.

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

September 17, 2024

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to lay out the features of the Potato Properties website. This is accomplished by establishing the requirements of each feature, associated constraints, and providing design diagrams.

1.2 Intended Audience and Use

This document holds valuable information for developers and stakeholders. Developers may utilize this document to further implement features and future patch requirements. While stakeholders may utilize this document to obtain a comprehensive understanding of what is going into the web application and how it functions.

1.3 Project Scope

The scope of the Potato Properties web application encompasses user authentication for multiple users and users types. The management user should be able to manage current and upcoming vacancies, manage tenant information, add/update/remove unit information, store and organize documents, manage payment system, and manage maintenance requirements. The potential 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.

2 Overall Description

2.1 User Needs

All users:

- Create and login to a unique user profile

Management users:

- Manage unit vacancies
- Add/Update/Remove tenant information
- Add/Update/Remove unit information
- Store and organize documents
- Manage payment system
- Manage maintenance requirements

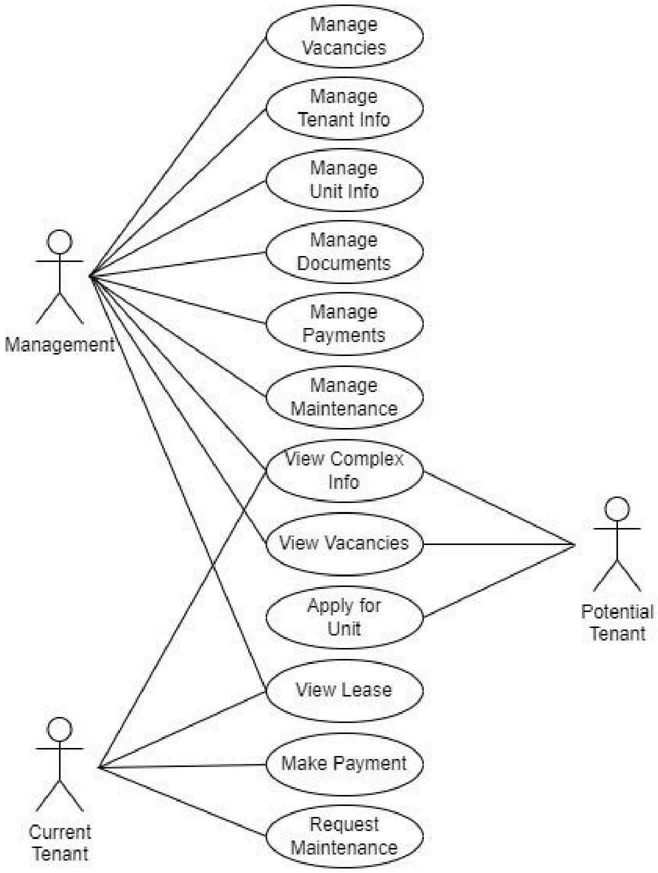
Potential tenant user:

- View apartment complex and unit information
- View current and upcoming vacancies
- Apply for a specific apartment unit

Current tenant user:

- View lease documentation
- Make payments
- Request unit maintenance

2.1.1 Use Case Diagram



2.2 Assumptions and Dependencies

The Potato Properties web application must have a centralized database to store all of the required user information. In addition, this application must utilize a payment gateway to handle financial transactions. It is assumed that users will access the system through a web-base interface.

2.3 Constraints

The website must comply with all required privacy and data protection regulations. This application should be able to efficiently manage increasing numbers of users and units. In addition, the user interface must be accessible for users with varying technical abilities.

3 System Features and Requirements

3.1 User Interface

The interface of the Potato Properties web application is designed and developed with HTML/CSS and Javascript. The website is designed with a focus on accessibility, consistency, and user-friendly navigation. The layout is clean and minimalistic, so users can get the information they need as quickly and efficiently as possible.

The following describes the different pages of the site:

Homepage

The Potato Properties homepage serves as a welcoming entry point for users. It features a clean design with intuitive navigation options. Any of the latest news and updates from the apartment complex will also be shown here, ensuring that they have easy access to the latest information.

Tenant Dashboard

The tenant dashboard is a personalized space for residents that they'll see after logging in with their credentials. In the case that the user is not logged in, they will be prompted to do so before seeing any information. Upon logging in, users are greeted with an overview of their current lease, payments, and upcoming maintenance requests and schedules. The dashboard also includes sections for viewing and paying rent online, updating personal information, and accessing lease documents.

