THE UNIVERSITY OF WINNIPEG

Dept. of Applied Computer Science

ACS-4901

Senior Systems Development Project

Prove IT – Detailed Design 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: 15 January 2024

Contents

1. Introduction

1.1 Background

HomeTrumpeter LLC is a company based in Illinois, USA. 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 is tasked to construct a new platform built on the foundations of the HomeTrumpeter API gateway, to "Prove IT" can be done.

1.2 Objectives

The goal of this project is to develop a platform that wraps around the HomeTrumpeter API gateway.

We will design and build a web interface in React, integrating into the HomeTrumpeter API gateway. A CI/CD workflow will be developed using GitLab to streamline the testing and deployment processes, allowing for fast and safe releases. The app will be continuously deployed to publicly accessible clusters managed by HomeTrumpeter through the OpenShift Kubernetes Platform.

Our system may not be used to replace the existing HomeTrumpeter web app. The primary objective of the project is to evaluate the feasibility of the API created by HomeTrumpeter. By working closely with the team at HomeTrumpeter, our team will be able to identify points of strength, or areas of improvement in the existing API. From this point, HomeTrumpeter will be able to better tailor their API to the requirements of an independent platform.

The API economy has become a massively 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 will be the construction of a web app leveraging the existing API. If time permits, we will explore the development of new features not included in the current HomeTrumpeter platform. Such features could include a React Native mobile app available on iOS and Android, or a machine learning model to predict service provider behavior using publicly available information.

1.3 Scope

Project “Prove IT” aims to develop a platform interfacing with HomeTrumpeter’s API gateway. The scope encompasses the following key deliverables:

1.3.1 Medium Fidelity Figma Prototype

The team is developing a medium fidelity prototype to conceptualize and communicate our vision for the system. Such a prototype will allow our team to demonstrate the core functionalities of the system and solicit feedback from sponsors. The medium level of fidelity will allow us the flexibility to easily change UI deigns based on stakeholder feedback. The team will then be able to use this prototype as a guide for interface design throughout the development process. As functionality is added to the system, the team will iterate on the prototype to conceptualize and communicate new features.

1.3.2 GitLab CI/CD Pipeline

To facilitate continuous deployment of changes to the system and ensure product quality, the team has begun to develop a continuous integration / continuous deployment pipeline in GitLab. The continuous integration portion of the pipeline will allow our team to automate unit tests, functional testing, static code analysis, dependency scanning, and fuzz testing. These functionalities will improve the quality and security of the delivered product, by automatically identifying potential issues with every push of new code to the repository. The continuous deployment portion of the pipeline will automate the process of building our application, containerizing it through Docker, and deploying it to HomeTrumpeter’s private cloud infrastructure.

1.3.3 Front-End Web Application

The primary deliverable of the project will be a React web application leveraging HomeTrumpeter’s API. This will allow homeowners, tenants, and service providers to engage in secure communication and transactions for all services related to the property. Homeowners will have access to easy-to-use manual and automated payment collection systems. They may also consult service providers for repairs, as well as issue and track the progress of property maintenance requests from tenants and property managers.

1.3.4 Back-End Proxy

The back-end proxy will mediate communication between the web application and HomeTrumpeter's API. Security is ensured by storing the API key on the local server. Efficiency will be improved through rate-limiting and traffic splitting. Our back-end platform will leverage HomeTrumpeter's multiple worldwide datacenters to guarantee maximum uptime.

1.3.5 React Native Mobile App

If the team develops all functional requirements of the web app with time remaining, we will develop an Android/iOS app using React Native offering similar functionality to that offered by the web app.

1.3.6 Machine Learning Model to Predict Service Provider Behavior

Commented [AG1]: Make sure we are doing this. Even at a minimal level.

If the team develops all functional requirements of the web and React Native apps with time remaining, we will develop a machine learning model to evaluate the trustworthiness of a service provider. This model will use sentiment analysis on publicly available information related to the service provider such as news articles and social media posts to provide a prediction of behavior. This evaluation could serve as part of the background check process for public service providers or be presented directly to homeowners considering their services.

The team will be building against the 'Black Box' of HomeTrumpeter's API. Functionalities provided by the API are out of scope for the project and will not be developed or modified by team members. Little to no information regarding the inner workings of the API will be provided to the team. A conceptualization of assets, infrastructure, and threat agents that could be present in HomeTrumpeter's system will need to be created by the team for the purpose of threat modelling. Though we are not responsible for the development of the API, it is important to consider threats and assets that may be present in the black box to uphold the practice of developing a 'zero-trust' system.

2. System Architecture

2.1 Overview

The implementation of our system will utilize the architecture of a typical full-stack web application. Two layers will be hosted on the Prove IT server, consisting of the front-end and back-end. The popular framework ReactJS will provide a foundation for building the front-end web server. Building against the HomeTrumpeter API, a database is not required and only caching will be implemented in the persistence layer. ExpressJS and NodeJS will be used to create the back-end proxy and implement caching.

Commented [XZ2]: This addresses why we don't have a database

The main objective is to build microservices that are capable of handling workload variations with minimal configuration. This exposes the HomeTrumpeter API to a vast audience from start-ups to large corporations requiring the services provided.

2.2 Architecture Diagram

The front-end and back-end servers will be hosted as separate microservices on an OpenShift Kubernetes platform. The front-end will be accessed publicly to the users. The back-end system will mediate communication between the web server and HomeTrumpeter API, as an added layer of security.

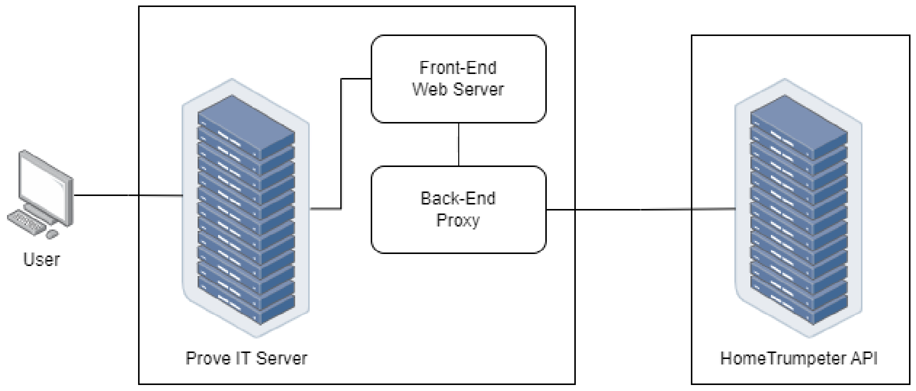


Figure 1. Architecture diagram for Prove IT infrastructure. The front-end web server and back-end proxy server will be hosted on the same cluster within the OpenShift private cloud. External connections will be made to the HomeTrumpeter API and the user accessing the system.

2.3 Components and Modules

The following components/modules are used in the full-stack application:

- Front-end web application
- Back-end proxy server
- HomeTrumpeter API

The separation into separate microservices allows each system to grow/shrink depending on resources usage. Users are able to scale the system to fit their needs, as well as reducing daily costs of cloud-based operations as the system is capable of scaling at a fine or coarse level.

3. User Interface Design

Commented [LK3]: Add a section on usability goals - visibility of the system status with loading spinner? How can we address change blindness with our error messages?

Commented [XZ4R3]: Added all design principles in 3.3, check if good enough

3.1 Wireframes

Wireframes were designed based on user stories gathered through interviews, questionnaires, and discussions. Workflows and the respective users involved were identified. Each user (homeowner, tenant, and/or service provider) was assigned various user stories (tasks). Through these stories, a set of pages were identified and built as wireframes concurrently with the user flow diagram.

