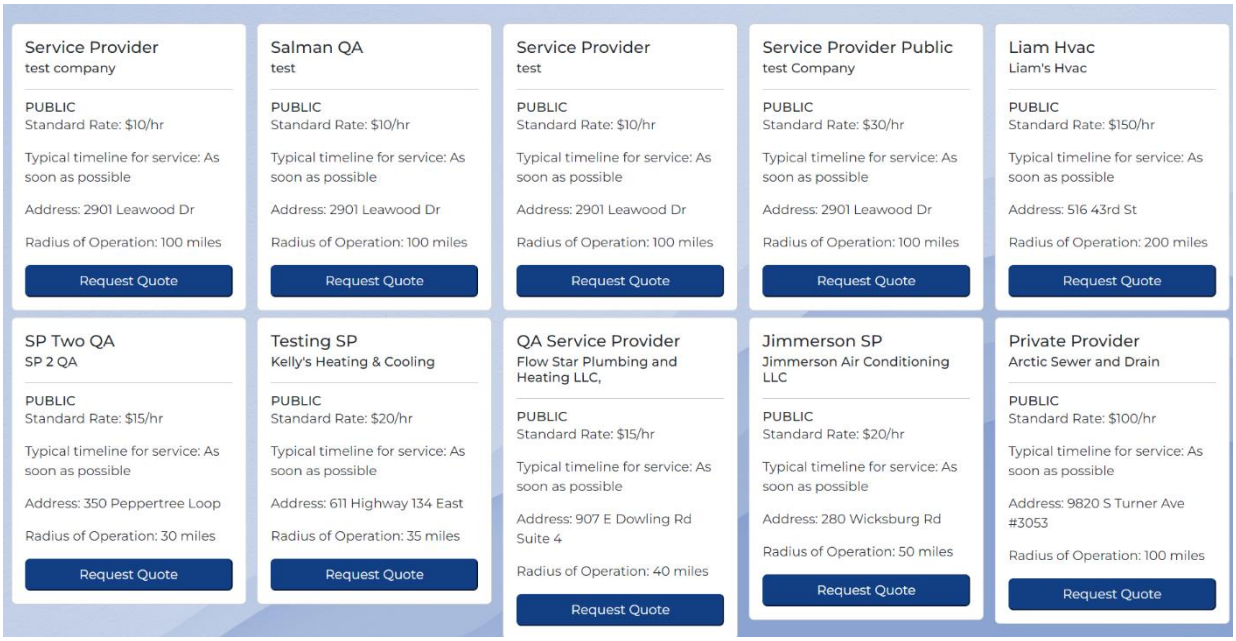


Figure 9.3.1 The initial layout of the request quote page. This design was scrapped as it was found to be too difficult to scan the cards according to price, service provider name, or radius of operation.

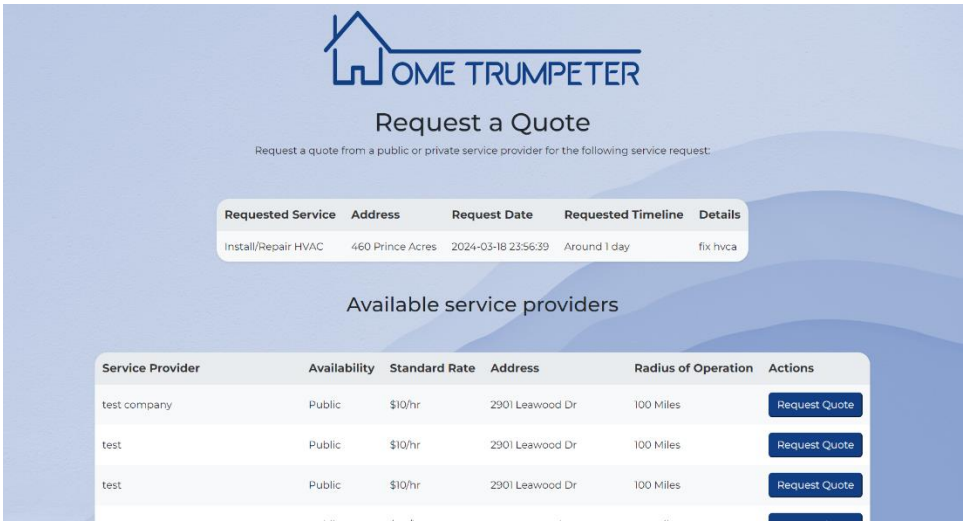
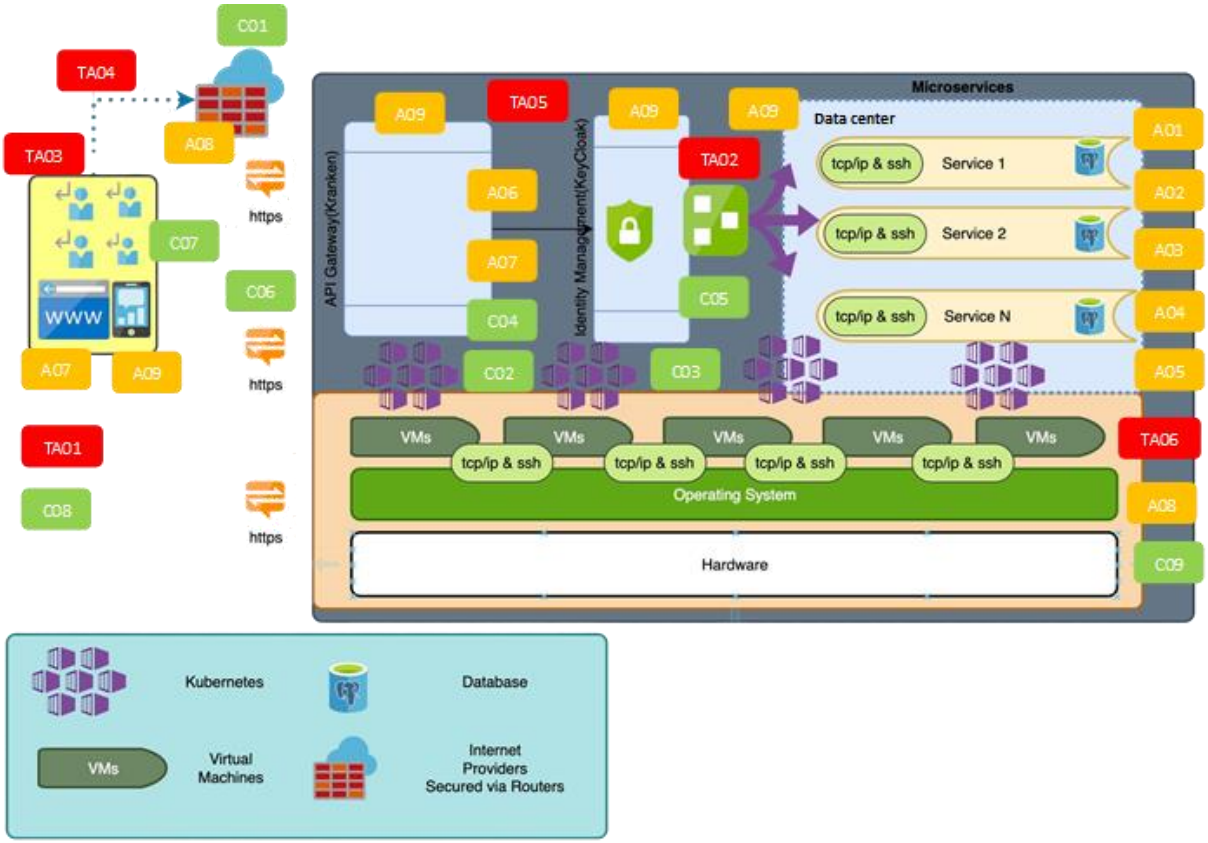


Figure 9.3.2 The final interface of the request quote page. Available service providers are formatted as a table, allowing homeowners to easily scan for a particular service provider, or the lowest price.

10. Security Design

10.1 Threat Modeling



Assets

A01: Homeowner Data

A02: Tenant Data

A03: Property Details

A04: Tickets and Tasks Details

A05: Payment Details and History

A06: Credentials

A07: Sensitive Application Data

A08: Infrastructure

A09: Applications

Threat Agents

TA01: Unauthorized External Users

TA02: Unauthorized Internal Users

TA03: Authorized Homeowners, Tenants, and Service Providers with Malicious Intent

TA04: Man-in-the-Middle Attackers

TA05: API Attackers

TA06: Viruses, Worms, Ransomware, etc.

Controls

C01: Network Security

C02: Container Security

C03: Kubernetes Security

C04: API Security

C05: Identity Management

C06: Encryption

C07: Regular Patching

C08: Monitoring and Logging

C09: User Training

Figure 10.1.1. Threat model of the HomeTrumpeter system. Microservices are run in clusters managed by OpenShift. Assets are identified in yellow tags with prefix "A0_". Threat agents are identified in red tags with prefix "TA0_" at various integration points. Controls are identified in green tags with prefix "C0_".

An initial threat model was constructed to monitor potential threat agents, attack surfaces, and find appropriate countermeasures for each. A copy of this threat model is given in Figure 10.1.1. This threat model served as the basis for the security measures taken early in the project.

As functionality was added and the project architecture evolved, new attack surfaces, assets, and control measures emerged. As such, the threat model evolved in step with the product, with the final version of the threat model given in figure 10.1.2.