Maintenance Requests

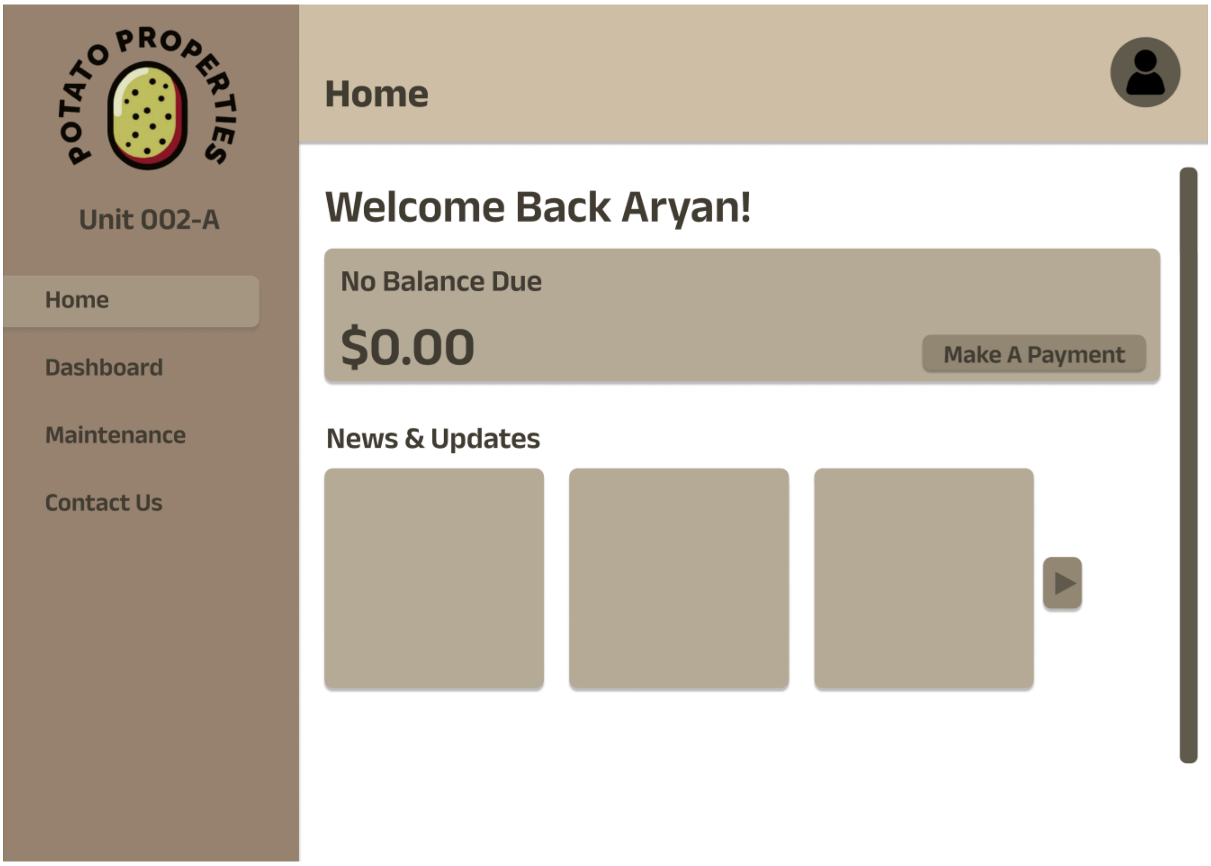
On the Maintenance Requests page, residents are able to submit and manage requests for any repairs or issues they're having in their apartments. Users are greeted with status updates on past requests and ones in the queue. The *Request Maintenance* button focuses on allowing users to describe their issue, the urgency of the issue, and a section to upload any photos as necessary.

Contact Us

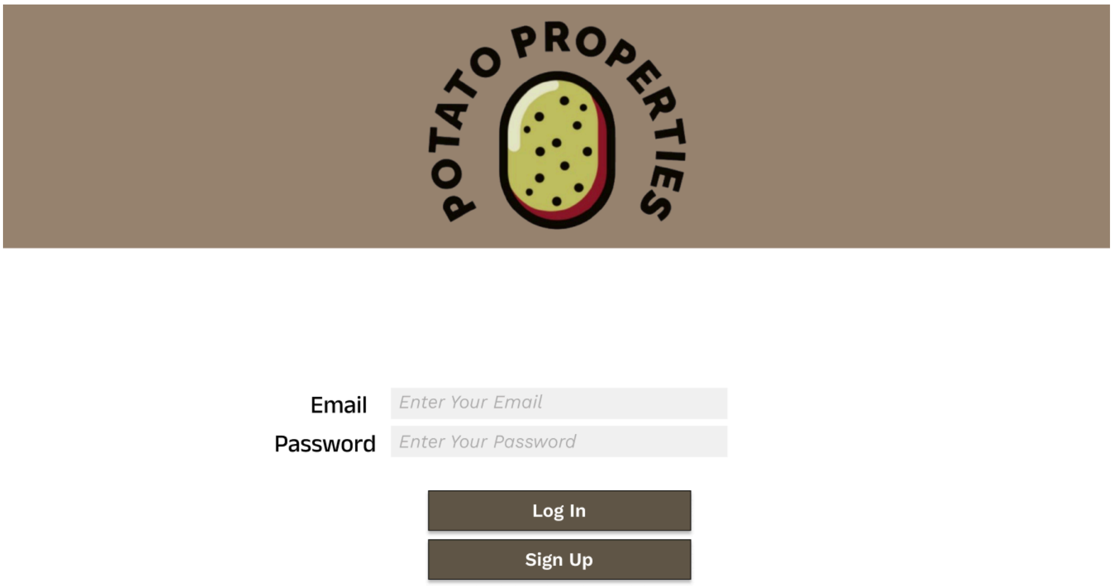
The Contact Us page provides users with multiple options to get in touch with property management. It includes a contact form for general inquiries, email addresses, and phone numbers for specific departments of the apartment complex. The page also features a map of the apartment complex with the property's location and office hours.