Entities	**Task**
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 allocate tasks to service providers, so that I can track who is working on what.
	As a homeowner, I would like to add/save/edit the status of a ticket, so that I am as flexible as possible with my services.
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 submit maintenance tickets, so that I can track my work.
	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 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.
	As a tenant, I would like to confirm the completion of service requests, so that the invoice can be finalized.

Components of each page was developed individually as modular components as to increase re-usability, which in turn promotes efficiency.

3.2 User Flow Diagram

The user flow diagram is as follows. Each box represents a page, and each line represents a bidirectional navigation between the pages. All pages shown below are accessed by routes protected by authentication.

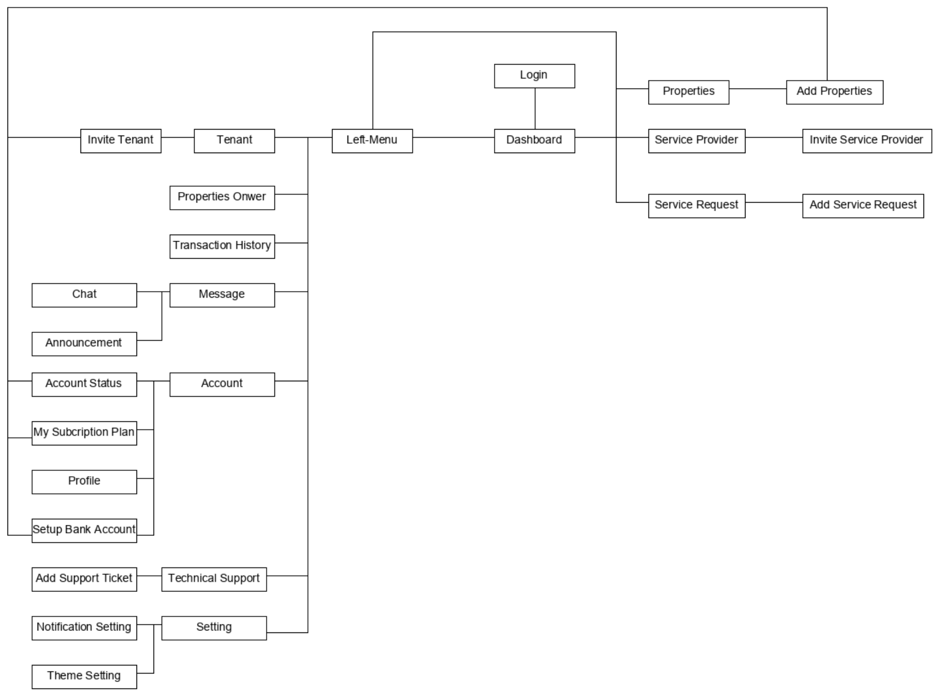


Figure 2. User flow diagram for Prove IT system users. All routes past the login authentication are protected routes. Security clearance is checked on each navigation to ensure permission is granted before allowing access to respective pages.

3.3 Design Mock-ups

An iterative approach was taken, following the interaction design (IxD) process. Prototypes were drafted quickly on paper and design principles of human-computer interaction were followed. Horizontal prototyping in Figma allowed users to provide feedback on a breadth of functionalities, despite lacking depth of functional requirements. Vertical prototyping integrated all functional requirements into the horizontal prototype after all feedback was considered.

This process was performed throughout each sprint, allowing for constant feedback from users to fine-tune the system to their desired needs.

Human Computer Interaction (HCI) design principles were considered during the prototyping process. Visibility is established by maintaining a brand kit, which provides consistent and 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. External consistency allows users to exploit pre-existing knowledge of similar systems, as well as hinting possible actions, which provides affordances.

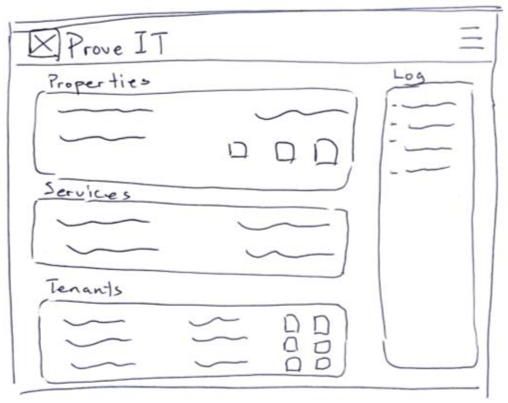
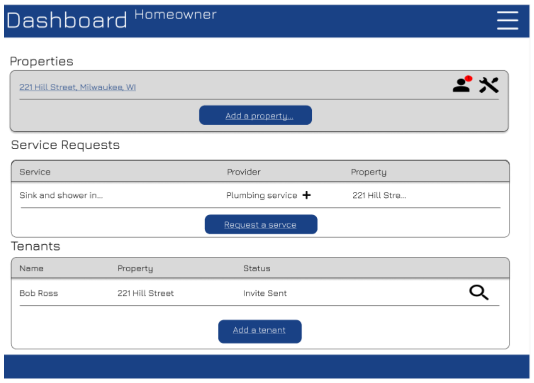


Figure 3. Paper prototype of the homeowner dashboard designed using user stories as the motivation. Rapid prototyping methods, such as crazy 8s, were used to create additional variations.



Service	Provider	Property
Sink and shower in...	Plumbing service +	221 Hill Stre...

Name	Property	Status
Bob Ross	221 Hill Street	Invite Sent

Figure 4. High fidelity prototype of the homeowner dashboard designed through vertical prototyping of the paper prototype. Figma was used to create the prototype and key features, such as adding a property, requesting a service, and adding tenants, are implemented for user testing.

4. Detailed System Design

4.1 Front-end

The front-end system used ReactJS as a foundational framework for building the user interfaces of the web application. Vite was used as a development server and bundler to ensure a quick and efficient workflow. This ensures that routes accessed by users are only required to load modules that are immediately used, which reduces loading latency significantly.

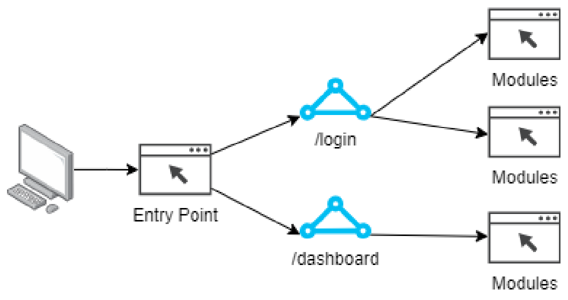


Figure 5. HTML and JavaScript module access using Vite to package front-end elements. Components used in different routes are packaged into modules to decrease loading time and increase performance. The modular design allows for re-usability of components as indicated in the */login* route.

Routing was implemented using ReactRouter to point URLs to respective web pages. React contexts were used in conjunction to enforce user access to their respective permission levels.

Design of the UI was done using React Bootstrap, which includes functional components. These were further manipulated through custom CSS to fit the Prove IT style guide.

4.2 Back-end

The back-end system uses ExpressJS. Routing is implemented using the built-in router and components are functionalized to allow maximum reusability. No database or filesystem will be used, as the back-end will only serve as a proxy between the web application and the HomeTrumpeter API.

