.

Potato Properties Web Application

The Potatoes - Group 03

Jade Lowe, Dakota Chanthakoummane, Aryan Kharva, Quintin Obey, Connor Smith, Edward Tobiasson

Professor Krasniqi

ITIS – 3300 – 092

October 29, 2024

Table of Contents

1 Requirements........ 3
1.1 Project Scope........ 3
1.2 Phase Progress........ 3
1.2 Phase Requirements........ 3
2 Diagrams........ 5
2.1 Class Diagram........ 5
2.2 Sequence Diagram........ 5
2.3 Use Case Diagram........ 6
3 User Manual........ 6
3.1 Installation........ 6
4 Instructions........ 6
4.1 Compilation........ 6
4.2 Test Cases........ 7
5 Reflection........ 7
6 Member Contributions........ 7

1 Requirements

1.1 Project Scope

The scope of the Potato Properties web application encompasses user authentication for client users types. The tenant user should be able to view apartment complex information, view current and upcoming vacancies, and apply for an apartment. The current tenant user should be able to view their lease documentation, make payments, and request unit maintenance.

1.2 Phase Progress

This phase has required some modification as we had some group members become ill with COVID as well as one sustaining a hand injury. For these reasons we have reconfigured our timeline as planned in our risk mitigation. Furthermore, we decided to cut out the admin facing system as we do not have enough time to implement both client and admin. Client side is more important, so we decided to pursue that.