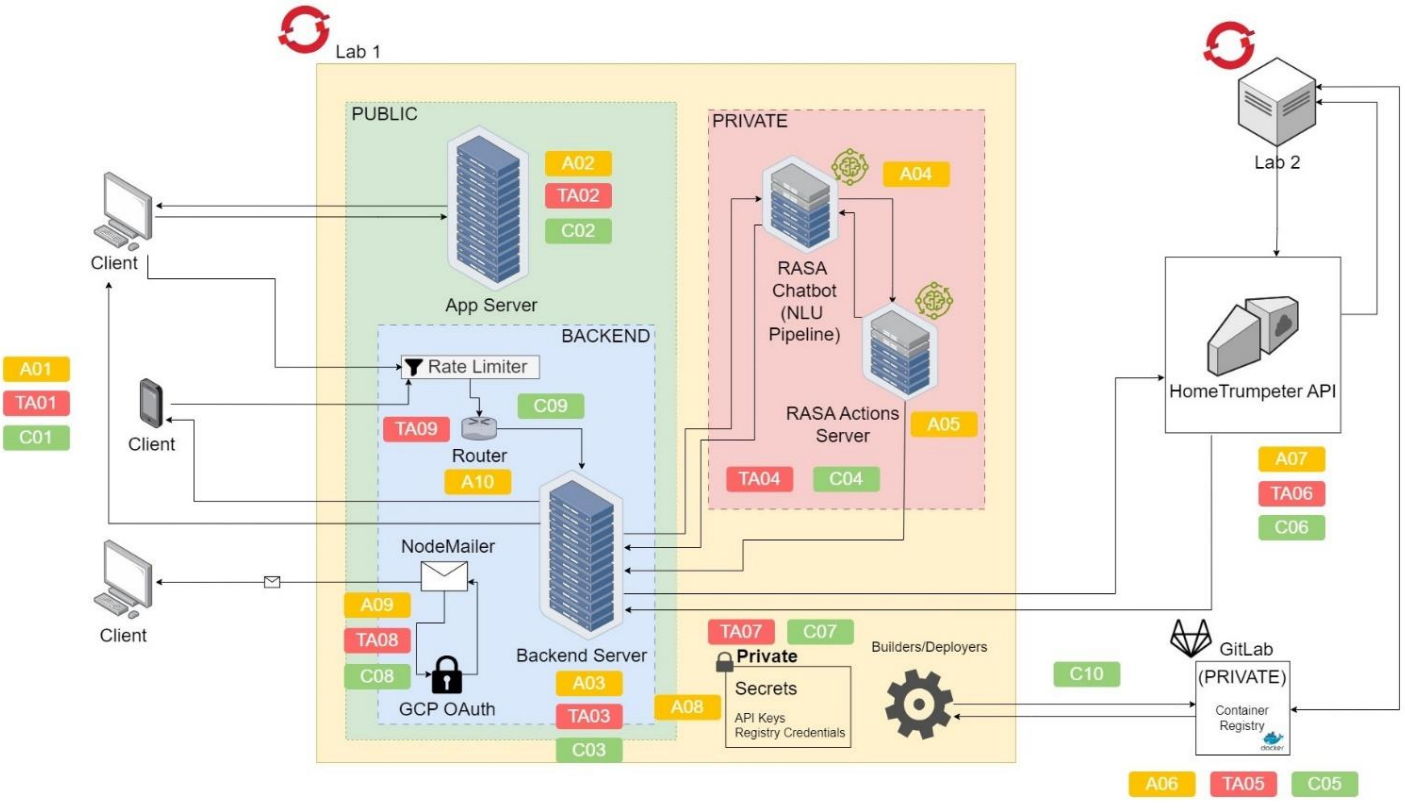


Figure 10.1.2. Updated threat model of the HomeTrumpeter (ProveIT) system.

10.2 Penetration Testing – OWASP ZAP

The penetration testing for this project was conducted using OWASP ZAP (Zed Attack Proxy). OWASP ZAP is an open-source web application security scanner. It is designed to find a variety of security vulnerabilities in web applications while they are in development and testing phases. The tool acts as a proxy between the tester's browser and the web application, allowing it to inspect and manipulate the traffic passing through. The following diagram illustrates this flow:

Figure 10.2.1 Penetration testing workflow using OWASP ZAP as a proxy between the web browser and web server. Packets are intercepted and examined for security risks, which are reported to the development team.

In this project, OWASP ZAP is being used for manual penetration testing, which includes exploring the application, understanding its flow and features, and creating or modifying requests sent by the browser to the server. The test is conducted on the application pushed to the develop branch in GitLab by cloning it to a computer (localhost). The following are screenshots from the ZAP interface:

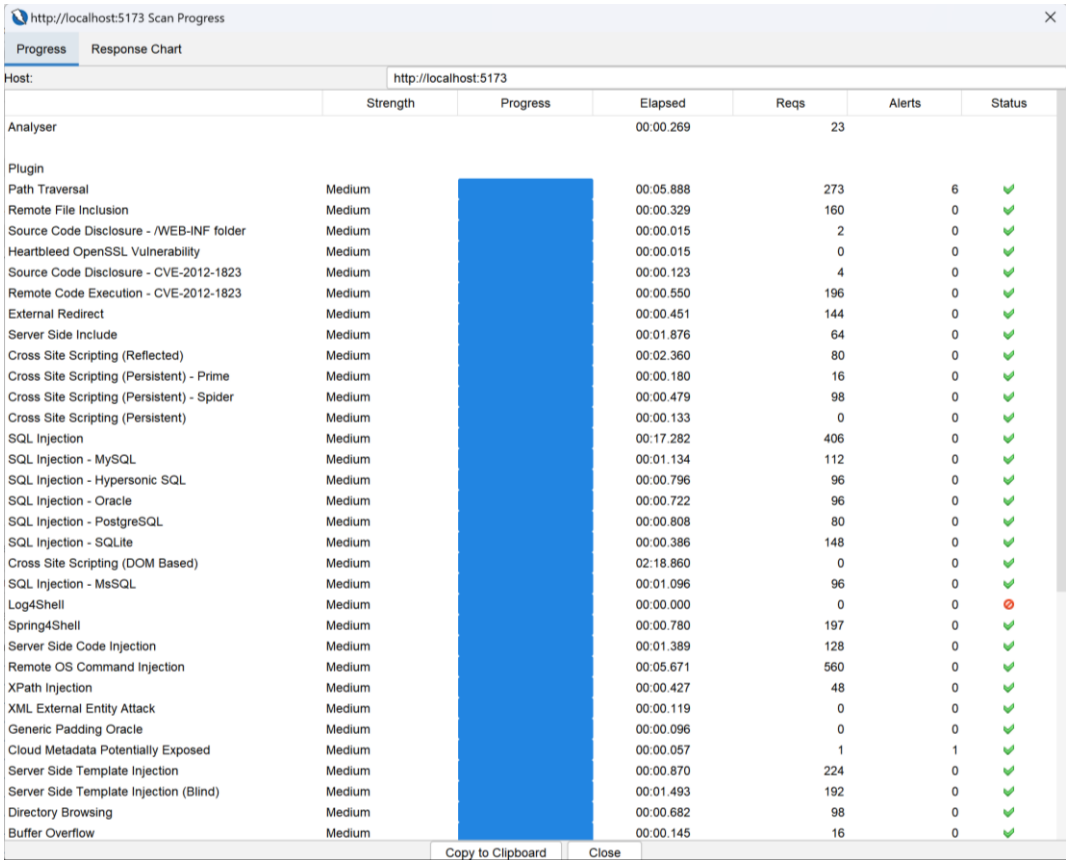


Figure 10.2.2 OWASP ZAP scan performed on the Prove IT website. Vulnerabilities such as Cross Site Scripting (XSS) and SQL Injection are tested in the scan.