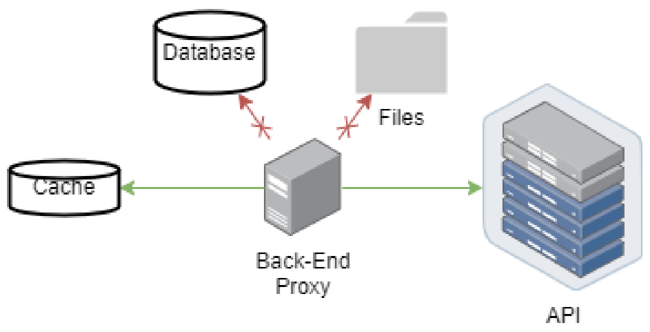


Figure 6. Data storage and access within the Prove IT back-end system. The persistence layer does not include a database nor file storage system. All data will be retrieved from the HomeTrumpeter database and partially cached in the Prove IT servers.

The ExpressJS app will serve the same identical API endpoints as the HomeTrumpeter API, routing API calls to the respective endpoints in the HomeTrumpeter API. Query information and authorization protocols will be passed through normally to enforce security.

5. API Design

5.1 HomeTrumpeter API

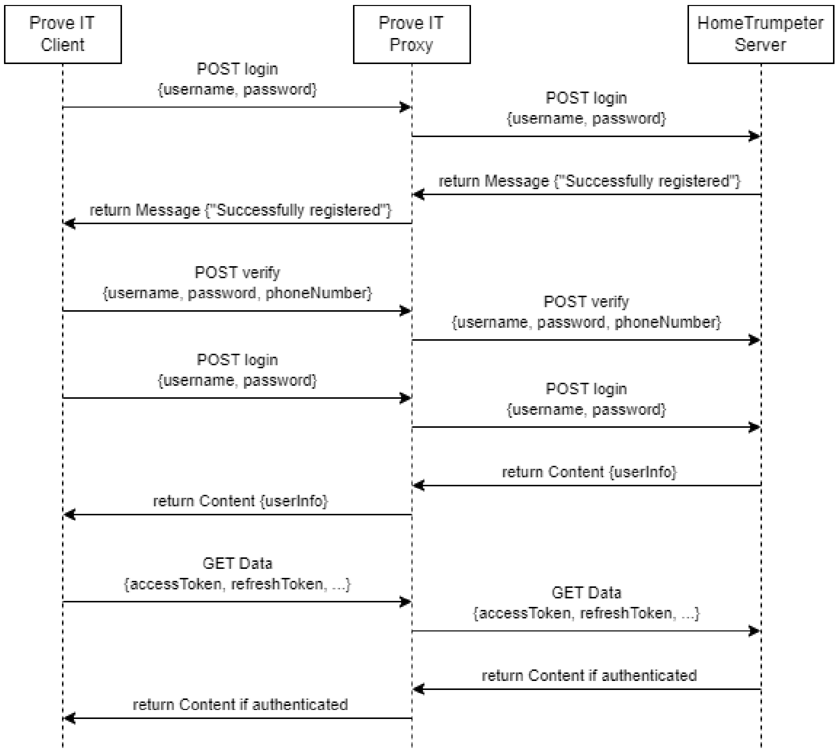
HomeTrumpeter has built a Restful API that includes features required to build a comprehensive property management system for property managers, homeowners, tenants, and service providers. Address verification through geographic information systems is provided by the Environmental Systems Research Institute (ESRI).

The HomeTrumpeter API is categorized as follows:

- user
- verification
- customer
- files
- plan
- intro-slider
- location
- school
- admin
- settings
- template
- support-ticket
- property-management
- service-provider
- ticket
- job
- service-request
- notification
- chat
- video
- payment-processing
- stripe
- agreement
- background-check
- general
- mail

5.2 JWT Session Management

Session management is implemented using the JWT tokens provided in the HomeTrumpeter API. The access tokens will be stored in the local storage of the web browser and retrieved for POST and GET requests.



Prove IT Client
Prove IT Proxy
HomeTrumpeter Server
POST login {username, password}
POST login {username, password}
return Message {"Successfully registered"}
return Message {"Successfully registered"}
POST verify {username, password, phoneNumber}
POST verify {username, password, phoneNumber}
POST login {username, password}
POST login {username, password}
return Content {userInfo}
return Content {userInfo}
GET Data {accessToken, refreshToken, ...}
GET Data {accessToken, refreshToken, ...}
return Content if authenticated
return Content if authenticated

Figure 7. System sequence diagram for JWT session management. User login information is delivered as a POST request, which returns an access token and refresh token to be stored in LocalStorage. Data will be accessed through GET requests which will include access tokens for authentication. Access tokens that expire are renewed with refresh tokens.

5.3 Proxy

A common issue when using private API keys is including the API key in the source code or client-side HTTP requests. To avoid exposing the API key, a proxy server is used as a middleman between the application and API server. This will redirect all API calls to the HomeTrumpeter API, along with the required authorization headers and bearer tokens.

Furthermore, the proxy server will allow scaling of the system through methods such as rate-limiting, load-balancing, and traffic splitting. It also permits us to manipulate the API responses to include the minimal information needed by the client, thus exposing less surface for attackers to exploit.

Additionally, the proxy allows careful control of information passed between users and the HomeTrumpeter API. As such, attack surfaces of our system are reduced, and privilege escalation is prevented by not exposing unused API endpoints in the Prove IT system.

6. Dependencies

6.1 External Dependencies

Front-end dependencies

- Bootstrap & React Bootstrap – Components for styling front-end using CSS
- React Router – Managing navigation and permission to access various pages

Back-end dependencies

- CORS – Allows access to resources in other domains
- Axios – Streamline HTTP request process

HomeTrumpeter API

- ESRI – GIS system for retrieving geographic data

6.2 Integration Points

There are various integration points within the full-stack application. The first point of integration is between the front-end and back-end server where the user's actions are converted to API calls. The back-end proxy server will connect with the HomeTrumpeter API to verify authority and return the response.

Commented [LK5]: Talk about how this impacts project timelines & deliverables

Commented [XZ6R5]: Done (sorta?), check if it's enough

Our services completely rely upon the HomeTrumpeter API, therefore their API uptime is crucial for our operations. As such, our application is redundantly deployed to both of their data centers, such that down time may be kept at a minimum.

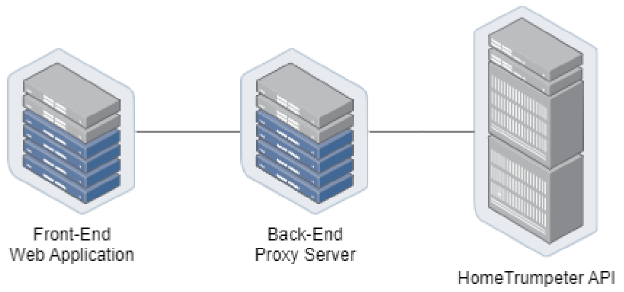


Figure 8. Integration points within the Prove IT system. The front-end and back-end servers communicate within the same local network. Users can access the front-end web application from the public web. The back-end system communicates with the HomeTrumpeter API using an API key for authentication.

