



## **SOEN 390 - Sprint 4 Documentation**

**April 14th, 2024**

**Department of Computer Science & Software Engineering**

Team Members	
Tanya So Tin Yan	40208954
Fadoua Doghmane	40198495
Asmae Loulidi	40210936
Muiz Madadi	40226708
Shamma Markis	40211998
Ihana Fahmy	40209686
Sonali Patel	40176580
Mehdi Fouzail	40101308
Valeria Dolgaliova	40212218
Robayth Shahrin Dhrubo	40123503

**Concordia University**

**Winter 2024**

# Table of Contents

Software Product Vision.....	4
1. Introduction.....	4
2. Positioning.....	4
3. Stakeholder and User Descriptions.....	4
4. Product Overview.....	10
5. Product Features.....	10
6. Other Product Requirements.....	12
User Stories Backlog.....	14
User Story 1.....	16
User Story 2.....	17
User Story 3.....	17
User Story 4.....	17
User Story 5.....	19
User Story 6.....	19
User Story 7.....	20
User Story 8.....	20
User Story 9.....	21
User Story 10.....	21
User Story 11.....	22
User Story 12.....	22
User Story 13.....	23
User Story 14.....	23
User Story 15.....	24
User Story 16.....	24
User Story 17.....	25
User Story 18.....	25
User Story 19.....	26
User Story 20.....	26
User Story 21.....	27
Software Architecture Document.....	28
Domain Model.....	28
Class Diagram.....	29
Component Diagram.....	30
Deployment Diagram.....	31
Activity Diagrams.....	32
Use Case Diagrams.....	41
Risk Assessment & Management Plan.....	50
Sprint 4 Testing Report.....	53
Short Sprint #4 Retrospective.....	58

Sprint 5 Release Plan.....	60
UI Prototypes.....	62
Home Page.....	62
Login.....	63
Sign Up.....	63
Property Page.....	64
Notification.....	65
Payment.....	65
Maintenance Requests.....	66
Financial Management Dashboard.....	66
Membership Information.....	67
Property Profile Management.....	68
Reservation System.....	69
Employees.....	70
Registration Key Page.....	71
Reservation Page for Condo Company.....	72
Manager Employee Page.....	72
Daily Operation Employee Page.....	73
User Story Coverage Analysis.....	74
Sprint 5 Testing Plan.....	76
Link to Github Testing:.....	79
Code Management.....	80
Deployment.....	95

# Software Product Vision

## 1. Introduction

The purpose of this document is to collect, analyze, and define high-level needs and features of the Urbankey. It focuses on the capabilities needed by the stakeholders, and the target users, and **why** these needs exist. The details of how the Urbankey fulfills these needs are detailed in the use-case and supplementary specifications.

## 2. Positioning

### 2.1 Problem Statement

The problem of	inefficient procedures for managing condominiums
affects	public users, condo owners, rental users, condo management companies
the impact of which is	issues of integration, accessibility and complexity for the users, as well as the lack of essential features.
a successful solution would be	better communication between the users and condo owners, simplified financial and reservation system which will ultimately lead to an increased satisfaction among all stakeholders.

### 2.2 Product Position Statement

For	condo management companies and tenants
Who	need a simple and efficient condominium management system
The Urbankey	is a Condominium Management System
That	enables effective property management, simplified financial tracking, simplified reservations system
Unlike	current alternatives which are missing user-centric features and design
Our product	guarantees a user-friendly experience, catering to the particular requirements of condo owners, tenants, and management companies.

## 3. Stakeholder and User Descriptions

### 3.1 Stakeholder Summary

Name	Description	Responsibilities
Architects and Engineers	Professionals involved in the design of the property, responsible for developing architectural plans and designs for the condominium.	Design architectural layouts that optimize space utilization and adhere to safety regulations. Ensure that the physical structure of the condominium supports the functionality required by the management system.
Tech companies and programmers	Professionals involved in the creation and maintenance of the software of the system, responsible for software development, database management, updates, and maintenance.	<ul style="list-style-type: none"><li>• Software development</li><li>• Database management</li><li>• Updates and maintenance</li><li>• Frontend interfaces, and integrations with third-party services</li></ul>

Financial institutions	Institutions or banks providing loans or financial services, responsible for assessing risks related to lending and investments and conducting credit evaluations.	<ul style="list-style-type: none"> <li>Assess risks related to lending and investments</li> <li>Credit evaluation</li> </ul>
Legal advisors	Professionals ensuring any operations regarding the condos complies with the law, and handle any legal issues related to privacy and data protection.	<ul style="list-style-type: none"> <li>Privacy and Data Protection</li> </ul>

### 3.2 User Summary

Name	Description	Responsibilities	Stakeholder
Condo owners	Individuals who own condominium units - responsible for managing unit details, submitting requests (moving in/out, violation reports, etc.), accessing/updating their property dashboards, reserving common facilities, and engaging in financial transactions.	<ul style="list-style-type: none"> <li>manages unit details</li> <li>submit requests (moving in/out, violation reports, etc.)</li> <li>access/update their property dashboards</li> <li>reserve common facilities</li> <li>engaging in financial transactions</li> </ul>	Condo management companies
Tenants	Individuals occupying condominium units through rental agreements, responsible for occupying and maintaining the rented units and engaging in some financial transactions related to rental payments.	<ul style="list-style-type: none"> <li>occupies and maintains the rented units</li> <li>may engage in some financial transactions related to rental payments</li> </ul>	Condo management companies
Condo management companies	Companies responsible for managing condominium properties, responsible for creating and managing property profiles, uploading condo files, managing financial aspects, setting up reservation systems for common facilities, assigning roles to employees, and handling requests from condo owners/tenants.	<ul style="list-style-type: none"> <li>Create and manage property profiles</li> <li>Upload condo files</li> <li>Manage financial aspects</li> <li>Set up reservation system for common facilities</li> <li>Assign roles to employees</li> <li>Handle requests from condo owners/tenants</li> </ul>	N/A
Managers	Individual responsible for overseeing daily operations of condominium properties, supervising property management tasks, ensuring smooth functioning of common facilities, managing employee roles and responsibilities, and addressing escalated issues from condo owners/tenants.	<ul style="list-style-type: none"> <li>Supervises property management tasks</li> <li>Ensures smooth functioning of common facilities</li> <li>Manages employee roles and responsibilities</li> <li>Addresses escalated issues from condo owners/tenants</li> </ul>	Condo management companies

Operations Managers	Individuals responsible for the day-to-day operations of condominium properties, coordinating maintenance and repairs, ensuring compliance with regulations and standards, overseeing staff scheduling and training, and resolving operational issues and emergencies.	<ul style="list-style-type: none"> <li>Coordinates maintenance and repairs</li> <li>Ensures compliance with regulations and standards</li> <li>Oversees staff scheduling and training</li> <li>Resolves operational issues and emergencies</li> </ul>	Condo management companies
Finance Officers	Individuals responsible for financial aspects of condominiums management, managing condo fees and operational budgets, tracking financial transactions and expenses, generating financial reports, and ensuring financial compliance and accuracy.	<ul style="list-style-type: none"> <li>Manages condo fees and operational budget</li> <li>Tracks financial transactions and expenses</li> <li>Generates financial reports</li> <li>Ensures financial compliance and accuracy</li> </ul>	Condo management companies

### 3.3 User Environment

- The number of people involved might vary, from condo owners or tenants taking care of their apartments to staff members of condo management companies handling several buildings. Depending on the participation of users or the property offers, the user base may grow or shrink.
- Task cycles change based on the type of activity of different users. For example, a condo owner can dedicate more time to financial transactions or managing reservations
- Users access the condo management system through devices such as laptops and smartphones, creating a mostly digital environment.
- Currently, the condo management system is accessible via web browsers. In the future, there are plans to develop dedicated Android and iOS applications, expanding the platform compatibility to include mobile devices.
- Users can use multiple applications, including messaging and communication platforms like email, and financial apps for transactions.

### 3.4 Key Stakeholder or User Needs

Need	Priority	Concerns	Current Solution	Proposed Solution
Better Communication	High	Slow responses and not clear	Emails, phone calls, in-person meeting	One place/page for all messages and notifications
Clearer Finances	High	Confusing fees and hard to understand	Using spreadsheets and done manually	Easy-to-read financial info and clear breakdowns
Easy Booking	High	Difficulty booking and prone to errors	Online booking with real-time updates	Simplified booking process with intuitive interface and personalized recommendations

Quicker Resolution	High	Delays in resolving issues and lack of visibility	Sending emails and making phone calls	Tracking status of requests and receiving updates in real-time
Broadcast Messages	Medium	Messages not reaching everyone and lack of central news hub	Notices and physical notes	Centralized platforms for important announcements and updates
User-friendly interface	High	Difficulty in navigating the system	Complex menus and unintuitive layout	Design an intuitive and responsive interface with clear navigation paths and consistent design elements
Data security	High	Concerns about privacy and data protection	Lack of encryption or data breaches	Implement robust data encryption protocols and regular security audits to ensure user data protection
Mobile accessibility	High	Limited access to features on mobile devices	Limited mobile app functionality or lack of compatibility across devices	Develop a responsive mobile application with feature parity and seamless user experience across different devices
Timely notifications	Medium	Missed updates or important events	Manual checking of emails or notifications	Enable push notifications and customizable alert settings to ensure timely updates and reminders
Streamlined workflows	Medium	Inefficient or redundant processes	Manual data entry and disjointed workflows	Automate repetitive tasks and streamline processes to improve efficiency and productivity

***Table 1: Key Stakeholders' and User's Needs***

Description	Centralized messaging system	Description	User-friendly financial module
Type	System enhancement	Type	System Enhancement
Responsibilities	Develop system for sending/receiving messages and notification	Responsibilities	Design financial module with clear fee breakdowns
Success Criteria	Easily accessible messages for stakeholders	Success Criteria	Understandable financial information
Involvement	Product manager, development team	Involvement	Product manager, financial analyst and development team
Deliverables Comments/Issues	Ensure compatibility across various devices. Address scalability concerns to accommodate potential increases in user base and message volume.	Deliverables Comments/Issues	Ensure scalability for future updates and additions. Implement features for exporting financial data in multiple formats to facilitate analysis and auditing processes.
<b><i>Table 2: Better Communication</i></b>		<b><i>Table 3: Clearer Financial Information</i></b>	

Description	Intuitive booking system	Description	Real-time request tracking
Type	New feature	Type	System enhancement
Responsibilities	Develop user-friendly booking system with personalized recommendations	Responsibilities	Implement system for real-time tracking and updates on requests
Success Criteria	Simplified, error-free booking process	Success Criteria	Easy request tracking and timely updates

<b>Involvement</b>	Product manager, UX/UI designer
<b>Deliverables Comments/Issues</b>	Ensure integration with existing reservation system. Address compatibility issues with different request types to provide a seamless booking experience for users.

*Table 4: Easy Booking*

<b>Involvement</b>	Product manager and development team
<b>Deliverables Comments/Issues</b>	Implement features for tracking the status of requests and providing real-time updates to users. Ensure compatibility with various communication channels (email, SMS, etc.) to reach users effectively.

*Table 5: Quicker Resolution*

<b>Description</b>	Centralizes messaging platform
<b>Type</b>	New feature
<b>Responsibilities</b>	Develop platform for broadcasting announcements to all users
<b>Success Criteria</b>	Effective communication of announcements
<b>Involvement</b>	Product Manager, Communication specialist and development team
<b>Deliverables Comments/Issues</b>	Ensure scalability for large user base. Implement features for scheduling and targeting broadcast messages to specific user groups or individuals.

*Table 6: Broadcast Message*

<b>Description</b>	Design an intuitive and responsive interface with clear navigation paths and consistent design elements
<b>Type</b>	UI/UX Design
<b>Responsibilities</b>	Designing user interface elements, navigation flows, and visual design
<b>Success Criteria</b>	User satisfaction with interface usability and navigation
<b>Involvement</b>	UI/UX Designers, Developers
<b>Deliverables Comments/Issues</b>	Ensure compliance with accessibility standards. Address performance issues related to interface responsiveness and loading times to enhance user experience.

*Table 7: User-friendly interface*

<b>Description</b>	Implement robust data encryption protocols and regular security audits to ensure user data protection
<b>Type</b>	Security
<b>Responsibilities</b>	Implementing encryption algorithms, securing databases, and conducting security audits
<b>Success Criteria</b>	Absence of data breaches and compliance with privacy regulations
<b>Involvement</b>	Security Experts, Developers
<b>Deliverables Comments/Issues</b>	Ensure adherence to industry security standards. Implement features for regular security audits and vulnerability assessments to identify and mitigate potential security risks.

*Table 8: Data Security*

<b>Description</b>	Develop a responsive mobile application with feature parity and seamless user experience across different devices
<b>Type</b>	Development
<b>Responsibilities</b>	Developing mobile application features and ensuring cross-device compatibility
<b>Success Criteria</b>	User engagement and satisfaction with mobile app functionality
<b>Involvement</b>	Developers, UI/UX Designers
<b>Deliverables Comments/Issues</b>	Ensure optimization for various screen sizes and resolutions. Address compatibility issues with different mobile platforms (Android, iOS, etc.) to reach a wider audience of mobile users.

*Table 9: Mobile accessibility*

<b>Description</b>	Enable push notifications and customizable alert settings to ensure timely updates and reminders	<b>Description</b>	Automate repetitive tasks and streamline processes to improve efficiency and productivity
<b>Type</b>	Notification	<b>Type</b>	Workflow Improvement
<b>Responsibilities</b>	Implementing push notification functionality and user preference settings	<b>Responsibilities</b>	Analyzing existing processes, identifying automation opportunities, and implementing workflow improvements
<b>Success Criteria</b>	Users receiving timely notifications and reminders	<b>Success Criteria</b>	Increased efficiency and productivity, reduced manual intervention
<b>Involvement</b>	Developers, Product Managers	<b>Involvement</b>	Business Analysts, Developers
<b>Deliverables</b> <b>Comments/Issues</b>	Ensure compatibility across different notification channels. Implement user training and adoption strategies to encourage users to enable and customize notification settings.	<b>Deliverables</b> <b>Comments/Issues</b>	Address usability issues related to workflow automation features to ensure intuitive user interaction. Implement features for monitoring and analyzing workflow efficiency to identify areas for further optimization.
<i><u>Table 10: Timely notifications</u></i>		<i><u>Table 11: Streamlined workflows</u></i>	

### 3.5 Alternatives and Competition

An alternative for the stakeholders would be to build their own management system. They could also buy an existing condo management system, add additional features and maintain it.

#### **Alternatives:**

- Build their own management system
- Buy an existing one, implement additional features and maintain it

These alternatives would allow the stakeholders to have full control on the system.

#### **Competition:**

- Existing condo management systems
- Real estate firms

Some companies that offer competition in the condo management systems market:

**Buildium:** Buildium offers a comprehensive property management solution designed for property managers, landlords, and homeowner associations. It includes features such as accounting, lease tracking, maintenance requests, and online payments.

**AppFolio Property Manager:** AppFolio is a cloud-based property management software that caters to residential, commercial, student housing, and community association property managers. It provides features like online rent payments, maintenance requests, and vacancy advertising.

**Condo Control Central:** Condo Control Central provides web-based property management software designed specifically for condominiums and HOAs. It offers features such as visitor management, amenity booking, maintenance tracking, and document management.

**TOPS Software:** TOPS Software offers a suite of community management solutions, including software for condominiums, homeowner associations, and co-ops. Their products include features like accounting, communication tools, architectural request management, and community websites.

**Yardi Voyager:** Yardi Voyager is a property management platform that serves various real estate sectors, including residential, commercial, and condominium management. It offers modules for accounting, leasing, maintenance, and reporting tailored to the specific needs of each property type.

To distinguish itself as a top condo management software from competitors such as Buildium, AppFolio Property Manager, Condo Control Central, TOPS Software, and Yardi Voyager, UrbanKey will focus on a smooth user experience with an intuitive UI and configurable functionality. UrbanKey would excel in financial management by providing full tools for fee tracking, budgeting, and transparent reporting. Mobile accessibility and strong data security safeguards would enable easy access and user data safety. UrbanKey will also innovate by offering unique features such as enhanced amenities booking and smart home integration, as well as superior customer service, to create user happiness and loyalty.

## 4. Product Overview

This section presents both product perspective and assumptions and dependencies of the Urbankey.

### 4.1 Product Perspective

The purpose of the condo management app and website is to simplify condo management procedures by activating as separate, self-contained platforms. The application enables users to interact with it, while the website provides further features that can be accessed by web browsers. User profiles, finance systems, property management tools, reservation capabilities, and request submission features are important parts. These platforms work together flawlessly to give management firms, condo owners, and renters all the resources they need to manage properties effectively. Widespread accessibility is ensured via external interfaces' interaction with several operating systems.

### 4.2 Assumptions and Dependencies

The availability of registration keys from management organizations, correct property data input, and the smooth operation of the finance and reservation systems are some of the assumptions and dependencies for the condo management system. Any modifications to these elements may have an impact on the characteristics listed in the Vision report. Furthermore, assumptions on the app's compatibility with multiple platforms, language options, and login methods like Gmail or Single Sign-On are critical, since changes may need adjustments to the Vision document.

## 5. Product Features

### 5.1 User Profile

Public users can create their own unique profile. This profile should include a profile picture, user name, contact email, phone number. Moreover, public users are required to provide a registration key obtained by their condo

management company to become a condo owner. To become rental users in the system, public users must input a registration key obtained from their condominium management company.

## **5.2 Condo Owner Dashboard**

Condo owner can view features of their properties such as general information, personal profile, condo information, financial status, status of the submitted request, etc in a dashboard

## **5.3 Condo Management Companies Profile**

The property profile in the system requires a property name, unit count, parking count, locker count, and address. Condo management companies can upload files for each property, and those files are accessible to all condo owners. They can include detailed information about condo units, parking spots, and lockers, encompassing unit size, owner details, occupant information, and associated condo fees. Furthermore, management companies can send registration keys to unit owners or rental users, allowing them to link their profiles with specific condo units.

## **5.4 Financial System**

Management companies input condo fees per square foot and parking spot. Condo fees for each unit are calculated and presented to owners, recorded in the financial system, along with operational budgets and costs. An annual report can be generated, summarizing condo fee collections for the year. The system also includes a reservation feature for common facilities, like a sky lounge or spa fitness. Condo owners and rental users can use a calendar-like interface to reserve facilities, with real-time availability display. Reservations operate on a first-come-first-serve basis, rendering a facility unavailable once booked.

## **5.6 Reservation System**

The condominium management system features a simple reservation system where condo management companies establish reservations for common facilities such as a sky lounge and a spa fitness center. This system is presented in a calendar-like interface, and allows both condo owners and renters to reserve these common facilities. Availability for these facilities is displayed, and reservations are processed on a first-come-first-serve basis. Once a facility is booked, it becomes temporarily unavailable for the reserved duration.

## **5.7 Reservation System**

Condo management firms have the ability to assign distinct roles to various employees overseeing the same property. These roles may include a manager, responsible for day-to-day operations, and an employee handling financial responsibilities.

## **5.8 Requests**

Condo owners can submit various requests, such as move-in/out dates for reserving elevators, intercom changes, access requests for fobs or keys, reporting violations, highlighting deficiencies in common areas, or seeking information. Each request is directed to the appropriate employee based on its type.

## 5.9 Notifications

Every user has a notifications page where they can view the most recent activities related to their submitted or assigned requests.

## 6. Other Product Requirements

### Standards, Hardware, or Platform Requirements

The software product must adhere to industry-standard security protocols to ensure the protection of user data. Additionally, it should be compatible with widely used operating systems including Android, iOS, Linux, MacOS, or Windows.

### Performance Requirements

Response time for critical user interactions, such as profile creation and reservation submissions, should be within 2 seconds to ensure a seamless experience. And system uptime must be maintained at 99.9% to minimize service disruptions, with downtime limited to scheduled maintenance windows.

### Environmental Requirements

The system should be built with optimal resource use to save CPU and battery consumption, and it should work well on both desktop and mobile devices. It is important to take into account how much bandwidth is used, particularly for customers who have spotty internet access, and strive for data transfer speeds that are suited for 3G and 4G networks.

### Quality Ranges

- **Performance:** average response time, peak response time, and throughput are important variables to consider when optimizing response times for effective user interactions.
- **Robustness:** metrics like error rates, crash frequencies, and mean time to failure (MTTF) should be used to gauge how well the system withstands unforeseen mistakes and handles heavy traffic loads.
- **Fault tolerance:** recovery time objective (RTO) and recovery point objective (RPO) metrics, together with error recovery and data backup mechanisms, should be in place to guarantee system dependability. The RTO provides the maximum permissible delay for system restoration following an incident, while RPO establishes the maximum allowable data loss in the case of a failure or interruption.
- **Usability:** tested by usability experts and user satisfaction questionnaires, the interface should be simple to use, easy to navigate, and provide insightful feedback.

### Design Constraints and Dependencies

Compatibility testing may be necessary to ensure smooth data interchange, and integration with current databases and condo management software may be necessary. Adherence to legal and regulatory obligations for privacy and data processing, with frequent audits carried out to confirm compliance with standards.

### Documentation Requirements

To guarantee widespread adoption, thorough user manuals and online help resources covering subjects like account creation, feature usage, and troubleshooting techniques should be made available. Accessibility metrics

for the material should also be monitored. In order to maintain uniformity across platforms, labeling and packaging regulations may incorporate branding rules.

## Priority and Attributes

- **Stability:** maintaining user trust and satisfaction requires a sturdy and reliable system.
- **Benefit:** with key performance indicators set up to track the effects of features on user happiness and productivity, the system's features should offer users observable advantages like increased efficiency and communication
- **Effort:** ensure that the project deadlines are fulfilled, development efforts should be concentrated on integrating key features and guaranteeing peak performance. Metrics for allocating resources, such as development hours and job completion rates, should be tracked.
- **Risk:** risk assessment matrices are used to prioritize risk in order to mitigate the risks associated with data security breaches and system failures, which are crucial for safeguarding user information and preserving business continuity.
- **Communication:** Establishing clear and effective communication channels between stakeholders, development teams, and end-users is essential for ensuring alignment of goals and expectations throughout the project lifecycle. Regular status updates, meetings, and feedback sessions should be conducted to facilitate transparent communication and foster collaboration. Additionally, providing user-friendly interfaces for communication within the system can enhance user engagement and satisfaction.

# User Stories Backlog

## *User Story #1*

As a public user, I want to personalize my profile with a picture, username, and contact details for community engagement.

[Issue #43](#)

## *User Story #2*

As a public user, I want to use a registration key from the management company to securely become a condo owner in the system.

[Issue #44](#)

## *User Story #3*

As a public user, I want to use a registration key from the management company to securely become a rental user in the system.

[Issue #45](#)

## *User Story #4*

As a condo owner, I want a comprehensive dashboard displaying property and financial details, allowing efficient property management.

[Issue #18](#)

[Issue #46](#)

## *User Story #5*

As a property manager, I want to create a property profile with essential details (property name, unit count, parking count, locker count, address) for accurate property management.

[Issue #17](#)

[Issue #30](#)

[Issue #47](#)

## *User Story #6*

As a property manager, I want to upload condo files for each property so that all condo owners can access important information, enhancing transparency and communication.

[Issue #48](#)

## *User Story #7*

As a property manager, I want to enter detailed information for each condo unit, parking spot, and locker, including unit ID, size, owner details, occupant information, and condo fees, ensuring a comprehensive and accurate property database.

## [Issue #49](#)

### *User Story #8*

As a property manager, I want to be able to generate and send registration keys to unit owners and rental users, so that they can link a condo unit to their profile.

## [Issue #50](#)

### *User Story #9*

As a property manager, I want to enter condo fee per square foot and per parking spot, so that I can accurately calculate the fee for each condo unit.

## [Issue #51](#)

### *User Story #10*

As a property manager, I want to calculate and present the fee for each unit, so that I can present it to the unit owner.

## [Issue #52](#)

### *User Story #11*

As a financial system agent, I want the system to record both the operational budget and the associated costs for each operation, so that we can maintain transparent financial management.

## [Issue #53](#)

### *User Story #12*

As a financial system agent, I want to generate an annual report summarizing all the fees collected for a given year.

## [Issue #54](#)

### *User Story #13*

As a property manager, I want to set up a system for managing reservations of common facilities, to allow residents to schedule and access these amenities.

## [Issue #55](#)

### *User Story #14*

As a condo owner/rental user, I want to be able to reserve common facilities within the complex through a calendar-like interface, to efficiently plan my activities.

## [Issue #56](#)

### *User Story #15*

As a condo owner/rental user, I want to know the availability of common facilities, so that I can easily plan and book the shared amenities for my personal or guest use.

## [Issue #57](#)

*User Story #16*

As a property manager, I want the reservation system to operate on a first-come-first-serve basis so that once a facility is booked, it becomes unavailable for the reserved time and ensures fair access to common amenities.

[Issue #58](#)

*User Story #17*

As a property manager, I want to be able to set up different roles (e.g., manager, finance) for employees who are responsible for the same property so that the efficiency of assigning and managing tasks within the team becomes more organized.

[Issue #59](#)

*User Story #18*

As a condo owner, I want to submit requests for tasks like moving, intercom changes, access items, violation reports, common area issues, and questions so that addressing needs becomes easy and convenient.

[Issue #60](#)

*User Story #19*

As a property manager, I want each request to be assigned to the appropriate employee based on the type of request so that the resolution process for residents remains smooth and efficient.

[Issue #61](#)

*User Story #20*

As a public user, I want to have a notification page where I can view the latest activities related to my submitted or assigned requests so that I can stay informed on the progress and updates of the condo.

[Issue #62](#)

*User Story #21*

As a user, I want to be able to navigate the website efficiently so that I can access the information I need quickly.