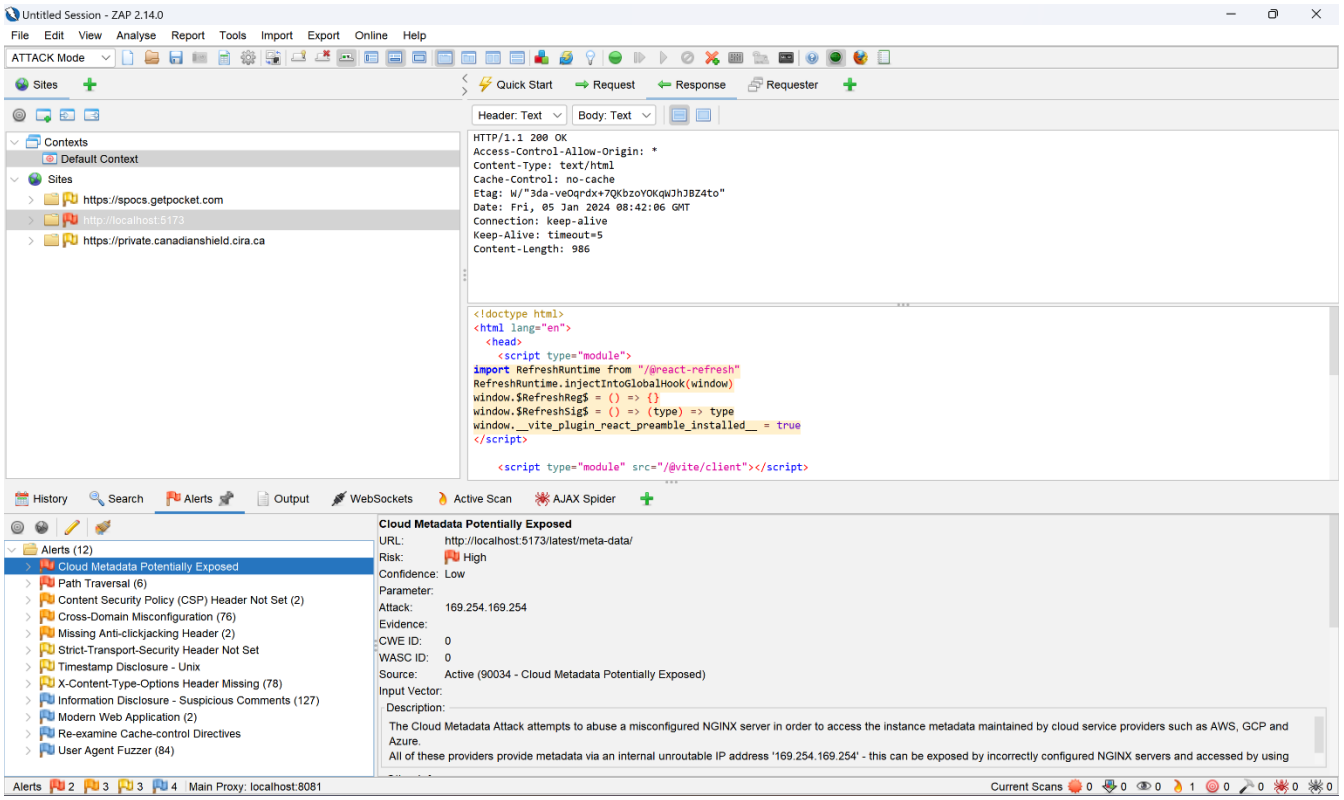


Figure 10.2.3 OWASP Zap scan results performed on an early version of the Prove IT website. Alerts show security risks in order of high, medium, low, and informational priorities. Each alert contains a description of the vulnerability and preventative measures.

However, the OWASP ZAP scan results are not absolute. There may be “false positives” which may mislead about a vulnerability that may not actually cause any threat, while in other cases it may not fully detect a vulnerability which can cause significant issues if unattended. Thus, multiple testing tools were employed and most importantly any alerts or vulnerabilities found after the test were carefully investigated by the development team so that any real threat are accurately determined and addressed. Finally, a report for the alerts and vulnerabilities in greater detail can be generated, a part of which looks like the image below:

Alert type	Risk	Count
Cloud Metadata Potentially Exposed	High	1 (12.5%)
Path Traversal	High	6 (75.0%)
Content Security Policy (CSP) Header Not Set	Medium	1 (12.5%)
Cross-Domain Misconfiguration	Medium	72 (900.0%)
Missing Anti-clickjacking Header	Medium	1 (12.5%)
X-Content-Type-Options Header Missing	Low	72 (900.0%)
Information Disclosure - Suspicious Comments	Informational	126 (1,575.0%)
Modern Web Application	Informational	1 (12.5%)
Total		8

Figure 10.2.4 OWASP ZAP scan report for an early version of the Prove IT system. Alert type indicates the attack type that may result in penetration of the system. Risks are classified as high, medium, low, and informational priorities. Count is provided to track size of the specific vulnerability relative to others.

Later in the project, the ZAP penetration was implemented into the CI/CD pipeline so that all the tests are automated, and vulnerabilities could be more efficiently addressed.

ZAP Scanning Report

Site: http://ht-website-ht-uat-coop.apps.prod.htuslab2.com

Generated on Fri, 8 Mar 2024 07:33:04

ZAP Version: 2.14.0

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	5
False Positives:	0

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	3
Missing Anti-clickjacking Header	Medium	3
Sub Resource Integrity Attribute Missing	Medium	8
Cookie without SameSite Attribute	Low	3
Permissions Policy Header Not Set	Low	4
X-Content-Type-Options Header Missing	Low	5
Information Disclosure - Suspicious Comments	Informational	5
Modern Web Application	Informational	3
Non-Storable Content	Informational	3
Session Management Response Identified	Informational	4

Medium	Missing Anti-clickjacking Header
Description	The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.
URL	http://ht-website-ht-uat-coop.apps.prod.htuslab2.com/
Method	GET
Parameter	x-frame-options
Attack	
Evidence	
Other Info	
URL	http://ht-website-ht-uat-coop.apps.prod.htuslab2.com/robots.txt
Method	GET
Parameter	x-frame-options
Attack	
Evidence	
Other Info	
URL	http://ht-website-ht-uat-coop.apps.prod.htuslab2.com/sitemap.xml
Method	GET
Parameter	x-frame-options
Attack	
Evidence	
Other Info	
Instances	3
Solution	Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
CWE Id	1021
WASC Id	15
Plugin Id	10020

Figure 10.2.5 OWASP ZAP scan report for the Prove IT system as of March 8th. These reports are generated as an artifact after an automated scan is completed in the CI/CD pipeline. Similar to the manual scan, the report classifies all the vulnerabilities according to the type and gives insight on the risk level, number of instances and details for mitigation.

10.3 Static Code Analysis - Snyk

Another tool used for analyzing code for vulnerabilities is Snyk, which is a code analysis tool used for identifying and fixing vulnerabilities in source code. It integrates into the development workflow to scan code repositories or build processes. In this project, Snyk is being used to analyze the codebase

for known vulnerabilities in its source code or dependencies. It scans the project's dependency files, like package.json in Node.js projects, to identify libraries with known vulnerabilities. Once identified, it provides information on the severity of the vulnerability, along with guidance for remediation, such as updating to a safer version of the library. This continuous monitoring ensures that new vulnerabilities are detected and addressed promptly as part of the development and deployment process.

The following is a screenshot of the Snyk extension for VS Code after scanning the code base pushed to the develop branch in GitLab:

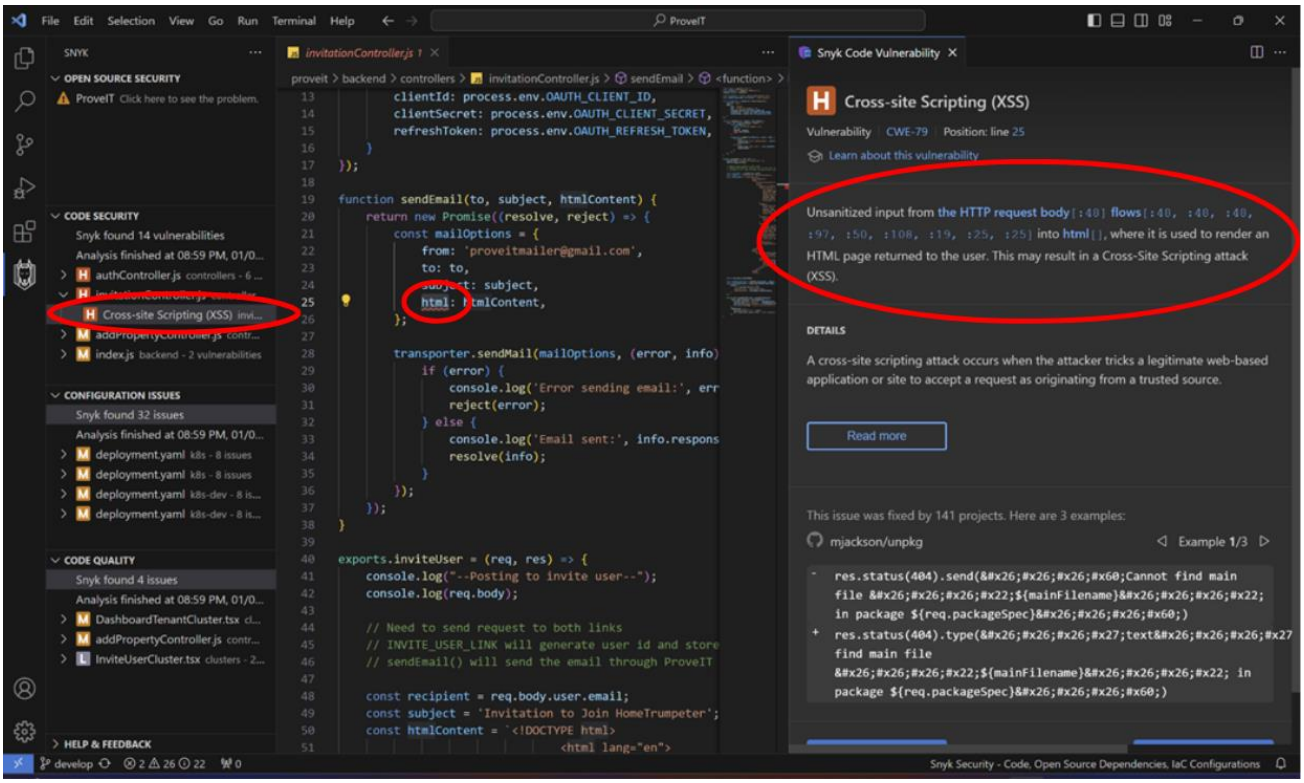


Figure 10.3.1 Snyk scan of Prove IT source code for the back-end system. Vulnerabilities are identified, along with a description of the attack and potential solutions. Vulnerability CWE-79 identified a Cross-Site Scripting (XSS) attack vulnerability a line 25 in the *invitationController.js* file. Examples of solutions from other open-source projects on GitHub are shown.

Static code analysis with Snyk was integrated into the CI/CD pipeline to ensure consistent testing with every code change. During each run of the pipeline, the repository would be scanned, a report of potential vulnerabilities would be produced, and the code would not be deployed if any critical security vulnerabilities were detected.