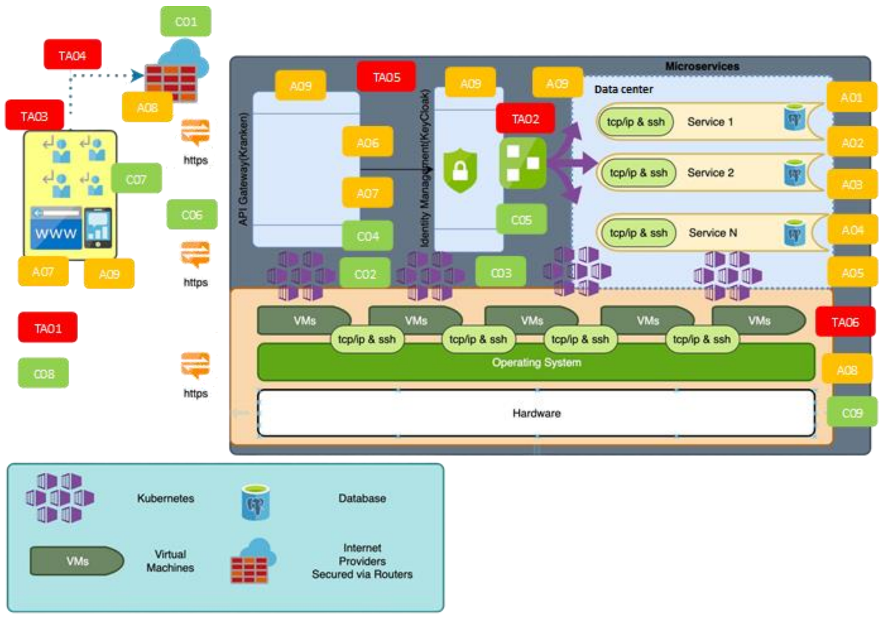
7. Security Design

Commented [LK7]: Shadman - make this more robust, talk about pen testing, screenshots, etc.

Commented [WS8R7]: I've added everything I intended to showcase for the DDR. Let me know if there's something that's missing or if there's something I need to fix.

7.1 Threat Modeling

A threat model has been constructed to monitor potential threat agents, attack surfaces, and find appropriate countermeasures for each. A full copy of our essential assets, abuse cases, and potential threat agents can be found in this threat model in the appendix. This threat model serves as the basis for the security measures taken in this project. As functionality continues to be added, potential attack surfaces will emerge. As such, the threat model will continually evolve as the project progresses.



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 9. 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_".

7.2 Penetration Testing

The penetration testing for this project is being 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:

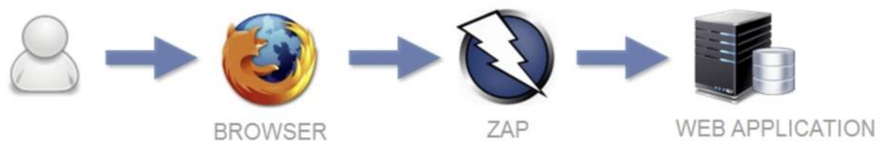
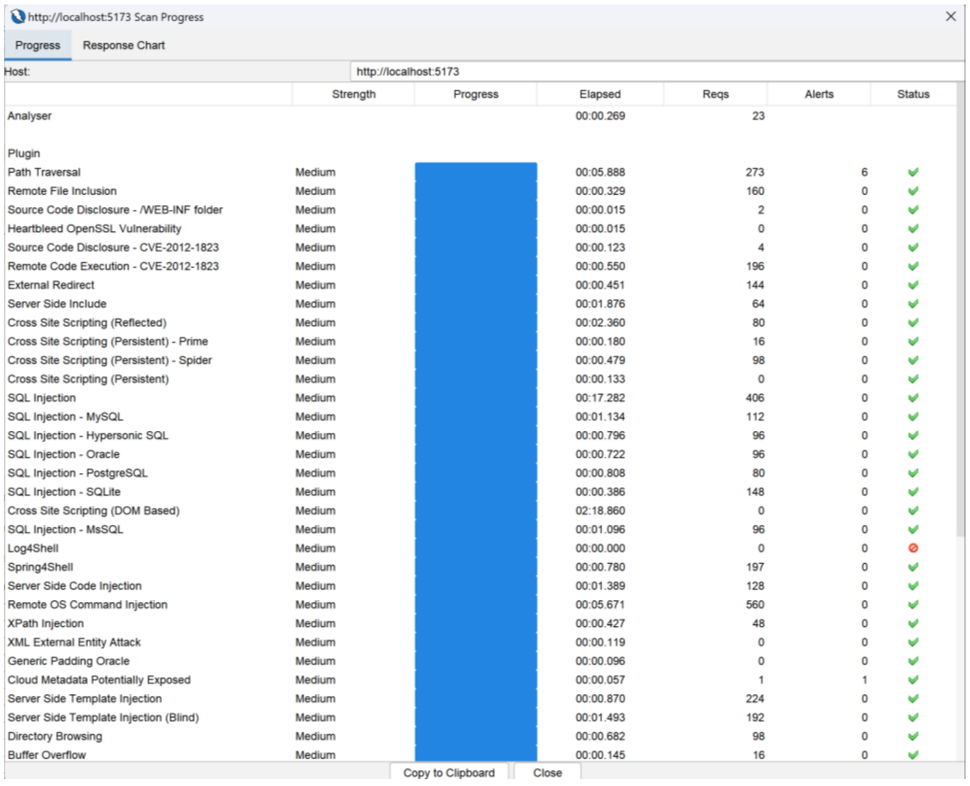


Figure 10. Penetration testing workflow using OWASP ZAP as a proxy between the web browser and web server. Packets are intercepted and examined for security risks, which are reported to the development team.

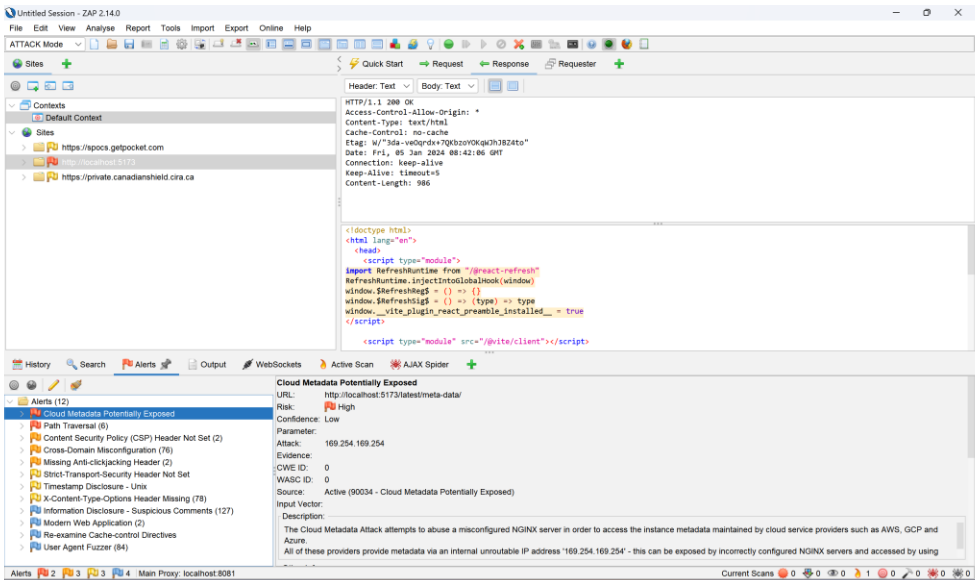
In this project, OWASP ZAP is being used for manual penetration testing, which includes exploring the application, understanding its flow and features, and creating or modifying requests sent by the browser to the server. The test is conducted on the application pushed to the develop branch in GitLab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