[Issue #62](#)

## User Story 1

001	Public User Profile Creation					
<b>As</b> a public user						
<b>I want</b> to personalize my profile with a picture, username, and contact details						
<b>Because</b> I am looking for community engagement						
User Profile Feature						
Must	Bus. Value: M	Risk: M	Effort: 5			
Status						
Front-end: Completed						
Backend: Completed						

## User Story 2

002	Becoming a Condo Owner					
<b>As</b> a public user						
<b>I want</b> to use a registration key from the management company						
<b>To</b> securely become a condo owner in the system						
Registration Key Feature						
Must	Bus. Value: H	Risk: M	Effort: 6			
Status						
Front-end: Completed						
Backend: Completed						

## User Story 3

003	Becoming a Rental User		
<b>As</b> a public user			
<b>I want</b> to use a registration key from the management company			
<b>To</b> securely become a rental user in the system.			
Registration Key Feature			
Must	Bus. Value: H	Risk: M	Effort: 6

Status

Front-end: Completed

Backend: Completed

## User Story 4

004	Condo Owner Profile		
<b>As</b> a condo owner			
<b>I want</b> a comprehensive dashboard displaying property and financial details			
<b>Because</b> it allows for efficient property management.			
Condo Owner Dashboard Feature			
Must	Bus. Value: H	Risk: M	Effort: 8

Status

Front-end: Completed

Backend: Completed

## User Story 5

005	Property Profile Creation					
<b>As</b> a property manager						
<b>I want</b> to create a property profile with essential details (property name, unit count, parking count, locker count, address)						
<b>For</b> accurate property management						
Condo Management Company Feature						
Must	Bus. Value: H	Risk: M	Effort: 8			
<b>Status</b>						
Front-end: Completed						
Backend: Completed						

## User Story 6

006	File Upload for Condo Management Companies					
<b>As</b> a property manager						
<b>I want</b> to upload condo files for each property so that all condo owners can access important information						
<b>Because</b> I would like to enhance transparency and communication						
File Upload Feature						
Could	Bus. Value: M	Risk: M	Effort: 5			
<b>Status</b>						
Front-end: Completed						
Backend: Completed						

## User Story 7

007	Condo Management Company Information					
<b>As a property manager</b>						
<b>I want to</b> enter detailed information for each condo unit, parking spot, and locker, including unit ID, size, owner details, occupant information, and condo fees						
<b>To</b> ensure a comprehensive and accurate property database						
Condo Management Company Feature						
Must	Bus. Value: H	Risk: M	Effort: 8			

Status

Front-end: Completed

Backend: Completed

## User Story 8

008	Generate and send registration code					
<b>As a property manager</b>						
<b>I want to</b> be able to generate and send registration keys to unit owners and rental users						
<b>To</b> link a condo unit to the user's profile.						
Registration Key Feature						
Must	Bus. Value: M	Risk: H	Effort: 5			

Status

Front-end: Completed

Backend: Completed

## User Story 9

009	Enter condo unit fee					
<b>As</b> a property manager						
<b>I want to</b> enter condo fees per square foot and per parking spot						
Must	Bus. Value: M	Risk: H	Effort: 3			

Status

Front-end: Completed

Backend: Completed

## User Story 10

010	Calculate condo unit fee					
<b>As</b> a property manager						
<b>I want to</b> calculate and present the fee for each unit,						
Must	Bus. Value: M	Risk: M	Effort: 8			

Status

Front-end: Completed

Backend: Completed

## User Story 11

011	Record operational budget					
<b>As</b> a financial system agent						
<b>I want</b> the system to record both the operational budget and the associated costs for each operation						
<b>To</b> maintain transparent financial management.						
Operational Budget Feature						
Should	Bus. Value: M	Risk: H	Effort: 5			

Status

Front-end: Completed

Backend: Completed

## User Story 12

012	Generate annual report					
<b>As</b> a financial system agent						
<b>I want to</b> generate an annual report summarizing all the fees collected for a given year						
<b>To</b> organize the finances for each year.						
Annual Report Feature						
Should	Bus. Value: M	Risk: H	Effort: 13			

Status

Front-end: Completed

Backend: Completed

## User Story 13

013	Set up facility reservation system					
<b>As</b> a property manager						
<b>I want to</b> set up a system for managing reservations of common facilities						
<b>To</b> allow residents to schedule and access these amenities.						

### Status

Front-end: Completed

Backend: Completed

## User Story 14

014	Reserve common facilities					
<b>As</b> a condo owner/rental user						
<b>I want to</b> be able to reserve common facilities within the complex through a calendar-like interface						
<b>To</b> efficiently plan my activities.						
Facility Reservation Feature						
Must	Bus. Value: M	Risk: M	Effort: 5			

### Status

Front-end: Completed

Backend: Completed

## User Story 15

015	Knowing the Common Facilities' Availability					
<b>As</b> a condo owner/rental user						
<b>I want</b> to know the availability of common facilities						
<b>So</b> that I can easily plan and book the shared amenities for my personal or guest use						

Status

Front-end: Completed

Backend: Completed

## User Story 16

016	First-Come-First-Serve System					
<b>As</b> a property manager						
<b>I want</b> the reservation system to operate on a first-come-first-serve basis						
<b>So</b> that once a facility is booked, it becomes unavailable for the reserved time and ensures fair access to common amenities						
Reservation System Feature						
M	Bus. Value: M	Risk: M	Effort: 6			
Status						
Front-end: Completed						
Backend: Completed						

## User Story 17

017	Setting Up Different Roles					
<b>As</b> a property manager						
<b>I want</b> to set up different roles (e.g., manager, finance) for employees who are responsible for the same property						
<b>So</b> that the efficiency of assigning and managing tasks within the team becomes more organized						

Status

Front-end: Completed  
Backend: Completed

## User Story 18

018	Submission of Requests					
<b>As</b> a condo owner						
<b>I want</b> to submit requests for tasks like moving, intercom changes, access items, violation reports, common area issues, and questions						
<b>So</b> that addressing needs becomes easy and convenient						
Request Feature						
M	Bus. Value: M	Risk: M	Effort: 6			
Status						
Front-end: Completed Backend: In Progress						

## User Story 19

019	Request Assignment					
<b>As</b> a property manager						
<b>I want</b> each request to be assigned to the appropriate employee based on the type of request						
<b>So</b> that the resolution process for residents remains smooth and efficient						
Request Feature						
M	Bus. Value: H	Risk: H	Effort: 8			

Status

Front-end: Completed

Backend: In Progress

## User Story 20

020	Notification Page					
<b>As</b> a public user						
<b>I want</b> to have a notification page where I can view the latest activities related to my submitted or assigned requests						
<b>So</b> that I can stay informed on the progress and updates of the condo						
Notification Page Feature						
C	Bus. Value: M	Risk: M	Effort: 7			

Status

Front-end: Completed

Backend: In Progress

## User Story 21

027	Navigation Bars
As a user	
I want to be able to navigate the website efficiently	
Because I want to quickly access the information I need.	
Navigation Bars Feature	
Should	Bus. Value: H
Risk: M	Effort: 8

Status

Front-end: Completed

Backend: Completed

We have removed User Stories #22 to #27 seeing as those are additional and optional features that are not part of the main requirements.

# Software Architecture Document

## Domain Model

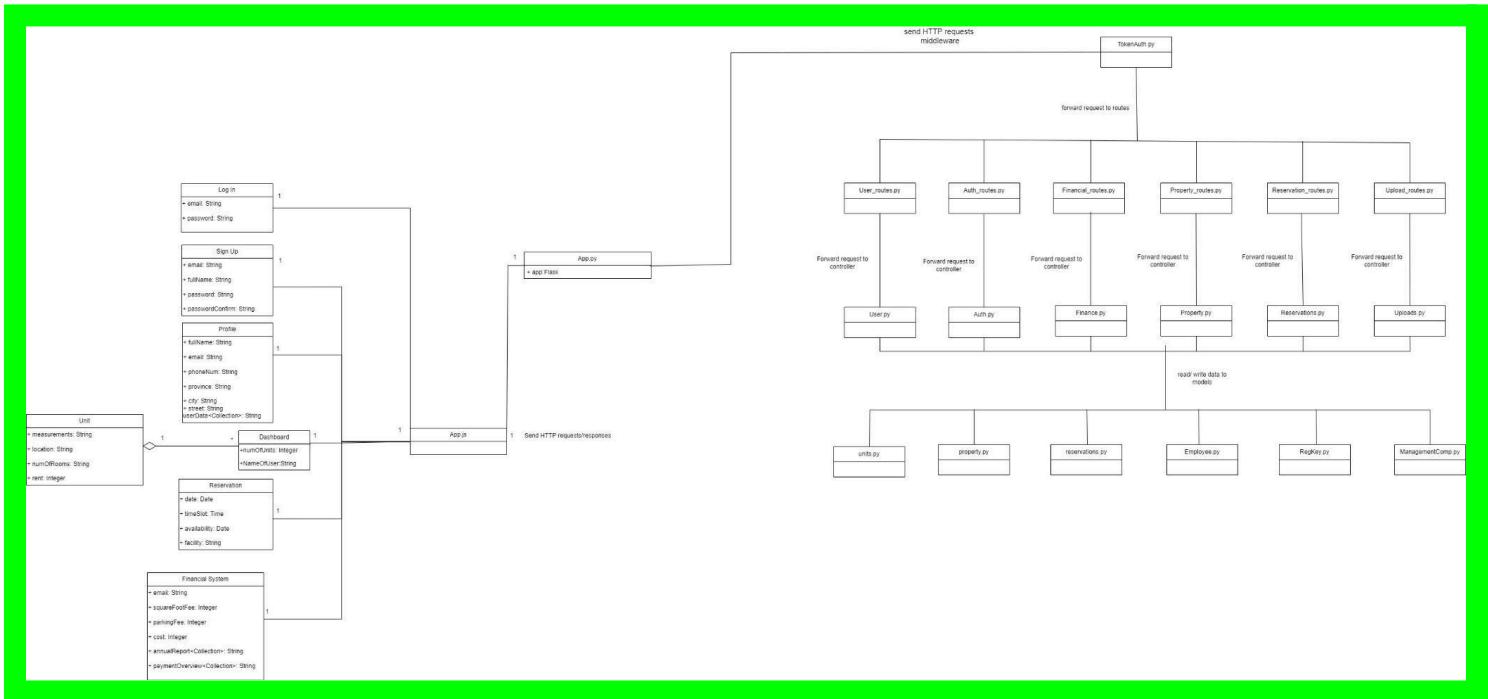


Figure 1: Domain Model

# Class Diagram

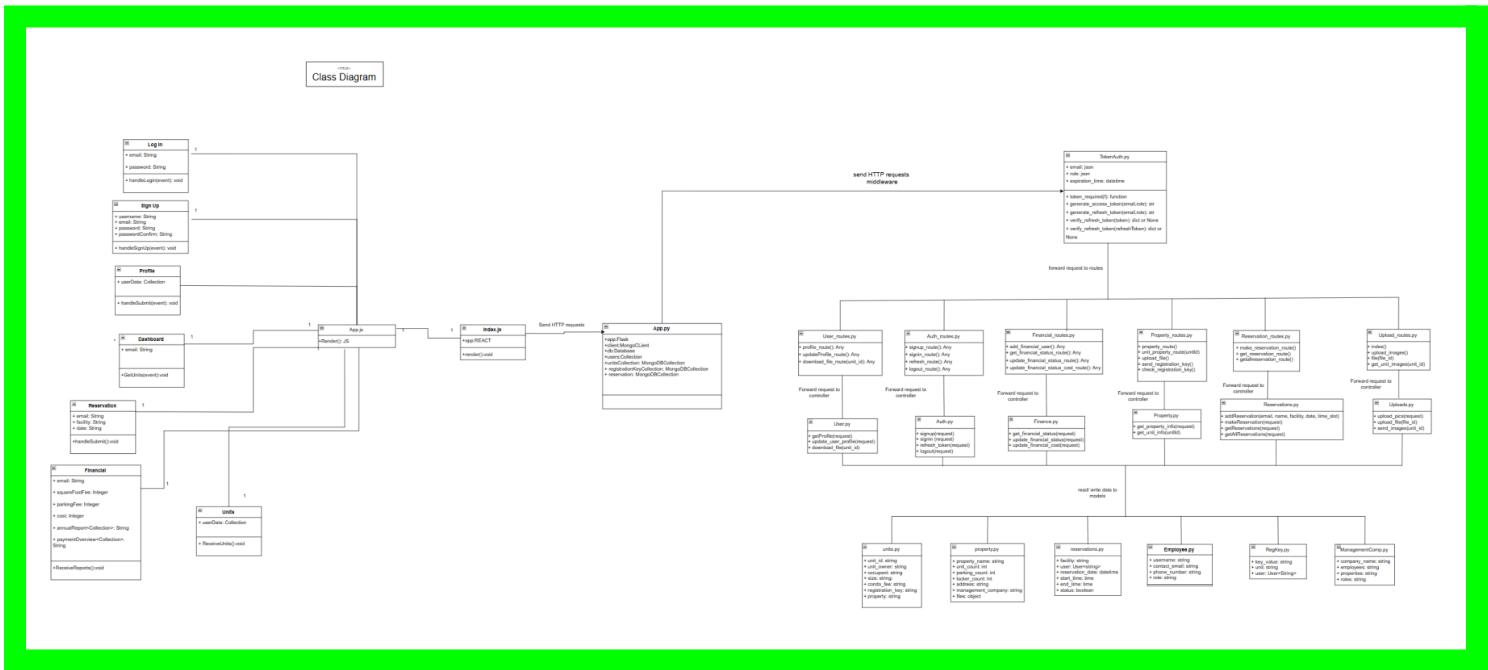


Figure 2: Class Diagram

# Component Diagram

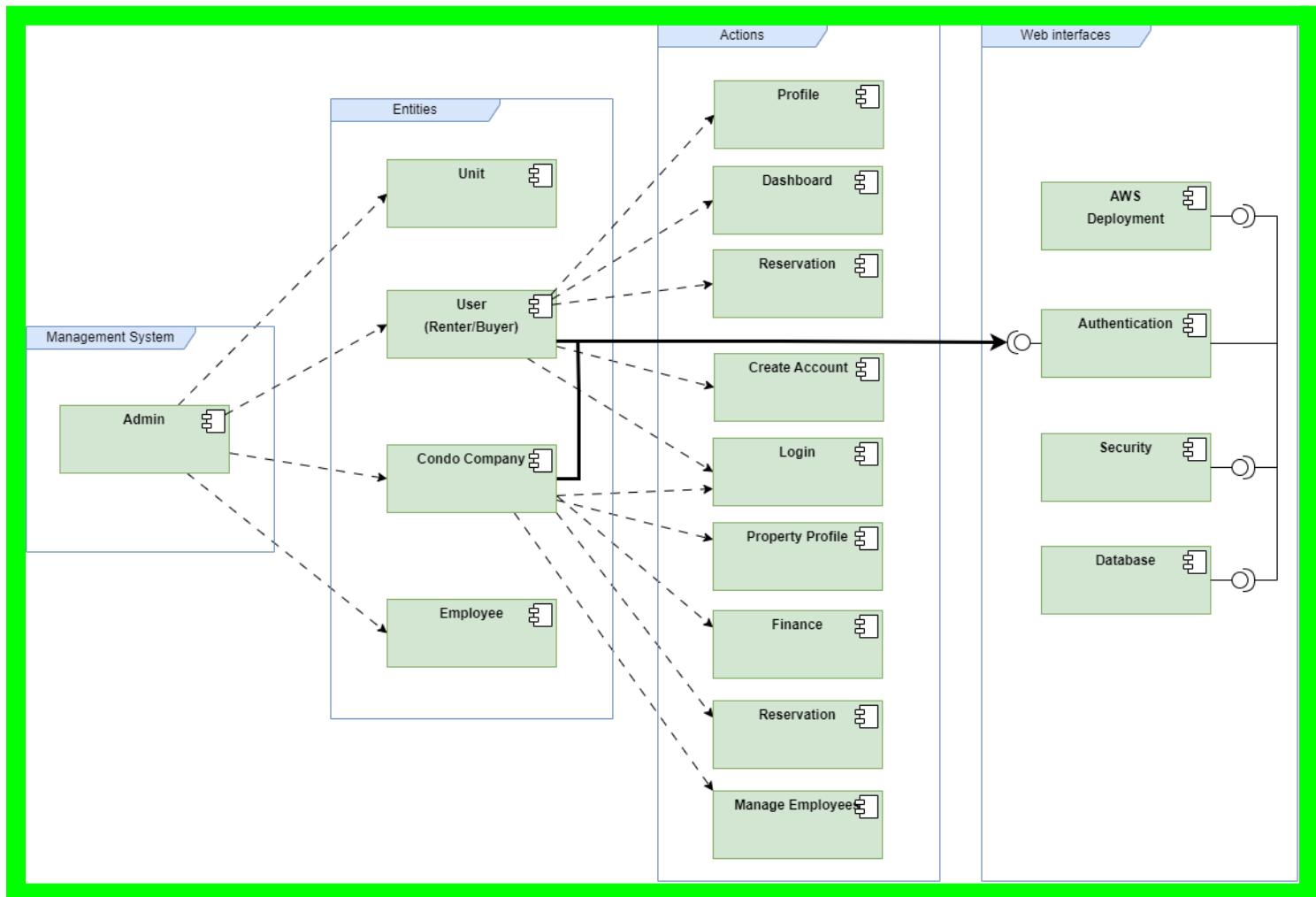


Figure 3: Component Diagram

# Deployment Diagram

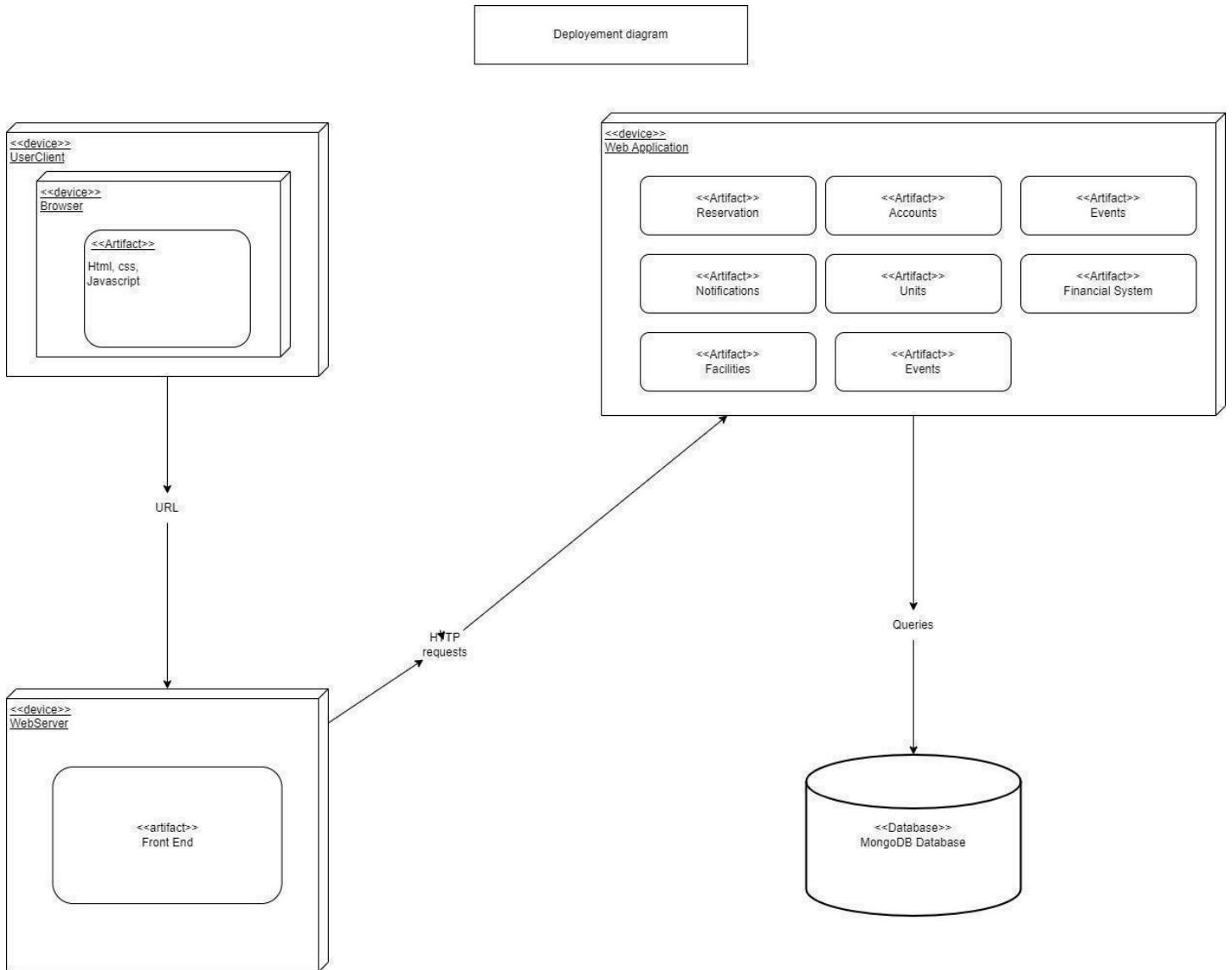
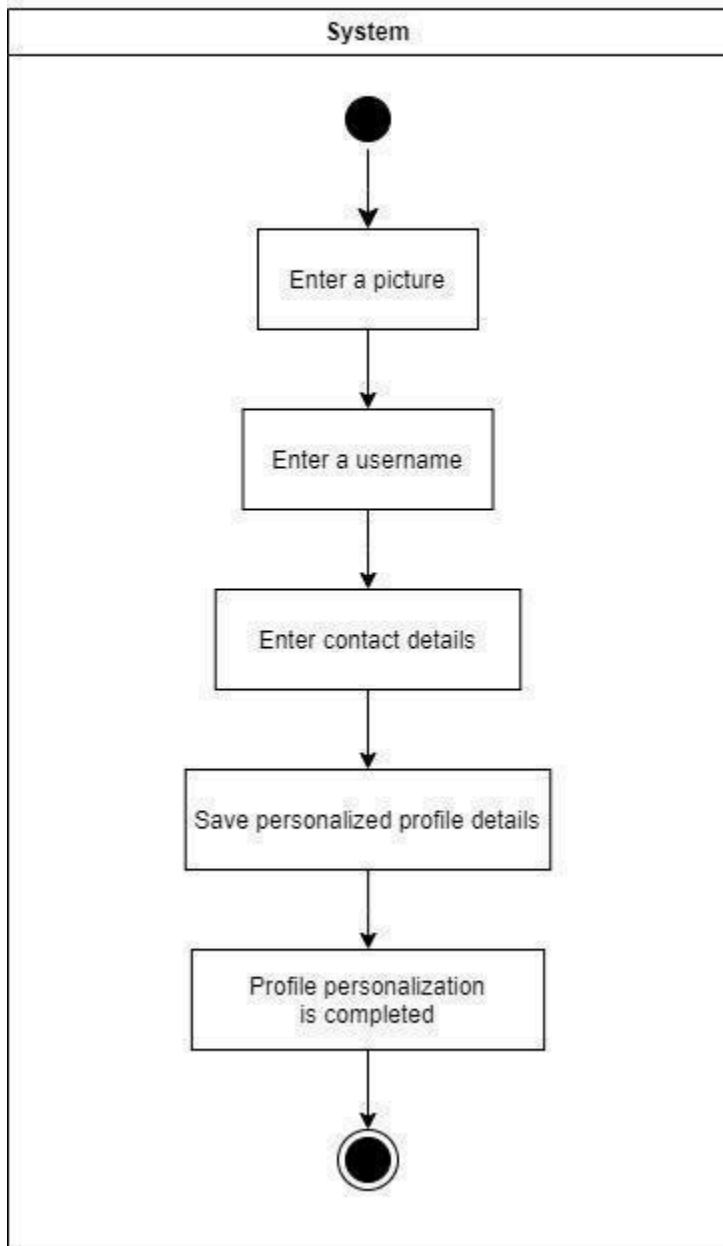