3.1.1 Wireframes

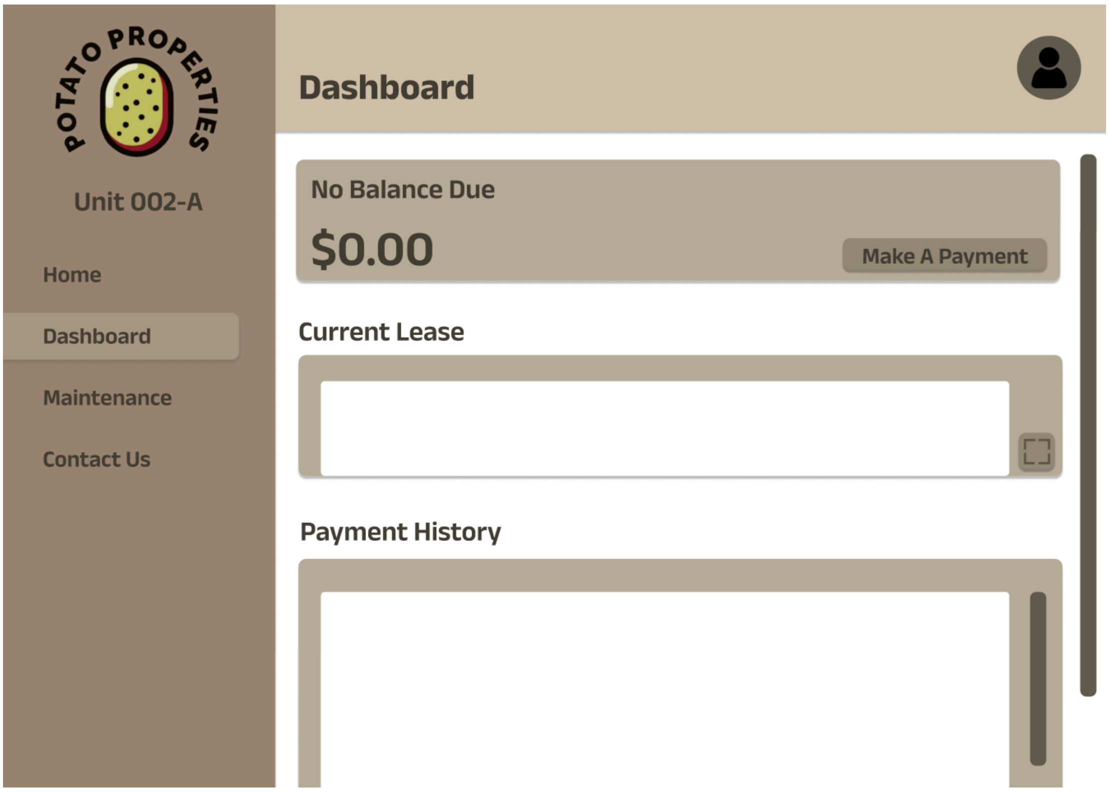
3.1.1.1 Homepage



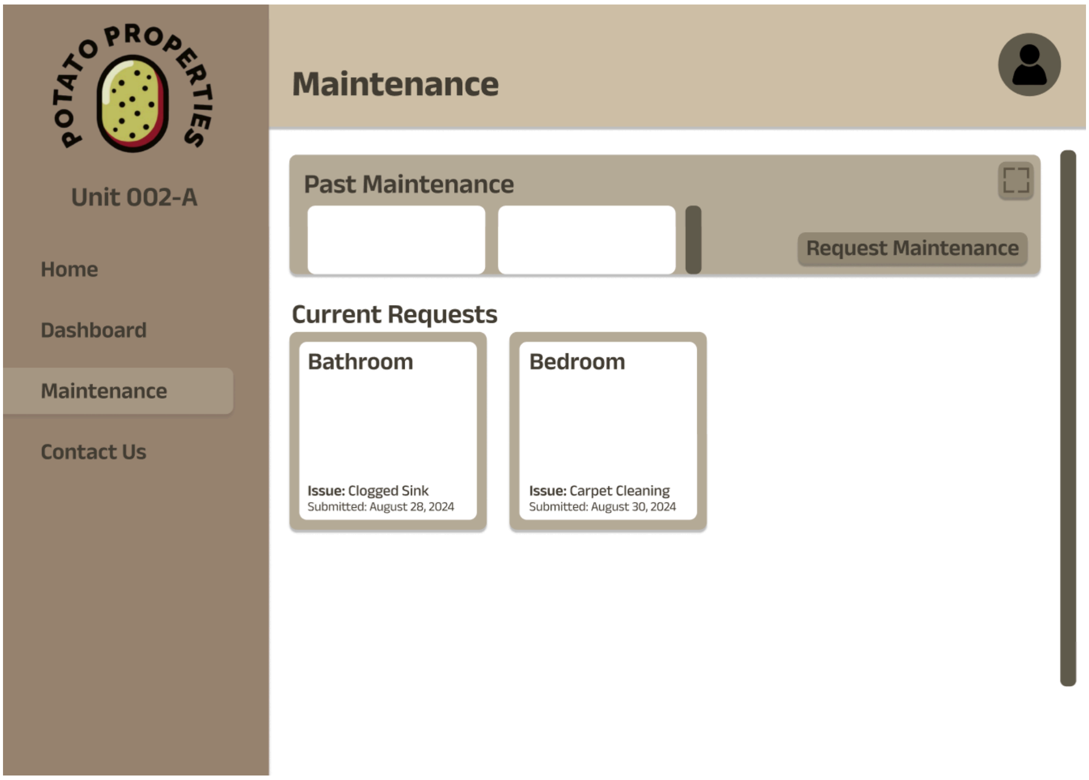
3.1.1.2 Login Page



3.1.1.3 Tenant Dashboard



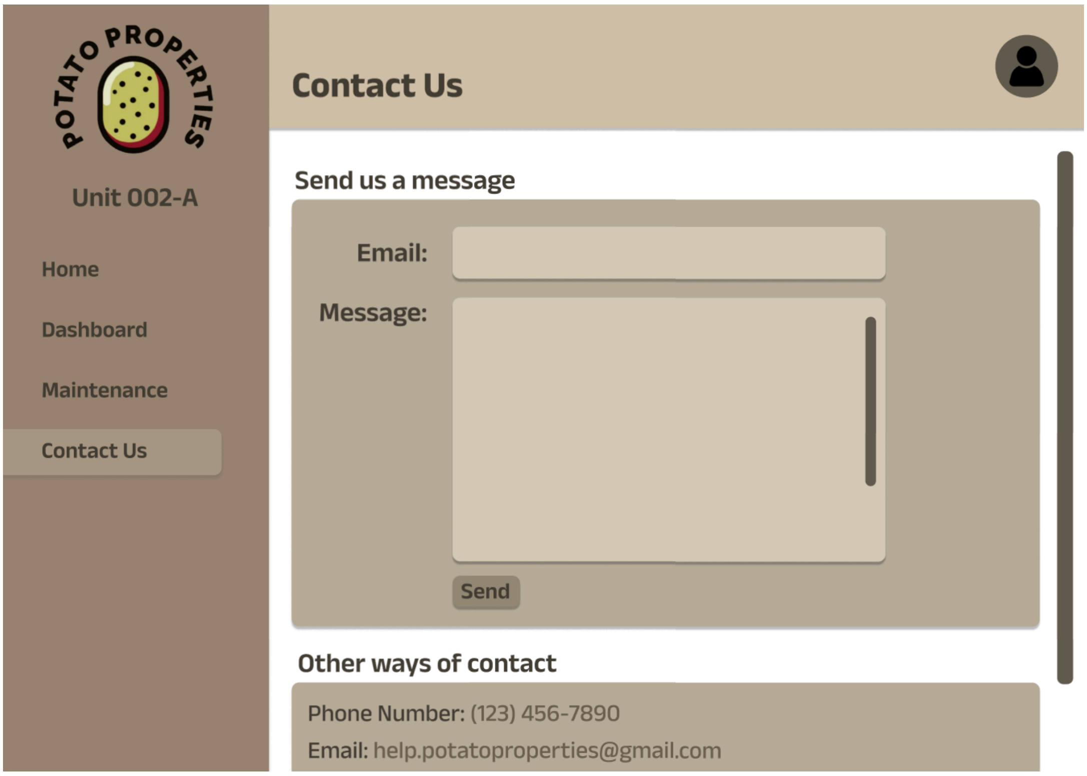
3.1.1.4 Maintenance Request Page



Issue: Clogged Sink
Submitted: August 28, 2024

Issue: Carpet Cleaning
Submitted: August 30, 2024

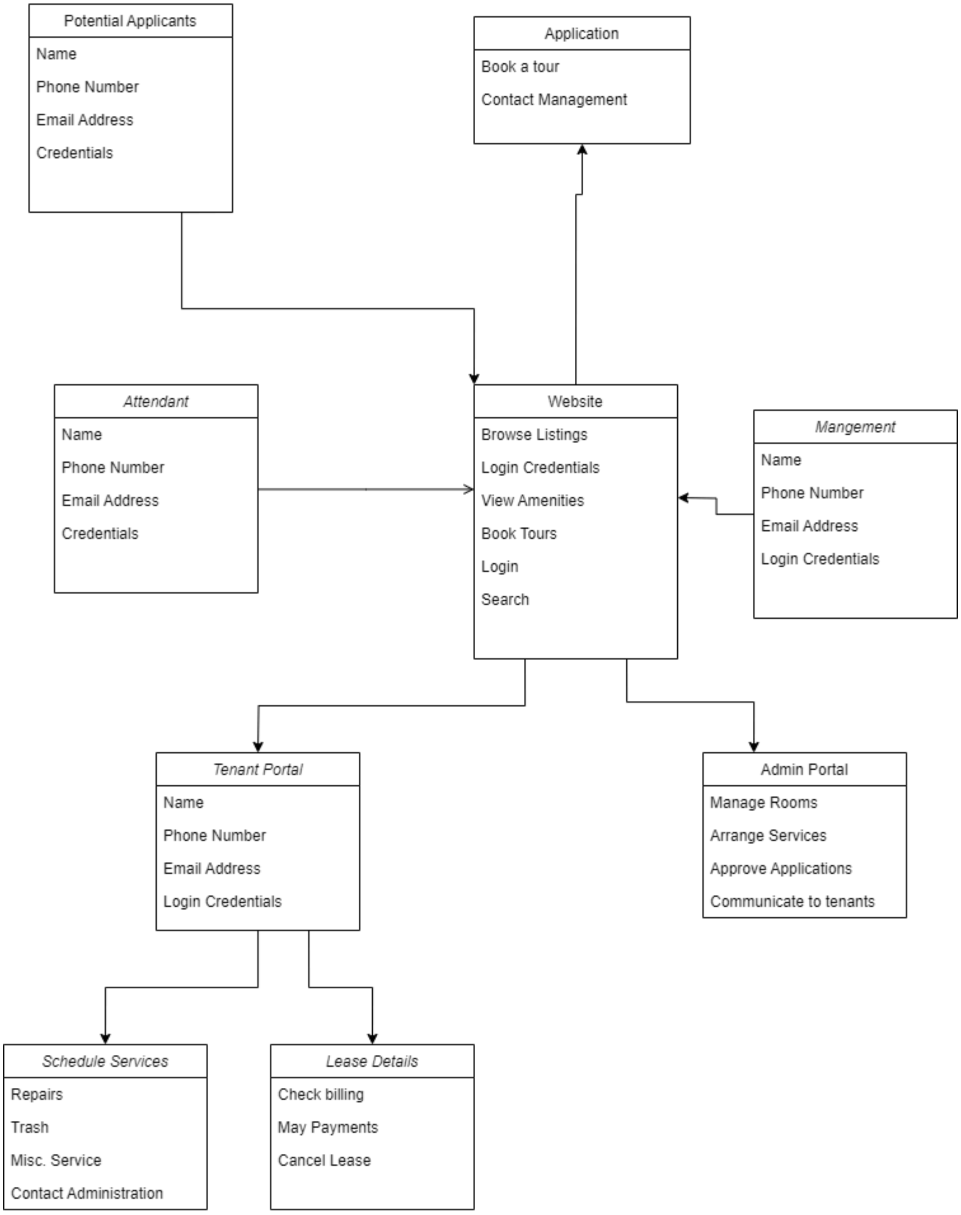
3.1.1.5 Contact Page



Phone Number: (123) 456-7890
Email: help.potatoproperties@gmail.com

3.2 Functional Requirements (by system feature)

3.2.1 UML Diagram



3.2.2 Feature 1

Tennant Administration

3.2.2.1 Purpose

The purpose of this is to compose an administration portal for the facility to operate changes and needs for the establishment.

3.2.2.2 Function

Able to log into the admin portal

3.2.2.3 Function

Real-time availability updates on available rooms, and manage reservations.

3.2.2.4 Function

Track status requests and receive updates

3.2.3 Feature 2

Tenant Dashboard

3.2.3.1 Purpose

To cater to ongoing needs during their stay. Access services and communicate with management

3.2.3.2 Function

Access lease details, rent status, and upcoming maintenance.

3.2.3.3 Function

Availability to request repairs and services.

3.2.4 Feature 3

Potential Tenants

3.2.4.1 Purpose

Showcases the available apartments, listing details on what is available on-site.

3.2.4.2 Function

Schedule tours to view the available listings.

3.2.4.3 Function

Highlight available amenities and the benefits of the establishment.