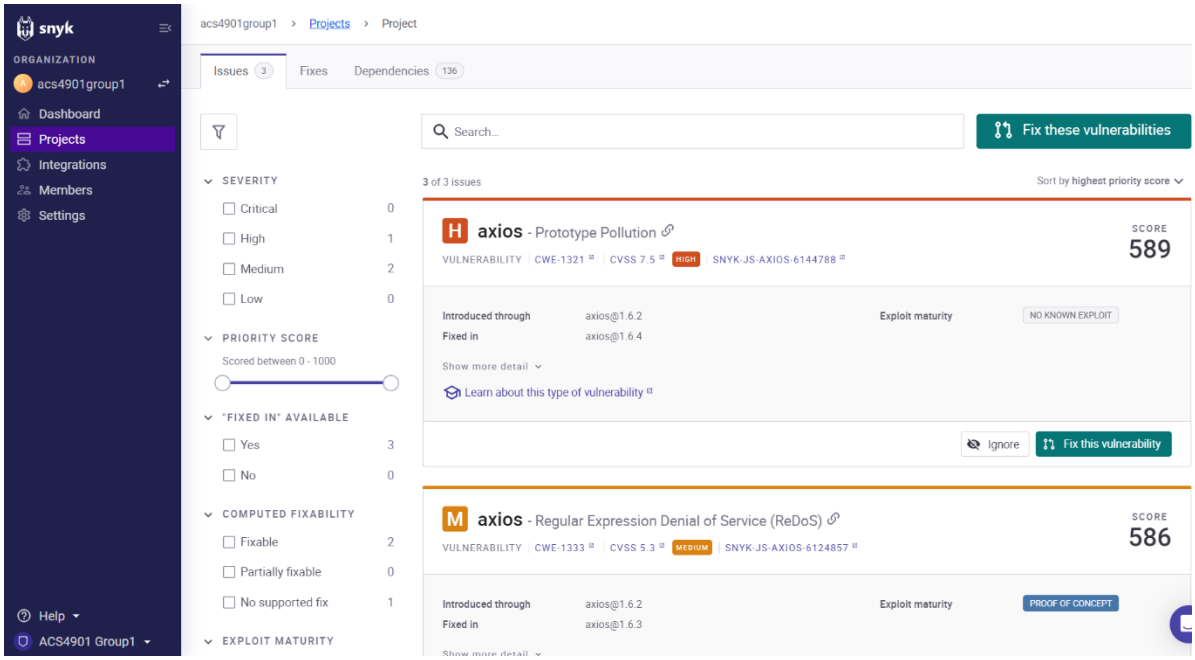


Figure 10.3.2 A screenshot of a report produced by Snyk after an automated scan in the CI/CD pipeline. Snyk has detected two vulnerabilities in the project dependency Axios. The report provides information on the severity of the exploit, and the package version where the vulnerability is fixed.

10.4 Security Controls

In addition to security testing and dependency scanning, we are implementing multiple security controls to prevent unauthorized use of our system. The API key given to us by HomeTrumpeter is securely stored in an environment variable in our deployment architecture. This prevents threat agents from gaining access to HomeTrumpeter's system directly if they were to gain access to Project Prove IT's source code. To further reduce the likelihood of our system being used as an attack surface towards HomeTrumpeter's API, we have implemented rate limiting in our backend proxy to prevent the effects of a DDOS attack on our system from propagating to HomeTrumpeter's infrastructure. Our backend proxy system is constructed such that only the HomeTrumpeter API endpoints that are necessary for our application to function are exposed. This measure will help prevent escalation of privilege attacks by not allowing threat agents to access sensitive administrator-level API endpoints. HomeTrumpeter endpoints returning personal data require authorization using a JSON Web Token issued by HomeTrumpeter. These tokens are issued, revoked, and expired by HomeTrumpeter's existing security architecture. Since these tokens expire after a fixed amount of time, they are effective at preventing session hijacking attacks. If an unauthorized user were to discover an old open session of the Prove IT app, they would not be able to exploit the session token to gain access to information, as it would be

expired. All communication between the Prove IT system and HomeTrumpeter's API is secured with HTTPS.

10.5 Risks and Mitigation Strategies

Identifying potential risks and challenges and outlining a plan to monitor, assess, and mitigate them is a crucial component in the modern software development approach. The following are some of the risks that were identified, along with their corresponding monitoring and assessment techniques and mitigation plans that we implemented:

Risk	Monitoring	Assessment	Mitigation
Technical Challenges	Monitor development progress and track integrated system performance	Review technical task status and assess task completion against schedule	Schedule extra time for troubleshooting, engage HomeTrumpeter's technical team, conduct thorough testing
Scope Creep	Monitor project progress and review product backlog	Assess requirement changes and review stakeholder feature requests	Implement formal change request process, prioritize new requests for future sprints
Resource Constraints	Monitor team workloads and track external dependencies	Evaluate resource impact on timelines	Manage commitments, reallocate tasks as needed, communicate with stakeholders about external delays

Security Vulnerabilities	Conduct security assessments and review security measures	Identify and evaluate security vulnerabilities	Perform regular security testing and code reviews, use automated security checks in CI/CD, penetration testing, implement strong authentication measures
External Dependencies	Track availability and performance of external services	Assess impact of external service disruptions or changes	Develop contingency plans, implement fallback mechanisms, maintain communication with third-party providers
Scope Uncertainty	Seek clarification from HomeTrumpeter and stakeholders	Evaluate gaps in stakeholder requirements	Maintain open communication with HomeTrumpeter, hold regular requirement clarification meetings, document all requirements
Time Management	Monitor sprint progress and review project timeline	Assess task completion within timeframes	Adjust sprint planning, allocate additional time as needed, refine sprint planning process

11. Deployment Architecture

11.1 Infrastructure as Code

Setting up infrastructure as code (IaC) allows the development team to maintain a consistent testing environment throughout the development process. All environments and dependencies are provisioned automatically through scripts that setup and maintain docker containers on the OpenShift platform.

Separate namespaces are used throughout the pipeline: testing, uat, and prod.

11.1.1 Testing – Development Testing

The testing namespace allows individual developers to test their build in an isolated environment. The developers will use test data to secure all edge-cases before merging to other branches.

11.1.2 UAT – User Acceptance Testing

The UAT namespace allows for integration testing after passing all build tests. The most up-to-date data and services/dependencies will be maintained in this namespace. This enables the users and developers to utilize the system in a non-critical environment while performing real-world tasks. Feedback will be used to determine whether the system is ready to be pushed to production.

11.1.3 Prod – Production

The prod namespace holds the latest release of the system. This should ensure the system is 100% functional and ready for users to use.

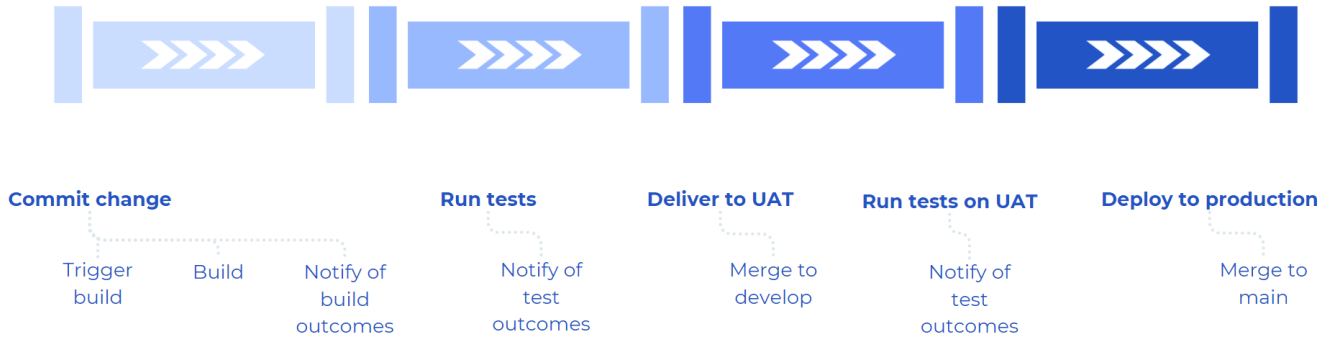


Figure 11.1.4 Deployment diagram for the CI/CD pipeline of Prove IT. Successful build of committed changes are tested within a testing namespace. Successful testing causes the commits to be delivered to the user acceptance testing (UAT) namespace for final integration testing prior to deployment to production, which merges the develop branch into the main branch.

11.2 CI/CD Pipeline

We have integrated continuous integration (CI) and continuous deployment (CD) into our development workflow using the GitLab CI/CD pipeline. This powerful tool streamlines various tasks, including building our product and deploying it onto the OpenShift platform. Here's how our pipeline operates:

Feature Development and Bug Fixes: When our developers work on new features or bug fixes, they branch out from the main repository. After coding is completed, they push their changes from their local environments to the GitLab repository.

Automated Building and Testing: GitLab automatically triggers the build process and runs a battery of tests. We have an array of testing processes that ensure the code quality, and we'll dive deeper into these tests shortly.

Test Failure Handling: If the code fails any of the tests, the developer receives feedback and is required to address the issues promptly.

Merging to Development Branch: Once the code is fixed and passes the tests, it is pushed back to be merged into the 'develop' branch. This branch acts as an intermediary staging area before deployment to the main branch.

Develop Branch Verification: When the code is merged into the 'develop' branch, it undergoes another round of building and testing (double-checking) to ensure its stability. An image is then released for deployment.

Deployment: The deployment to OpenShift is managed by a dedicated 'deploy' job. This deployment is distinct from the live deployment and allows Quality Assurance (QA) teams and other stakeholders to review the software to ensure it functions as expected.

Main Branch Update: Following the successful review of the deployment in the 'develop' branch, it is merged into the 'main' branch. This process implements a Rolling Update strategy, seamlessly replacing the previous public deployment with the new one.