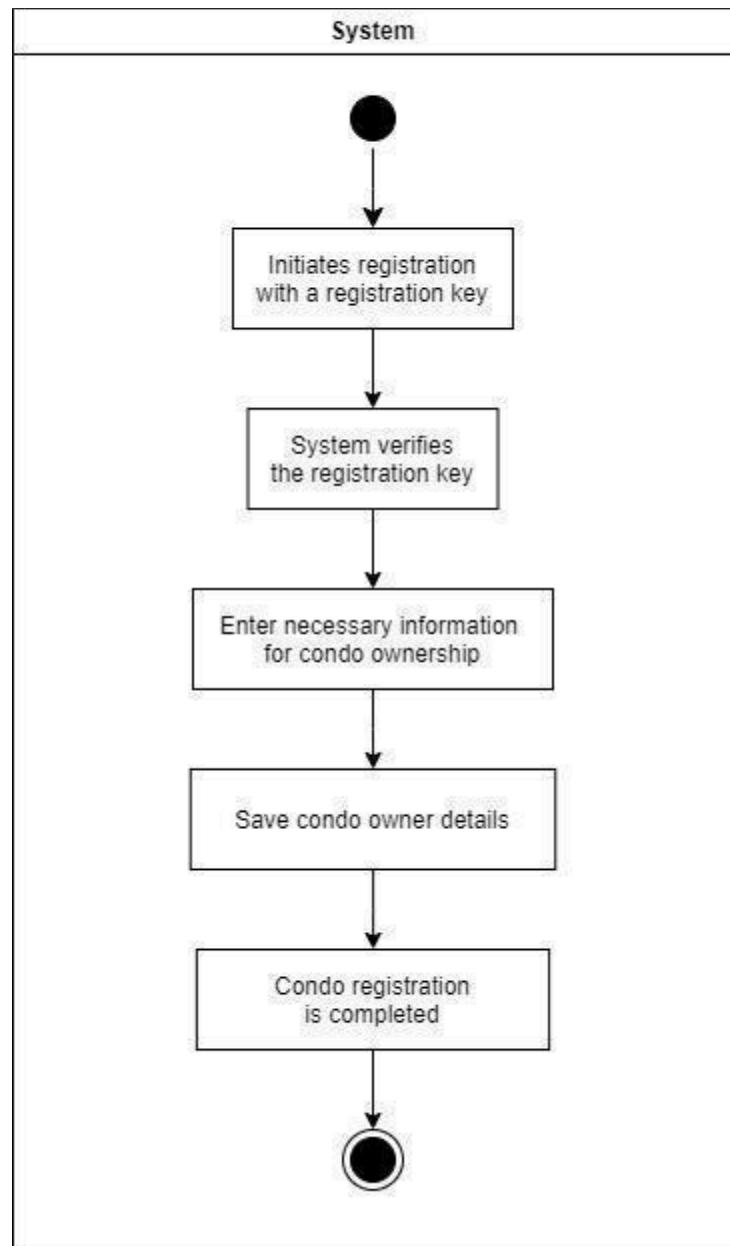
Figure 4: Deployment Diagram

## Activity Diagrams

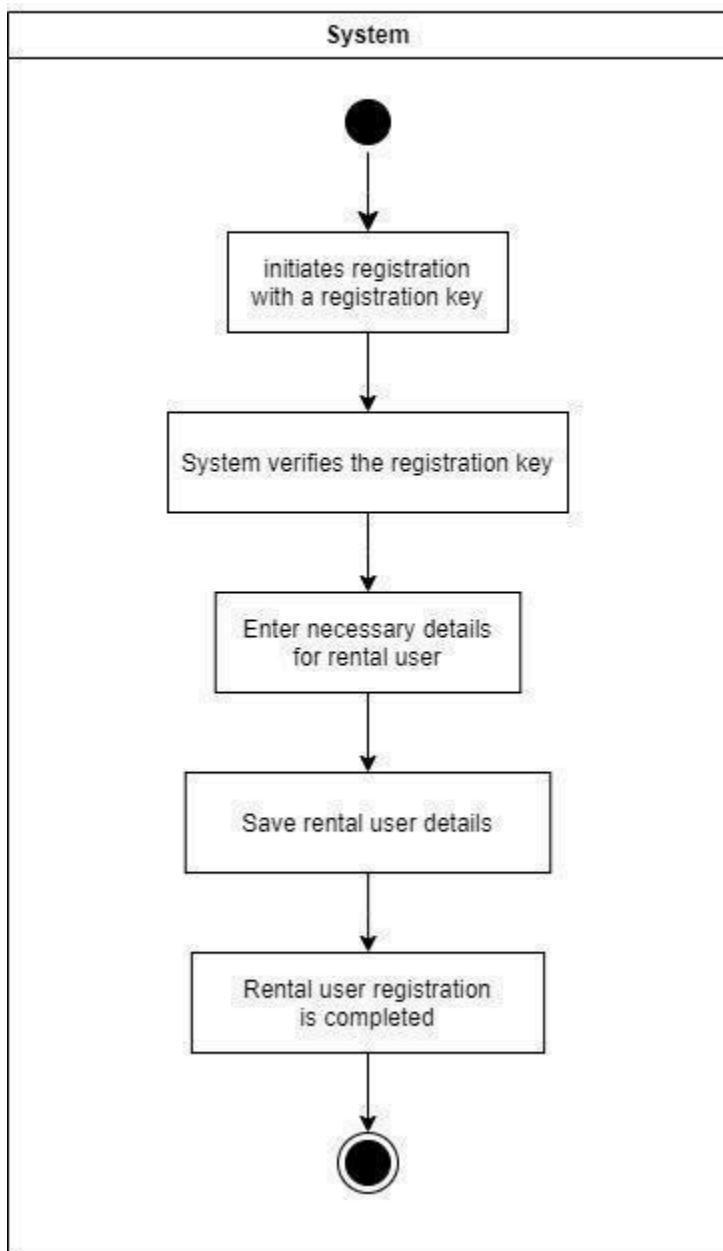
ID: 001



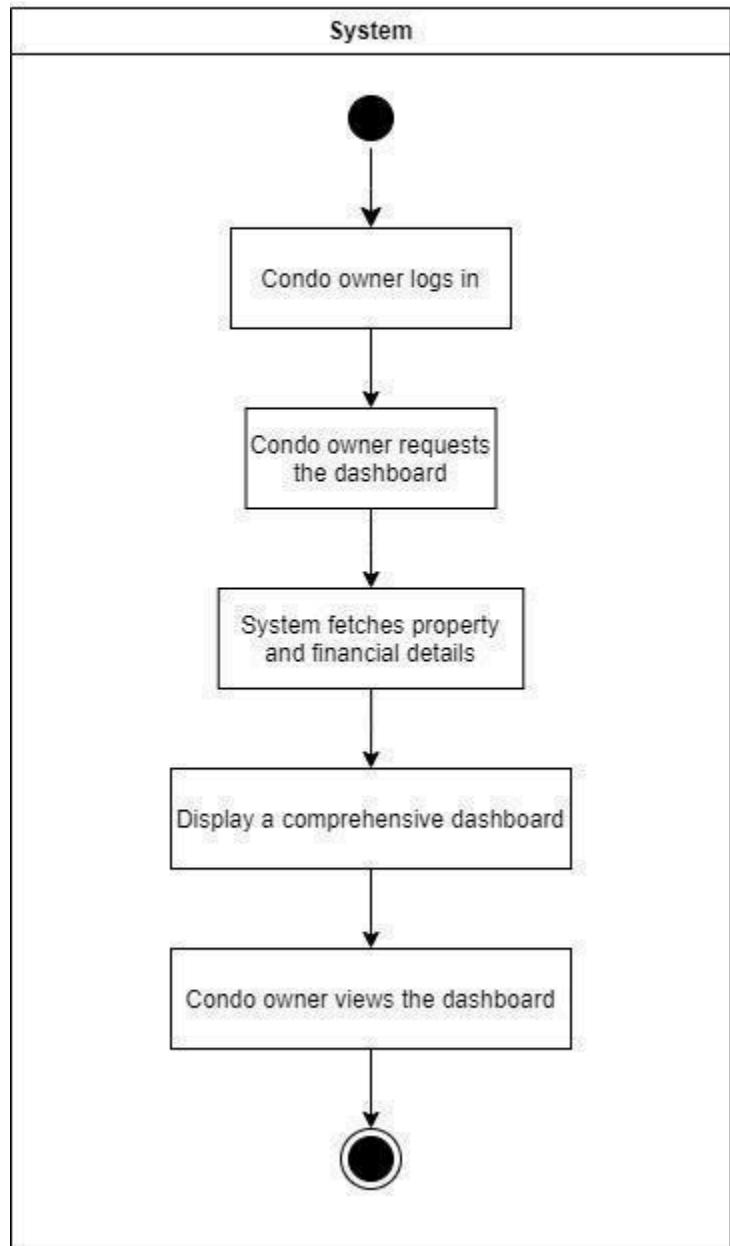
ID: 002



ID: 003

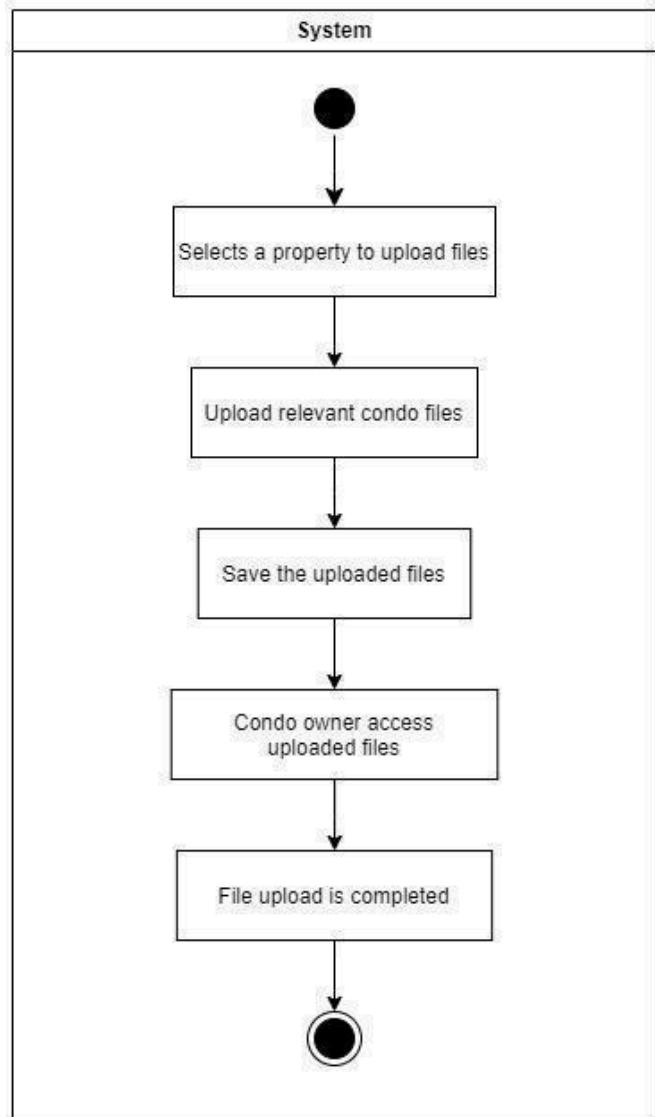
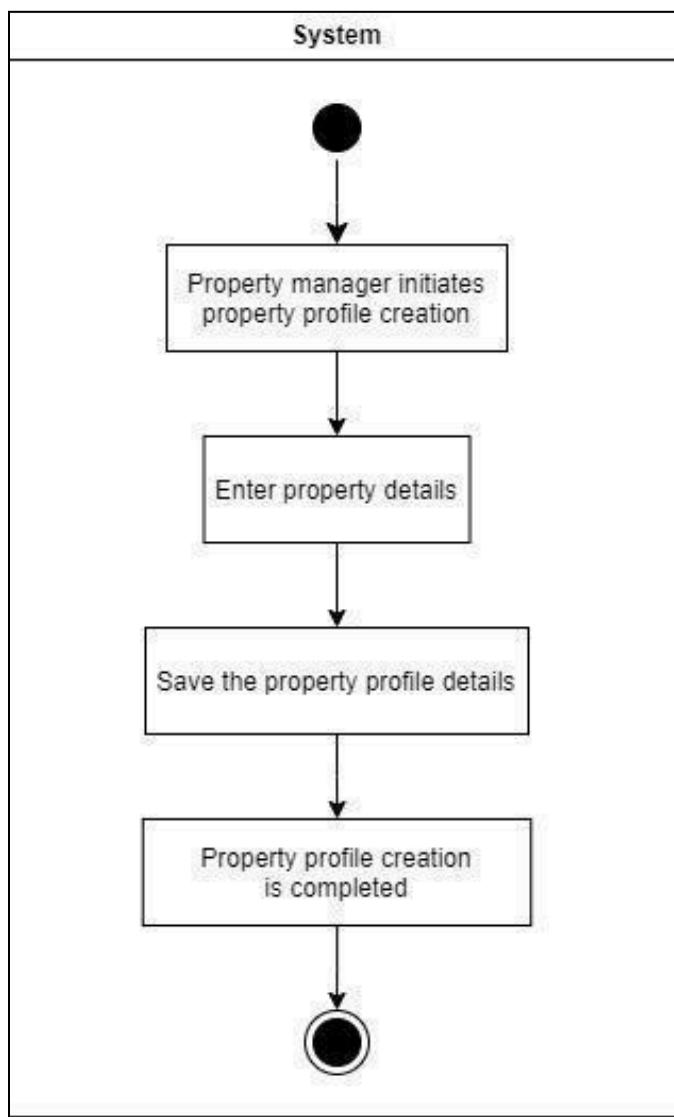


ID: 004



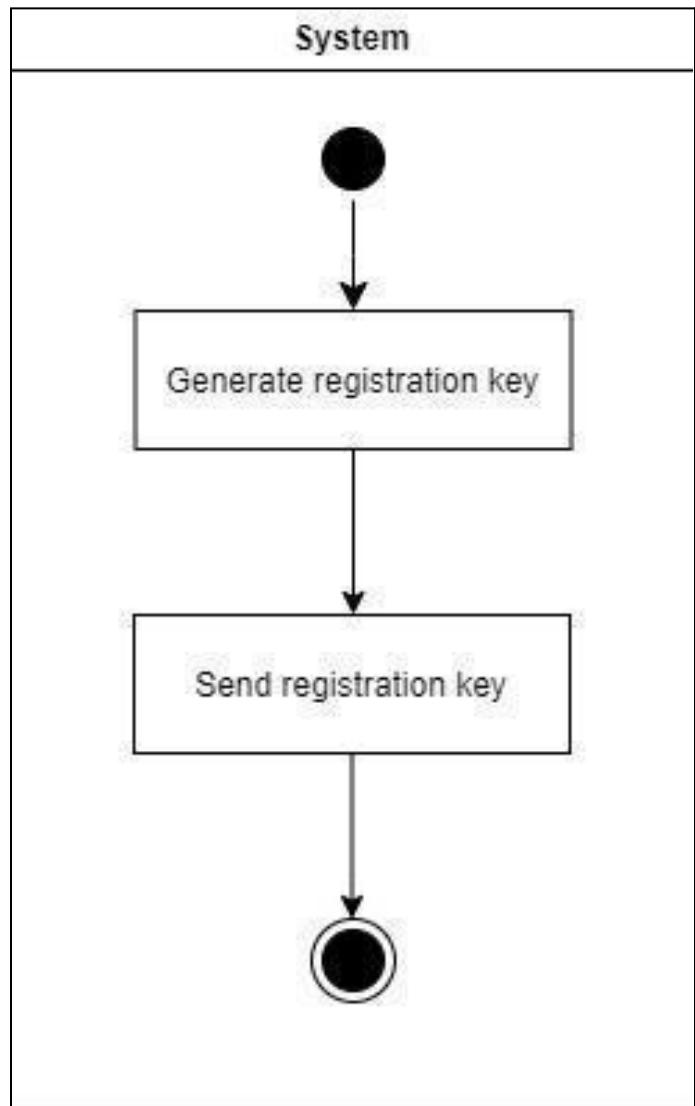
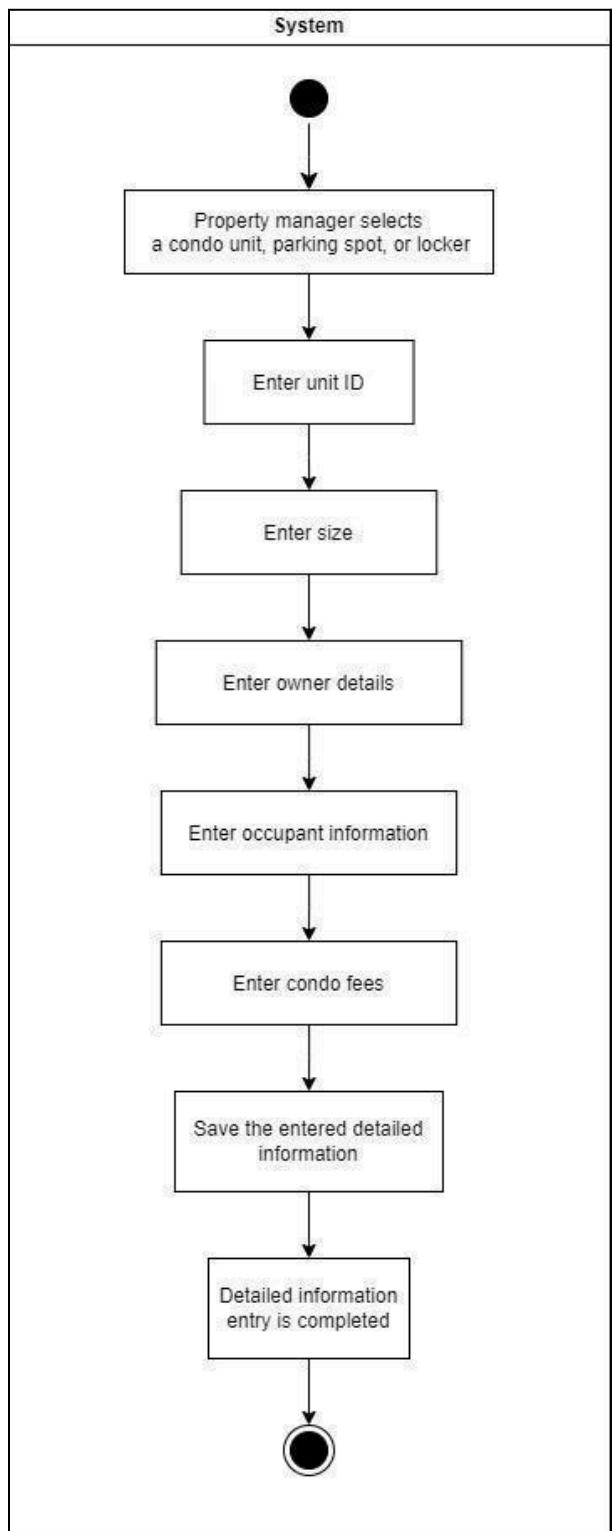
ID: 005