3.3 System Nonfunctional Requirements

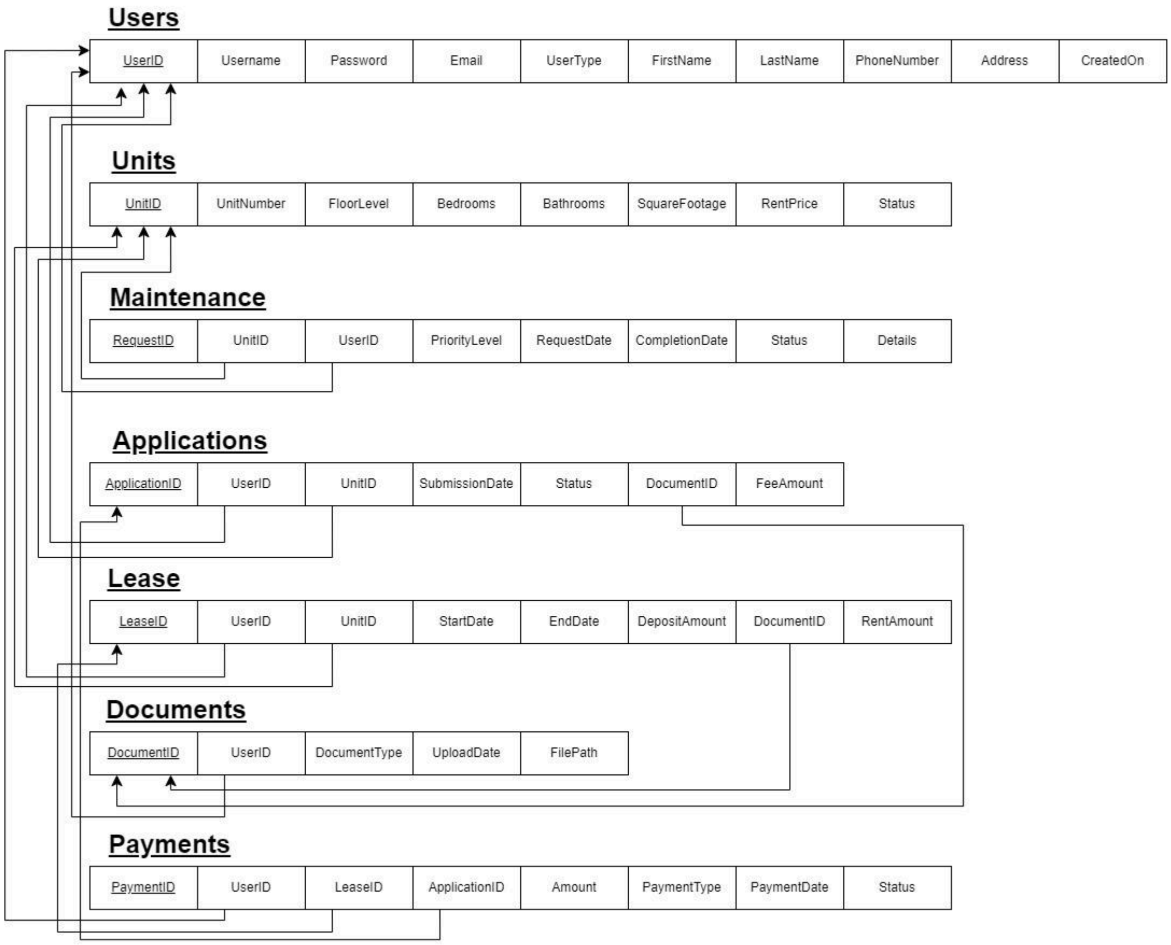
3.3.1 Space Requirements

We will need a set amount of space to host the database of potential and current tenants, as well as an amount of space for the website

3.4 Database

We plan to host the database for the apartment management system on SQL. The database will be a key component in the project to hold important information, possibly including resident attendance, apartment details, maintenance records, billing, and other key elements necessary for efficient apartment management. This approach will help us maintain organized, accurate, and up-to-date records, ultimately enhancing the overall management experience for both administrators and residents.

3.4.1 Database Schema



4 Supporting Information

4.1 Risk Management

For a successful project, it is crucial to anticipate possible risks that may occur throughout the course of the project.

4.1.1 Severe Illness

4.1.1.1 Probability

The probability that there will be a severe illness within our team is medium.

4.1.1.2 Impact

There would be a serious impact on the project should this risk occur.

4.1.1.3 Monitoring

To monitor this risk, our team can initiate immediate communication among group members should one fall ill.

4.1.1.4 Mitigation

To mitigate this risk, our team can meet every few days to assess the conditions of each affected member to analyze progress speed. Should it be necessary, our team can change the requirements for each member of the team or adjust to what is needed depending on the member's condition.

4.1.2 Unfamiliar Technology language or tech software

4.1.2.1 Probability

The probability that there will be unfamiliarity with a technology language or tech software is medium.

4.1.2.2 Impact

There would be a moderate impact on the project should this risk occur.

4.1.2.3 Monitoring

To monitor this risk, our team can implement regular team meetings to check in on how everyone is doing and progressing.

4.1.2.4 Mitigation

To mitigate this risk, our team can provide resources of the current software or language to members who are not as familiar while still assigning light work to practice. In addition, our team can provide members with the most expertise on the subject of lead operation, while also using documentation for members to provide guidance.