12. Testing Strategy

12.1 Unit Testing

We employed Jest and React Testing Library for basic unit testing. These tests validated individual components, functions, or units of code to ensure they work correctly. This process ensured that code changes were verified against expected behavior. We employed different test suites which contains multiple unit tests related to a specific module.

12.2 Integration Testing

Integration testing verifies the interactions and compatibility between different components of our application. It ensures that various parts of the system work harmoniously together, simulating real-world usage scenarios. The tests suites we have in our project are written in a way that they perform unit testing on single components and at the end, we mock the API calls to check the integration of that module with the rest of the project.

12.3 Regression Testing

Regression testing is crucial in CI/CD pipelines. It involves re-running previously executed tests to detect and prevent the introduction of new bugs or issues as code evolves. It helps maintain the stability of our application. Because we have unit tests setup in the project, the pipeline re-runs all the tests with a single command and every component in the application is checked. Because our develop branch is a merge of every feature branch, so there is a possibility that someone could have changed the logic of a module, and these tests ensure the stability of the older components.

12.4 User Acceptance Testing (UAT)

User acceptance testing is the last stage of testing where change requests are accepted. Users perform real-world tasks in real-world environments and provide feedback. This confirms all user needs are met before pushing to production.

12.5 End-to-End Testing (E2E)

For E2E testing, we use Selenium. E2E tests simulate user interactions with the application, checking the entire flow from start to finish. This testing approach ensures that all components of the system work together as expected. We have created some automated workflows which mimics the actual user interaction and provide us an easier way of performing the same manual tasks. These tests can be run on a machine with GUI however in the pipeline we use the headless browser drivers.

13. Release Planning

13.1 Agile Release Plan

Adhering to the Scrum methodology, our team strived to continuously deliver working software to our stakeholders throughout the project. Our release plan was closely tied to the product backlog, as we provided a new release of the product to our stakeholders at the conclusion of each sprint. At each sprint planning meeting, the development team and stakeholders collaborated to select a list of features to be included in the next release. This list formed the sprint backlog.

12.0	**Sprint - 9 - Service Request Creation II**			**24**			
12.1	Sprint 9 - Planning / Sprint 8 Demo	ALL		0	18 Jan 2024 6:30PM	18 Jan 2024 7:30PM	Not Started
12.2	Send Proposal	Unassigned		2			Not Started
12.3	View Proposed Quotes	Unassigned		3			Not Started
12.4	View Proposed Quote Details	Unassigned		2			Not Started
12.5	Approve Proposed Quote	Unassigned		2			Not Started
12.6	Proposal Notification	Unassigned		2			Not Started
12.7	Proposal Approval Notification	Unassigned		2			Not Started
	Mark Job as In Progress, Completed	Unassigned		3			Not Started
12.8	Service Request - Testing	Unassigned		8			Not Started
13.0	**Sprint - 10 - Background Check I**			**26**			

Figure 13.1.1 The sprint backlog for Sprint 9. The list of tasks in this backlog would be presented to stakeholders in a sprint planning meeting. At the conclusion of the sprint, all newly developed features would be delivered to stakeholders in a new release.

Following the sprint planning meeting, individual tasks from the sprint backlog were assigned to team members. Each member would develop their feature in their own feature branch. Upon completion of the feature, the branch would be deployed to the development environment. This deployment would be tested by the developer to ensure the feature was functional in the OpenShift environment.

After a feature was deployed to the development environment and tested, it could be deployed to the UAT environment. This environment would contain all new features developed during the current sprint. User acceptance testing would be carried out at the sprint demo meeting using this deployment. If the demonstrated changes were accepted by HomeTrumpeter stakeholders, the changes would be pushed to the production environment, constituting a new release of the product.

13.2 Definition of Done

An important aspect of release planning in agile methodology is agreeing upon a "Definition of Done". The definition of done is the set of conditions that must be met for individual work items to be considered completed. Our team's definition of done included 5 conditions:

1. The feature branch must be deployed to the development environment and tested by the developer.

2. The changes to the develop branch must be reviewed by the QA Engineer.

3. The feature must be merged to the develop branch using a GitLab merge request.

4. The development branch must pass all automated tests and successfully deploy to the UAT environment with the feature included.

5. The feature must be tested in the UAT environment by the QA Engineer.

Once these conditions were met for a particular feature, the corresponding backlog item would be marked completed, and the feature would be included in the next release of the product.

14. Documentation

14.1 Documentation in Agile Methodology

The agile manifesto provides a set of values and principles meant to increase the efficiency and quality of the software development process. One of these core values is valuing working software ahead of comprehensive documentation. The manifesto states that there is value in documentation, but the delivery of working software should be prioritized. As we strived to follow agile methodology, documentation would only be produced if it facilitated the delivery of working software, or better enabled collaboration with stakeholders. Such documentation includes the system architecture diagram, domain class diagram, threat modelling diagram, and user story diagram seen previously in this document.

14.2 Product Backlog / WBS

The most important document to the scrum framework is the product backlog. This document lists all tasks that need to be completed throughout the project, the sprint they are planned to be completed, and the team member assigned to them. The product backlog is like the work breakdown structure used by the waterfall approach, as it combines the tasks, resources, and timelines of the project into a single document. To enable adaptability to change, the product backlog was treated as a living document, with items being updated, added, or removed throughout the project. This document facilitated collaboration with HomeTrumpeter stakeholders, as the backlog would be shared and updated collaboratively every two weeks in the sprint planning meeting. The product backlog is available in the appendix.

14.2 Developer Documentation

Prove IT developers have access to documentation that outlines coding and integration standards. Coding standards enforce basic syntactic patterns to guarantee a uniform code base. Functional programming guidelines are also in place for ReactJS development to increase reusability of various components. A workflow for Git integration is provided to embrace the idea of "infrastructure as code", enabling the team to easily test functionality and integration prior to delivering the source code. This documentation facilitates delivery of working software by offering a source of truth for the agreed-upon coding standards and development processes.

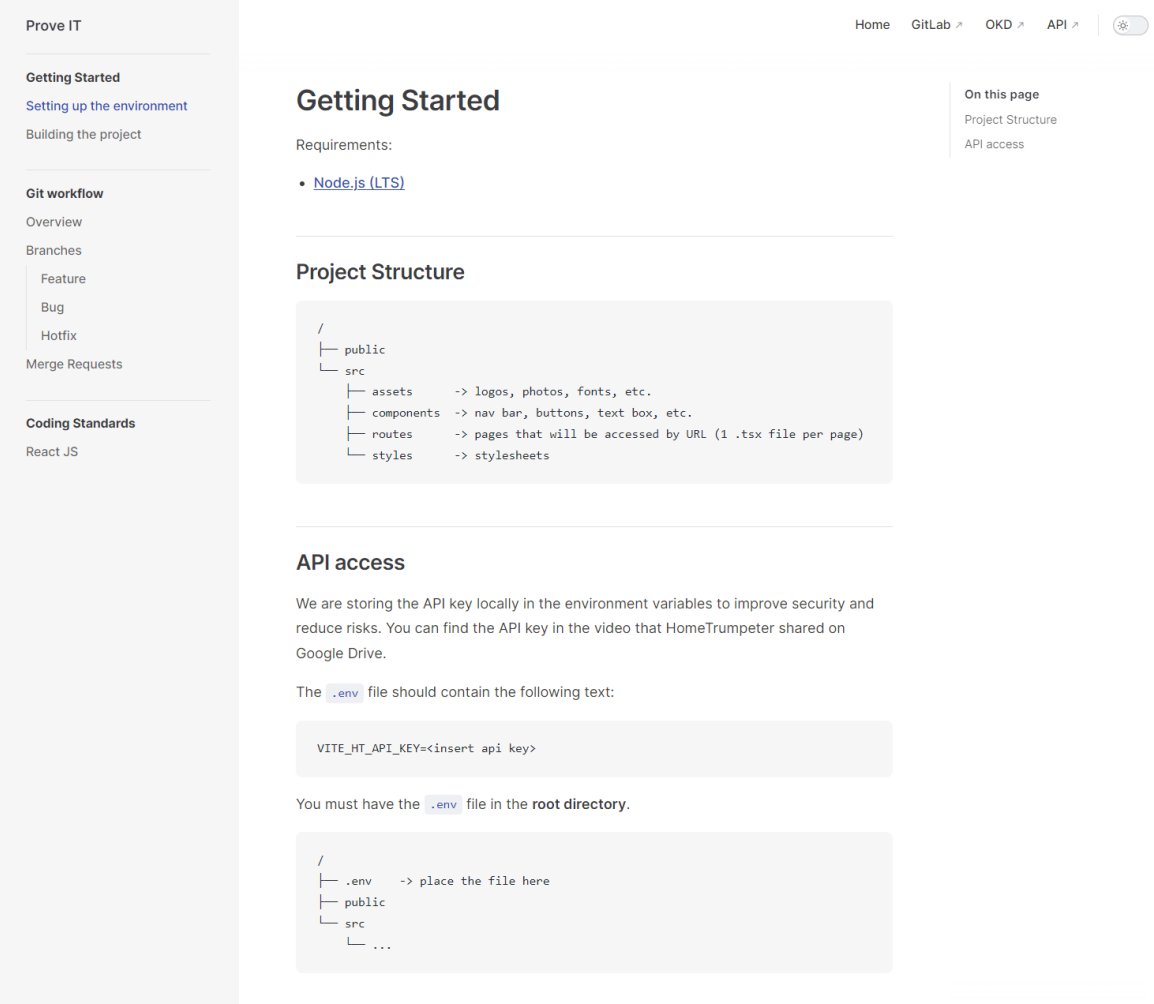


Figure 19. Prove IT development guidelines for developers. Instructions are provided for setting up an environment for development. CI/CD Git workflow is included for branching and merging. Coding standards for naming conventions, file structure, and component usage are established under the React JS sections.