ID: 006

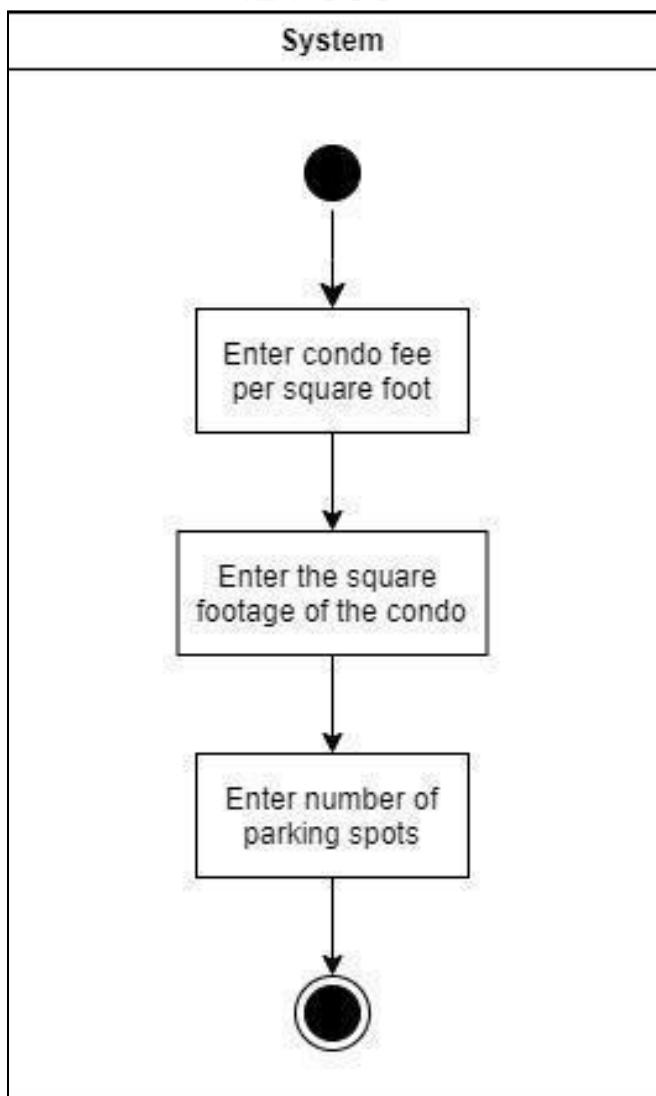


ID: 007

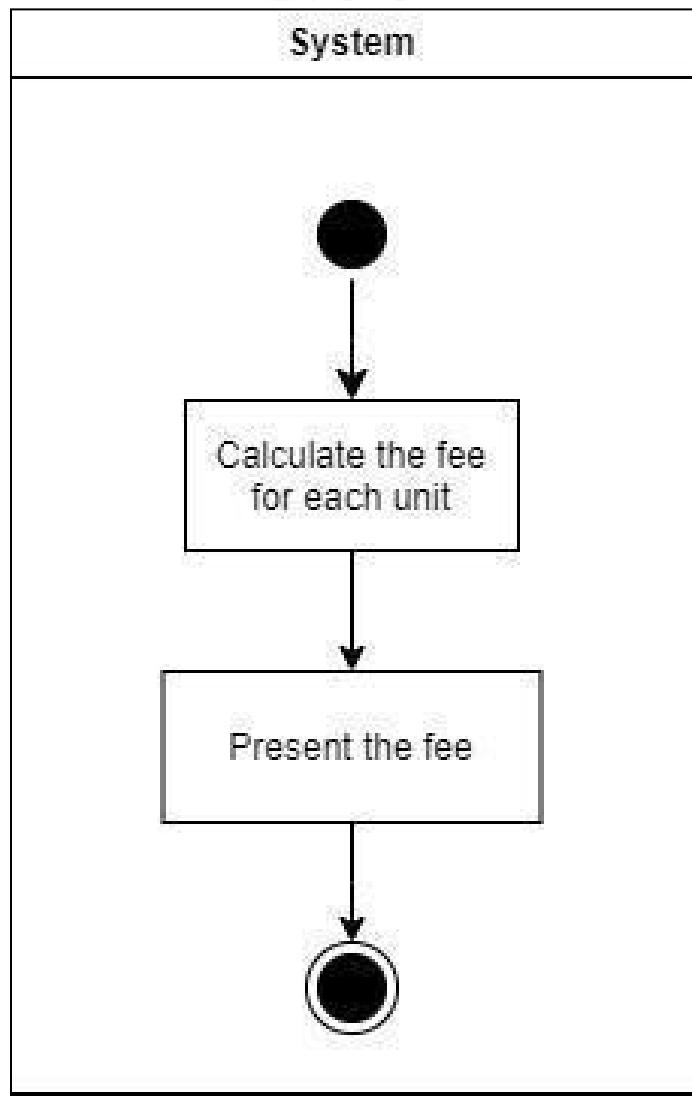
ID: 008



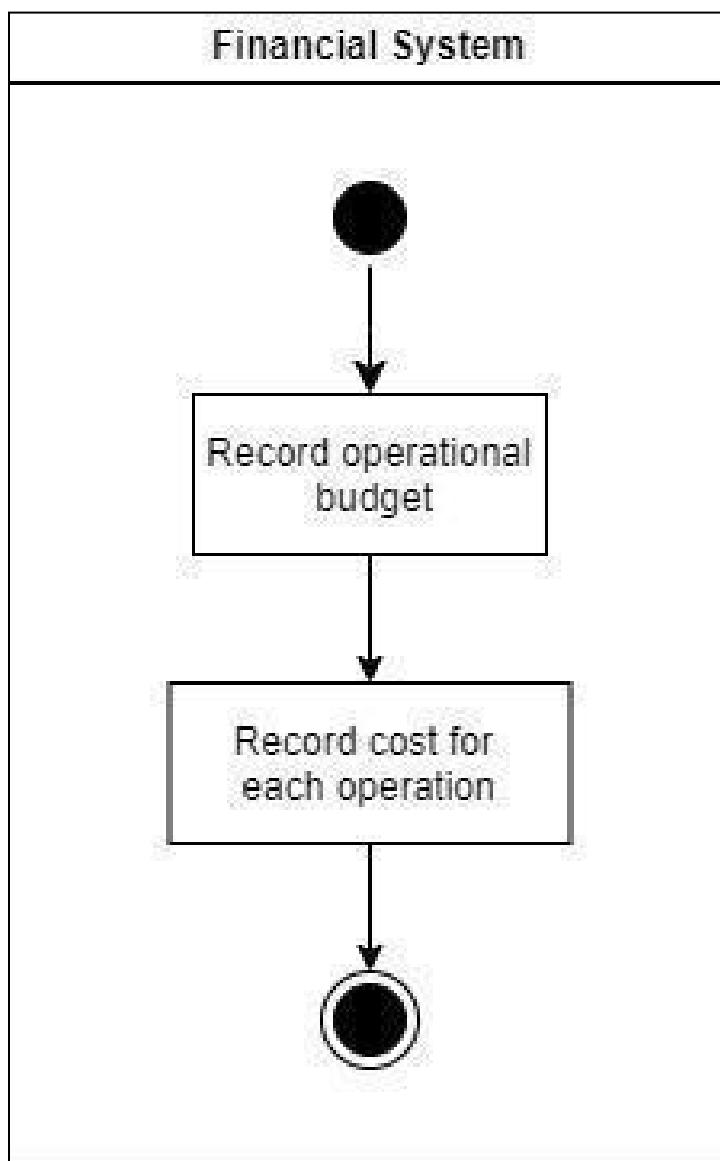
ID: 009



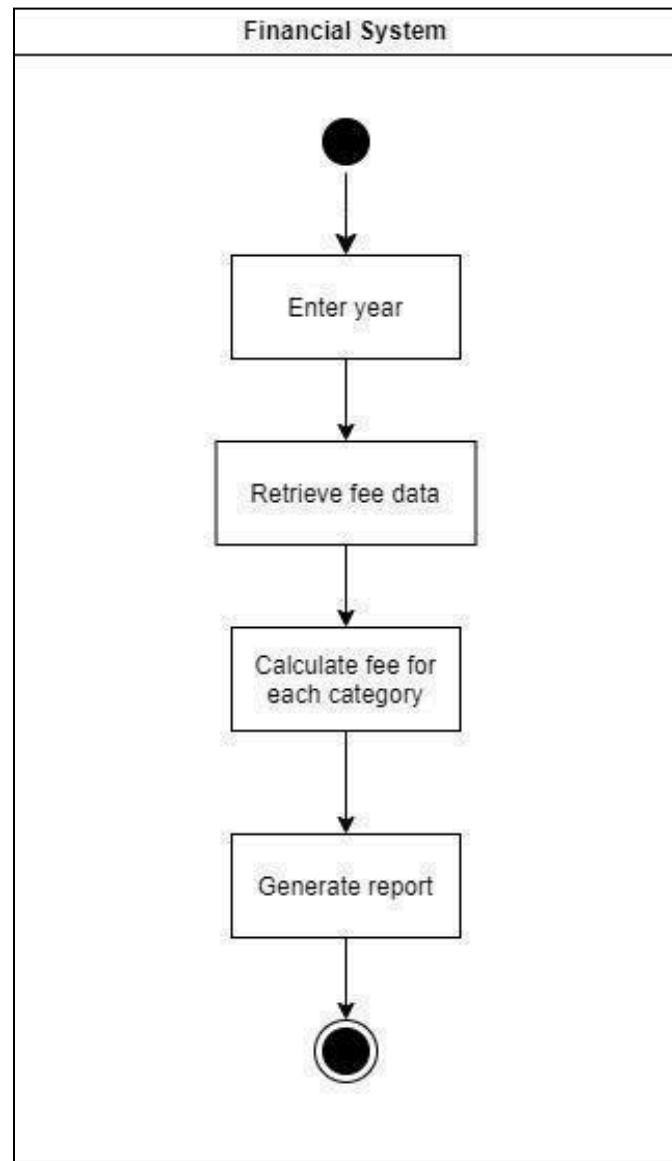
ID: 010



ID: 011

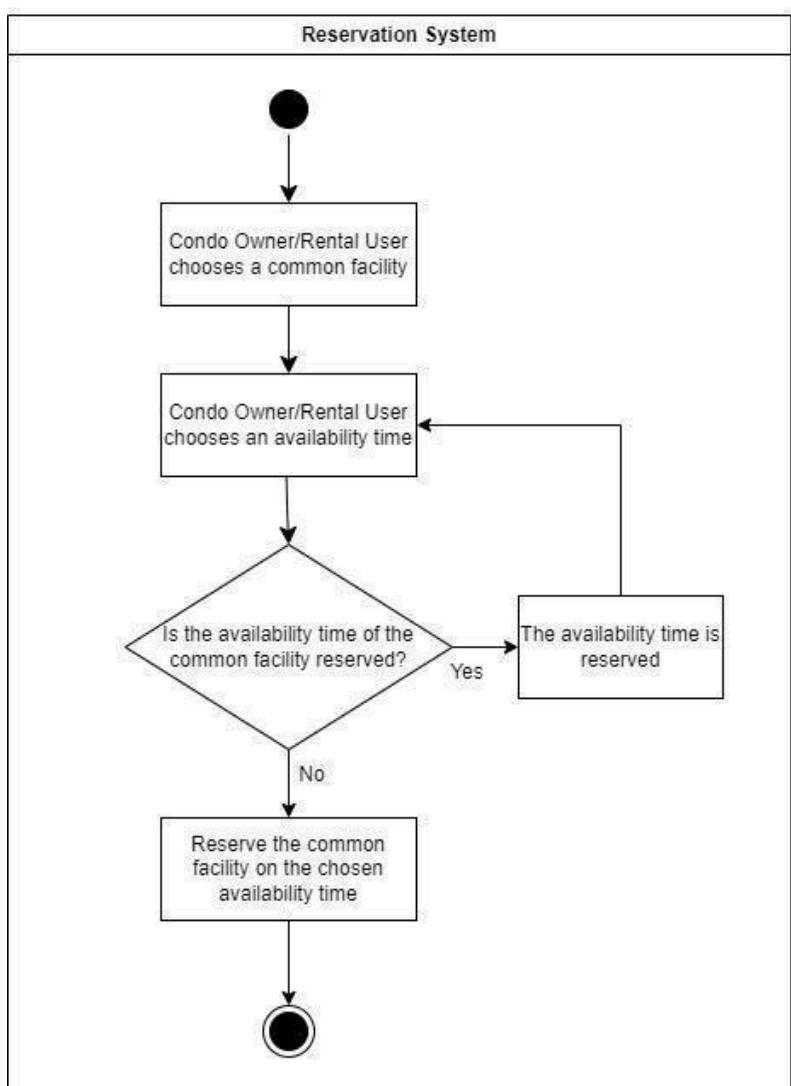
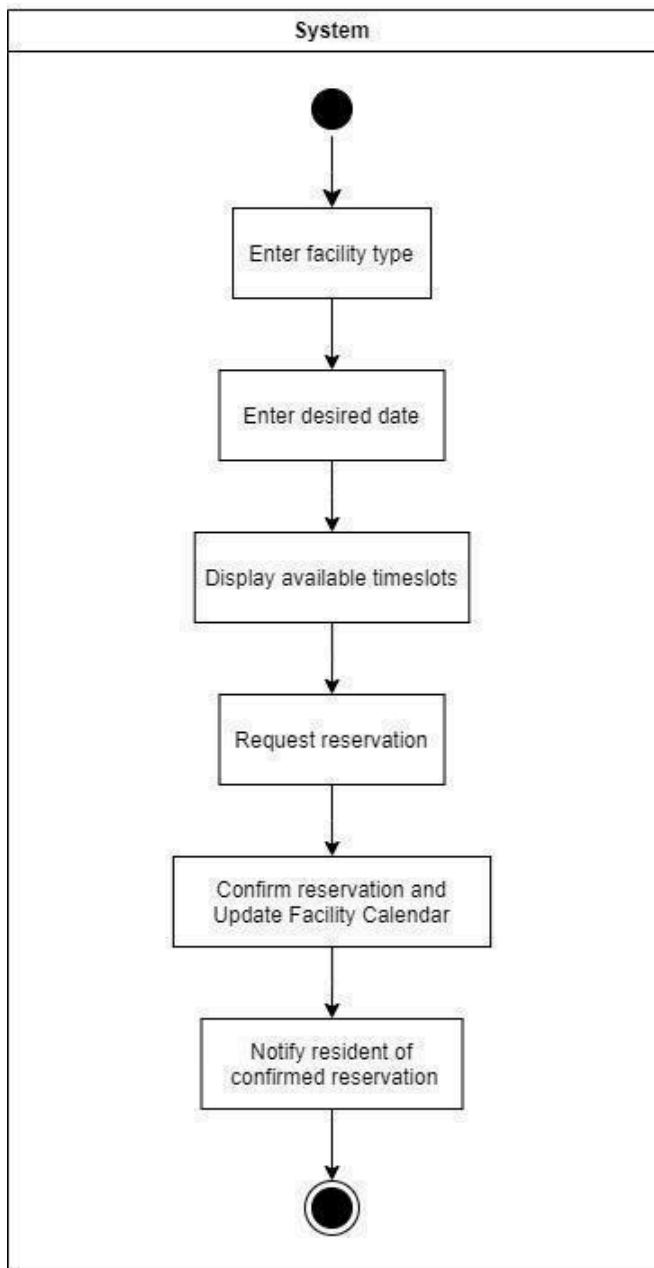