Host: http://localhost:5173

	Strength	Progress	Elapsed	Reqs	Alerts	Status
Analyser			00:00.269	23		
Plugin						
Path Traversal	Medium	🟦	00:05.888	273	6	✓
Remote File Inclusion	Medium	🟦	00:00.329	160	0	✓
Source Code Disclosure - /WEB-INF folder	Medium	🟦	00:00.015	2	0	✓
Heartbleed OpenSSL Vulnerability	Medium	🟦	00:00.015	0	0	✓
Source Code Disclosure - CVE-2012-1823	Medium	🟦	00:00.123	4	0	✓
Remote Code Execution - CVE-2012-1823	Medium	🟦	00:00.550	196	0	✓
External Redirect	Medium	🟦	00:00.451	144	0	✓
Server Side Include	Medium	🟦	00:01.876	64	0	✓
Cross Site Scripting (Reflected)	Medium	🟦	00:02.360	80	0	✓
Cross Site Scripting (Persistent) - Prime	Medium	🟦	00:00.180	16	0	✓
Cross Site Scripting (Persistent) - Spider	Medium	🟦	00:00.479	98	0	✓
Cross Site Scripting (Persistent)	Medium	🟦	00:00.133	0	0	✓
SQL Injection	Medium	🟦	00:17.282	406	0	✓
SQL Injection - MySQL	Medium	🟦	00:01.134	112	0	✓
SQL Injection - Hypersonic SQL	Medium	🟦	00:00.796	96	0	✓
SQL Injection - Oracle	Medium	🟦	00:00.722	96	0	✓
SQL Injection - PostgreSQL	Medium	🟦	00:00.808	80	0	✓
SQL Injection - SQLite	Medium	🟦	00:00.386	148	0	✓
Cross Site Scripting (DOM Based)	Medium	🟦	02:18.860	0	0	✓
SQL Injection - MsSQL	Medium	🟦	00:01.096	96	0	✓
Log4Shell	Medium	🟦	00:00.000	0	0	⊘
Spring4Shell	Medium	🟦	00:00.780	197	0	✓
Server Side Code Injection	Medium	🟦	00:01.389	128	0	✓
Remote OS Command Injection	Medium	🟦	00:05.671	560	0	✓
XPath Injection	Medium	🟦	00:00.427	48	0	✓
XML External Entity Attack	Medium	🟦	00:00.119	0	0	✓
Generic Padding Oracle	Medium	🟦	00:00.096	0	0	✓
Cloud Metadata Potentially Exposed	Medium	🟦	00:00.057	1	1	✓
Server Side Template Injection	Medium	🟦	00:00.870	224	0	✓
Server Side Template Injection (Blind)	Medium	🟦	00:01.493	192	0	✓
Directory Browsing	Medium	🟦	00:00.682	98	0	✓
Buffer Overflow	Medium	🟦	00:00.145	16	0	✓

Figure 11. OWASP ZAP scan performed on the Prove IT website. Vulnerabilities such as Cross Site Scripting (XSS) and SQL Injection are tested in the scan.



```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: text/html
Cache-Control: no-cache
Etag: W/"3da-veOqrdx+7QKbzoYOKqWJhJBZ4to"
Date: Fri, 05 Jan 2024 08:42:06 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 986
```

```
<!doctype html>
<html lang="en">
  <head>
    <script type="module">
import RefreshRuntime from "/@react-refresh"
RefreshRuntime.injectIntoGlobalHook(window)
window.$RefreshReg$ = () => {}
window.$RefreshSig$ = () => (type) => type
window.__vite_plugin_react_preamble_installed__ = true
</script>

    <script type="module" src="/@vite/client"></script>
```

Alerts (12)
- Cloud Metadata Potentially Exposed
- Path Traversal (6)
- Content Security Policy (CSP) Header Not Set (2)
- Cross-Domain Misconfiguration (76)
- Missing Anti-clickjacking Header (2)
- Strict-Transport-Security Header Not Set
- Timestamp Disclosure - Unix
- X-Content-Type-Options Header Missing (78)
- Information Disclosure - Suspicious Comments (127)
- Modern Web Application (2)
- Re-examine Cache-control Directives
- User Agent Fuzzer (84)

Cloud Metadata Potentially Exposed

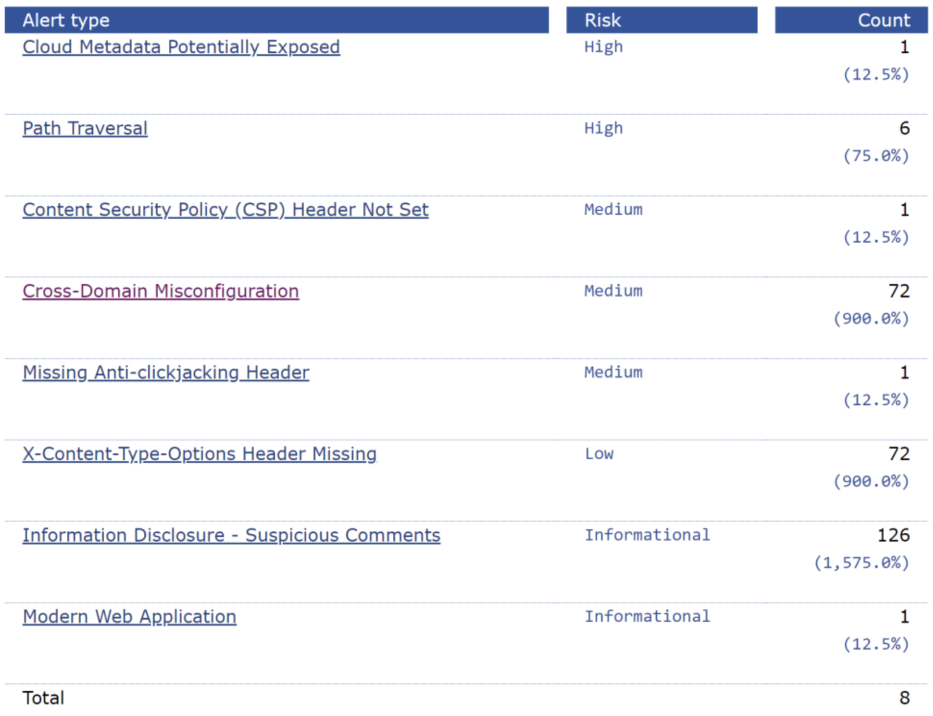
URL:	http://localhost:5173/latest/meta-data/
Risk:	High
Confidence:	Low
Parameter:	
Attack:	169.254.169.254
Evidence:	
CWE ID:	0
WASC ID:	0
Source:	Active (90034 - Cloud Metadata Potentially Exposed)
Input Vector:	

Description:
The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure.
All of these providers provide metadata via an internal unroutable IP address '169.254.169.254' - this can be exposed by incorrectly configured NGINX servers and accessed by using

Alerts 2 3 3 4 Main Proxy: localhost:8081 Current Scans 0 0 0 1 0 0 0 0

Figure 12. OWASP Zap scan results performed on 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 are recommended and most importantly any alerts or vulnerabilities found after the test are 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:



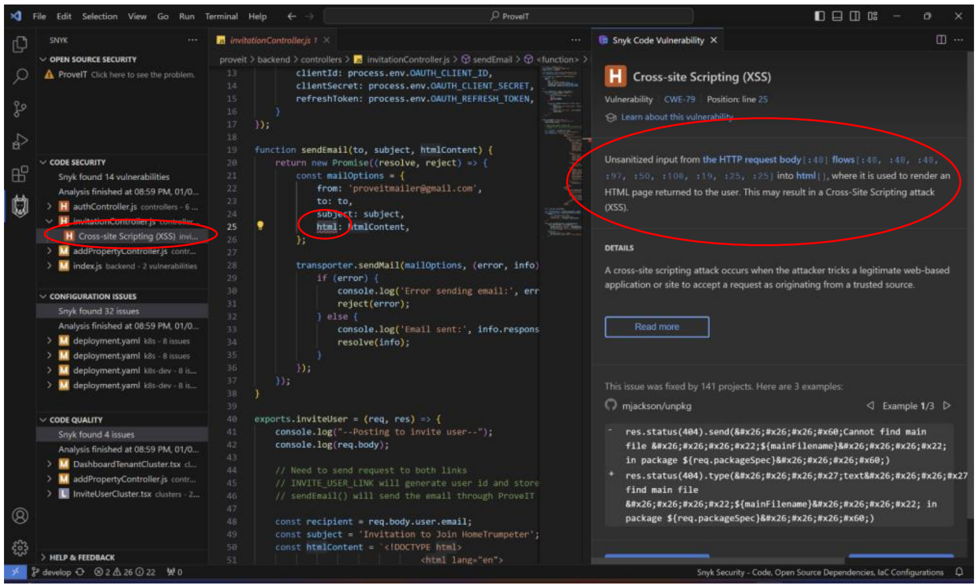
Alert type	Risk	Count
Cloud Metadata Potentially Exposed	High	1 (12.5%)
Path Traversal	High	6 (75.0%)
Content Security Policy (CSP) Header Not Set	Medium	1 (12.5%)
Cross-Domain Misconfiguration	Medium	72 (900.0%)
Missing Anti-clickjacking Header	Medium	1 (12.5%)
X-Content-Type-Options Header Missing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

Figure 13. OWASP ZAP scan report for 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.

As the project progresses, the ZAP penetration test will be implemented into the CI/CD pipeline so that all the tests are automated, and vulnerabilities are more efficiently addressed.

Another tool used for analyzing code for vulnerabilities is Snyk, which is a code analysis tool used for identifying and fixing vulnerabilities in dependencies. 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 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:


File Edit Selection View Go Run Terminal Help
ProveIT
SNYK
OPEN SOURCE SECURITY
ProveIT Click here to see the problem.
CODE SECURITY
Snyk found 14 vulnerabilities
Analysis finished at 08:59 PM, 01/0...
authController.js controllers - 6 ...
invitationController.js controller...
Cross-site Scripting (XSS) invi...
addPropertyController.js contr...
index.js backend - 2 vulnerabilities
CONFIGURATION ISSUES
Snyk found 32 issues
Analysis finished at 08:59 PM, 01/0...
deployment.yaml k8s - 8 issues
deployment.yaml k8s - 8 issues
deployment.yaml k8s-dev - 8 is...
deployment.yaml k8s-dev - 8 is...
CODE QUALITY
Snyk found 4 issues
Analysis finished at 08:59 PM, 01/0...
DashboardTenantCluster.tsx cl...
addPropertyController.js contr...
InviteUserCluster.tsx clusters - 2...
HELP & FEEDBACK
develop ⊗ 2 ⚠ 26 ⓘ 22 0
invitationController.js 1
proveit > backend > controllers > invitationController.js > sendEmail > <function> >
Snyk Code Vulnerability
Read more
Snyk Security - Code, Open Source Dependencies, IaC Configurations

```
13              clientId: process.env.OAUTH_CLIENT_ID,
14              clientSecret: process.env.OAUTH_CLIENT_SECRET,
15              refreshToken: process.env.OAUTH_REFRESH_TOKEN,
16          }
17      });
18
19      function sendEmail(to, subject, htmlContent) {
20          return new Promise((resolve, reject) => {
21              const mailOptions = {
22                  from: 'proveitmailer@gmail.com',
23                  to: to,
24                  subject: subject,
25                  html: htmlContent,
26              };
27
28              transporter.sendMail(mailOptions, (error, info)
29                  if (error) {
30                      console.log('Error sending email:', err
31                      reject(error);
32                  } else {
33                      console.log('Email sent:', info.respons
34                      resolve(info);
35                  }
36              });
37          });
38      }
39
40      exports.inviteUser = (req, res) => {
41          console.log("--Posting to invite user--");
42          console.log(req.body);
43
44          // Need to send request to both links
45          // INVITE_USER_LINK will generate user id and store
46          // sendEmail() will send the email through ProveIT
47
48          const recipient = req.body.user.email;
49          const subject = 'Invitation to Join HomeTrumpeter';
50          const htmlContent = `<!DOCTYPE html>
51                              <html lang="en">
```

H Cross-site Scripting (XSS)

Vulnerability | CWE-79 | Position: line 25

Learn about this vulnerability

Unsanitized input from the HTTP request body[:40] flows[:40, :40, :40, :97, :50, :108, :19, :25, :25] into html[], where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

DETAILS

A cross-site scripting attack occurs when the attacker tricks a legitimate web-based application or site to accept a request as originating from a trusted source.

This issue was fixed by 141 projects. Here are 3 examples:

mjackson/unpkg ◁ Example 1/3 ▷

```
-  res.status(404).send(&#x26;#x26;#x26;#x60;Cannot find main
   file &#x26;#x26;#x26;#x22;${mainFilename}&#x26;#x26;#x26;#x22;
   in package ${req.packageSpec}&#x26;#x26;#x26;#x60;)
+  res.status(404).type(&#x26;#x26;#x26;#x27;text&#x26;#x26;#x26;#x27
   find main file
   &#x26;#x26;#x26;#x22;${mainFilename}&#x26;#x26;#x26;#x22; in
   package ${req.packageSpec}&#x26;#x26;#x26;#x60;)
```

Figure 14. 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.

In the image above shows that after scanning and analyzing the code it categorizes the issues into Code Security, Configuration Issues and Code Quality and lists all of them accordingly. Upon selection it will give a greater view of the vulnerability by providing more information about the issue, where the issue is occurring and suggestions with references that may potentially fix the issue.

7.3 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. We intend to implement HTTPS on the Prove IT system itself to further secure the system against eavesdropping and man-in-the-middle attacks.

7.4 Encryption and Decryption

For encryption and decryption, the project employed JWT (JSON Web Tokens) and SHA256 encryption for securing tokens. The secret keys used for this encryption were managed by the HomeTrumpeter system. This approach ensured a robust method of securing and verifying user data and sessions, which is particularly important in web-based applications where session management and data integrity are paramount. The use of SHA256, a strong hashing algorithm, provided an additional layer of security, making it extremely difficult for potential attackers to decipher or tamper with the data. The combination of JWT and SHA256 encryption formed a solid foundation for the project's security, ensuring that sensitive information remained protected during transmission and storage.

7.5 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 have been identified, along with their corresponding monitoring and assessment techniques and mitigation plans that we will implement:

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

8. Testing Strategy

8.1 Unit Testing

We employ Jest and React Testing Library for basic unit testing. These tests validate individual components, functions, or units of code to ensure they work correctly. We are transitioning towards a Test-Driven Development (TDD) approach where our QA team writes tests for functional components before the development team writes the corresponding code. This process ensures that code changes are verified against expected behavior. We have different test suites which contains multiple unit tests related to a specific module.

8.2 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.

8.3 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.

8.4 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.

8.5 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.

9. Deployment Architecture

9.1 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.

9.1.1 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.

9.1.1 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.

9.1.2 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.