4.1.3 Team members leave class or withdraw

4.1.3.1 Probability

The probability that a team member will leave or withdraw from the course is relatively low.

4.1.3.2 Impact

There would be a moderate impact on the project should this risk occur.

4.1.3.3 Monitoring

To monitor this risk, our team can monitor team member absences and project contributions.

4.1.3.4 Mitigation

To mitigate this risk, our team can split up the newly unoccupied tasks equally amongst the remaining team members to ensure that the project can be completed according to the deadlines.

4.1.4 Time Constraints

4.1.4.1 Probability

The probability that our team is met with time constraints through this project is high.

4.1.4.2 Impact

There would be a serious impact on the project should this risk occur.

4.1.4.3 Monitoring

To monitor this risk, our team can implement regular team meetings to ensure that all team members are on track to complete their tasks by the deadlines.

4.1.4.4 Mitigation

To mitigate this risk, our team can prioritize essential functions that would directly impact user experiences. In addition, our team can break the project into manageable phases with clear deadlines for each release. With this approach, we are able to track progress and adjust the workload when necessary.

4.2 Project Timeline

4.2.1 Milestones

4.2.1.1 System Specifications & Requirements - 8/19 - 9/17

- Establish project technologies and software that will be used
  - Backend language, web framework
- Design various web page mockups
  - Admin facing UI mockup
  - Client facing UI mockup

4.2.1.2 Phase I - Admin Focus - 9/18 - 10/29

- Complete Database Schemas for various entities
  - Come to consensus w/ team on needed fields for each entity
- Implement User Authentication
  - Admins should be able to sign in and logout
  - User shouldn't be able to see admin pages without being signed into an admin account
- Core backend functionality working
  - Major operations that should work for the admin side (subject to change):
    - Assigning a user/tenant to a unit
    - Interacting with maintenance requests
    - Seeing any outstanding balances for any users
- Basic UI established; correctly integrated with backend
  - Ensure API endpoints work and data is inserted into the database

4.2.1.3 Phase 2 - Client Focus - 10/30 - 11/18

- Add any additional entities if needed
- Update backend functionality to include added client actions
  - Major operations that should work for the client side (subject to change):
    - Creating a user account
    - Creating a maintenance request
    - Seeing and having the ability to "pay" a bill
    - Canceling a maintenance request
- Tweak / Update UI
  - Enhance the look of UI (make it look better)
- Test backend & client functionality to ensure both sides are still working correctly
  - <u>ALL</u> endpoints are working as intended

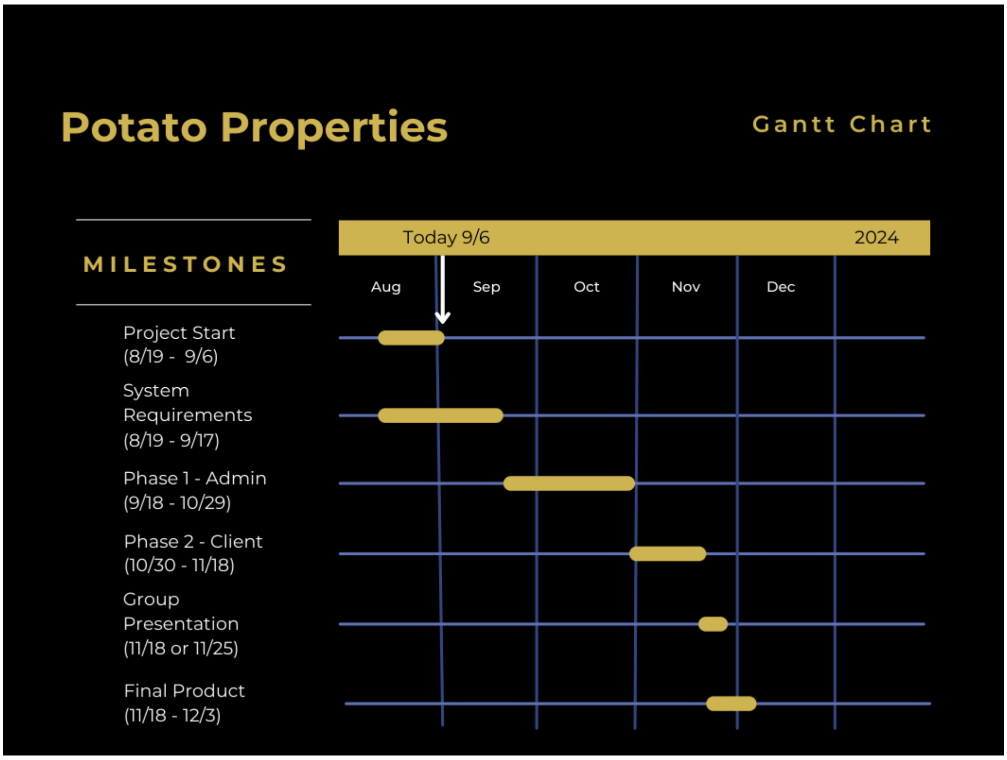
4.2.1.4 Group Presentation - Due 11/18 OR 11/25

- Use feedback to make any small changes for final product

4.2.1.3 Phase 3 - The Final Product - 11/18 - 12/3

- Make any suggested updates (from presentation) to product that are manageable given limited time remaining
- If time is available, potential for hosting web application on a hosting platform
- Add quality of life features if not added already (input validation, 404 screens, etc.)