15. Lessons Learned

The senior systems development project has been an excellent opportunity to learn hands-on with industry-level technology, as well as industry-standard processes for software development. It was a great privilege to be among the pioneers of Agile methodology in the project course.

15.1 Adjustments to the Agile Process

Consistent face-to-face communication between the development team and stakeholders is a key principle of Agile methodology. However, facilitating such communication between six students and multiple stakeholders with different schedules was challenging at times. To easier facilitate the agile ceremonies, we decided to combine our sprint demo and sprint planning into a single meeting. In each demo/planning meeting, the development team would first demonstrate the progress made in the previous sprint. Following the demo, the team would solicit feedback and plan the next sprint collaboratively with the stakeholders. By combining these two important phases into a single 60-minute meeting, the amount of schedule co-ordination required by both the development team and stakeholders was significantly reduced.

The development team also found success in adopting a flexible approach to our regular scrum meetings. It was paramount to have a regular check-in where team members could share their progress, and plans for the upcoming days. Whenever possible, the team would meet face-to-face as recommended by the Agile manifesto. As students with unpredictable availability between class, work, and extracurricular commitments, meeting face-to-face was not always possible. By expanding our scrum practices to include meeting synchronously or asynchronously on Microsoft Teams, we were able to consistently benefit from the benefits of the scrum, despite our different schedules.

15.2 Development with HomeTrumpeter API – Points of Strength

By leveraging HomeTrumpeter's API, the team was able to develop a functional property management platform in 28 weeks. HomeTrumpeter's API accelerated our development process, allowing us to abstract processes like authentication, data persistence, and address verification. By having these services available through an external API, we were able to build a stateless backend proxy instead of an extensive database architecture. This saved us a significant amount of time, allowing us to focus on development of our front end. The responses from HomeTrumpeter's API endpoints had a consistent format, making the development process easier. HomeTrumpeter's API offered an extensive suite of services. We did not require use of all these endpoints for the requirements of our system. Instead, we were able to expose only the necessary endpoints and add additional security measures through our

backend proxy. We would recommend a similar approach be taken for any future teams building an external application leveraging HomeTrumpeter's API.

The consistent availability of HomeTrumpeter team members, and their willingness to help were paramount to the success of the project. Whenever the development team experienced issues with HomeTrumpeter's API, stakeholders were consistently available to answer questions or pass them off to HomeTrumpeter's developers. This availability ensured that we did not spend extensive amounts of time debugging the "black box" API. We recommend that this availability of communication be maintained for any development teams leveraging HomeTrumpeter's API in the future.

15.3 Development with HomeTrumpeter API – Areas of Improvement

By relying on HomeTrumpeter's API for essential services such as authentication, our project is entirely reliant on the availability of the API. As HomeTrumpeter is itself growing and expanding its services, changes were constantly being made to their API. At times, this would be disruptive to the development of project Prove IT, as downtime of HomeTrumpeter's API would make our system inoperable. All deployments of Prove IT were tied to HomeTrumpeter's QA API. Future projects leveraging HomeTrumpeter's API could benefit from a private API deployment insulated from changes being made to HomeTrumpeter's flagship system.

Throughout the project, the development team has collaborated with HomeTrumpeter to resolve issues with API endpoints, and properly denote deprecated endpoints. HomeTrumpeter has requested a list of endpoints still in need of attention. This table is given in figure 15.3.1 below.

Endpoint	**Issue**
GET /job/active-jobs/	This endpoint currently only returns active jobs when logged in as a service provider. This endpoint could be more useful if expanded to Homeowners and Tenants. This would allow active jobs to be tracked through a dedicated endpoint rather than through the service requests endpoint.
POST /service-request/search-result GET /service-request/search-result	In our experimentation with these endpoints, they both return massive JSON objects that are

	too large to process efficiently. Perhaps they are deprecated?
GET /service-provider/approve-proposal/{id}	Based on its functionality, this endpoint should use the POST method instead of GET.

Figure 15.3.1 A table of endpoints in HomeTrumpeter's API that need attention or could be improved.

16. Stakeholder Feedback

16.1 Summary of Feedback

Stakeholder feedback was very positive both throughout the project and upon its completion. HomeTrumpeter stakeholders were pleased at the breadth of technology that the development team was able to successfully learn and employ during the project. At each sprint meeting, HomeTrumpeter stakeholders commended the development team's work ethic, willingness to learn and dedication to success. Initially, HomeTrumpeter expected the team to deliver either a web or mobile application and conduct preliminary research towards an AI feature. The development team was able to exceed these expectations, developing both a web application and mobile application, as well as an AI chatbot with integrated functionality.

16.2 Adjustments Made

As an Agile team, it was important to adapt promptly to change and stakeholder feedback. An opportunity to make such an adjustment was presented early in the project. Our initial design was a frontend-only application that would communicate directly with HomeTrumpeter's API. HomeTrumpeter stakeholders expressed concern that this would compromise security of their API and expose their API key. As such, the development team reworked our vision for the product to include a backend proxy, to act as an intermediary between the frontend application and HomeTrumpeter's API. Thanks to the flexibility offered by Agile, we were able to implement this change. The backend proxy allowed us to clean up data returned from the backend, implement rate limiting, and expose only the necessary endpoints of HomeTrumpeter's API. Thanks to HomeTrumpeter's feedback, and our ability to respond to change, the quality of our final product was greatly improved.

17. Agile Methodology Assessment

Project Prove IT was among the first projects developed in the senior systems development course using the Agile methodology. As such, this project has been a proving ground not only for HomeTrumpeter's API, but for the agile methodology and its suitability for the senior systems development project. The strengths of the agile methodology have made it a great fit for the needs of our specific project, and for the needs of the senior systems development course in general.

17.1 Emphasis on Collaboration

A key principle of agile development is consistent collaboration between the development team and business stakeholders. The Scrum framework encourages this collaboration through regular meetings where progress, challenges, and ideas can be shared. These pillars of the agile scrum methodology made it a great fit for our project. A core objective of the project was to identify any points of difficulty in developing an independent system leveraging HomeTrumpeter's API. The bi-weekly sprint planning meetings provided an ideal setting for the development team to share challenges experienced with the API. HomeTrumpeter could also offer potential solutions to these challenges during the regular meetings. With these consistent touch points between the developers and stakeholders, issues identified in the API could be addressed, and questions regarding the API's usage could be quickly answered.

17.2 Facilitation of Learning

An emphasis on continuous learning is another strength of agile that make it a good fit for the project course. As a newly formed group of students developing a software project together for the first time, there was a lot to learn for our development team. Most of us were unfamiliar with the technology we would be using throughout the project. By allowing us to start hands-on development as early as possible, we were able to gain valuable experience early in the project. This experience would help us understand what would be possible for our team to accomplish with the given time and resources, allowing us to better refine and prioritize goals and requirements for the project.

The scrum framework allowed us to define requirements at a high level at the beginning of the project and refine them collaboratively with the stakeholders at each sprint planning meeting. This approach allowed the project plan to evolve as our understanding of the technology and requirements grew. This flexibility prevented unnecessary work in planning requirements that would eventually change and allowed us the freedom to implement new ideas as the project progressed. In an environment such as the project course with young, growing minds, it is important to facilitate an environment

where change is accepted, and new ideas can be explored. In all, the agile scrum methodology is a great fit for our project, and the senior system development course in general thanks to its emphasis on collaboration, learning, and adaptability to change.

18. Recommendations

18.1 Start Development Early!

Early development is a key recommendation of Agile methodology. In the context of the project course, an early start to development is even more beneficial. Project groups will likely be required to work with several unfamiliar technologies and frameworks as part of their project. In our experience, the best approach to learning new technologies was through hands-on development. Official documentation and training manuals can be helpful to get started with a new technology, but hands-on experimentation will best develop the specific skills needed within the scope of the project. Early development is also the best way to understand rate at which development work can be completed. Such an understanding will help future teams in their continuous planning process and help them to set realistic goals for their respective projects.

18.2 Be Well-Rounded!

The roles assigned to our team members at the beginning of the project were Project Manager, Technical Leader, Lead Designer, Lead Developer, QA Engineer, and DevSecOps Engineer. Each of these roles bring a much-needed skillset to the project. Rather than restricting each team member's work to tasks related to their specific role, we suggest that every member of the team be involved in each of these areas. Each team member can be a leader in their specific domain and should be supported by all members of the team as required. If all members of the team understand all aspects of the project, more team members can be allocated to security, quality, design, or development as necessary. Because this is a learning project, members shouldn't stay restricted to their domain. They should explore what other teammates are doing and should have a solid understanding of the system. This will lead the team to the stairs of success.

18.3 Dream Big!

Embark on this project course with an open heart and a daring spirit. Remember, there are no limits to what you can achieve. Trust in your abilities and be willing to embrace new technologies, challenges,

and innovative ideas without fear. The scope of your project should not confine your ambition. Instead, see it as a starting point from which you can expand and explore uncharted territories.

Understand that no one will hand you all the knowledge you need on a silver platter. The journey ahead requires you to be proactive, seeking out information and learning independently. This course is an opportunity for you to discover your potential and the strengths of your team members. Our experience taught us that what might seem overwhelmingly ambitious at first—like integrating web and mobile applications with machine learning components—can indeed be achieved through perseverance and teamwork.