9.2 Deployment Diagram

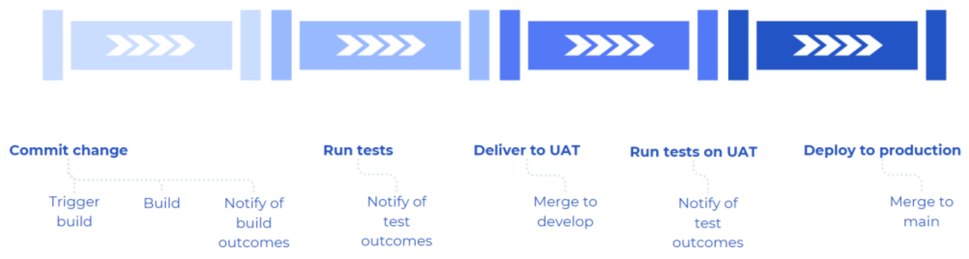


Figure 15. Deployment diagram for the CI/CD pipeline of Prove IT. Successful build of committed changes are tested within a testing namespace. Successful testing causes the commits to be delivered to the user acceptance testing (UAT) namespace for final integration testing prior to deployment to production, which merges the develop branch into the main branch.

9.3 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.

10. Release Planning

10.1 Agile Release Plan

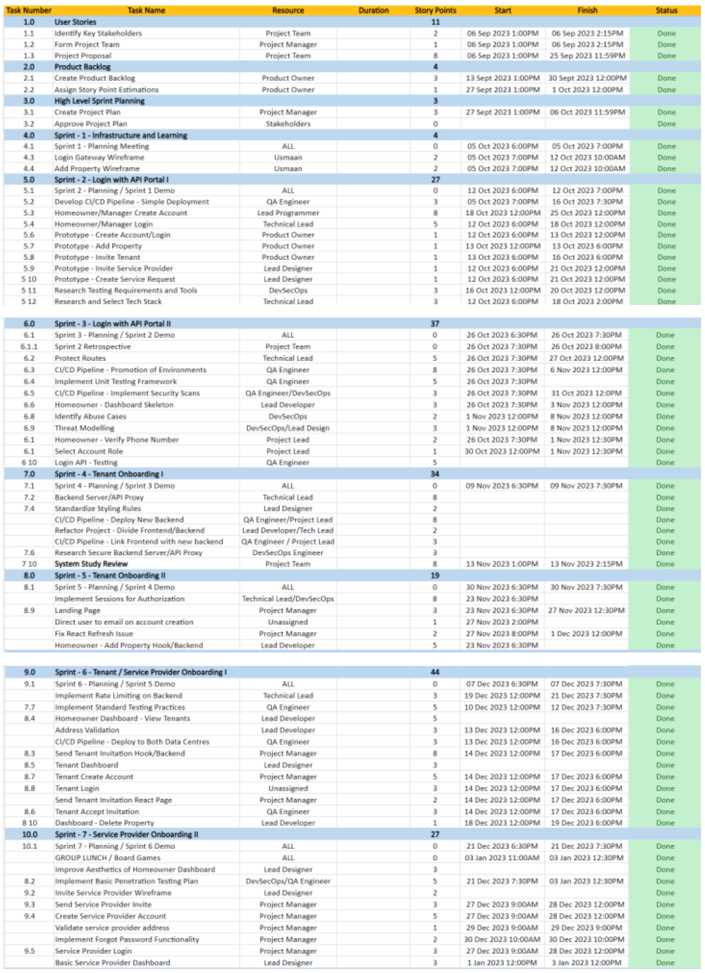
Sprint planning meetings were held every two weeks, involving the development team and stakeholders. These meetings allowed the team the opportunity to present our current progress and proposed objectives for the upcoming sprint. Stakeholders would then be able to offer input on the planned objectives, which could be adjusted accordingly.

The current development branch will be pushed to production (main) during the bi-weekly sprint planning meetings. Once deemed to be a significant update from the previous release of the system, the main branch will be tagged with a new release, named following Semantic Versioning (SemVer) guidelines.

10.2 Sprint Backlog

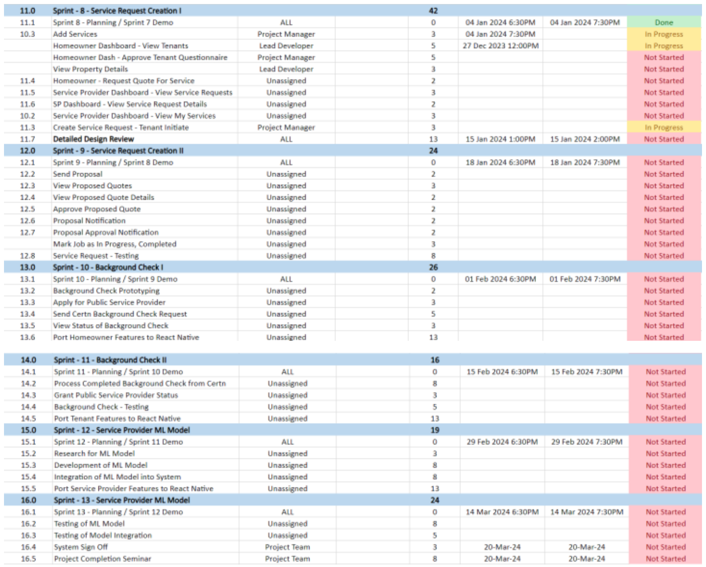
Each work item was assigned a story point estimation' according to the estimated amount of effort required to complete the work item. Work items were assigned to sprints based on the number of story points the team had historically been able to complete in a two-week sprint. Each high-level requirement was allotted two sprints to be completed. If we complete a requirement ahead of schedule, we will be able to pull backlog items from later sprints into the current sprint backlog to work ahead.

The agile methodology enables the team to add, remove, and shift tasks between sprints as necessary. In sprint 3, backend development was added as an additional task. The HomeTrumpeter team deemed it necessary to secure the API key using a proxy, thus maintaining the integrity of the system. Sprint 5 also included the addition of email sender on the backend system. Testing the invitation system in UAT exposed the issue of email redirects in the HomeTrumpeter system—all emails would redirect to the HomeTrumpeter system, rather than Prove IT. Our implementation of the Agile methodology has allowed us to adapt to these changes in requirements without the need for major changes to existing project plans. The current state of the product backlog, and the release plan for upcoming sprints is given below:



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

Figure 16. Product backlog for sprint 1 to 7, consisting of all completed sprint up to date. Tasks are numbered following a work breakdown structure. Story points are assigned relative to effort required for completion. Members are assigned to tasks and completion time is tracked through start and finish dates.



11.0	**Sprint - 8 - Service Request Creation I**			**42**			
11.1	Sprint 8 - Planning / Sprint 7 Demo	ALL		0	04 Jan 2024 6:30PM	04 Jan 2024 7:30PM	Done
10.3	Add Services	Project Manager		3	04 Jan 2024 7:30PM		In Progress
	Homeowner Dashboard - View Tenants	Lead Developer		5	27 Dec 2023 12:00PM		In Progress
	Homeowner Dash - Approve Tenant Questionnaire	Project Manager		5			Not Started
	View Property Details	Lead Developer		3			Not Started
11.4	Homeowner - Request Quote For Service	Unassigned		2			Not Started
11.5	Service Provider Dashboard - View Service Requests	Unassigned		3			Not Started
11.6	SP Dashboard - View Service Request Details	Unassigned		2			Not Started
10.2	Service Provider Dashboard - View My Services	Unassigned		3			Not Started
11.3	Create Service Request - Tenant Initiate	Project Manager		3			In Progress
11.7	**Detailed Design Review**	ALL		13	15 Jan 2024 1:00PM	15 Jan 2024 2:00PM	Not Started
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**			
13.1	Sprint 10 - Planning / Sprint 9 Demo	ALL		0	01 Feb 2024 6:30PM	01 Feb 2024 7:30PM	Not Started
13.2	Background Check Prototyping	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
13.6	Port Homeowner Features to React Native	Unassigned		13			Not Started
14.0	**Sprint - 11 - Background Check II**			**16**			
14.1	Sprint 11 - Planning / Sprint 10 Demo	ALL		0	15 Feb 2024 6:30PM	15 Feb 2024 7:30PM	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
14.5	Port Tenant Features to React Native	Unassigned		13			Not Started
15.0	**Sprint - 12 - Service Provider ML Model**			**19**			
15.1	Sprint 12 - Planning / Sprint 11 Demo	ALL		0	29 Feb 2024 6:30PM	29 Feb 2024 7:30PM	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
15.5	Port Service Provider Features to React Native	Unassigned		13			Not Started
16.0	**Sprint - 13 - Service Provider ML Model**			**24**			
16.1	Sprint 13 - Planning / Sprint 12 Demo	ALL		0	14 Mar 2024 6:30PM	14 Mar 2024 7:30PM	Not Started
16.2	Testing of ML Model	Unassigned		8			Not Started
16.3	Testing of Model Integration	Unassigned		5			Not Started
16.4	System Sign Off	Project Team		3	20-Mar-24	20-Mar-24	Not Started
16.5	Project Completion Seminar	Project Team		8	20-Mar-24	20-Mar-24	Not Started