4.2.2 Gantt Chart



4.3 Incremental Development Plan

Feature	Phase	Task	Description	Technologies
Admin UI & Pages	1	Create pages for the admin facing side of the application.	Base UI for starting admin 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
View Units - Available & Unavailable	1	Allow admin user to view ALL units	The user can view all available & unavailable units.	SQLAlchemy, Python
View Maintenance Requests	1	Allow admin users to view all tenant maintenance requests.	The user can view all maintenance requests made by tenants.	SQLAlchemy, Python
View Outstanding Charges	1	Allow admin users to view all tenant maintenance requests.	The user can view all tenant outstanding charges (who hasn't paid yet).	SQLAlchemy, Python
User Authentication	1	Implement user login	Implement login functionality for management, tenants, and applicants	SQLAlchemy, Python
Unit Management - Viewing a Unit	1	Allow admin user to view a unit	The admin user can view unit associated data, such as which user is currently leasing a unit.	SQLAlchemy, Python
Unit Management - Editing a Unit	1	Allow admin user to edit unit data	The admin user can edit unit data such as removing a tenant from a unit.	SQLAlchemy, Python, SQL
Tenant Management - Viewing Tenants	1	Allow admin users to view a list of tenants.	The admin user can view a list of tenants and select one.	SQLAlchemy, Python
Tenant Management - Viewing tenant account information	1	Allow admin users to view a specific tenant's information.	The admin user can view a specific tenant's account information.	SQLAlchemy, Python
Tenant Management - Deactivating tenant account	1	Allow admin users to deactivate a registered tenant (delete their account).	The admin user can deactivate a tenant's account.	SQLAlchemy, Python, SQL
Tenant UI & Pages	2	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
Maintenance Management	2	Implement maintenance request system	Implement functionality for management to view and manage maintenance requests	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 payment processing	SQLAlchemy, Python
Payment Tracking	2	Implement payment tracking	Allow management to track all payments and tenants to track their payments	SQLAlchemy, Python
Application Submission	2	Implement application submission system	Allow potential tenants to submit their application	SQLAlchemy, Python
Application Tracking		Implement application tracking	Allow potential tenants to track their application	SQLAlchemy, Python
Document Submission	2	Implement document upload and storage system	Allow all users to upload relevant documents	SQLAlchemy, Python
Document Management - Viewing	2	Implement document viewing	Allow management to view all documents and tenants to see relevant documents	SQLAlchemy, Python
Document Management - Assign and organize	2	Implement document assigning and organizing	Allow management to assign documents to a specific user, lease, and unit	SQLAlchemy, Python, SQL
Document Management - Delete	2	Implement document deletion	Allow management to delete documents	SQLAlchemy, Python, SQL
Lease Management - Create	2	Implement lease creation	Allow management to create leases	SQLAlchemy, Python, SQL
Lease Management - Edit	2	Implement lease editing	Allow management to edit and manage leases	SQLAlchemy, Python, SQL
Lease Viewing	2	Display lease information	Allow management to view all leases and tenants to view their lease	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

4.4 Team Roles and Contributions

4.4.1 Team Member Roles

Member Name	Role
Jade Lowe	Project Management; Web Developer, Database Developer
Dakota Chanthakoummane	Web Developer, Database Developer
Aryan Kharva	Web Developer, Database Developer
Quintin Obey	Web Developer, Database Developer
Connor Smith	Web Developer, Database Developer
Edward Tobiasson	Web Developer, Database Developer

4.4.2 Team Member Contributions Table

Member Name	Contribution Description	Overall % of Contribution	Note (if applicable)
Jade Lowe	Project Outline, Introduction Parts 1-3, Overall Description Parts 1-3, Database Schema, References, Incremental Development Plan	16.66	
Dakota Chanthakoummane	Risk Management Part 1-3, Database	16.66	
Aryan Kharva	User Interface, Graphics/Prototypes, Wireframes	16.66	
Quintin Obey	Documentation, Planning, Website Stylesheet	16.66	
Connor Smith	Project Milestones, Gantt Chart, Incremental Development Plan	16.66	
Edward Tobiasson	Risk Management Part 4, Functional Requirements, UML Diagram.	16.66	

4.5 References

Krüger, G., & Lane, C. (2023, January 17). *How to Write an SRS (Software Requirements Specification Document)*. Perforce Software. Retrieved August 28, 2024, from https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document