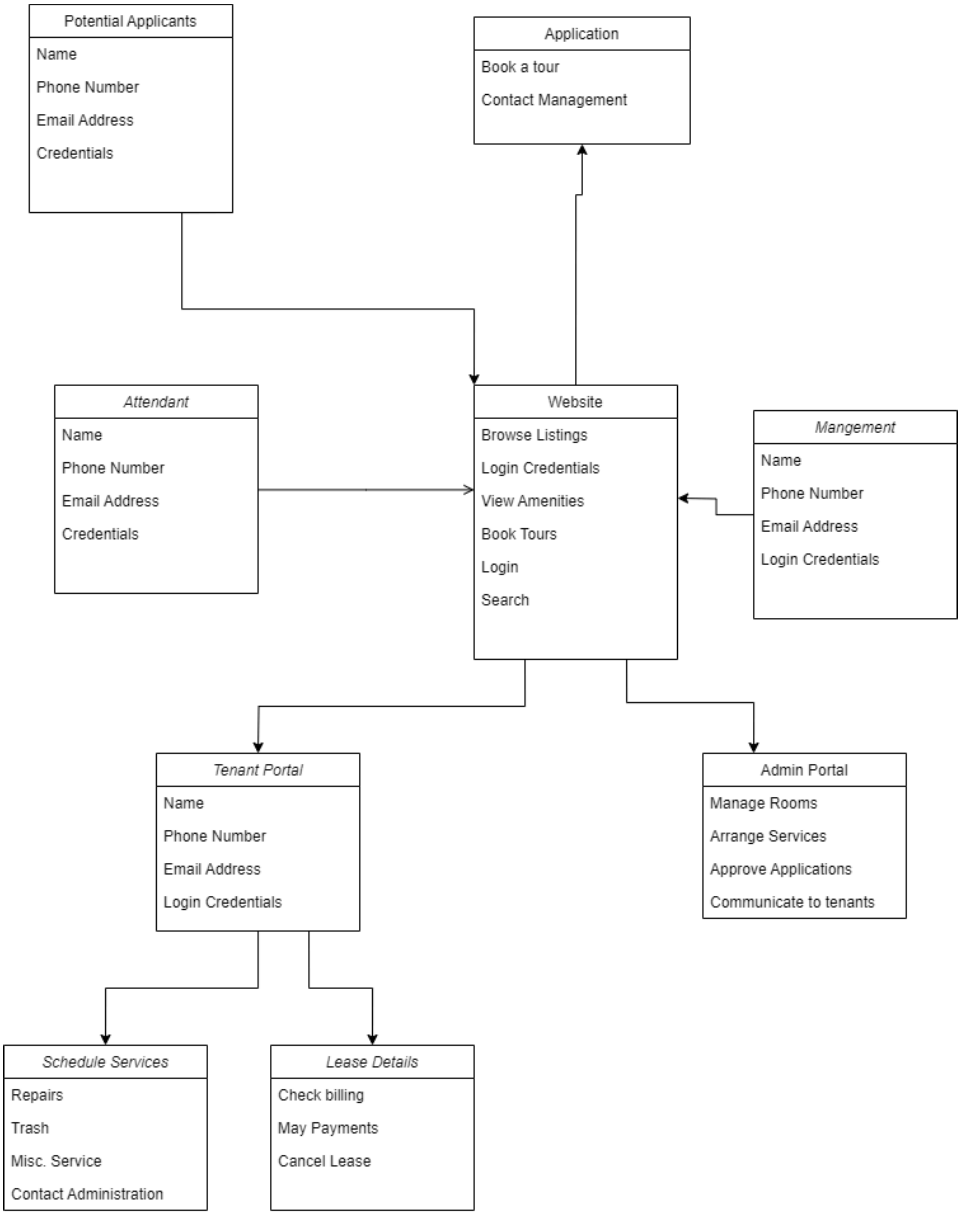
1.2 Phase Requirements

Feature	Phase	Task	Description	Technologies
Tenant UI & Pages	1	Create pages for the tenant facing side of the application.	Base UI for starting tenant pages (Subject to change as new functionality is introduced)	HTML/CSS, JS
Create User Account	1	Allow user to create an account	The user can create an account.	SQLAlchemy, Python, SQL
User Authentication	1	Implement user login	Implement login functionality for tenants. Tenants should be redirected to the dashboard page upon successful login.	SQLAlchemy, Python
Authentication Validation	1	Create flash messaging for users that are logging in and/or creating an account.	Users should see a flash message for invalid input on registration and login as well as a flash message for successful account creation.	SQLAlchemy, Python
Tenant Maintenance	2	Implement tenant maintenance requests	Implement functionality for tenants to submit and track maintenance requests	SQLAlchemy, Python
Payment	2	Implement payment processing system	Implement functionality for simulated payment processing	SQLAlchemy, Python
Payment Tracking	2	Implement payment tracking	Allow tenants to track payments	SQLAlchemy, Python

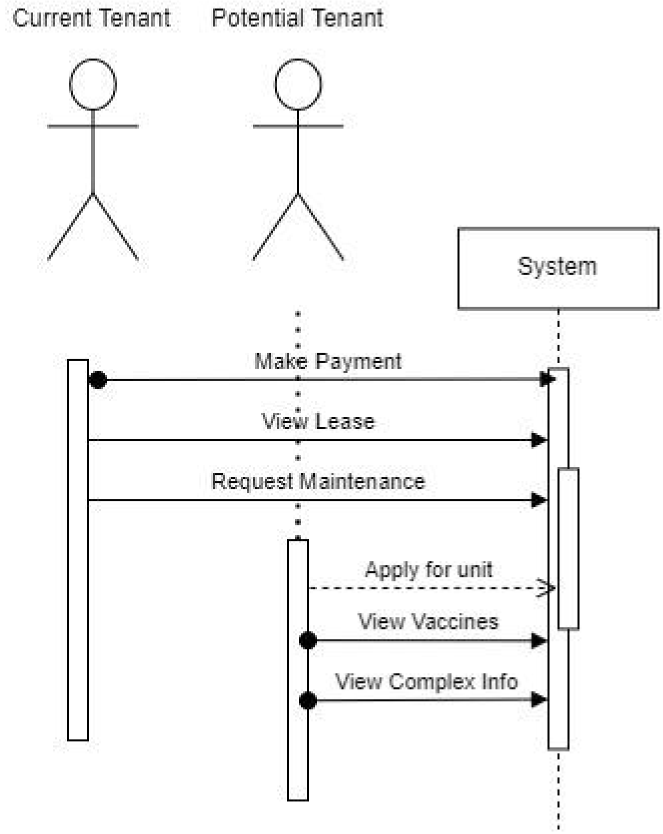
Application Submission	2	Implement application submission system	Implement functionality for simulated application submission from potential tenants	SQLAlchemy, Python
Application Tracking	2	Implement application tracking	Allow potential tenants to track application	SQLAlchemy, Python
Contact admin via the contact page	2	Implement a function to show a message that the user wants to send admin via contact us page.	Users should be able to send a message to admin via the contact us page. A function can be implemented to show this message, simulating an email.	SQLAlchemy, Python
Document Submission	2	Implement document upload and storage system	Allow both tenant types to upload relevant documents	SQLAlchemy, Python
Document Viewing	2	Implement document viewing	Allow both tenant types to view relevant documents	SQLAlchemy, Python
Lease Information	2	Display lease information	Allow current tenants to view their lease information	SQLAlchemy, Python
Improve UI	3	Update UI to look better	All UI Pages receive updated styling to improve flow of the pages.	HTML/CSS, JS
Security Update	3	Implement user data encryption	Encrypt user data for enhanced security	SQLAlchemy, Python, SQL
Optimize performance	3	Optimize website performance	Optimize web page loading times and page responsiveness	SQLAlchemy, Python, SQL, HTML/CSS, JS