ID: 012

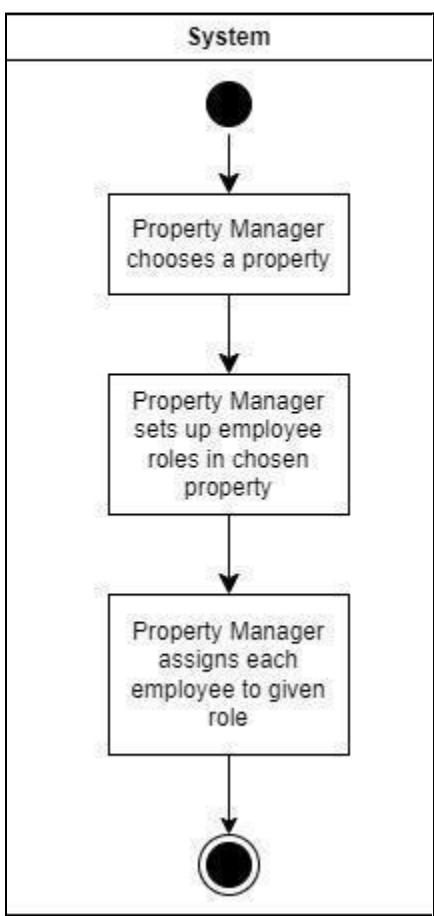


ID: 013-014

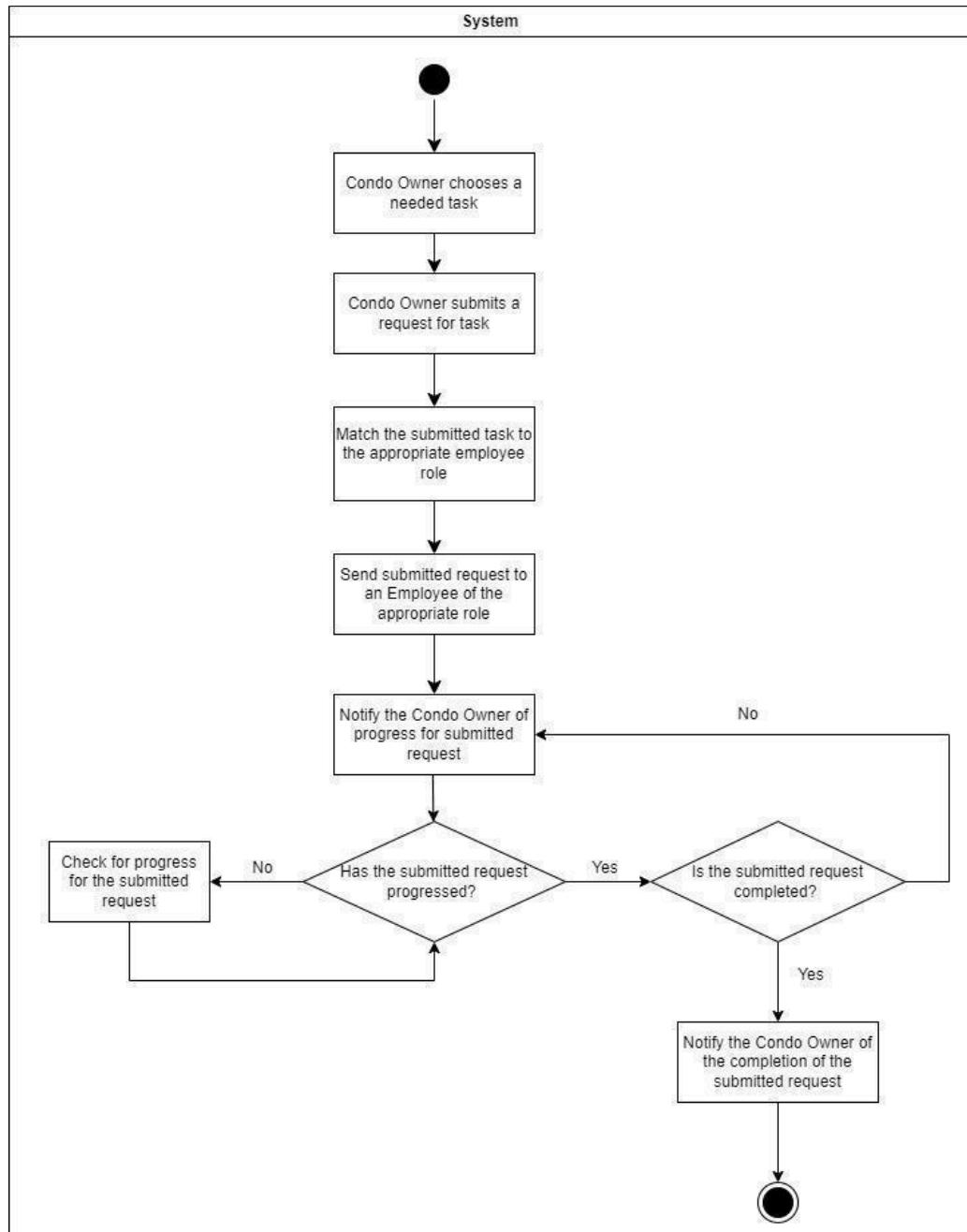
ID: 015-016



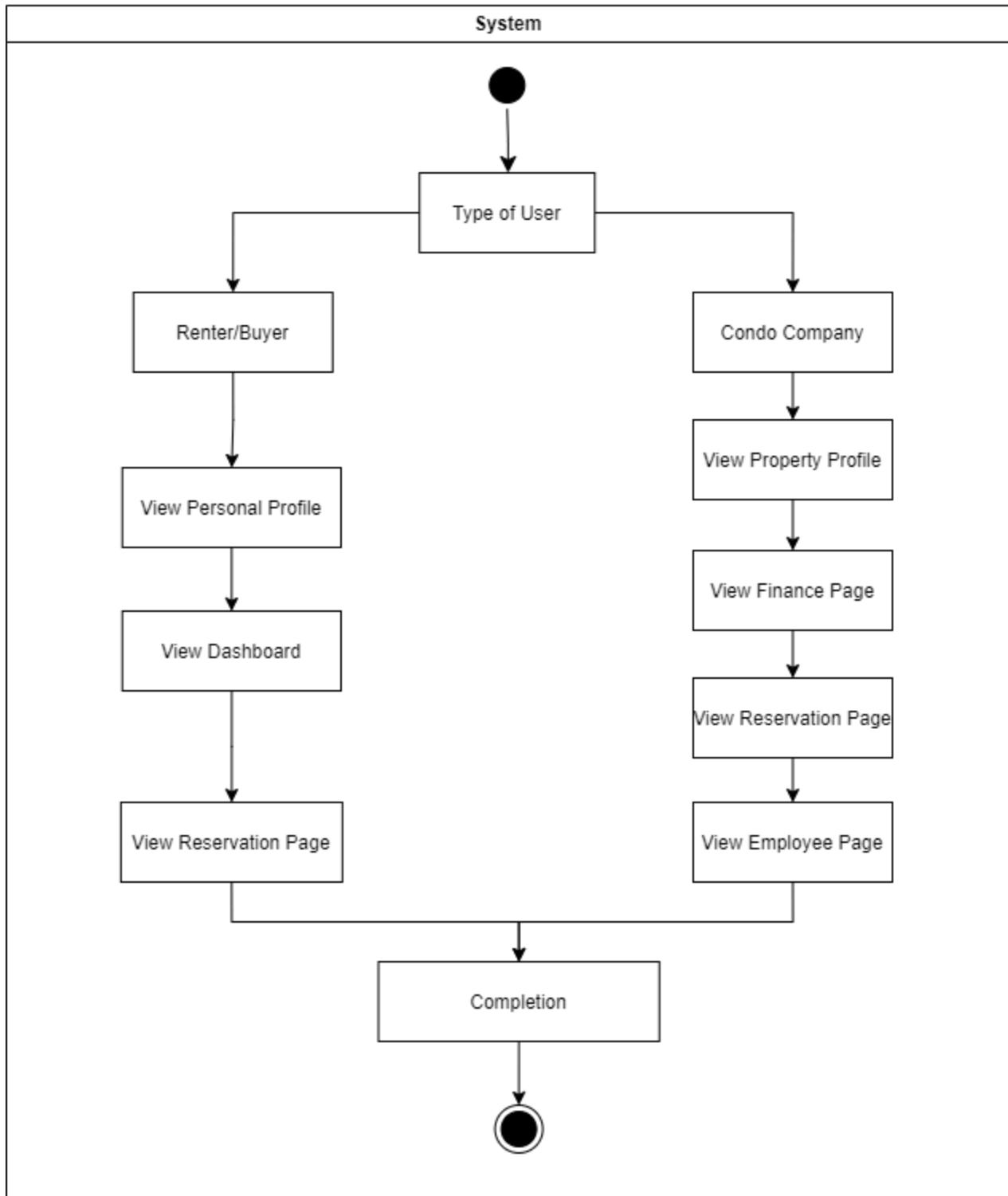
ID: 017



ID: 018 - 019 - 020

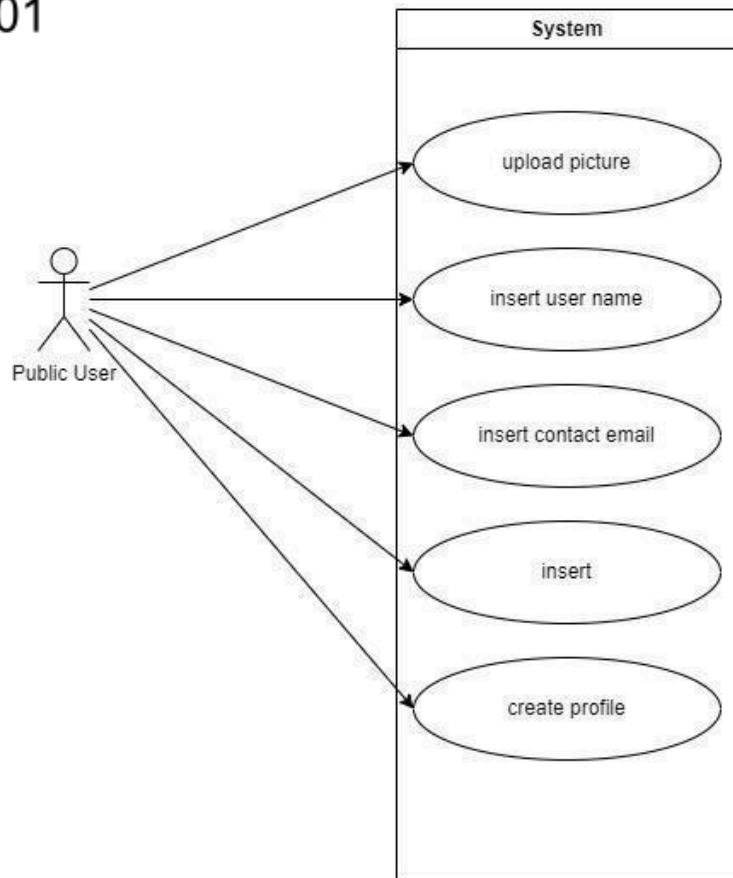


ID: 021

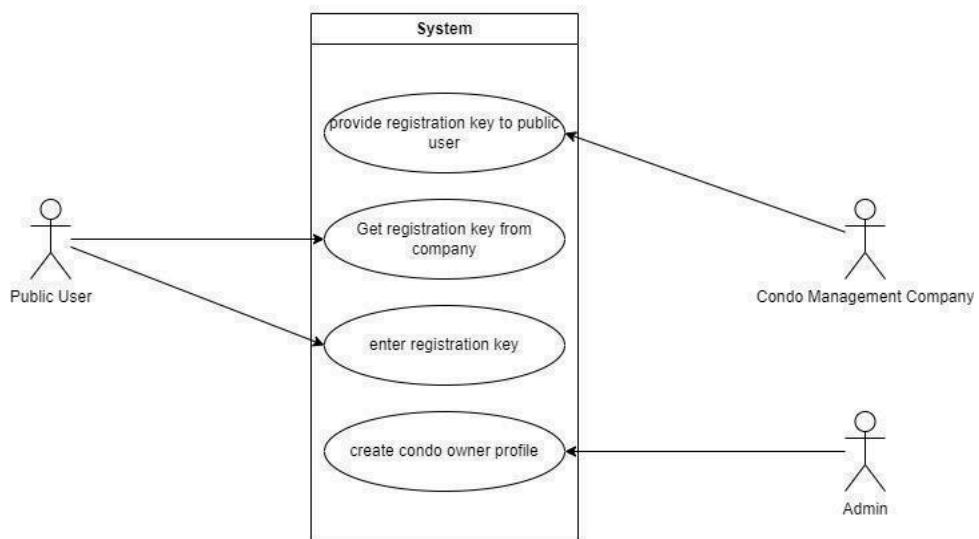


# Use Case Diagrams

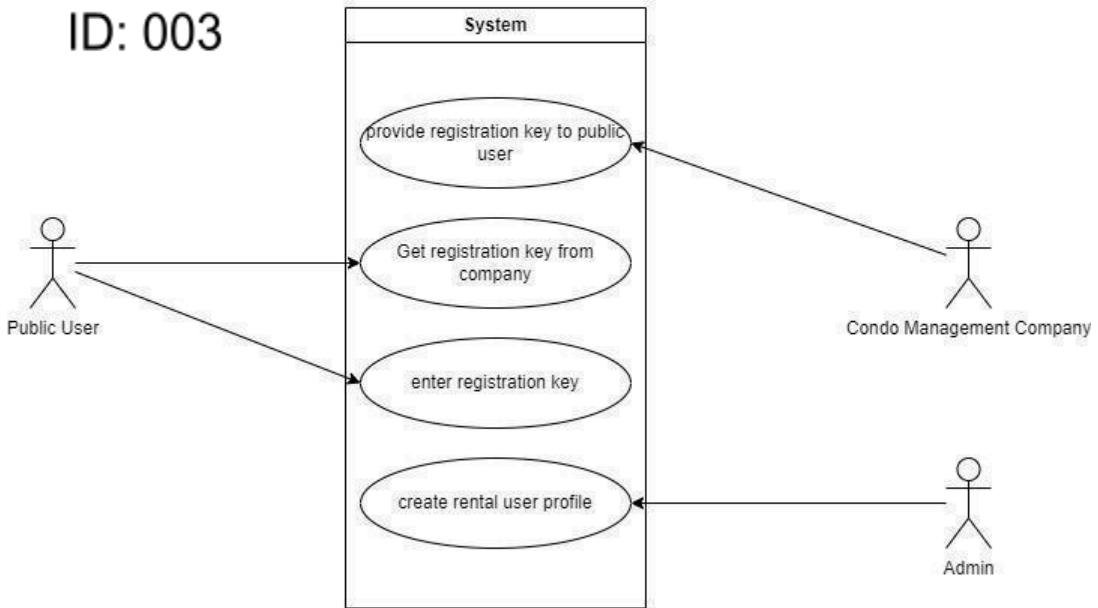
ID: 001



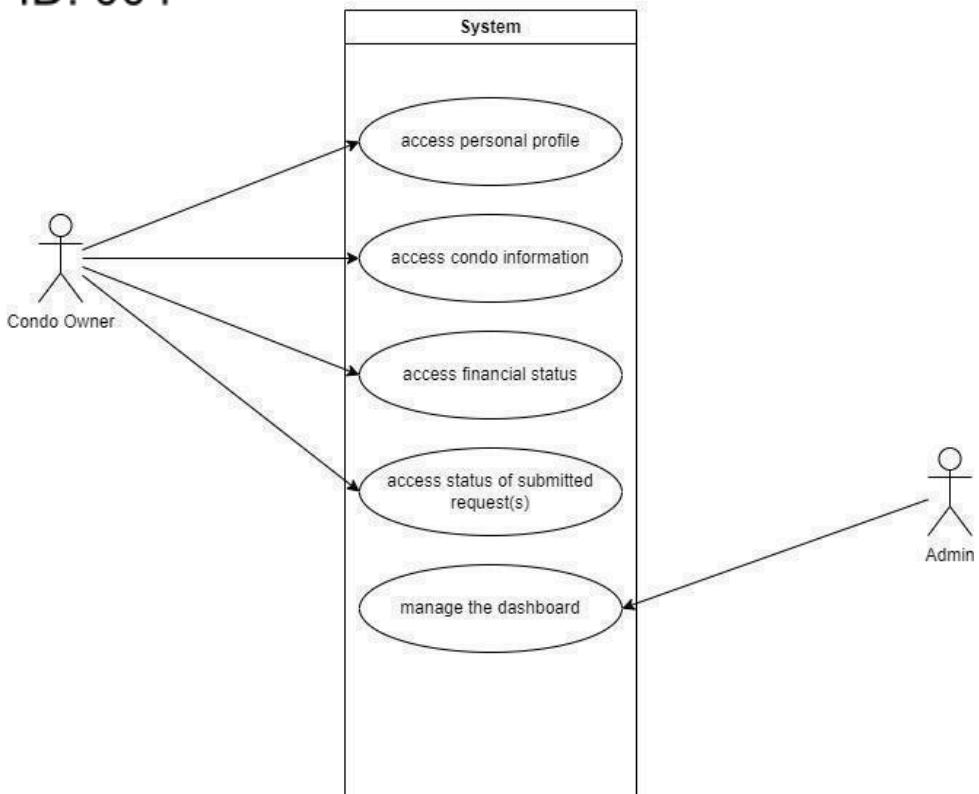
ID: 002



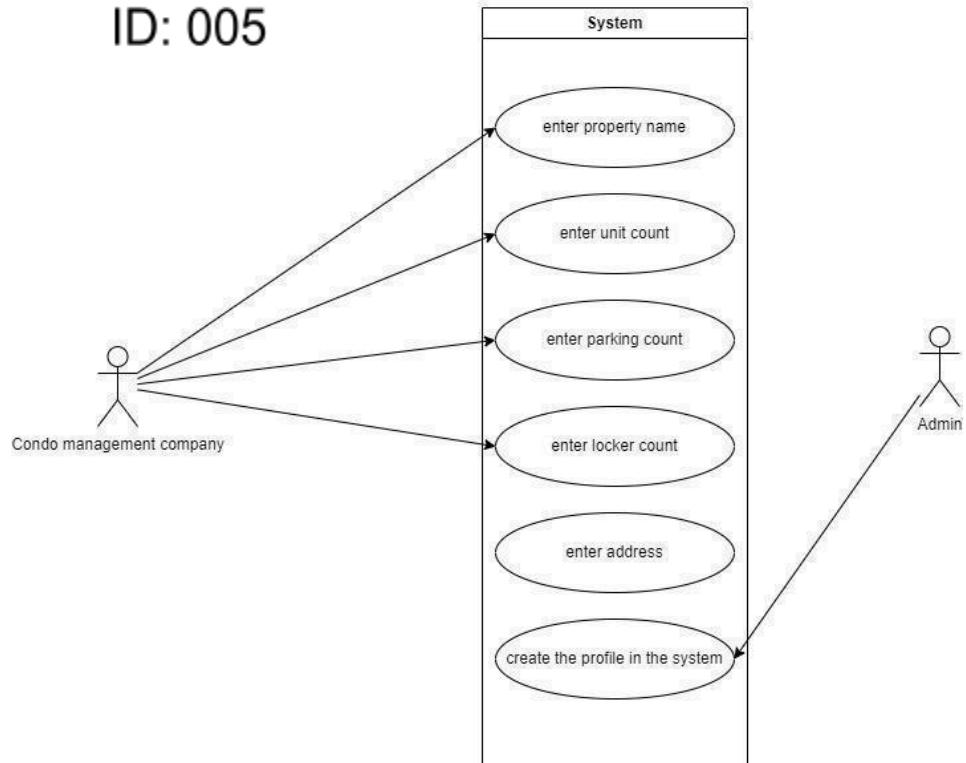
ID: 003



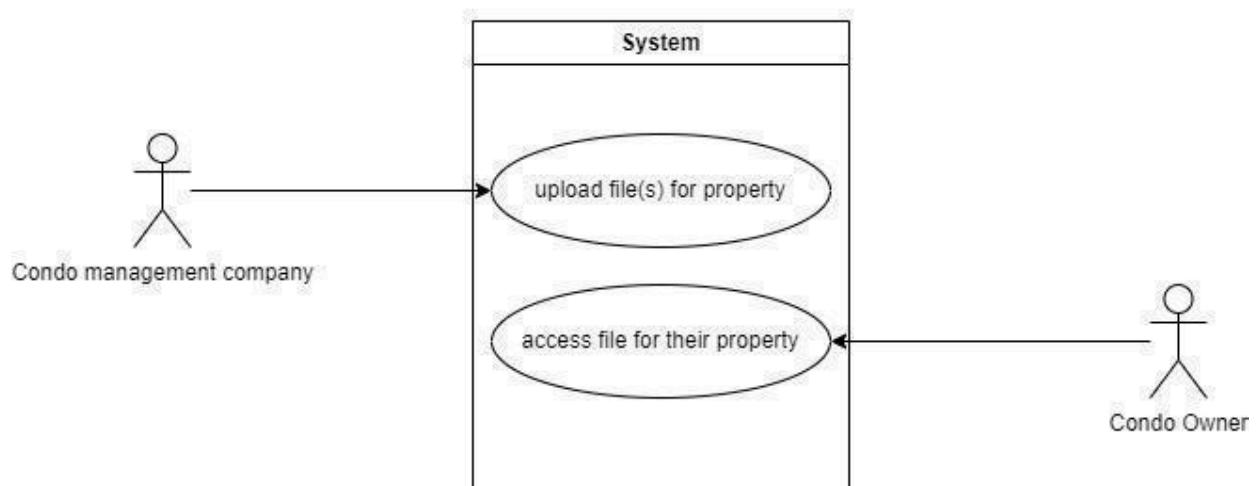
ID: 004



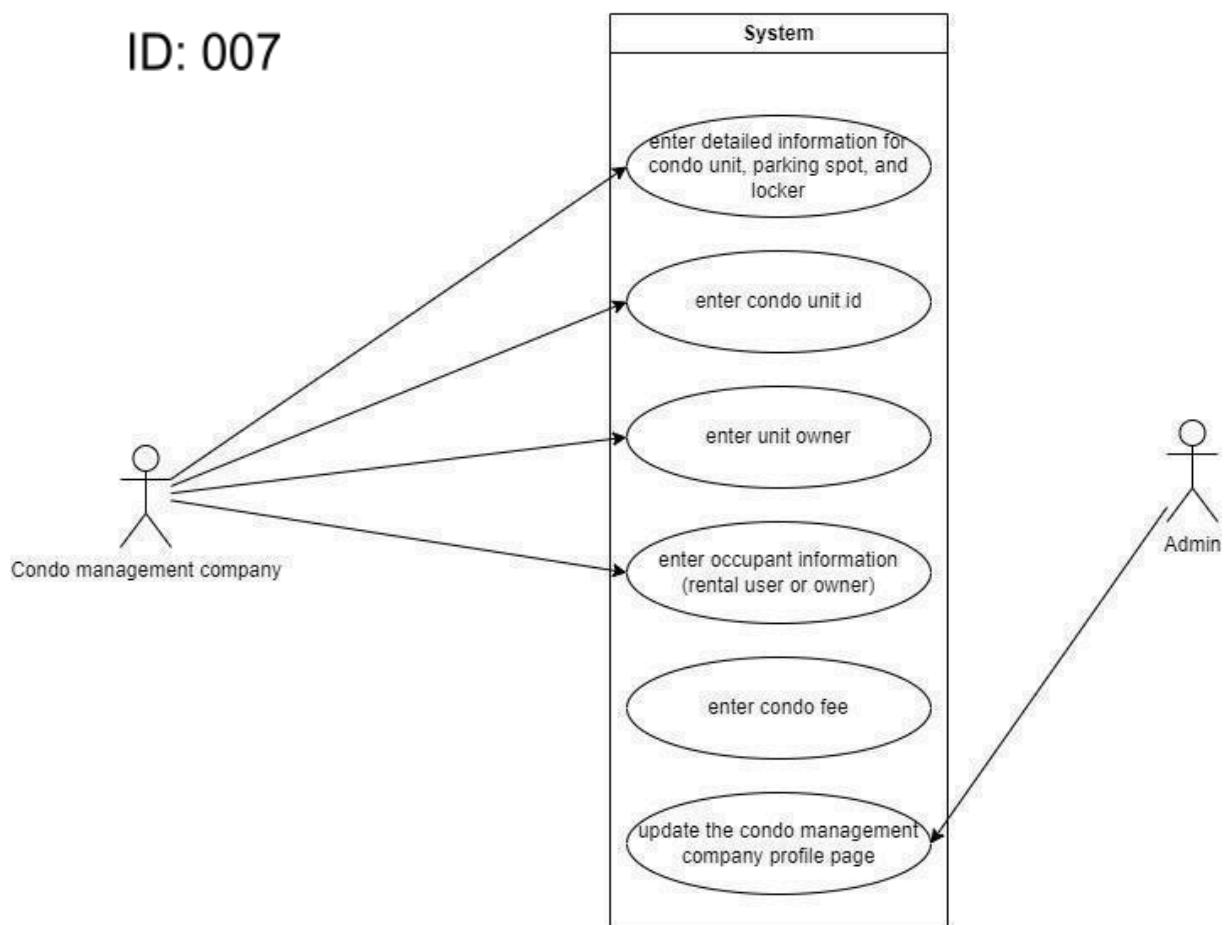
ID: 005



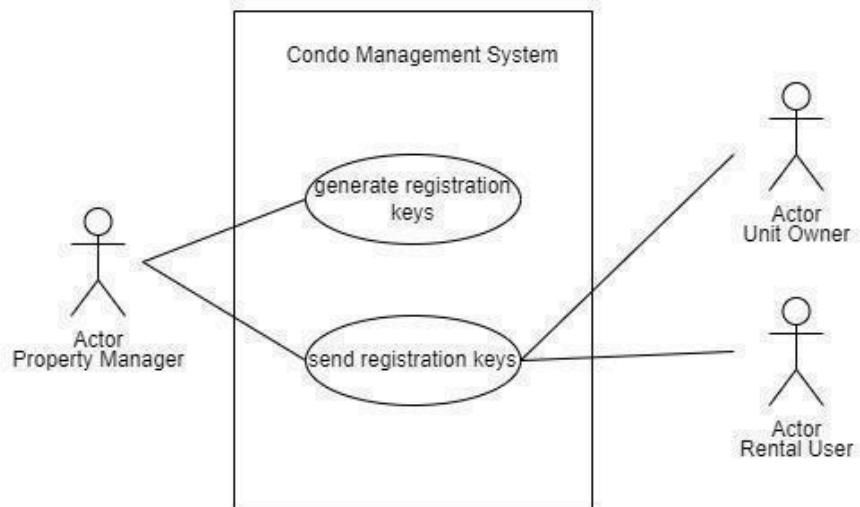
ID: 006



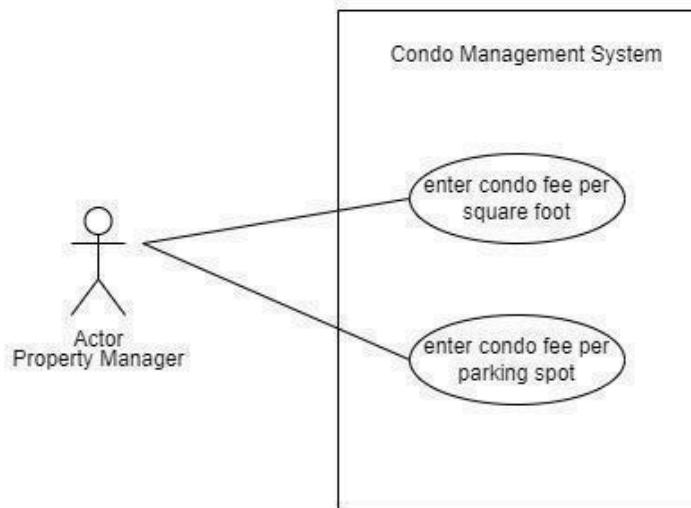
ID: 007



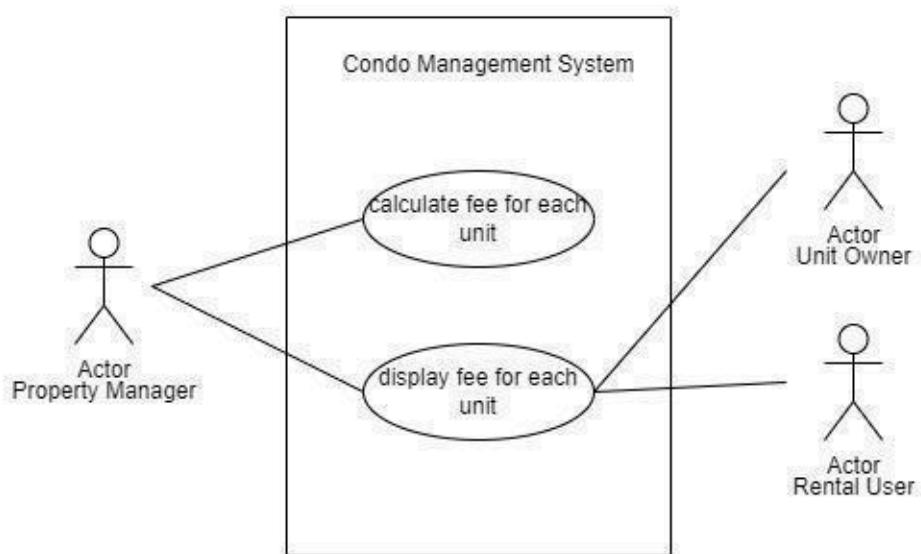
ID: 008



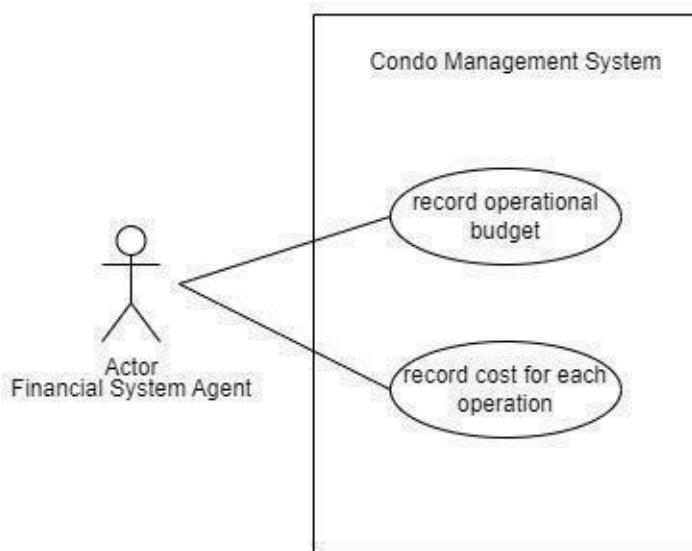
ID: 009



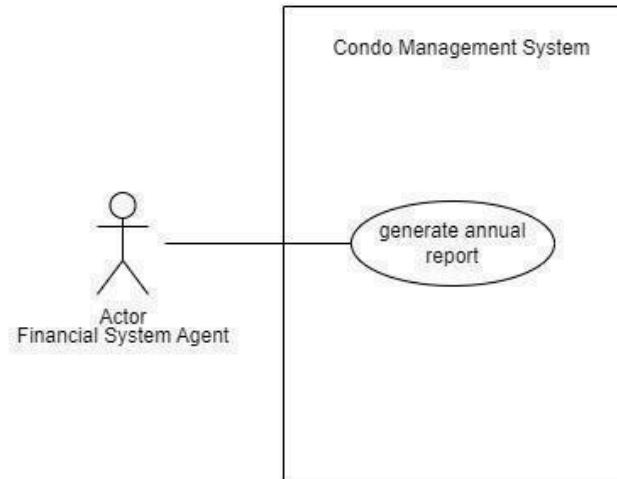
ID: 010



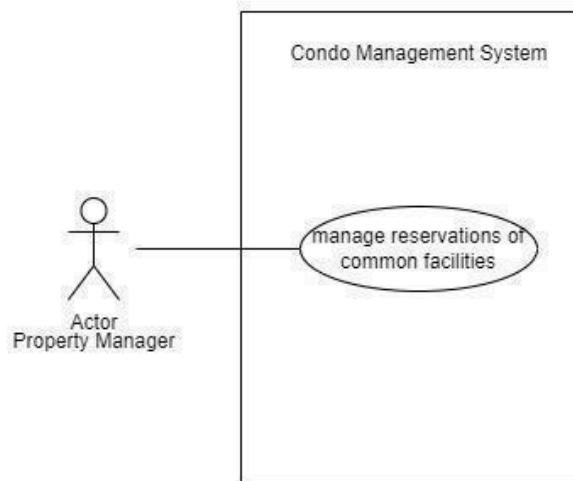
ID: 011



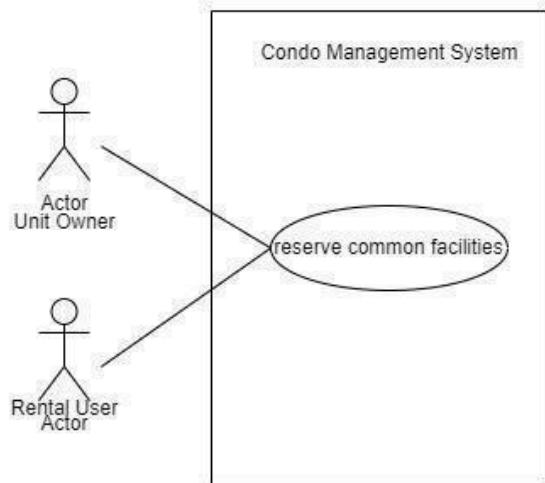
User Story #12



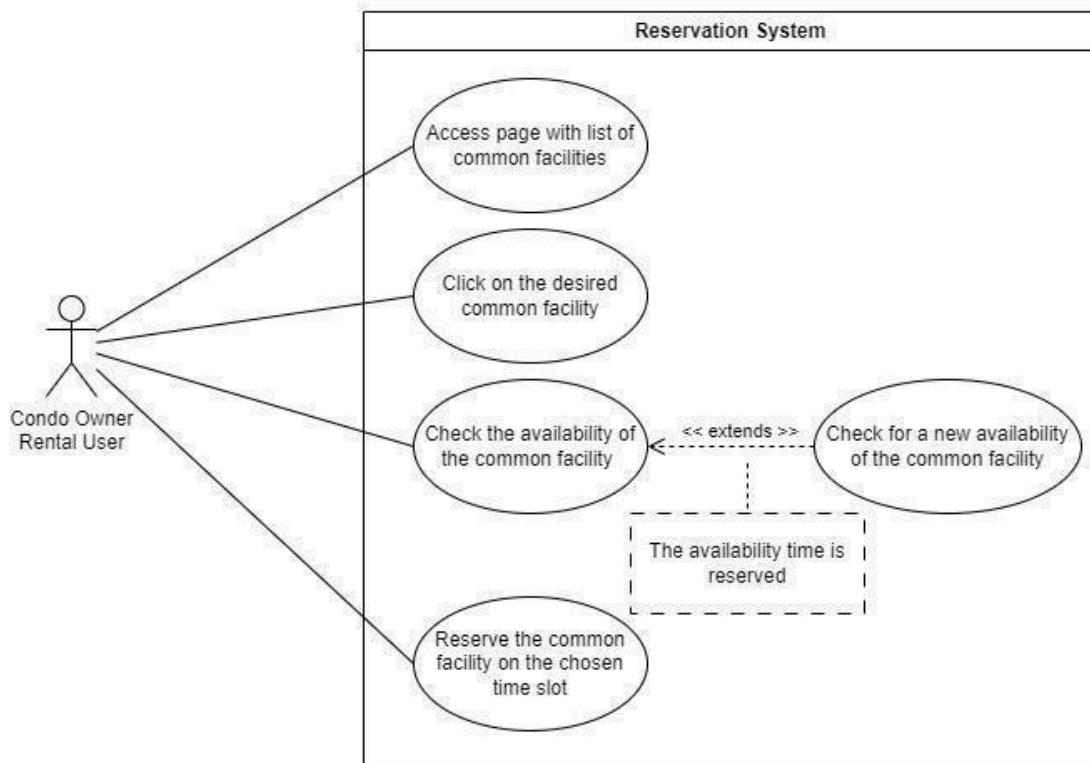
User Story #13



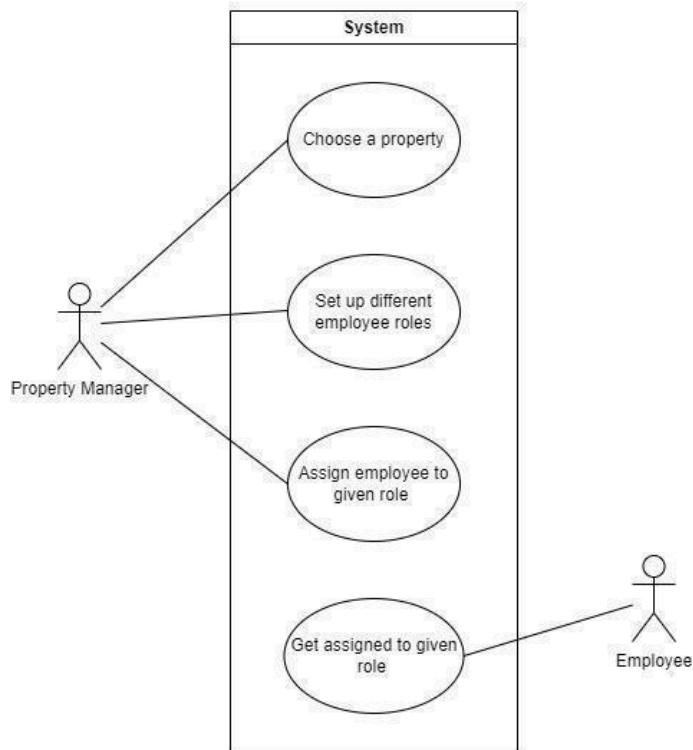
User Story #14



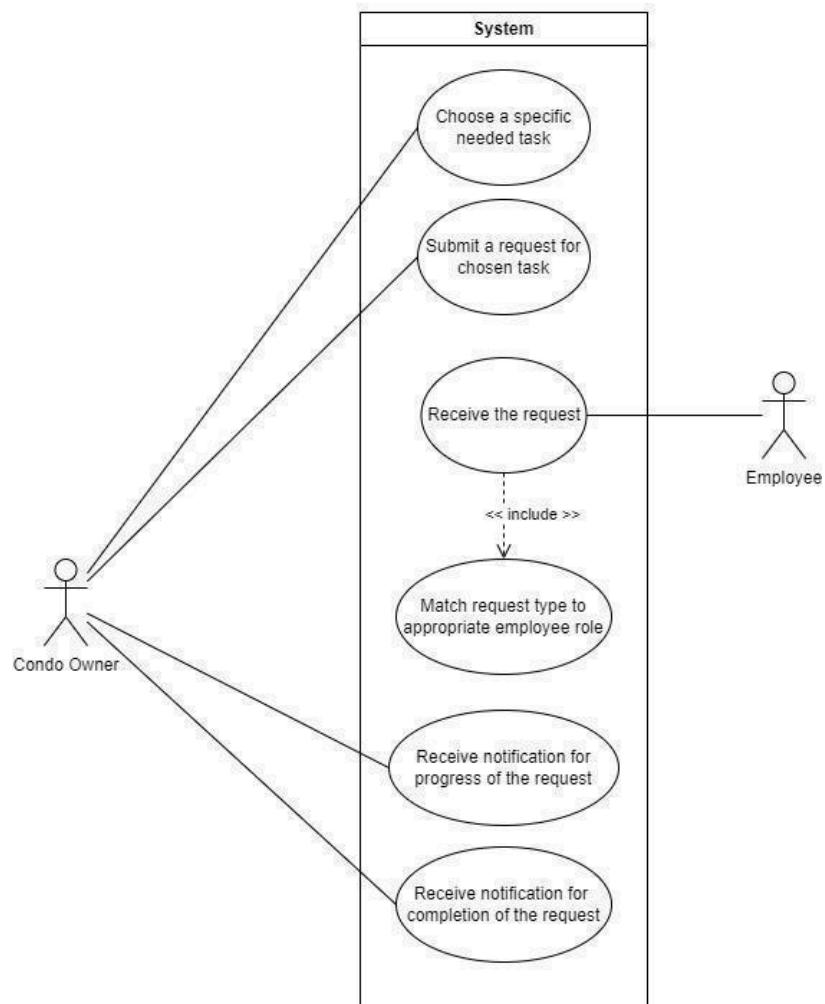
## User Story #15-16



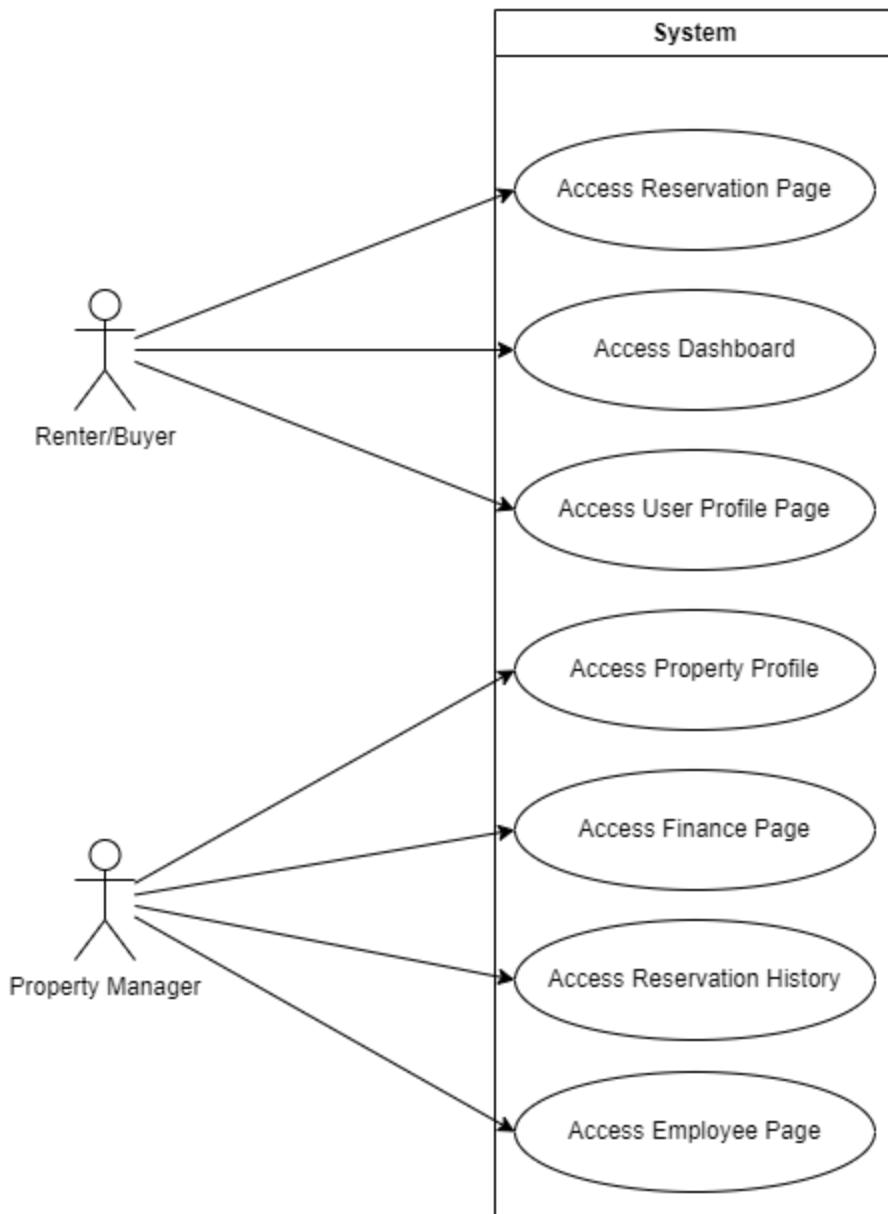
## User Story #17



## User Story #18-19-20



ID: 021



# Risk Assessment & Management Plan

## Purpose of the risk assessment and management plan along with its benefits

The Risk Assessment and Management Plan serves as a proactive approach to identify, assess, prioritize, and mitigate potential risks that may arise during the project lifecycle. Its purpose is to anticipate challenges, minimize negative impacts, and ensure the successful completion of the project within the defined constraints of time, budget, and quality. By systematically identifying and addressing risks, the plan helps to enhance project visibility, stakeholder confidence, and overall project resilience.

## How the risks and assessment were identified among our team

The identification and assessment of risks within our project were predominantly conducted through regular team meetings. During these meetings, team members engaged in open discussions and collaborative brainstorming sessions to identify potential risks across various aspects of the project, such as technical complexities, resource constraints, and communication challenges. Drawing upon the diverse perspectives and expertise of each team member, we collectively examined project documentation, shared insights from past experiences, and solicited input from stakeholders to comprehensively identify and assess risks. Through this iterative process of dialogue and reflection, we prioritized risks based on their likelihood and potential impact, laying the groundwork for effective risk management strategies and mitigation plans.

Impact	Low	Medium	High
Probability			
Low			#6, #16
Medium		#20	#1, #9, #11, #13
High		#25	#24

Figure 5: Risk management chart

## **Legend:**

High – Greater than <70%> probability of occurrence

Medium – Between <30%> and <70%> probability of occurrence

Low – Below <30%> probability of occurrence

ID	User Story ID	Risk	Type	Probability	Impact	Mitigation
1	1	Poor usability due to complex profile setup	User Experience	Medium	High	Conduct user testing and feedback sessions throughout development. Prioritize simplicity and clarity in design. Provide intuitive tooltips and guidance for profile setup.
2	6	Vulnerabilities in condo file upload leading to unauthorized access	Technical	Low	High	Implement input validation and file type restrictions. Use secure file storage mechanisms and regularly audit access controls.
3	9	Incorrect calculation of condo fees due to coding errors	Technical	Medium	High	Implement thorough unit testing and code reviews for fee calculation logic. Utilize standardized libraries or frameworks for financial calculations.
4	11	Inadequate logging of financial transactions leading to auditing challenges	Technical	Medium	High	Implement comprehensive logging of financial operations with proper data encryption. Regularly review and monitor logs for anomalies.
5	13	Performance issues in reservation system due to inefficient code	Technical	Medium	High	Optimize database queries and server-side processing. Implement caching mechanisms for frequently accessed data. Conduct load testing to identify and address performance bottlenecks.
6	16	Security vulnerabilities in reservation system leading to data breaches	Management	Low	High	Follow secure coding practices, such as input validation and parameterized queries, to prevent SQL injection and other attacks.
7	20	Lack of notification updates leading to user frustration	Management	Medium	Medium	Implement real-time notification system, provide manual refresh option. Optimize

						notification delivery, prioritize critical updates.
8	N/A	Deployment failure leading to website downtime	Technical	Medium	High	Conduct thorough testing in staging environment before deployment. Implement automated deployment scripts with rollback procedures. Utilize a gradual deployment approach to minimize impact.
9	N/A	Security vulnerabilities introduced during deployment	Security	Low	High	Implement secure deployment practices, such as code signing and secure transmission protocols. Regularly update and patch deployment tools and dependencies. Conduct security audits before and after deployment.
10	2,3	Registration key generation errors leading to user authentication issues	Technical	Low	High	Implement thorough validation checks during registration key generation process. Provide error handling mechanisms and user-friendly error messages. Regularly monitor key generation logs for anomalies and discrepancies.