Figure 17. Product backlog for sprint 8 to 13, consisting of upcoming sprints. Sprint 8 is currently in progress. Tasks are numbered following a work breakdown structure. Story points are assigned relative to effort required for completion. Members are assigned to tasks and completion time is tracked through start and finish dates.

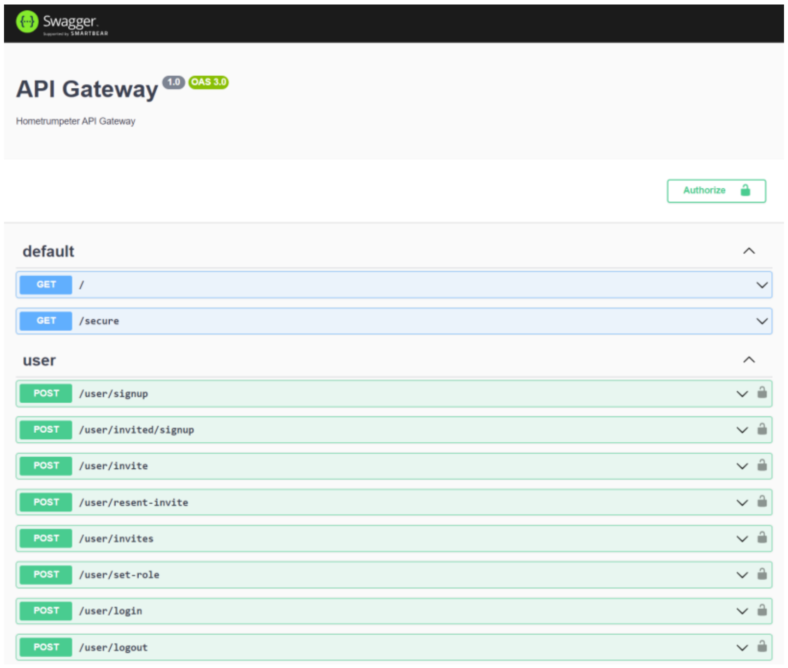
10.3 Definition of Done

The definition of done (DoD) is outlined by our sponsors during the bi-weekly sprint planning meetings. Once all acceptance criteria have been satisfied, the current sprint is adjourned, and the system is pushed to production.

11. Documentation

11.1 API Documentation

HomeTrumpeter has provided comprehensive documentation on their API in the form of tutorial videos, as well as an interactive UI using the Swagger API platform.



API Gateway 1.0 OAS 3.0

Hometrumpeter API Gateway

default

```
GET   /
GET   /secure
```

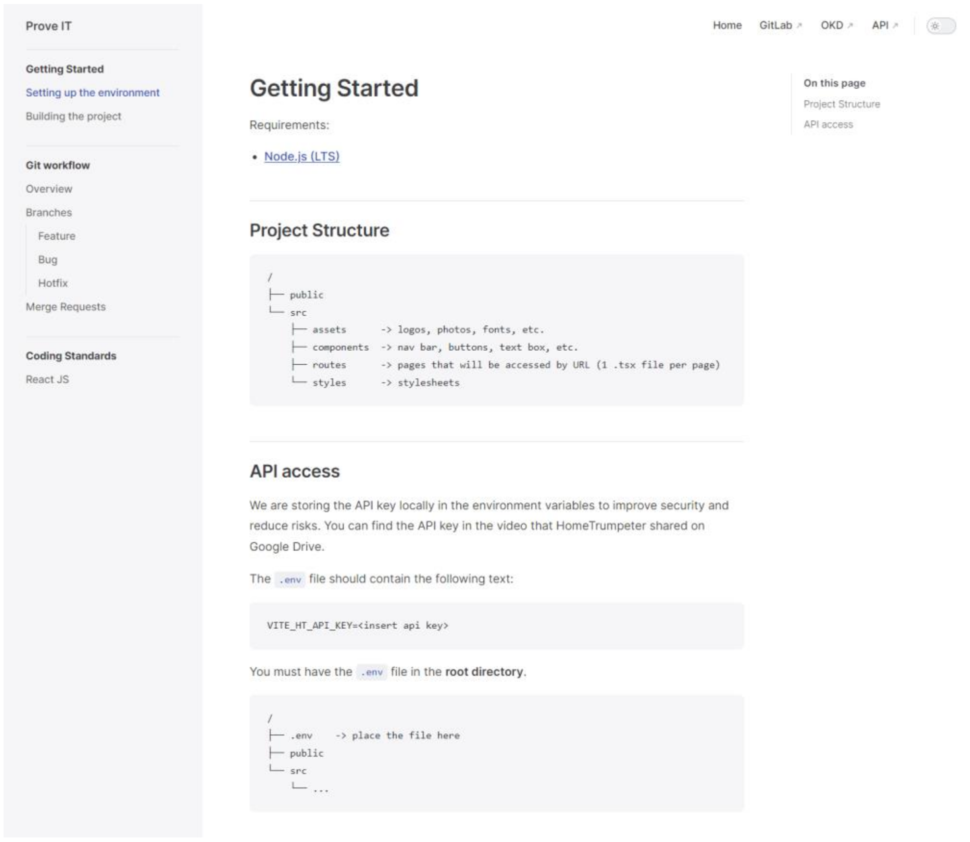
user

```
POST  /user/signup
POST  /user/invited/signup
POST  /user/invite
POST  /user/resent-invite
POST  /user/invites
POST  /user/set-role
POST  /user/login
POST  /user/logout
```

Figure 18. HomeTrumpeter API documentation. API endpoints are categorized based on usage and labelled as POST or GET requests.

11.2 Code 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.



Getting Started

Requirements:

- Node.js (LTS)

Project Structure

```
/
├─ public
└─ src
   ├─ assets      -> logos, photos, fonts, etc.
   ├─ components  -> nav bar, buttons, text box, etc.
   ├─ routes      -> pages that will be accessed by URL (1 .tsx file per page)
   └─ styles      -> stylesheets
```

API access

We are storing the API key locally in the environment variables to improve security and reduce risks. You can find the API key in the video that HomeTrumpeter shared on Google Drive.

The `.env` file should contain the following text:

```
VITE_HT_API_KEY=<insert api key>
```

You must have the `.env` file in the **root directory**.

```
/
├─ .env    -> place the file here
├─ public
└─ src
   └─ ...
```

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.