2 Diagrams

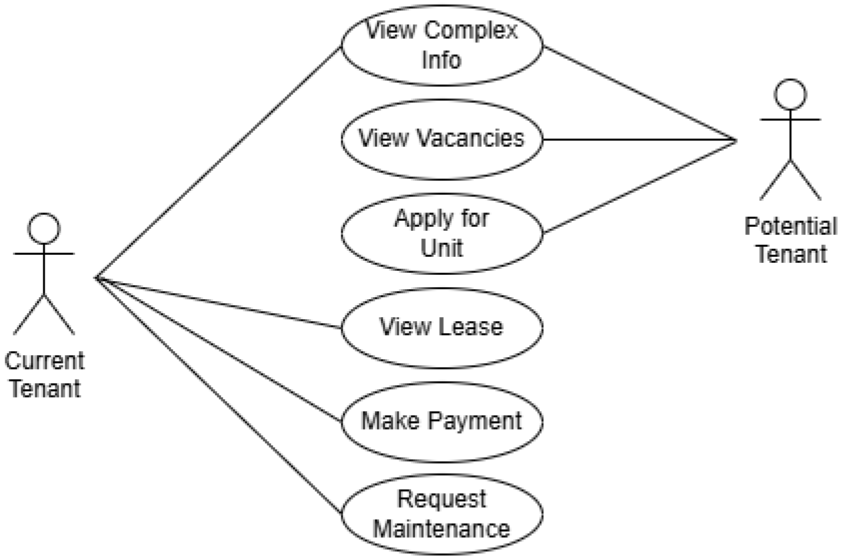
2.1 Class Diagram



2.2 Sequence Diagram



2.3 Use Case Diagram



3 User Manual

3.1 Installation

Clone this repository to your local machine using git clone https://github.com/jlowe093/ITIS3300-GroupProject.git

4 Instructions

4.1 Compilation

Change directory into the project folder.

Open a terminal in this project folder and run python -m venv venv to set up a virtual environment.

Run source venv/Scripts/activate to start the virtual environment.

Now that you are inside the virtual environment, run pip install -r requirements.txt to install dependencies.

Now to start the web server, run flask --app src run --debug.

Go to http://localhost:5000 to see the running application.

Note: You must have git and python installed to run these commands.

Note: This project uses PostgresQL as a local database. Please install postgres before trying to login or create an account. Set the admin password (the password for default user 'postgres' as '1234'). Alternatively, you can set whatever password for the postgres admin user that you desire, but be sure to update the *password* variable in the __init__.py file on line 8 located in the src directory.

4.2 Test Cases

To test this project, we simply ran the app to ensure that the user interface and login functions are working correctly.

5 Reflection

In this phase we were able to successfully implement the user interfaces, create the necessary database, and the login functionality. Some things that went well in this phase would be team collaboration and communication, maintaining frequent code review for continuous improvement, keeping the documentation up to date, and error handling. However, we could improve by creating better test coverage, increasing response time, further polishing the user interfaces, and better planning for the inevitable risk factors.

6 Member Contributions

Member Name	Contribution Description	Overall % of Contribution	Note (if applicable)
Jade Lowe	Report formatting, use case diagram, report contribution, team collaboration	16.66	
Dakota Chanthakoummane	Report contribution, team collaboration	16.66	
Aryan Kharva	frontend for the login, signup, and dashboard pages, team collaboration	16.66	
Quintin Obey	Report contribution, sequence diagram, team collaboration	16.66	
Connor Smith	Slotted html, created database, report contribution, team collaboration	16.66	
Edward Tobiasson	Report contribution, team collaboration	16.66	