**Table 1: List of identified risks**

***Link to the Risk Analysis UrbanKey:***

[Risk analysis UrbanKey.xlsx](#)

# Sprint 4 Testing Report

## Front-End Testing

### **Introduction:**

Front-end testing is a crucial aspect of ensuring the reliability and functionality of web applications. In our Condo Management System Website project, we employed Jest, a widely used testing library compatible with React, to conduct front-end testing. This report outlines the tools utilized, the testing approach adopted, and presents the code coverage achieved through our testing efforts.

### **Tools Used:**

We employed Jest as our primary testing library due to its seamless integration with React. Jest offers a robust framework for creating and executing tests, making it an ideal choice for our front-end testing needs.

### **Testing Approach:**

To organize our testing efforts, we established a dedicated folder named "Front-end Testing" within the src directory of our project repository. Within this folder, we created individual testing files corresponding to each front-end page of our website. Following the naming convention recommended by Jest, each testing file was named after its respective page, suffixed with .test.js.

For example, if a page in our application is named Employee.js, we created a corresponding test file named Employee.test.js. Within these test files, we crafted unit tests focusing on the key features and functionalities of each page. These tests were designed to verify the expected behavior of the front-end components and ensure that they operate as intended.

Upon completing the creation of test files for each front-end page, we executed the tests using the command “npm test” in the terminal. Additionally, to assess the code coverage achieved by our testing suite, we utilized the command “npm test -- --coverage”. This command generated detailed insights into the extent of code covered by our tests, allowing us to identify areas that require further testing or optimization.

### **Code Coverage:**

The code coverage analysis provided valuable insights into the effectiveness of our testing efforts. The following screenshots illustrate the code coverage metrics obtained from our front-end testing:

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	81.75	52.23	73.94	82.23	
Components/CondoOwnerDashboard	100	100	100	100	
OwnerDashboard.js	100	100	100	100	
Components/DailyOperations	100	100	100	100	
DailyOperations.js	100	100	100	100	
Components/EmployeePage	64.51	0	38.46	64.51	
Employee.js	50	0	20	50	63-109,148,153-161,184-190
NewEmployeePopup.js	100	100	100	100	
Components/FinanceDashboard	100	100	100	100	
Finance.js	100	100	100	100	
Components/HomePage	100	100	100	100	
Home.js	100	100	100	100	
Components/ManagerEmployeePage	84.61	20	60	84.61	
ManagerEmployeePage.js	84.61	20	60	84.61	44-85
Components/NavBar	84.31	71.42	78.26	84.44	
NavBar.js	69.56	50	55.55	71.42	20,72-79
NavBar_Company.js	100	100	100	100	
NavBar_HomePage.js	100	100	100	100	
NavBar_User.js	91.66	100	83.33	90	20
Components/Popups	100	100	100	100	
MaintenanceRequest.js	100	100	100	100	
Notification.js	100	100	100	100	
PaymentHistory.js	100	100	100	100	
ReservationSuccess.js	100	100	100	100	
Components/PropertyProfileManagement	60	0	50	66.66	
PropertyProfileManagement.js	60	0	50	66.66	17-21
Components/RegistrationKey	80	100	50	80	
RegistrationKey.js	80	100	50	80	42
Components/ReservationPageCompany	80	73.91	70	80.59	
ReservationPageCompany.js	80	73.91	70	80.59	92,107-109,129-132,163,185,214-224,241-249
CustomHooks	0	100	0	0	
useLogout.js	0	100	0	0	4-19

Test Suites: 1 failed, 18 passed, 19 total  
 Tests: 84 passed, 84 total  
 Snapshots: 0 total  
 Time: 7.662 s

**Figure 6: Front-End Code Coverage**

We have three JavaScript components— Login.js, Profile.js, and SignUp.js — for which we have created corresponding .test.js files using Jest. However, these tests consistently fail due to an issue with the `'import axios from 'axios';'` statement. After exhausting all possible solutions to resolve this import issue, we have decided to exclude these components from our coverage calculations. Despite this, our overall coverage remains robust. Specifically, our Statement Coverage and Line Coverage both exceed 80%, while our Function Coverage stands at 73.94%, nearing our target of 80%. This demonstrates significant progress in enhancing our testing framework and overall coverage metrics. Thus, we consider this phase of our testing strategy to be largely successful, closely aligning with our goal of achieving 80% coverage.

### ***Link to Github Front-End Testing (Test Cases):***

<https://github.com/Tanya-STY/UrbanKey/tree/main/urbankey/src/FrontEndTesting>

## **Back-End Testing**

### **Introduction:**

Backend testing is essential for ensuring the functionality, performance, and security of web applications. In our Condo Management System Website project, we employed Pytest, a powerful testing framework for Python, to conduct backend testing. This report outlines the tools utilized, the testing approach adopted, and presents the code coverage achieved through our testing efforts.

### **Tools Used:**

Pytest was selected as the primary testing framework for our backend testing needs. With its simplicity and flexibility, Pytest provides a comprehensive suite of features for writing and executing tests in Python applications, making it an ideal choice for our project.

### **Testing Approach:**

To organize and streamline our backend testing efforts, we established a dedicated folder named "backend" within the project repository. Within this folder, we created individual testing files corresponding to different backend functionalities and pages of our website.

For instance, if a backend module is responsible for handling employee-related operations, we created a test file named `test_employee.py` within the `backend` folder. Following the conventions of Pytest, we crafted test functions within these files to validate the behavior of backend components and functionalities.

Our testing approach focused on covering various aspects of backend functionality, including data validation, business logic, API endpoints, and database interactions. Each test function was designed to verify specific scenarios and edge cases, ensuring comprehensive test coverage.

Upon completing the creation of test files and test functions, we executed the tests using Pytest's command-line interface. Pytest automatically discovered and executed the test cases within the `backend` folder, providing detailed feedback on test results and identifying any failures or errors encountered during testing.

### **Code Coverage:**

The code coverage analysis provided valuable insights into the effectiveness of our backend testing efforts. By integrating code coverage measurement into our testing workflow, we gained visibility into the percentage of code executed by our test suite and identified areas that require additional testing or optimization.

## Coverage report: 90%

coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

Module	statements	missing	excluded	coverage ↓
__init__.py	0	0	0	100%
config.py	6	0	0	100%
model\__init__.py	0	0	0	100%
model\handleRefresh.py	22	0	0	100%
routes\__init__.py	0	0	0	100%
services\__init__.py	0	0	0	100%
tests\__init__.py	0	0	0	100%
tests\test_auth.py	111	0	0	100%
tests\test_tokens.py	0	0	0	100%
tests\test_todo.py	135	1	0	99%
model\user.py	36	4	0	89%
routes\user_routes.py	27	3	0	89%
app.py	20	3	0	85%
services\token_service.py	27	6	0	78%
routes\auth_routes.py	18	5	0	72%
model\auth.py	81	26	0	68%
<b>Total</b>	<b>483</b>	<b>48</b>	<b>0</b>	<b>90%</b>

coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

**Figure 7: Back-End Code Coverage**

Coverage for **app.py**: 85%

20 statements | 17 run | 3 missing | 0 excluded

« prev ^ index » next coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

```

1 # app.py
2
3 from flask import Flask
4 from flask_cors import CORS
5 from flask_bcrypt import Bcrypt
6 from config import users, client
7 from routes.auth_routes import auth_routes # Import route blueprints
8 from routes.user_routes import user_routes
9
10 app = Flask(__name__)
11 app.config['SECRET_KEY'] = '0622d0d552f33f6309180901'
12 app.config['REFRESH_TOKEN_SECRET'] = '062444442d0d554442f33f63091804444901'
13 # CORS(app, origins="http://localhost:3000", supports_credentials=True)
14
15 cors_options = {
16     "origins": "http://localhost:3000",
17     "supports_credentials": True,
18     "options_succes_status": 200
19 }
20 CORS(app, **cors_options)
21
22
23 # Register route blueprints
24 app.register_blueprint(auth_routes)
25 app.register_blueprint(user_routes)
26
27 if __name__ == "__main__":
28     app.run(debug=True)
29
30 try:
31     client.admin.command('ping')
32     print("Pinged your deployment. You successfully connected to MongoDB!")
33 except Exception as e:
34     print(e)

```

Coverage for **model\handleRefresh.py**: 100%

22 statements | 22 run | 0 missing | 0 excluded

« prev ^ index » next coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

```

1 from flask import request, jsonify
2 from config import users
3 import jwt
4 from jwt.exceptions import ExpiredSignatureError, InvalidTokenError
5 from datetime import datetime, timedelta, timezone
6
7
8 def handle_refresh_token(app):
9     # Get the refresh token from cookies
10    cookies = request.cookies
11    if 'jwt' not in cookies:
12        return jsonify({'error': 'Refresh token not found in cookies'}), 401
13
14    refresh_token = cookies['jwt']
15
16    # Find the user with the given refresh token in the collection
17    found_user = users.find_one({'refreshToken': refresh_token})
18    if not found_user:
19        return jsonify({'error': 'User not found or invalid refresh token'}), 403
20
21    try:
22        # Verify the refresh token and extract username
23        decoded = jwt.decode(refresh_token, app.config['REFRESH_TOKEN_SECRET'])
24        email = decoded['username']
25    except ExpiredSignatureError:
26        return jsonify({'error': 'Refresh token has expired'}), 403
27    except InvalidTokenError:
28        return jsonify({'error': 'Invalid refresh token'}), 403
29
30    # Create new access token
31    token = jwt.encode({
32        'email': email,
33        'exp': datetime.now(timezone.utc) + timedelta(minutes=10) # Adjust the expiration time as needed
34

```

Coverage for **routes\user\_routes.py**: 89%

27 statements | 24 run | 3 missing | 0 excluded

« prev ^ index » next coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

```

1 from flask import Blueprint, request, jsonify
2 from functools import wraps
3 from model.user import getProfile, update_user_profile
4 from services.token_service import verify_token
5
6 user_routes = Blueprint('user_routes', __name__)
7
8 def token_required(f):
9     @wraps(f)
10    def decorated(*args, **kwargs):
11        token = request.headers.get('Authorization')
12
13        if not token or not token.startswith('Bearer '):
14            return jsonify(message='Unauthorized'), 401
15
16        token = token.split(' ')[1]
17
18        decoded_token = verify_token(token)
19        if not decoded_token:
20            return jsonify(message='Invalid or expired token'), 403
21
22        # Attach token payload to request object for easy access in route functions
23        request.email = decoded_token.get('email')
24        request.role = decoded_token.get('role')
25
26        return f(*args, **kwargs)
27
28    return decorated
29
30
31 @user_routes.route("/Profile", methods=['GET'])
32 @token_required
33 def profile_route():
34

```

Coverage for **tests\test\_auth.py**: 100%

111 statements | 111 run | 0 missing | 0 excluded

« prev ^ index » next coverage.py v7.4.4, created at 2024-03-21 05:29 -0400

```

1 from flask import Flask, jsonify, make_response, request
2 import flask
3 import jwt
4 from backend.config import users # Import the MongoDB collection and bcrypt instance
5 from backend.services.token_service import generate_access_token, generate_refresh_token, verify_refresh_token
6 from flask_bcrypt import Bcrypt
7 from unittest.mock import MagicMock, patch
8 import pytest
9 import sys
10 sys.path.append('C:\\\\Users\\\\Shamma\\\\Desktop\\\\UrbanKey-2\\\\backend')
11 from app import app
12 from backend.model.auth import signup, signin, refreshToken, handle_logout
13 from backend.model.handleRefresh import handle_refresh_token
14 from backend.model.user import getProfile, update_user_profile
15 from unittest.mock import MagicMock, patch
16 from pymongo import MongoClient
17
18 # commands to run tests: pytest --cov=. tests/ --cov-report xml:cov.xml
19
20 # Create a mock MongoDB collection
21 mongo_client_mock = MagicMock()
22
23 bcrypt_mock = MagicMock(return_value='test_password')
24
25 # Create a mock bcrypt instance
26 bcrypt = Bcrypt()
27
28 # Mock the Flask application and configuration
29 app = Flask(__name__)
30 app.config['SECRET_KEY'] = '0622d0d552f33f6309180901'
31 app.config['REFRESH_TOKEN_SECRET'] = '062444442d0d554442f33f63091804444901'
32
33 # Apply the configuration to the app
34 app.testing = True

```

**Figure 8**

New test files and cases, specifically "test\_reservations.py", "test\_reservationsCompany.py", and "test\_user.py", have been created and implemented to enhance backend testing. These additions complement our existing suite of tests. As a result, we have achieved an 85% coverage rate for our `app.py`, demonstrating the effectiveness of our testing efforts.

### ***Link to Github Backend Testing (Test Cases):***

- For "test\_reservations.py", "test\_finance.py" and "test\_user.py" files:  
<https://github.com/Tanya-STY/UrbanKey/tree/main/backend/tests>

- For "test\_todo.py" and "test\_auth.py" files:  
<https://github.com/Tanya-STY/UrbanKey/tree/backend-testing/backend/tests>

# Short Sprint #4 Retrospective

## Crafting Excellence

### Key Takeaways from Our Project Postmortem

#### Introduction

In this sprint retrospective, we reflect on the successes and challenges encountered as we navigated through our latest development cycle. One notable achievement was the seamless integration of front-end and back-end tasks, facilitated by proactive role adjustments within our team. By addressing communication gaps between developers, we optimized resource utilization and fostered a more cohesive workflow. This strategic realignment not only enhanced collaboration but also allowed us to capitalize on the skills and expertise of our team members more effectively.

Furthermore, this sprint highlighted the importance of structured deadlines in managing interdependent tasks. By setting clear timelines and priorities, we successfully mitigated bottlenecks and delays arising from concurrent development requirements. Additionally, we celebrated the stability and relevance of our existing documentation and diagrams, sparing valuable time and resources that would have been spent on extensive updates. Through equitable task distribution, we ensured that each team member had sufficient workload to contribute meaningfully to sprint objectives while maintaining a healthy balance with other commitments.

#### What went wrong

##### *1 - Communication between Front-End and Back-End Developers*

During this sprint, a notable challenge emerged due to communication gaps between our front-end and back-end developers. This resulted in inefficiencies and missed opportunities, particularly in terms of team allocation and resource utilization. Had communication been smoother, we could have seamlessly integrated more team members into the back-end tasks, enhancing our capacity to tackle the workload effectively. Recognizing the critical nature of streamlined communication, proactive measures were taken to address this issue promptly.

Consequently, to mitigate the impact of communication breakdowns, we implemented strategic changes within the team structure. Specifically, we realigned the roles of several individuals from front-end to back-end responsibilities. This adjustment not only facilitated better collaboration between the two teams but also optimized our workflow by leveraging the skills and expertise of team members more efficiently. Through these adaptations, we aim to foster a more cohesive and productive environment for future sprints.

##### *2 - Task Interdependence*

This sprint presented a challenge with numerous tasks interdependent on each other, particularly those requiring concurrent development of both front-end and back-end components for specific pages. This interdependence often led to bottlenecks and delays in completing tasks, hampering overall progress. To address this issue effectively, we implemented a structured approach by setting shorter deadlines within the team. Front-end developers were allocated a dedicated week to focus on their respective pages, enabling them to finalize their

work efficiently. This timeframe allocation allowed back-end developers the remaining duration of the sprint to seamlessly integrate their components, resulting in smoother coordination and improved task completion.

By establishing these clear deadlines and task priorities, we aimed to streamline the development process and mitigate delays caused by interdependencies. This approach not only enhanced team efficiency but also fostered better communication and collaboration, ensuring that both front-end and back-end tasks could progress harmoniously within the sprint timeframe.

## **What went right**

### **1 - Documentation and Diagrams**

This sprint saw a notable achievement as the majority of our diagrams and documentation remained unchanged, reflecting the stability and accuracy of our initial planning and execution. This spared valuable time and resources that would have otherwise been allocated to extensive updates.

Additionally, a significant portion of our documentation remained relevant, indicating effective communication and alignment throughout the team. For the sections that did require updates, we implemented a strategic approach to ensure efficiency. Tasks were carefully divided among team members, allowing those responsible for updates to focus their efforts without excessive time investment. This approach ensured that necessary revisions were completed promptly, while minimizing disruptions to ongoing development tasks.

### **2 - Clear Task Separation**

This sprint demonstrated effective task allocation, ensuring that workload distribution was equitable and manageable for all team members. Tasks were clearly delineated, allowing each member to have sufficient work to contribute meaningfully to sprint goals while balancing other commitments such as classes and projects.

This approach fostered a productive and sustainable work environment, where team members could efficiently fulfill their responsibilities without feeling overwhelmed or stretched thin. As a result, we were able to achieve our sprint objectives while also maintaining a healthy work-life balance for all involved.

## **Conclusion**

In conclusion, this sprint retrospective emphasizes the value of effective communication, strategic planning, and equitable workload distribution in achieving our project goals. By addressing challenges head-on and implementing proactive measures, we have strengthened our team dynamics and enhanced our productivity. Moving forward, we will continue to prioritize clear communication, efficient task management, and collaborative problem-solving to drive success in future sprints. With a shared commitment to continuous improvement and mutual support, we are well-positioned to tackle new challenges and achieve even greater milestones in the journeys ahead.

# Sprint 5 Release Plan

In Sprint 5 of the Condo Management System, our strategic direction is to enhance operational efficiency and user responsiveness through the introduction of sophisticated request management functionalities and further refinement of user interaction features. This sprint will focus on four main user stories, covering a breadth of functionalities from issue reporting by condo owners to advanced request handling by property managers. More specifically, 4 Sub User Stories were pushed from Sprint 4 to Sprint 5, those being 18.2, 19.1, 19.2 and 20.1. In addition to that, there are 5 general tasks that need to be completed, being the improvement of the front-end of the web app, the completeness of the backend testing, the deployment of the backend, the responsiveness of the website and the final documentation.

The primary objectives for this sprint include implementing a streamlined process for condo owners to report common area issues, enabling these issues to be promptly addressed. We are also set to enhance the request categorization process, employing automation to ensure efficient assignment of requests based on their nature. Alongside automation, we will maintain flexibility through the capability for manual reassignment, ensuring requests are always handled by the most suitable personnel. Moreover, our commitment extends to improving communication with users; we plan to implement robust notification systems to keep users informed about the status of their requests, enhancing transparency and user satisfaction. These developments will be supported by critical tasks such as backend testing to ensure reliability, comprehensive documentation to track and assess project progress, and deployment activities to make these features readily available to our users. Our commitment to enhancing frontend usability and responsiveness remains strong, with significant efforts dedicated to ensuring the system's interface is intuitive and accessible on various devices. This sprint will also see us finalizing preparations for a seamless deployment of backend improvements, aiming for minimal disruption and maximum performance efficiency.

As we proceed with these enhancements, we anticipate a significant boost in the system's functionality and user engagement. By the end of Sprint 5, we expect to deliver a complete, deployed and functional system that is not only more efficient in handling requests but also more engaging and user-friendly, ultimately supporting a cohesive and empowered condo community, and meeting all the necessary requirements for this project.

## Release Plan Legend (on the Excel sheet):

User Story ID	User Story Points (USP)	Priority	Status
User Stories are from # 18 to #20. The sub-user stories are in the format 2.1, 2.2, 3.1, etc.	The user story points are done based on the Fibonacci sequence.	<ul style="list-style-type: none"><li>● High</li><li>● Medium</li><li>● Low</li></ul>	<ul style="list-style-type: none"><li>● NOT STARTED</li><li>● TODO IN SPRINT 5</li><li>● DONE</li><li>● PUSHED TO SPRINT 5</li><li>● REMOVED</li></ul>

There are a total of 43 story points for Sprint 5, and approximately 46 hours of work.

**Link to the Release Plan Sprint 5 Excel file:**

[https://docs.google.com/spreadsheets/d/13q1o12V8uahZp1lSYp9ZLLBz-c9Qpgi/edit?usp=sharing&ouid=112342122863028184816&tfo=false&sd=true](https://docs.google.com/spreadsheets/d/13q1o12V8uahZp1lSYp9ZLLBz-c9Qpgi/edit?usp=sharing&ouid=112342122863028184816&tpo=false&sd=true)

# UI Prototypes

## Home Page

The screenshot displays a user interface for a real estate service. At the top, there is a navigation bar with links for Home, Property, About, Service, Contact, and a Log In button. To the right of the navigation is a user icon. Below the navigation is a large banner featuring a photograph of a house with the text "Your dream house is here." and a search bar.

Below the banner is a section titled "Featured Properties" showing four apartment listings, each with a thumbnail image, price (\$290,000), and a "View Property" button. There is also a "All" link.

Following this is a section titled "Featured Rental" showing four rental listings, each with a thumbnail image, price (\$3,000), and a "View Property" button. There is also a "All" link.

Next is a section titled "Search Your Dream House On The Map" which includes a placeholder text "Find the house you are looking for easily according to location information." and a "Search On Map" button. To the right of this text is a photograph of a person holding a smartphone displaying a map application.

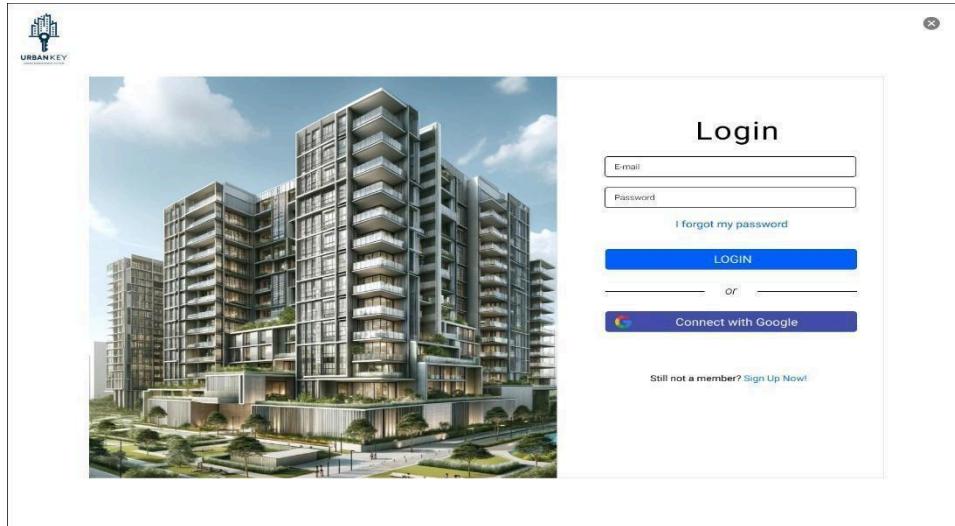
Finally, there is a section titled "Featured Property" featuring three property management companies: ProManage Property, Prime Property Management, and MasterKey Property. Each company has a thumbnail image of a building, a "View Property" button, and a brief description of its location and features.

The footer of the page contains three columns of links: "Homefast" (About Us, Our Awards, Client Materials, Advertisement, Human Resources, Sitemap), "Our Services" (Our Special Services, Membership, Company Membership, Projects, Advertisers For Free, Search On Map), and "Other" (Posting Rules, Terms of Use, User Agreement and Privacy Policy, Operation Guide, Cookie Policy, Collection and Use of Personal Data, Explicit Consent Text, Contact).

Figure 8

The homepage design for a real estate service features a sophisticated, user-centric interface. It includes an intuitive navigation bar, a striking banner for property searches, neatly organized listings of featured properties and rentals, and a map search functionality.

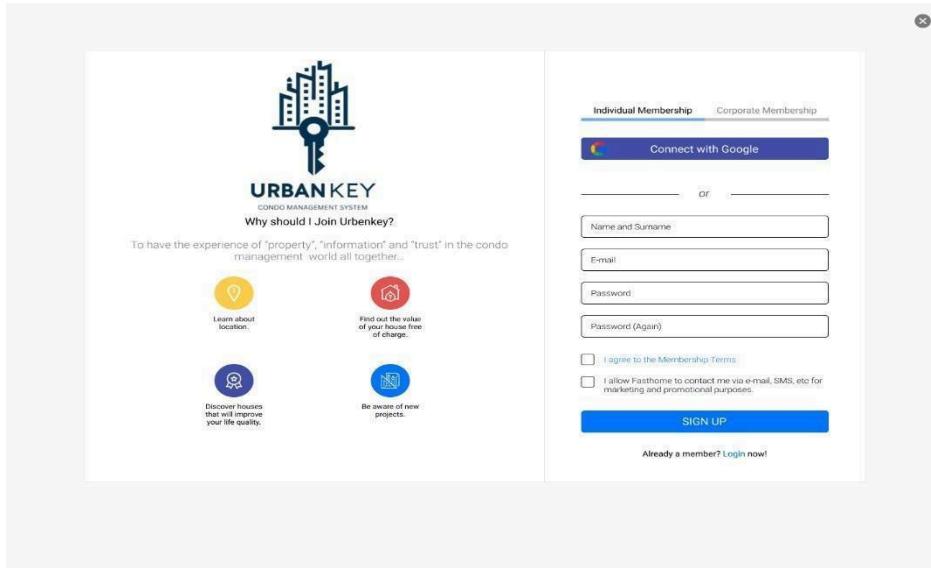
## Login



**Figure 9**

It offers users the choice to log in using their email and password or via Google. A link for those who have forgotten their password and a prompt to sign up for new members enhance the user experience with convenience and accessibility.

## Sign Up



**Figure 10**

The form provides a choice between individual or corporate membership and the option to register with options for Google sign up, and clearly presents terms of service and promotional contact opt-ins, ensuring an informative yet streamlined user journey.

# Property Page

**Single Person House**  
Montreal, QC

**General Information**

Condo No	0-0002	Floor Location	2
Purchase Date	20 November 2020	Furnished	Yes
Housing Shape	Apartment	Front	Northwest
Room + Living Number	1 + 1		
Gross / Net M <sup>2</sup>	50 M <sup>2</sup> / 110 M <sup>2</sup>		
Warming Type	Natural Gas		
Building Age	6		

**Financial Status**

- Monthly Condo Fees: \$4568 (paid)
- Outstanding Balances: \$0 (paid)

**Explanation**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Egestas ac convallis tellus pellentesque non odio consectetur bibendum. Auctor leo risus in tristique sit enim nec sed. Ridiculus vulputate facilisi a velit cursus sapien egestas nec, accumsan.