So, dream big! If there's something you believe should be part of your project, pursue it with conviction. Don't let the fear of failure hold you back. Failures are stepping stones to learning and ultimately to success. This course is more than just a project; it's a chance to push your boundaries and realize your dreams. Go beyond what's expected and create something truly remarkable.

19. Conclusion

19.1 Overview

The Senior Systems Development Project has been an experience unlike any other course offered at the University of Winnipeg. Such an opportunity to develop a system for a real-world client has been an exceptional challenge, learning experience, and opportunity for growth. The requirements of the project have forced each member of the team out of their classroom comfort zone, and into the real-world challenges of software development. Our experience in this course has equipped us not only with technical knowledge, but also with the skills of teamwork, time management, and professionalism that will be invaluable in our future careers. No other academic experience could so effectively endow us with these capabilities.

19.2 Special Thanks

The success of this project would not have been possible without the help of several outstanding individuals. We as the development team would like to take this opportunity to express gratitude to some of these individuals for their support.

We would like to thank our industrial mentor, Gabriel O for the technical guidance he provided throughout the project. Gabriel's extensive technical knowledge was an invaluable resource to the success of the project. Gabriel provided feedback and guidance ensuring that we were on the right path with our technical approach to the wide range of technologies used as part of the project. His efforts in helping us understand new technologies will help us beyond the project, as we will carry this knowledge into our respective careers.

We would like to thank our primary HomeTrumpeter stakeholder, Adeyinka for his continuous support throughout the project. Adeyinka took time out of his busy schedule as a VP of HomeTrumpeter to attend every sprint demo and planning meeting throughout the year. His consistent positive feedback and belief in our abilities were exceptional motivators throughout the long and challenging project. His availability to answer questions and provide direction as the primary point of contact between HomeTrumpeter and the development team were invaluable to the success of the project.

Finally, we would like to thank our IS Director / Agile Coach, Dr. Victor Balogun. Dr. Balogun spearheaded great changes to the project course. He led the first agile projects to ever be completed as part of the Senior Systems Development Project, with great success. Dr. Balogun expects great things from the students under his guidance. In so doing, he has extracted a great, successful effort from each member of the agile teams. We thank him for his tireless efforts to ensure that the first agile projects not only met but exceeded expectations.

19.3 Final Remarks

The Senior Systems Development Project has been an exciting, challenging, and irreplicable experience. Such an opportunity to collaborate with talented and experienced industry professionals has left each student better equipped to enter their future career. The technical and non-technical skills acquired throughout the project will greatly serve us all in whatever path we take in our lives. As we reflect on the accomplishments of the past six months, we would like to express our gratitude to all those who worked so hard to make this project possible, as well as our excitement for the future, as this project has opened countless doors for all of us.

Appendix A: Completed Product Backlog

Task Number	Task Name	Resource	Duration	Story Points	Start	Finish	Status
1.0	**User Stories**			**11**			
1.1	Identify Key Stakeholders	Project Team		2	06 Sep 2023 1:00PM	06 Sep 2023 2:15PM	Done
1.2	Form Project Team	Project Manager		1	06 Sep 2023 1:00PM	06 Sep 2023 2:15PM	Done
1.3	Project Proposal	Project Team		8	06 Sep 2023 1:00PM	25 Sep 2023 11:59PM	Done
2.0	**Product Backlog**			**4**			
2.1	Create Product Backlog	Product Owner		3	13 Sept 2023 1:00PM	30 Sept 2023 12:00PM	Done
2.2	Assign Story Point Estimations	Product Owner		1	27 Sept 2023 1:00PM	1 Oct 2023 12:00PM	Done
3.0	**High Level Sprint Planning**			**3**			
3.1	Create Project Plan	Project Manager		3	27 Sept 2023 1:00PM	06 Oct 2023 11:59PM	Done
3.2	Approve Project Plan	Stakeholders		0			Done
4.0	**Sprint - 1 - Infrastructure and Learning**			**4**			
4.1	Sprint 1 - Planning Meeting	ALL		0	05 Oct 2023 6:00PM	05 Oct 2023 7:00PM	Done
4.3	Login Gateway Wireframe	Usmaan		2	05 Oct 2023 7:00PM	12 Oct 2023 10:00AM	Done
4.4	Add Property Wireframe	Usmaan		2	05 Oct 2023 7:00PM	12 Oct 2023 10:00AM	Done
5.0	**Sprint - 2 - Login with API Portal I**			**27**			
5.1	Sprint 2 - Planning / Sprint 1 Demo	ALL		0	12 Oct 2023 6:00PM	12 Oct 2023 7:00PM	Done
5.2	Develop CI/CD Pipeline - Simple Deployment	QA Engineer		3	05 Oct 2023 7:00PM	16 Oct 2023 7:30PM	Done
5.3	Homeowner/Manager Create Account	Lead Programmer		8	18 Oct 2023 12:00PM	25 Oct 2023 12:00PM	Done
5.4	Homeowner/Manager Login	Technical Lead		5	12 Oct 2023 6:00PM	18 Oct 2023 12:00PM	Done
5.6	Prototype - Create Account/Login	Product Owner		1	12 Oct 2023 6:00PM	13 Oct 2023 12:00PM	Done
5.7	Prototype - Add Property	Product Owner		1	13 Oct 2023 12:00PM	13 Oct 2023 6:00PM	Done
5.8	Prototype - Invite Tenant	Product Owner		1	13 Oct 2023 6:00PM	16 Oct 2023 6:00PM	Done
5.9	Prototype - Invite Service Provider	Lead Designer		1	12 Oct 2023 6:00PM	21 Oct 2023 12:00PM	Done
5 10	Prototype - Create Service Request	Lead Designer		1	12 Oct 2023 6:00PM	21 Oct 2023 12:00PM	Done
5 11	Research Testing Requirements and Tools	DevSecOps		3	16 Oct 2023 12:00PM	20 Oct 2023 12:00PM	Done
5 12	Research and Select Tech Stack	Technical Lead		3	12 Oct 2023 6:00PM	18 Oct 2023 2:00PM	Done
6.0	**Sprint - 3 - Login with API Portal II**			**37**			
6.1	Sprint 3 - Planning / Sprint 2 Demo	ALL		0	26 Oct 2023 6:30PM	26 Oct 2023 7:30PM	Done
6.1.1	Sprint 2 Retrospective	Project Team		0	26 Oct 2023 7:30PM	26 Oct 2023 8:00PM	Done
6.2	Protect Routes	Technical Lead		5	26 Oct 2023 7:30PM	27 Oct 2023 12:00PM	Done
6.3	CI/CD Pipeline - Promotion of Environments	QA Engineer		8	26 Oct 2023 7:30PM	6 Nov 2023 12:00PM	Done
6.4	Implement Unit Testing Framework	QA Engineer		5	26 Oct 2023 7:30PM		Done
6.5	CI/CD Pipeline - Implement Security Scans	QA Engineer/DevSecOps		3	26 Oct 2023 7:30PM	31 Oct 2023 12:0PM	Done
6.6	Homeowner - Dashboard Skeleton	Lead Developer		3	26 Oct 2023 7:30PM	3 Nov 2023 12:00PM	Done
6.8	Identify Abuse Cases	DevSecOps		2	1 Nov 2023 12:00PM	8 Nov 2023 12:00PM	Done
6.9	Threat Modelling	DevSecOps/Lead Design		3	1 Nov 2023 12:00PM	8 Nov 2023 12:00PM	Done
6.1	Homeowner - Verify Phone Number	Project Lead		2	26 Oct 2023 7:30PM	1 Nov 2023 12:30PM	Done
6.1	Select Account Role	Project Lead		1	30 Oct 2023 12:00PM	1 Nov 2023 12:30PM	Done
6 10	Login API - Testing	QA Engineer		5			Done
7.0	**Sprint - 4 - Tenant Onboarding I**			**34**			
7.1	Sprint 4 - Planning / Sprint 3 Demo	ALL		0	09 Nov 2023 6:30PM	09 Nov 2023 7:30PM	Done
7.2	Backend Server/API Proxy	Technical Lead		8			Done
7.4	Standardize Styling Rules	Lead Designer		2			Done
	CI/CD Pipeline - Deploy New Backend	QA Engineer/Project Lead		8			Done
	Refactor Project - Divide Frontend/Backend	Lead Developer/Tech Lead		2			Done
	CI/CD Pipeline - Link Frontend with new backend	QA Engineer / Project Lead		3			Done
7.6	Research Secure Backend Server/API Proxy	DevSecOps Engineer		3			Done
7 10	**System Study Review**	Project Team		8	13 Nov 2023 1:00PM	13 Nov 2023 2:15PM	Done
8.0	**Sprint - 5 - Tenant Onboarding II**			**19**			
8.1	Sprint 5 - Planning / Sprint 4 Demo	ALL		0	30 Nov 2023 6:30PM	30 Nov 2023 7:30PM	Done
	Implement Sessions for Authorization	Technical Lead/DevSecOps		8	23 Nov 2023 6:30PM		Done
8.9	Landing Page	Project Manager		3	23 Nov 2023 6:30PM	27 Nov 2023 12:30PM	Done
	Direct user to email on account creation	Unassigned		1	27 Nov 2023 2:00PM		Done
	Fix React Refresh Issue	Project Manager		2	27 Nov 2023 8:00PM	1 Dec 2023 12:00PM	Done
	Homeowner - Add Property Hook/Backend	Lead Developer		5	23 Nov 2023 6:30PM		Done