**Interior Features**

- ✓ ADSL
- ✓ Alarm
- ✓ Balcony
- ✓ Barbecue
- ✓ Laundry room
- ✓ Wallpaper
- ✓ Dressing Room
- ✓ Video Intercom
- ✓ Shower
- ✓ Laminate
- ✓ Panel Door
- ✓ Blinds
- ✓ Satin Plaster
- ✓ Satin Color
- ✓ Ceramic Floor

**External Features**

- ✓ Elevator
- ✓ Gardened
- ✓ Fitness
- ✓ Sauna
- ✓ Thermal Insulation
- ✓ Generator
- ✓ Tennis Court
- ✓ Car Park
- ✓ PVC
- ✓ Basketball Field
- ✓ Market

**Maintenance Requests**

Request #001	In Progress
Request #002	Completed
Request #003	Pending

**Location Information**

**Homefast**

- About Us
- Our Awards
- Corporate Materials
- Advertisement
- Human Resources
- Sitemap

**Our Services**

- Our Special Services
- Membership
- Corporate Membership
- Projects
- Advertise For Free
- Search On Map

**Other**

- Posting Rules
- Terms of Use
- Membership Agreement and Privacy Policy
- Opinion Guide
- Cookie Policy
- About Protection of Personal Data
- Explicit Consent Text
- Contact

Figure 11

## Notification

The screenshot shows a notification inbox interface. At the top, there are tabs for 'Inbox' (with a red badge '2'), 'Archived', 'All', and a help icon. To the right are settings and search icons. Below the tabs are two buttons: 'Mark all as read' and 'Archive read'. The main area lists three notifications from 'Micheal James': 'you have submitted your request.' (11 hours ago, Task List). The notifications are displayed in a list with a blue circular profile picture next to each name.

- Micheal James you have submitted your request.  
11 hours ago • Task List
- Micheal James you have submitted your request.  
11 hours ago • Task List
- Micheal James you have submitted your request.  
11 hours ago • Task List

Figure 12

## Payment

The screenshot shows a payment history table. The columns are: Payment Date, Amount, Payment Method, Reference Number, and Status. The data is as follows:

Payment Date	Amount	Payment Method	Reference Number	Status
September 2023	\$4568.00	Bank Transfer	12345	Paid
October 2023	\$4568.00	Credit Card	65078	Pending
November 2023	\$4568.00	Debit Card	78965	Rejected

At the bottom are 'Go Back' and 'See Invoices' buttons.

Figure 13

The property page layout is detailed, featuring a photo gallery, essential information about the property, and an interactive map. The design also includes sections for financial details for the user when they pay rent or mortgage and so on.

# Maintenance Requests

**Maintenance Requests**

Title \_\_\_\_\_

**Request Description**

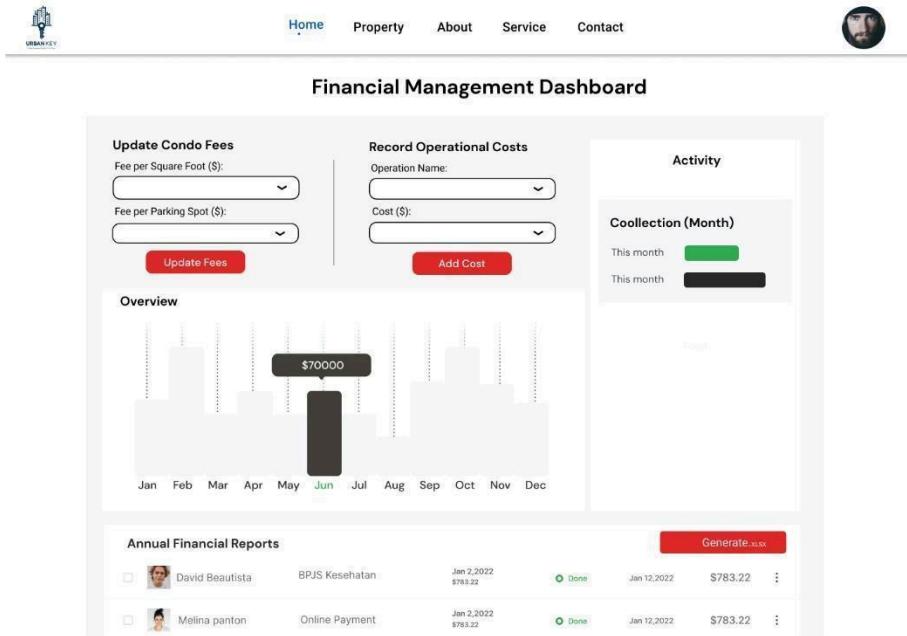
You can request about moving in/out (date for reserving elevators), intercom changes, requesting access (fobs, keys), reporting a violation, reporting deficiency found in common areas, or asking a question.

**Submit Your Request**

**Figure 14**

It encourages users to report issues or make requests related to property upkeep directly through the website.

# Financial Management Dashboard



**Figure 15**

That allows updating of condo fees, recording operational costs, and provides an overview of financial activities, including a bar graph and a section to get annual financial reports.

# Membership Information

The screenshot shows the 'Membership Information' page of the URBANKEY website. At the top, there is a navigation bar with links for Home, Property, About, Service, and Contact. To the right of the navigation is a user profile icon featuring a circular portrait of a man's face.

The main content area is titled 'Membership Information'. It contains several input fields:

- Name / Surname
- E-mail
- Province and City dropdown menus
- Mobile Number and Mobile Number 2 input fields
- Confirm Your password field (containing '\*\*\*\*\*')
- Address text area

To the right of the address field is a circular profile picture placeholder with a blue 'Upload Profile Picture' button below it.

Below the input fields is a section for preferences:

- E-mail and SMS buttons
- A checkbox for 'I want to be informed about all announcements and campaigns via commercial electronic mail.' followed by two radio buttons labeled 'No'.

At the bottom center is a red 'Save' button.

**Figure 16**

This allows users to manage their profile information, with fields for personal details, contact information, and preferences. It may also allow users to upload a profile picture and opt-in for notifications.

# Property Profile Management

The screenshot displays a web-based property listing form titled "Property Profile Management". At the top, there's a navigation bar with links for Home, Property, About, Service, and Contact. A logo for "Ion Structure Building Company" is visible in the top right corner.

**Property Details:**

- Category: Housing\*
- Unit ID\*
- Title\*
- Description\*
- Unit Owner\*
- Unit occupant information\*
- Price\*
- Number of rooms\*
- Number of Living Rooms\*
- Gross SF\*
- Nai SF\*
- Warming Type\*
- Building Age\*
- Floor location\*
- Available for Lease\*
- Furnished\*
- Parking\*
- Parking Spot ID
- Locker\*
- Rental Income\*
- Type\*

**Location Information:**

- Province\*
- City\*
- Neighborhood\*

**Upload Condo Files:**

You can upload your condo files which include: condo decorations, annual budgets, etc.

Buttons: Choose File, No file Chosen, Upload File.

**Posting Photos:**

You can add up to 30 photos.

Buttons: Browse From Computer.

**Advertise Features:**

**Interior Features:**

- Alarm
- Balcony
- Built-in Kitchen
- Basement
- Furnished
- Laundry Room
- Air Conditioning
- Wallpaper
- Dressing Room

**External Features:**

- Fence
- Gated
- Pools
- Security
- Thermal Insulation
- Generator

**Optional Features:**

- Video Intercom
- Jacuzzi
- Shower
- TV/Satellite
- Laminate
- Panel Door
- Mable Floor
- Blinds
- Seals
- Parent Bathroom
- Garage
- Sandstone
- Stain Glass
- Ceramic Floor
- Spotlight
- Fireplace
- Terrace
- Cobblestone
- Bedrock Heating
- Double Glazing
- Tennis Court
- Fire Escape
- Swimming Pool
- Football Field
- Basketball Field
- Market

**Buttons:**

- Send Registration Keys

Figure 17

This allows property owners to enter details about their property for listing purposes. It includes sections for property features, location mapping, file uploads for important documents, and a photo gallery.

# Reservation System

The screenshot shows a web-based facility booking system. At the top, there is a navigation bar with links for Home, Property, About, Service, and Contact. To the right of the navigation is a user profile icon. Below the navigation, the title "Facility Reservation System" is displayed. The main content area is titled "Book a Facility". It features two input fields: "Choose a Facility:" with a dropdown menu containing "Sky Lounge" (selected), "Spa & Fitness", and "Gym"; and "Select Date:" with a calendar for September 2021. The calendar highlights the 29th and 30th as reserved dates. A red "Book Now" button is located at the bottom of the form.

Figure 18



Congrats

Reservation has been successfully  
made.

[Back to Home](#)

Figure 19

This is the facility booking system, where users can select from available facilities and dates, followed by a confirmation message indicating a successful reservation.

# Employees

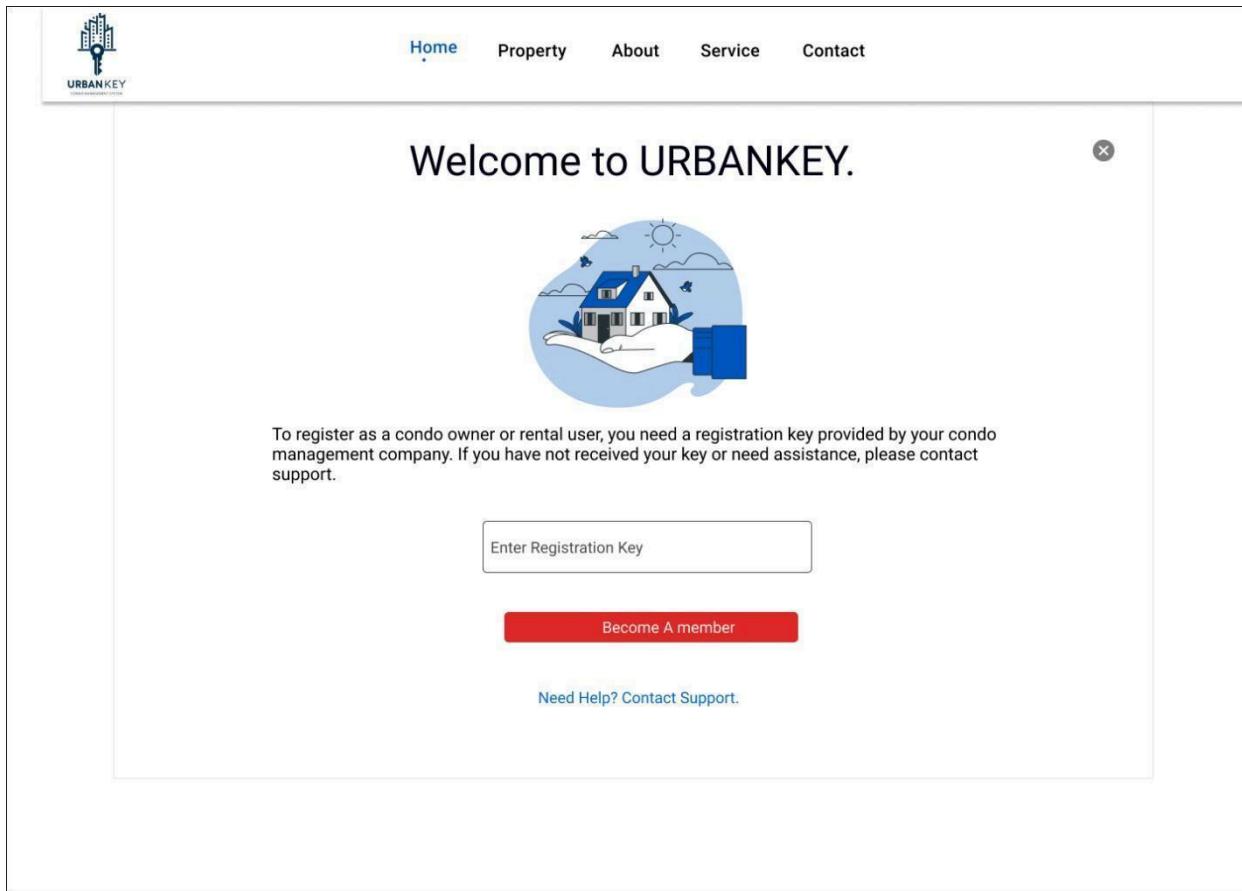
The screenshot shows a web-based employee management system. At the top, there is a navigation bar with links for Home, Property, About, Service, and Contact. On the far left is a logo for 'URBANKY'. On the far right is a user profile icon. Below the navigation bar, the page title is 'Employees' with a count of '100'. There are two buttons: 'Export' and 'New Employee'. Below the title, there are three tabs: 'Teams', 'Roles', and 'All Employees', with 'All Employees' being the active tab. A search bar allows users to search by name, role, ID, or keywords. To the right of the search bar are 'Filter' and 'View' buttons. The main content area displays a table of employee records. The columns are: Name, Employee ID, Role, Status, and Company Name. Each row contains a checkbox, a small profile picture, the employee's name, their email address, their Employee ID, their role (e.g., manager, associate), their status (Active or Inactive), and the company they work for. At the bottom of the table, there are navigation buttons for 'Previous' and 'Next' with page numbers 1 through 10.

Name	Employee ID	Role	Status	Company Name
Tanner Finsha @Tannerfisher@gmail.com	#23454GH6J7YT6	manager Full time	Active	Building Company
Emeto Winner Emetowinner@gmail.com	#23454GH6J7YT6	manager Contract	Active	Building Company
Tassy Omah Tassyomah@gmail.com	#23454GH6J7YT6	manager Associate	Inactive	Building Company
James Muriel JamesMuriel@Aerten.finance	#23454GH6J7YT6	manager Full time	Inactive	Building Company
Emeto Winner Emetowinner@gmail.com	#23454GH6J7YT6	daily operations Full time	Inactive	Building Company
Tassy Omah Tassyomah@gmail.com	#23454GH6J7YT6	daily operations Part time	Active	Building Company
James Muriel JamesMuriel@Aerten.finance	#23454GH6J7YT6	Finance Part time	Active	Building Company
Emeto Winner Emetowinner@gmail.com	#23454GH6J7YT6	Finance Part time	Inactive	Building Company

**Figure 20**

This is an employee management dashboard, listing employee details and providing functionality for filtering, searching, and managing employee records within a real estate or property management company.

## Registration Key Page



**Figure 21**

This is to prompt users to enter a registration key to gain access or to contact support for assistance.

# Reservation Page for Condo Company

The screenshot shows a web-based appointment booking system for a condo company. At the top, there's a navigation bar with links for Home, Property, About, Service, Contact, Appointments, and a user icon. Below the navigation is a search bar labeled "Search appointments...". The main area is titled "Appointments" and contains a table of scheduled appointments. The columns include Time, User Name, Owners type, Facility, Contact Number, Email, Creator, Status, waiting time, and Actions. The table lists various users like Braha Marlam Roh, Jordyn Dokidis, Carter Botosh, etc., along with their appointment details. A "New Appointment" button is located in the top right corner of the table header.

Time	User Name	Owners type	Facility	Contact Number	Email	Creator	Status	waiting time	Actions
7:00 - 7:10	Braha Marlam Roh	Condo	Sky Lounge	(912) 330-4635	caronni@optonline.net		Visited		
7:20 - 7:30	Jordyn Dokidis	Condo	Sky Lounge	(692) 656-8460	keijser@gmail.com		Visited		
7:40 - 7:50	Aspen Passaquindici Arcand	Condo	Spa & Fitness	(807) 317-8350	krueger@mac.com		Visited		
8:00 - 8:10	Carter Botosh	Condo	Spa & Fitness	(466) 233-0398	kosact@comcast.net		Scheduled		
8:20 - 8:30	Carter Smith	Condo	Sky Lounge	(435) 918-6920	jbartha@outlook.com		Waiting	05:54	
8:40 - 8:50	Ashlynn Botosh	Condo	Sky Lounge	(319) 263-6751	whimsy@outlook.com		Scheduled		
9:00 - 9:10	Ashlynn Botosh	Condo	Spa & Fitness	(796) 225-6190	pierce@yahoo.ca		Scheduled		
9:20 - 9:30	Talan Sepdmus	Condo	Spa & Fitness	(989) 440-1134	pakaste@sbcglobal.net		Scheduled		
9:40 - 9:50									
10:00 - 10:10	Alfonco Franci	condo	Spa & Fitness	(998) 440-1035	duncand@sbcglobal.net		Scheduled		

Figure 22

This interface displays all successful reservations made by condo owners. It features a table with reservation details including the user's email, the booked facility, and the selected date.

## Manager Employee Page

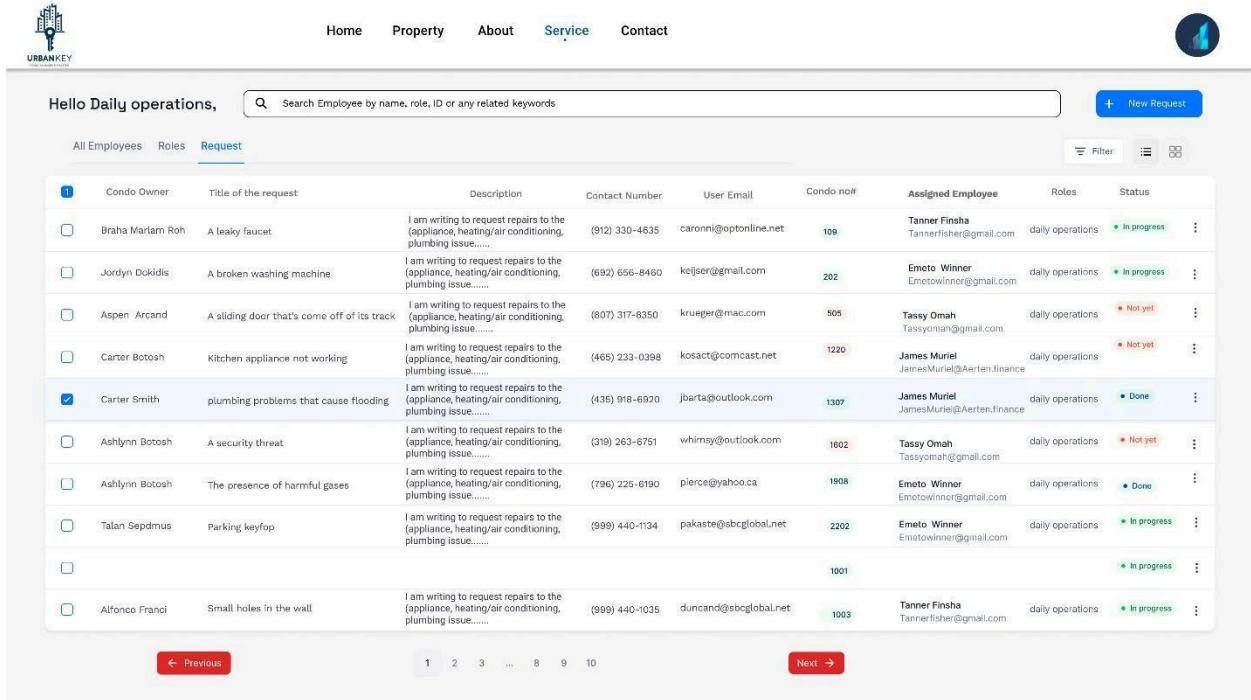
The screenshot shows a web-based request management system for a company. At the top, there's a navigation bar with links for Home, Property, About, Service, Contact, and a user icon. Below the navigation is a search bar labeled "Search Employee by name, role, ID or any related keywords". The main area is titled "Hello Manager," and contains a table of requests. The columns include Condo Owner, Title of the request, Description, Contact Number, User Email, Assigned Employee, Roles, Status, and ID. The table lists various users like Braha Marlam Roh, Jordyn Dokidis, Carter Botosh, etc., along with their request details. A "New Request" button is located in the top right corner of the table header.

Condo Owner	Title of the request	Description	Contact Number	User Email	Assigned Employee	Roles	Status	ID
Braha Marlam Roh	A leaky faucet	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(912) 330-4635	caronni@optonline.net	Tanner Finsha Tannerfisher@gmail.com	daily operations	Active	#234540H6J7YT6
Jordyn Dokidis	A broken washing machine	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(692) 656-8460	keijser@gmail.com	Emeto Winner Emetowinner@gmail.com	daily operations	Active	#234540H6J7YT6
Aspen Arcand	A sliding door that's come off of its track	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(807) 317-8350	krueger@mac.com	Tassy Omaha Tassyomaha@gmail.com	daily operations	Inactive	#234540H6J7YT6
Carter Botosh	Kitchen appliance not working	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(466) 233-0398	kosact@comcast.net	James Muriel JamesMuriel@Aerten.finace	daily operations	Inactive	#234540H6J7YT6
Carter Smith	plumbing problems that cause flooding	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(435) 918-6920	jbartha@outlook.com	James Muriel JamesMuriel@Aerten.finace	daily operations	Active	#234540H6J7YT6
Ashlynn Botosh	A security threat	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(319) 263-6751	whimsy@outlook.com	Tassy Omaha Tassyomaha@gmail.com	daily operations	Inactive	#234540H6J7YT6
Ashlynn Botosh	The presence of harmful gases	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(796) 225-6190	pierce@yahoo.ca	Emeto Winner Emetowinner@gmail.com	daily operations	Active	#234540H6J7YT6
Talan Sepdmus	Parking keyfop	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(989) 440-1134	pakaste@sbcglobal.net	Emeto Winner Emetowinner@gmail.com	daily operations	Active	#234540H6J7YT6
Alfonco Franci	Small holes in the wall	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(998) 440-1035	duncand@sbcglobal.net	Tanner Finsha Tannerfisher@gmail.com	daily operations	Active	#234540H6J7YT6

Figure 23

Allows a manager to assign daily operation tasks to employees. Shows owner names, request IDs, titles, descriptions, and request statuses.

## Daily Operation Employee Page



The screenshot shows a web-based application interface for managing daily operation tasks. At the top, there is a navigation bar with links for Home, Property, About, Service, Contact, and a user icon. Below the navigation bar is a search bar labeled "Search Employee by name, role, ID or any related keywords". A blue button labeled "+ New Request" is located to the right of the search bar. The main content area is titled "Hello Daily operations," followed by a table listing assigned requests. The table has columns for: Request ID (checkbox), Condo Owner, Title of the request, Description, Contact Number, User Email, Condo no#, Assigned Employee, Roles, and Status. The table lists 10 entries, each with a checkbox for selection. The status column includes icons indicating task status: green for "In progress", red for "Not yet", and blue for "Done". At the bottom of the table, there are navigation buttons for "Previous" and "Next" with page numbers 1 through 10.

All Employees	Roles	Request								
<input type="checkbox"/> Brahma Marlam Roh	Condo Owner	A leaky faucet	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(912) 330-4635	caronni@optonline.net	109	Tanner Finsha Tannerfisher@gmail.com	daily operations	<span>In progress</span>	<span>⋮</span>
<input type="checkbox"/> Jordyn Dokidis		A broken washing machine	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(692) 656-8460	keljser@gmail.com	202	Emeto Winner Emetowinner@gmail.com	daily operations	<span>In progress</span>	<span>⋮</span>
<input type="checkbox"/> Aspen Arcand		A sliding door that's come off of its track	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(807) 317-8350	krueger@mac.com	505	Tassy Omaha Tassyomaha@gmail.com	daily operations	<span>Not yet</span>	<span>⋮</span>
<input type="checkbox"/> Carter Botosh		Kitchen appliance not working	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(466) 233-0398	kosact@comcast.net	1220	James Muriel JamesMuriel@Aerten.financial	daily operations	<span>Not yet</span>	<span>⋮</span>
<input checked="" type="checkbox"/> Carter Smith		plumbing problems that cause flooding	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(436) 918-6020	jbarba@outlook.com	1307	James Muriel JamesMuriel@Aerten.financial	daily operations	<span>Done</span>	<span>⋮</span>
<input type="checkbox"/> Ashlynn Botosh		A security threat	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(319) 283-6751	whimsy@outlook.com	1602	Tassy Omaha Tassyomaha@gmail.com	daily operations	<span>Not yet</span>	<span>⋮</span>
<input type="checkbox"/> Ashlynn Botosh		The presence of harmful gases	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(796) 225-6190	pierce@yahoo.ca	1808	Emeto Winner Emetowinner@gmail.com	daily operations	<span>Done</span>	<span>⋮</span>
<input type="checkbox"/> Talan Sepdmus		Parking keyfop	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(999) 440-1134	pakaste@sbcglobal.net	2202	Emeto Winner Emetowinner@gmail.com	daily operations	<span>In progress</span>	<span>⋮</span>
<input type="checkbox"/>						1001			<span>In progress</span>	<span>⋮</span>
<input type="checkbox"/> Alfonco Franci		Small holes in the wall	I am writing to request repairs to the (appliance, heating/air conditioning, plumbing issue.....	(999) 440-1035	duncand@sbcglobal.net	1003	Tanner Finsha Tannerfisher@gmail.com	daily operations	<span>In progress</span>	<span>⋮</span>

Figure 24

For daily operation employees to view and manage their assigned tasks. They can update the status of tasks, which reflects on both the manager's page and the condo owner's dashboard.

# User Story Coverage Analysis

The Simplified Condo Management App is designed to facilitate the management of condos by providing a platform for public users, condo owners, rental users, and condo management companies to interact, manage properties, and access information. These functionalities are translated into user stories that describe the desired features from the perspective of the app's users. The aim is to ensure that the development efforts align with user needs and expectations, providing a clear and concise guide for the development team.

- Feature 1: Public User Profile Creation

- User Story 1.1: Ensures public users can engage with the community, highlighting the app's social integration and personalization capabilities.

## Registration and Verification

- Features 2 & 3: Registration Key for Ownership and Rental

- User Stories 2.1 & 3.1: Address the need for a secure and verified entry into the system as either condo owners or rental users, emphasizing the app's focus on security and proper authorization.

## Condo Owner Features

- Feature 4: Condo Owner Dashboard

- User Story 4.1: Allows condo owners to have a comprehensive view of their properties, ensuring a seamless management experience.

## Condo Management Company Features

- Features 5 to 7: Property Management

- User Stories 5.1, 6.1, & 7.1: Cater to the operational needs of condo management companies, facilitating document management, detailed property information entry, and profile management.

## Financial System Features

- Features 9 to 12: Financial Management

- User Stories 9.1, 10.1, & 11.1: Emphasize the importance of a simplified yet detailed financial system within the app, supporting budget management and financial transparency.

## Reservation System Features

- Features 13 to 16: Facility Reservation

- User Story 13.1: Ensures users can efficiently plan and book common facilities, enhancing user convenience and facility utilization.

## Role and Request Management Features

- Features 17 to 19: Role Assignment and Request Submission

- User Stories 17.1 & 18.1: Focus on the administrative aspect of condo management, allowing for efficient role assignment and request handling.