9.0	**Sprint - 6 - Tenant / Service Provider Onboarding I**			**44**			
9.1	Sprint 6 - Planning / Sprint 5 Demo	ALL		0	07 Dec 2023 6:30PM	07 Dec 2023 7:30PM	Done
	Implement Rate Limiting on Backend	Technical Lead		3	19 Dec 2023 12:00PM	21 Dec 2023 7:30PM	Done
7.7	Implement Standard Testing Practices	QA Engineer		5	10 Dec 2023 12:00PM	12 Dec 2023 7:30PM	Done
8.4	Homeowner Dashboard - View Tenants	Lead Developer		5			Done
	Address Validation	Lead Developer		3	13 Dec 2023 12:00PM	16 Dec 2023 6:00PM	Done
	CI/CD Pipeline - Deploy to Both Data Centres	QA Engineer		3	13 Dec 2023 12:00PM	16 Dec 2023 6:00PM	Done
8.3	Send Tenant Invitation Hook/Backend	Project Manager		8	14 Dec 2023 12:00PM	17 Dec 2023 6:00PM	Done
8.5	Tenant Dashboard	Lead Designer		3			Done
8.7	Tenant Create Account	Project Manager		5	14 Dec 2023 12:00PM	17 Dec 2023 6:00PM	Done
8.8	Tenant Login	Unassigned		3	14 Dec 2023 12:00PM	17 Dec 2023 6:00PM	Done
	Send Tenant Invitation React Page	Project Manager		2	14 Dec 2023 12:00PM	17 Dec 2023 6:00PM	Done
8.6	Tenant Accept Invitation	QA Engineer		3	14 Dec 2023 12:00PM	17 Dec 2023 6:00PM	Done
8 10	Dashboard - Delete Property	Lead Developer		1	18 Dec 2023 12:00PM	19 Dec 2023 6:00PM	Done
10.0	**Sprint - 7 - Service Provider Onboarding II**			**27**			
10.1	Sprint 7 - Planning / Sprint 6 Demo	ALL		0	21 Dec 2023 6:30PM	21 Dec 2023 7:30PM	Done
	GROUP LUNCH / Board Games	ALL		0	03 Jan 2023 11:00AM	03 Jan 2023 12:30PM	Done
	Improve Aesthetics of Homeowner Dashboard	Lead Designer		3			Done
8.2	Implement Basic Penetration Testing Plan	DevSecOps/QA Engineer		5	21 Dec 2023 7:30PM	03 Jan 2023 12:30PM	Done
9.2	Invite Service Provider Wireframe	Lead Designer		2			Done
9.3	Send Service Provider Invite	Project Manager		3	27 Dec 2023 9:00AM	28 Dec 2023 12:00PM	Done
9.4	Create Service Provider Account	Project Manager		5	27 Dec 2023 9:00AM	28 Dec 2023 12:00PM	Done
	Validate service provider address	Project Manager		1	29 Dec 2023 9:00AM	29 Dec 2023 9:00PM	Done
	Implement Forgot Password Functionality	Project Manager		2	30 Dec 2023 10:00AM	30 Dec 2023 10:00PM	Done
9.5	Service Provider Login	Project Manager		3	27 Dec 2023 9:00AM	28 Dec 2023 12:00PM	Done
	Basic Service Provider Dashboard	Lead Designer		3	1 Jan 2023 12:00PM	3 Jan 2023 12:00PM	Done

11.0	**Sprint - 8 - Service Request Creation I**			**35**			
11.1	Sprint 8 - Planning / Sprint 7 Demo	ALL		0	04 Jan 2024 6:30PM	04 Jan 2024 7:30PM	Done
10.3	Service Provider Add Service Listings	Project Manager		3	04 Jan 2024 7:30PM		Done
	Homeowner Dashboard - View Tenants	Lead Developer		5	27 Dec 2023 12:00PM		Done
	Homeowner Dash - Approve Tenant Questionnaire	Project Manager		5			Done
	View Property Details	Lead Developer		3			Done
10.2	Service Provider Dashboard - View My Services	Project Manager		3			Done
11.3	Create Service Request - Tenant Initiate	Project Manager		3			Done
11.7	**Detailed Design Review**	ALL		13	15 Jan 2024 1:00PM	15 Jan 2024 2:00PM	Done
12.0	**Sprint - 9 - Service Request Creation II**			**33**			
12.1	Sprint 9 - Planning / Sprint 8 Demo	ALL		0	18 Jan 2024 6:30PM	18 Jan 2024 7:30PM	Done
	Minor Adjustments to Dashboards	Lead Designer		2			Done
11.5	Tenant Dashboard - View Service Requests	Technical Lead		3			Done
11.4	Homeowner - Request Quote For Service	Project Manager		2			Done
	Homeowner Dashboard - View Service Requests	Technical Lead		1			Done
	Search for Available Service Providers Based on Tick	Lead Developer		5			Done
	Improve Viewing of Services	Lead Designer		2			Done
	Docker Configuration Changes/Fix	QA Engineer		5			Done
12.3	HO Dashboard - View Proposed Quotes	Project Manager		3			Done
12.5	HO Dashboard - Approve Proposed Quote	Unassigned		2			Done
11.6	SP Dashboard - View Service Request Details	DevSecOps Engineer		2			Done
	Penetration Testing In Pipeline	DevSecOps		3			Done
	Basic Research for ML Model	QA Engineer		1			Done
12.2	Service Provider- Send Proposal to HO	Technical Lead		2			Done
13.0	**Sprint - 10 - Background Check I / Machine Larning**			**31**			
13.1	Sprint 10 - Planning / Sprint 9 Demo	ALL		0	01 Feb 2024 6:30PM	01 Feb 2024 7:30PM	Done
13.3	SP Dashboard - Apply for Public Service Provider	Project Manager		3			Done
	Seperate Public and Private Service Providers in Quo	Lead Developer		2			Done
13.4	Send Certn Background Check Request	Project Manager		1			Done
	Mark Job as In Progress, Completed	Project Manager		3			Done
14.3	Grant Public Service Provider Status on Approval	Project Manager		3			Done
13.5	SP Dashboard - View Status of Background Check	Project Manager		3			Done
	Machine Learning Explorative Development	QA Engineer		3			Done
	HO/SP/TEN Dashboard - Cancel Service Requests	Technical Lead		3			Done
	UI For Tenant Background Check	Lead Designer		3			Done
	Fully implement tenant background check	Lead Developer		3			Done
	SP/HO/TEN Dashboard - View Job Details	Technical Lead		2			Done
	Homeowner Initiate Service Request	DevSecOps/QA Engineer		2			Done

Task Number	Task Name	Resource	Duration	Story Points	Start	Finish	Status
14.0	**Sprint - 11 - Background Check II / AI Chatbot**			**70**			
14.1	Sprint 11 - Planning / Sprint 10 Demo	ALL		0	15 Feb 2024 6:30PM	15 Feb 2024 7:30PM	Done
	HO/TEN Dashboard - Track Job Status	Project Manager		2			Done
14.2	Process Completed Background Check from Certn	Unassigned		8			Done
	Implement Tenant Aggreement	Lead Developer		3			Done
	Quality - Verify OTP page	Project Manager		2			Done
13.6	Basic React Native Dashboard?	QA Engineer		13			Done
	Quality - Separate service requests/jobs into dropdo	Technical Lead		2			Done
15.2	Research for ML Model	QA Engineer		3			Done
15.3	Development of ML Model	QA Engineer		8			Done
	Team meeting	ALL		2			Done
	Speed up inference	QA Engineer		3			Done
15.4	Integration of ML Model into System	QA Engineer		8			Done
	Fix - Cleanup Snyk Warnings	DevSecOps Engineer		3			Done
	Port HomeOwner Features to React Native	QA Engineer		5			Done
14.5	Port Tenant Features to React Native	QA Engineer		8			Done
15.0	**Sprint 12 - AI Chatbot / Quality Iteration**			**74**			
15.1	Sprint 12 - Planning / Sprint 11 Demo	ALL		0	29 Feb 2024 6:30PM	29 Feb 2024 7:30PM	Done
	Quality - Redesign Request Quote Page	Project Manager		2			Done
	Fix - Dashboard Sidebar Title	Project Manager		1			Done
	Tenant Dashboard - Replace Address with Request D	Project Manager		1			Done
	Tenant Dashboard - Show Job Status	Project Manager		1			Done
	Remove Dead Links From Side Menu	Project Manager		1			Done
	Quality - Tenant dashboard property listing	Unassigned		2			Done
H	HO Dashboard - Remove quote button on active job	Project Manager		1			Done
15.5	Port Service Provider Features to React Native	QA Engineer/Technical Lead		13			Done
	Quality - Add Property Page	Lead Designer		1			Done
	Quality - Make dashboard titles clickable	Lead Designer		2			Done
	Mobile Figma Prototypes	Lead Designer		1			Done
	Replace Alerts with Modals	Lead Designer		2			Done
	Quality - Make all dashboards properly Proportionec	Lead Designer		1			Done
	Fix - Error handling for forms	Lead Developer		2			Done
	Quality - Send Quote Page	Lead Designer		2			Done
	Quality - Tab menu for service request tables	QA Engineer		2			Done
	Quality - Request Quote Page	QA Engineer/Lead Designer		1			Done
	TEN/HO Dashboard - Fix footer to bottom	QA Engineer		1			Done
	Fix - Tenant Modal Size in HO Dashboard	QA Engineer		2			Done
	Deploy chatbot as a service	QA Engineer		13			Done
16.2	Testing of ML Model	QA Engineer		5			Done
	WebAPI(dictation) for Homie	QA Engineer		2			Done
	Homie on react-native	QA Engineer		3			Done
16.3	Testing of Model Integration on Openshift	QA Engineer		8			Done
	Major UI fixes	QA Engineer		3			Done

Appendix B: Developer and User Signoffs