## Notification

- Feature 20: Notifications
  - User Story 20.1: Keeps users informed of activities related to their submissions or assignments, fostering an environment of transparency and responsiveness.

## Navigation and User Interface

- Feature 21: Navigation and Navbar
  - User Story 21.1: Underlines the necessity of an intuitive and user-friendly interface, allowing for easy navigation throughout the app's features.

### **Status Update:**

All Figma UI pages have been completed. We have successfully covered all aspects of the user interface design, ensuring comprehensive development coverage.

# Sprint 5 Testing Plan

This project documentation refers to the outlining of a comprehensive strategy for verifying and validating the software developed during the fifth sprint of the Condo Management System project. In order to meet the requirements of the project, it is crucial to incorporate a multi-level testing strategy that includes unit tests, integration tests, system tests, and a combination of automated and manual testing processes.

As a reminder, the stack we have selected and used for our Condo Management System is React for the front-end of the website app, as well as HTML/CSS/ JavaScript. For the backend, we have used Python alongside Flask, a micro web framework. Moreover, MongoDB was used as our NoSQL database. No APIs were used.

## **Testing Tool Selection:**

As we've progressed in the development of our project, we have continued to employ Pytest, a testing framework for Python, as our tool for backend testing. Additionally, we have incorporated the usage of Jest, a JavaScript testing framework, alongside the *React Testing Library* in order to test our front-end React components. We will stick with these tools moving forward in our subsequent sprints.

### **Front-end Testing:**

The combination of Jest and the React Testing Library allows us to thoroughly test the behavior and functionality of our React components in isolation. Each React component is accompanied by a corresponding `.test.js` file, which contains the unit tests for that component. These tests can be found in the [FrontEndTesting](#) directory of our GitHub repository.

### **Back-end Testing:**

Using Pytest, our tests are designed to verify the functionality of our Flask-based REST API endpoints and the underlying business logic. The back-end tests for "test\_auth.py" and "test\_todo.py" can be found in this [backend/tests-1](#) directory of our repository. Additionally, there are 4 test files that were added since Sprint 4, being "test\_finance.py", "test\_reservations.py", "test\_reservationsCompany.py" and "test\_user.py", which can be found here [backend/tests-2](#).

### **Examples of test file:**

- The "app.py" file assesses the functionality and robustness of the application's user profile management and authentication token processes. It simulates critical user interaction scenarios, such as profile retrieval, profile updates, and handling of refresh tokens under various conditions. By using MagicMock and patch from the unittest.mock library, the tests avoid direct database interactions and cryptographic operations, instead simulating these processes to validate the application's behavior in controlled environments. For example, it tests the profile update functionality by simulating a user request to

update their profile information, expecting successful completion or handling of exceptions gracefully. The refresh token handling is properly tested under different circumstances, including valid tokens, expired tokens, and invalid tokens, ensuring the application correctly validates and responds to each scenario. Additionally, the file incorporates tests for custom decorators used for route protection, verifying that routes are accessible only with valid authentication tokens and appropriately rejecting unauthorized access attempts. This approach to testing not only guarantees the security and efficiency of the user management and authentication systems but also ensures that potential issues can be identified and resolved promptly, enhancing the overall integrity and reliability of our web app.

- The “test\_auth.py” file is structured to conduct automated tests for the Flask-based backend, specifically focusing on user authentication mechanisms, utilizing the Pytest framework. It effectively simulates various authentication-related scenarios including user sign-up, sign-in, token refresh, and logout processes, thereby ensuring the integrity and security of the user management system. By harnessing mocks for database interactions and bcrypt hashing, the tests are insulated from real database transactions, enabling a controlled testing environment that mirrors expected operational behaviors without actual data persistence. For instance, the tests examine the system's handling of duplicate email registrations, verify the correctness of generated JWT tokens upon user login, and assess the system's response to token refresh requests, amongst other functionalities. This approach not only safeguards the application against unauthorized access scenarios but also rigorously checks the system's response to both routine and exceptional user behaviors. Through strategic mocking of database responses and cryptographic operations, these tests offer a solid examination of the authentication workflows, highlighting any potential discrepancies in the authentication logic. This testing procedure is critical in preemptively identifying and rectifying any vulnerabilities or logic flaws, hence strengthening the application's reliability.
- The “test\_finance.py” file is designed to test the financial-related functionalities within a Flask web application, ensuring that components work correctly without interacting with a real database. It starts by setting up the Python environment to include necessary directories in the system path, followed by importing essential modules like Flask and pytest. Mock objects and patching are utilized to simulate interactions with a MongoDB instance, preventing actual database operations during testing. The ‘app’ object is configured for testing, and the ‘financial\_routes’ blueprint is registered to handle related routes. Testing functions are then defined to verify the behavior of API endpoints such as retrieving and updating financial data, focusing on response status and content validation. This setup illustrates a robust approach to backend testing by isolating the application from its external dependencies, thereby enabling reliable unit tests.
- The “test\_reservations.py” file ensures that the application's reservation system operates seamlessly by configuring Flask for testing and employing Flask's test client for isolated testing environments. The tests cover a range of functionalities, from fetching all reservations to handling specific reservation requests with valid and invalid registration keys. They validate API endpoints to ensure correct status codes and data formats are returned, and verify the application's robust handling of both valid inputs and edge cases such as deactivated keys. This thorough testing strategy is critical for maintaining the reliability and integrity of the reservation system, thereby enhancing overall application stability and user satisfaction.

- The “test\_user.py” file verifies critical user profile operations, such as updating and fetching user profiles. They use mock data and headers to simulate real requests, checking both the execution flow and the accuracy of the response data against expected outcomes, such as correct status codes and appropriate response messages. This rigorous testing confirms the reliability of user profile endpoints, ensuring that they handle data correctly and respond as expected under various scenarios, ultimately aiding in maintaining the robustness and reliability of the user management system within the application.

## Levels of Testing

**Unit Testing:** The backend testing, as seen in files like “test\_auth.py” and “test\_todo.py”, encompasses unit tests targeting individual modules or functionalities, such as the authentication processes and todo item management. These tests, by leveraging mocks for dependencies like database connections, aim to verify that specific backend functionalities perform as expected independently from the rest of the system. On the frontend side, the files ending in `.test.js` are geared towards unit testing of React components. These tests critically assess whether individual components behave correctly in isolation, including checks for correct rendering, responsive user input handling, and appropriate state management changes.

For Sprint 5, we plan on expanding test coverage to ensure all critical paths and edge cases are examined as paramount, alongside enhancing test isolation to guarantee that tests evaluate components in a vacuum, hence avoiding unintended dependencies. We also want to refactor tests for clarity and maintainability ensures that the test suite remains accessible and easy to update alongside application changes.

**Integration Testing:** On the backend, represented by files like “test\_auth.py” and “test\_todo.py”, integration tests go beyond individual unit tests by simulating the interaction between various modules, such as authentication processes and database operations. These tests validate the cohesive functionality of the system under test scenarios that mirror real-world usage, albeit with dependencies like databases being mocked. This ensures that different parts of the application work harmoniously to accomplish tasks like user registration and todo item management. In the frontend, the React component tests (`.test.js` files) subtly incorporate integration testing by examining how components interact with each other and with the overall application state. Through the React Testing Library, these tests assess not only the rendering and user interaction within individual components but also their ability to integrate and function within the larger application ecosystem, such as handling API responses or user input events. This approach to testing underscores the importance of both isolated functionality and integrated performance, ensuring that all parts of the application operate cohesively to deliver a seamless user experience.

For Sprint 5, we plan to broaden the scope to encompass more comprehensive interactions between components, improving the simulation of production environments, and incorporating real database interactions can uncover more subtle integration issues.

**System Testing:** For Sprints 5, if we have time, we plan on implementing system tests, which would involve testing the complete application, including real database interactions, through automated UI tests or API tests that hit the actual endpoints without mocking the database or internal logic. We plan on implementing End-to-End (E2E) testing tests in Sprint 4 to test the application as a whole, simulating real user scenarios from

start to finish. This includes testing the integration between the frontend, backend, and any external services. We might use a tool such as Cypress, which is a powerful tool for E2E testing that can simulate real user interactions with our React application. Cypress tests can cover scenarios such as signing up, logging in, navigating through the app, and interacting with web elements.

**Automation:** Currently, our development process does not include the use of a Continuous Integration (CI) tool, such as Jenkins, Travis CI, or GitHub Actions, which would automate the execution of our tests upon every commit or merge request. However, depending on the time available in Sprint 5, we plan on selecting the most suitable automation tool for our web application's needs. This will facilitate early detection of flaws, and including performance benchmarks within tests.

### **Testing Approach:**

The testing strategy for this application is comprehensive, encompassing both backend and frontend to ensure robust coverage across all functionalities. On the backend, the approach leans heavily on unit testing to verify the correctness of individual modules, such as authentication and todo list management, in isolation. These tests are instrumental in validating the business logic and integration points within the application, employing mocks for external dependencies like databases to simulate real-world interactions without the overhead of actual database operations. This layer ensures that each component performs as expected under various scenarios, laying a solid foundation for application reliability.

Transitioning to the frontend, the testing regime utilizes the React Testing Library to focus on unit tests for individual React components, as demonstrated in the various `.test.js` files. These tests scrutinize components in isolation, checking for correct rendering, user interaction handling, and state management. Beyond unit testing, the frontend tests subtly weave in aspects of integration testing, examining how components interrelate and function within the larger application context, such as integrating with APIs and managing application-wide state changes.

Together, these testing approaches form a multi-layered defense, ensuring that both discrete functionalities and their interactions are thoroughly vetted. This strategy not only facilitates early detection of issues within isolated units but also safeguards against integration pitfalls, paving the way for a more reliable and maintainable application. Through diligent application of both unit and integration testing, the development team can confidently evolve the application, knowing that each change is properly evaluated to maintain the high standards of quality and user experience.

### **Metrics and Coverage:**

For Sprint 4, we used coverage tools which are integrated with our testing frameworks to measure and report coverage. For instance, to assess our front end coverage using Jest, we incorporated a test coverage script, named "test:coverage", into our package.json file. This script generates a coverage report, including metrics like statement, branch, function, and line coverage.

All metrics and coverage for the front end and the back end testing can be found in the Sprint 4 Testing Report on page 53 of this documentation.

In Sprint 5, we aim to increase our coverage % as much as possible, both for front-end and backend testing.

### **Acceptance Tests:**

At this point in the development process, no acceptance tests were developed yet in order to validate that the system performs as expected from an end-user perspective. If we have the time in Sprint 5, we will implement acceptance tests that will cover a wide range of scenarios for our website, including user registration, property and financial management functionalities, and the use of common facilities, to simulate real-world use effectively. We might use Behavior-Driven Development (BDD) tools like Cucumber or SpecFlow to write acceptance tests in a language that is understandable, however it has not been decided yet..

### **Implementation Plan:**

As our project progresses through its development phases, we will integrate testing into the development workflow by setting up a CI pipeline in Sprint 5, as mentioned previously, that runs tests automatically on every commit. This will ensure that the pipeline includes steps for running unit tests, integration tests, and system tests. Additionally, we will schedule regular review sessions to assess test coverage and effectiveness, adjust testing strategies as needed, and ensure that testing keeps pace with development. Lastly, we will attempt to encourage collaboration between the teammates in charge of both development and testing to ensure that testing reflects the user's needs and project requirements.

### **Link to Github Front-End Testing (Test Cases):**

<https://github.com/Tanya-STY/UrbanKey/tree/main/urbankey/src/FrontEndTesting>

### **Link to Github Backend Testing (Test Cases):**

- For “test\_reservations.py”, “test\_finance.py” and “test\_user.py” files:  
<https://github.com/Tanya-STY/UrbanKey/tree/main/backend/tests>

- For “test\_todo.py” and “test\_auth.py” files:  
<https://github.com/Tanya-STY/UrbanKey/tree/backend-testing/backend/tests>

# Code Management

## Project Overview:

Sprint 4 focused on creating the missing front-end pages and working on the backend to enhance the functionality and user experience of the platform. These pages and functionalities include:

### **Front-End**

- Reservation page for Condo Company
- Manager Employee Page
- Daily Operations Page

### **Back-End**

- Unique navigation bar for each user types functionality
- Edit profile photo functionality
- Make new reservation functionality
- Annual financial report of users graph functionality
- Payment history functionality
- Operational costs reflected in the financial status of the user dashboard functionality
- Company sending registration keys to users

## Code Management Approach:

In Sprint 4, we adopted a systematic code management strategy to ensure efficient collaboration and maintain high code quality. Each team member was responsible for developing specific pages or backend functionalities within the project. 5 team members were working on the front end of the pages and 5 team members were working on the backend of the website. We also assigned 2 team members for the front-end and backend-testing of the website.

## Workflow Overview:

### *Frontend Development*

Upon completion of frontend development for a page, the respective team member created a pull request on our GitHub repository.

### *Pull Request Management*

Two designated team members oversaw the pull request process, ensuring the integration of frontend pages into the website and verifying functionality.

### *Backend Development*

Following successful frontend integration, backend development commenced to implement the necessary functionality.

## *Issue Resolution*

In cases of encountered problems, team meetings were held to discuss and find appropriate solutions. Our team will also start using the ‘Issues’ functionality of Github for future problems.

## *Branch Management*

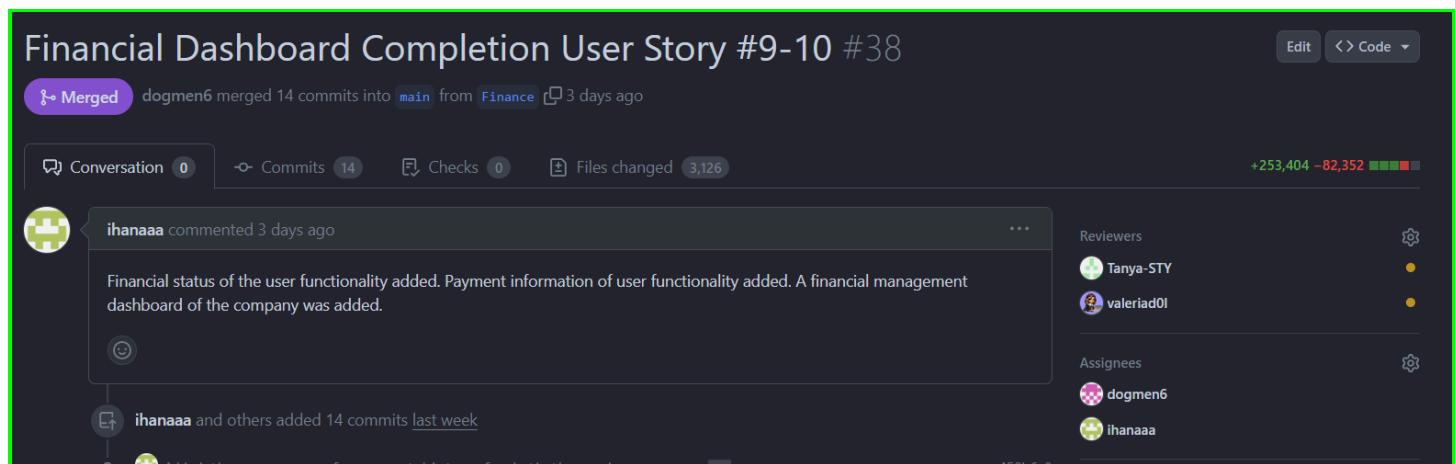
Our repository was organized with individual branches for each team member, facilitating clear tracking of contributions and merging branches for integration.

## **Code Quality and Management Metrics:**

### *Quality of Source Code Reviews*

During Sprint 4, our team implemented a robust peer code review process. This involved team members thoroughly examining each other's code to identify potential issues, provide feedback, and ensure adherence to project standards. This mandatory review process meant that each pull request had to be reviewed and approved by at least two team members before merging, as can be seen on Figure 26. By conducting rigorous code reviews, we aimed to maintain high code quality, improve overall understanding of the codebase, and foster knowledge sharing among team members.

To ensure comprehensive reviews, we enforced the practice of assigning reviewers to each pull request and issues. Assigning reviewers helped distribute responsibility for code inspection and ensured that multiple perspectives were considered during the review process. Additionally, we designated assignees to issues to clarify ownership and accountability for resolving specific tasks or bugs. By linking reviewers and assignees to pull requests and issues, we fostered a collaborative environment where feedback could be exchanged effectively, leading to improved code quality and faster resolution of issues.



**Figure 25**

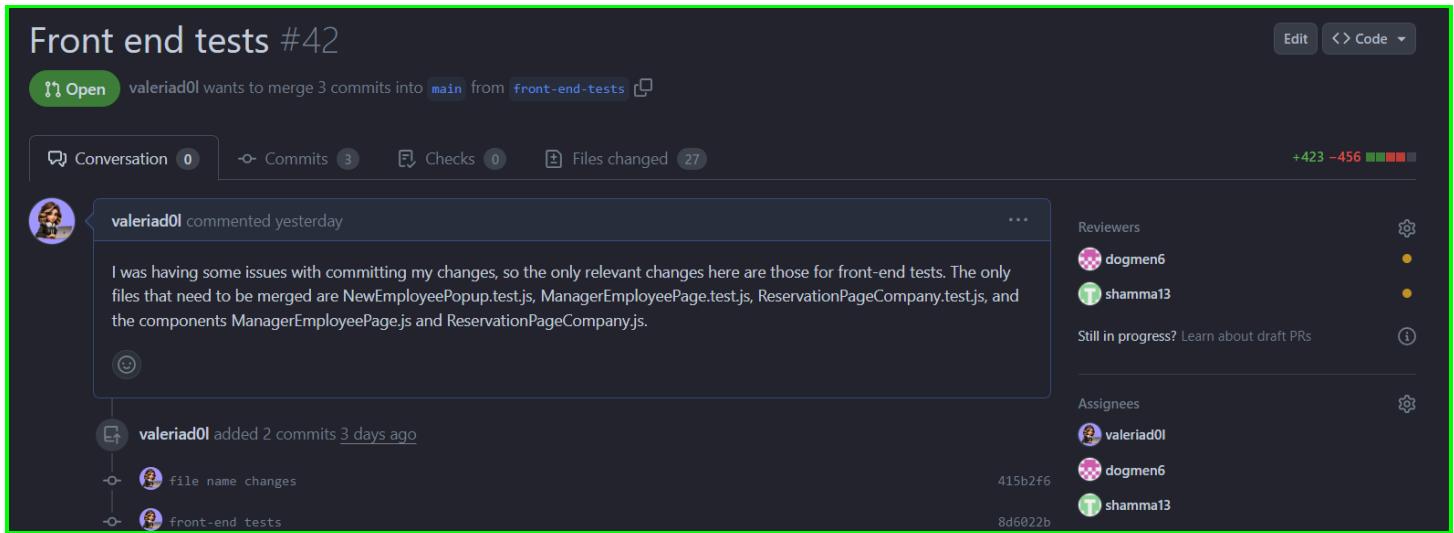


Figure 26

### Correct Use of Design Patterns

Design patterns were strategically employed throughout the development process to enhance code modularity and scalability. By leveraging established design patterns such as MVC (Model-View-Controller), Singleton, Factory, and Observer, we aimed to promote code reuse, improve maintainability, and facilitate future enhancements.

Additionally, we used the Component-Based Design Pattern, given our use of React and the creation of modular components. Examples of our components include the Registration Key Page, Homepage, Condo Owners Dashboard, Condo Management Company Profile for Property Under Their Management, Financial System, Reservation System, Notifications, and Employee Page. This approach not only ensured the efficient development of our web application but also laid the foundation for a scalable and easily maintainable system.

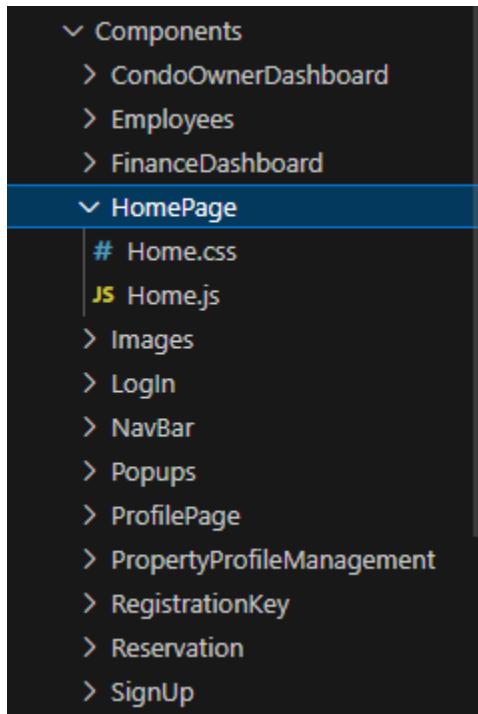


Figure 27

Moreover, we also used the In the "Arrange-Act-Assert" (AAA) pattern in our front-end testing code. This pattern is evident in the structure of each test case, where the setup (Arrange), action (Act), and assertion (Assert) phases are clearly delineated. Additionally, within each test case, there is a Given-When-Then structure, which is a variation of the AAA pattern, further organizing the tests into understandable sections. The use of this pattern enhances test readability, maintainability, and understandability by clearly separating concerns and making it easy to identify the purpose and flow of each test case. By adhering to the AAA pattern, we ensure that the tests remain focused, concise, and effective, facilitating better testing practices and overall code quality.

**Link:** [ReservationPageCompany.test.js](#)

```

test("renders date picker", () => {
  const { container } = render(<ReservationPageCompany />);
  const datePicker = container.querySelector(".date-picker");
  expect(datePicker).toBeInTheDocument();
});

test("renders facility checkboxes", () => {
  render(<ReservationPageCompany />);
  const facilityOptions = screen.getAllByRole("checkbox");
  expect(facilityOptions.length).toBeGreaterThan(0);
});

test("updates search term on user input", () => {
  render(<ReservationPageCompany />);
  const searchInput = screen.getByPlaceholderText(
    "Search by name, contact number, email"
  );
  fireEvent.change(searchInput, { target: { value: "John" } });
  expect(searchInput.value).toBe("John");
});

```

**Figure 28**

#### ***Justification for Design Patterns and Benefits:***

Overall, we employed several design patterns thoughtfully chosen to optimize code modularity, maintainability, and scalability. The MVC (Model-View-Controller) pattern was applied broadly, notably in the Condo Owners Dashboard and Financial System, to enhance separation of concerns and ease of maintenance, allowing for focused development and scalable enhancements. The Singleton pattern managed critical shared resources like user sessions and configuration settings, ensuring controlled access and maintaining consistent application state. Factory patterns facilitated the dynamic creation of user profiles and notifications, particularly within the Notifications and Employee Page modules, by allowing flexibility in object creation and easing the integration of new types as the system evolves. The Observer pattern was crucial in the Notifications system, supporting decoupled components and dynamic real-time updates. We also implemented Component-Based Design across all React components, including the Registration Key Page and Homepage, which promoted reusability and isolated maintenance efforts. For testing, the Arrange-Act-Assert (AAA) and Given-When-Then patterns structured our front-end tests, such as those in `ReservationPageCompany.test.js`, to enhance clarity, maintainability, and efficiency by clearly delineating setup, actions, and assertions. These design patterns collectively not only ensured efficient development and maintenance of our web application but also laid a strong foundation for future scalability.

## *Respect to Code Conventions*

Adherence to established code conventions was a cornerstone of our development approach. We enforced consistent coding styles, naming conventions, and formatting guidelines across the project to ensure readability and maintainability. By maintaining uniformity in coding practices, we aimed to minimize confusion, streamline collaboration, and promote code comprehensibility for all team members.

For example, in our components, such as the RegistrationKey component, we consistently apply camelCase naming conventions for variables and functions, maintain a clear folder structure, and use descriptive class names and comments to improve code readability and maintainability. This adherence to conventions ensures that our codebase remains cohesive and easy to understand, promoting efficient development and scalability.

```
<h1 className="welcomeTitle">Welcome to URBANKEY.</h1>
<div className="houseImage">
  <img src= {houseImage} alt="House Image"/>
</div>

<div className="registrationText">
  To register as a condo owner or rental user, you need a registration key provided by your condo management company.
  <br/>
  If you have not received your key or need assistance, please contact support.
</div>
```

**Figure 29**

## *Design Quality*

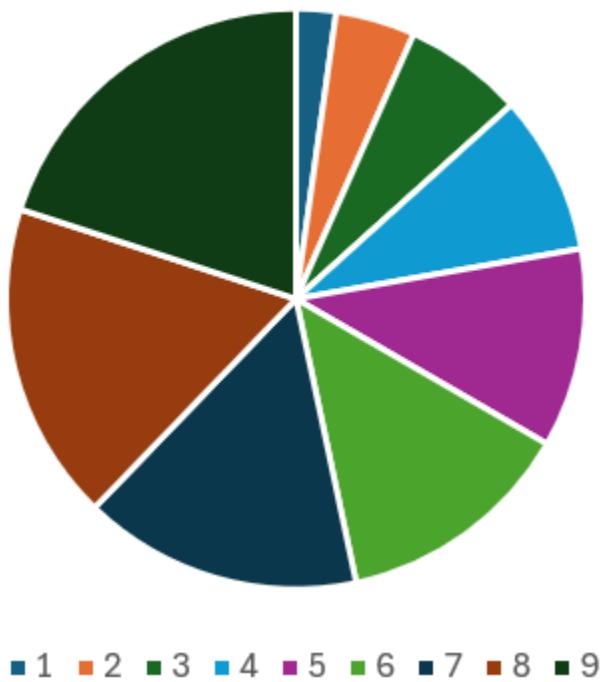
Emphasis was placed on optimizing the design of our codebase to enhance maintainability and extensibility. We carefully evaluated factors such as the number of classes/packages, code size, coupling, and cohesion to ensure a well-structured and scalable architecture. By maintaining a balance between granularity and cohesion, we aimed to facilitate easier navigation, debugging, and future enhancements of the codebase.

### *Code Metrics:*

Code metrics offer quantitative assessments of different facets of code quality, aiding developers in gauging the efficacy and maintainability of their codebase. Metrics like cyclomatic complexity, essential complexity, lines of code, and code duplication serve as indicators, pinpointing areas within the codebase that necessitate enhancement.

Cyclomatic complexity quantifies the number of linearly independent paths through a program's source code. In simpler terms, it measures how many unique paths there are through a program's code, which can indicate its overall complexity and potential difficulty to understand, maintain, and test.

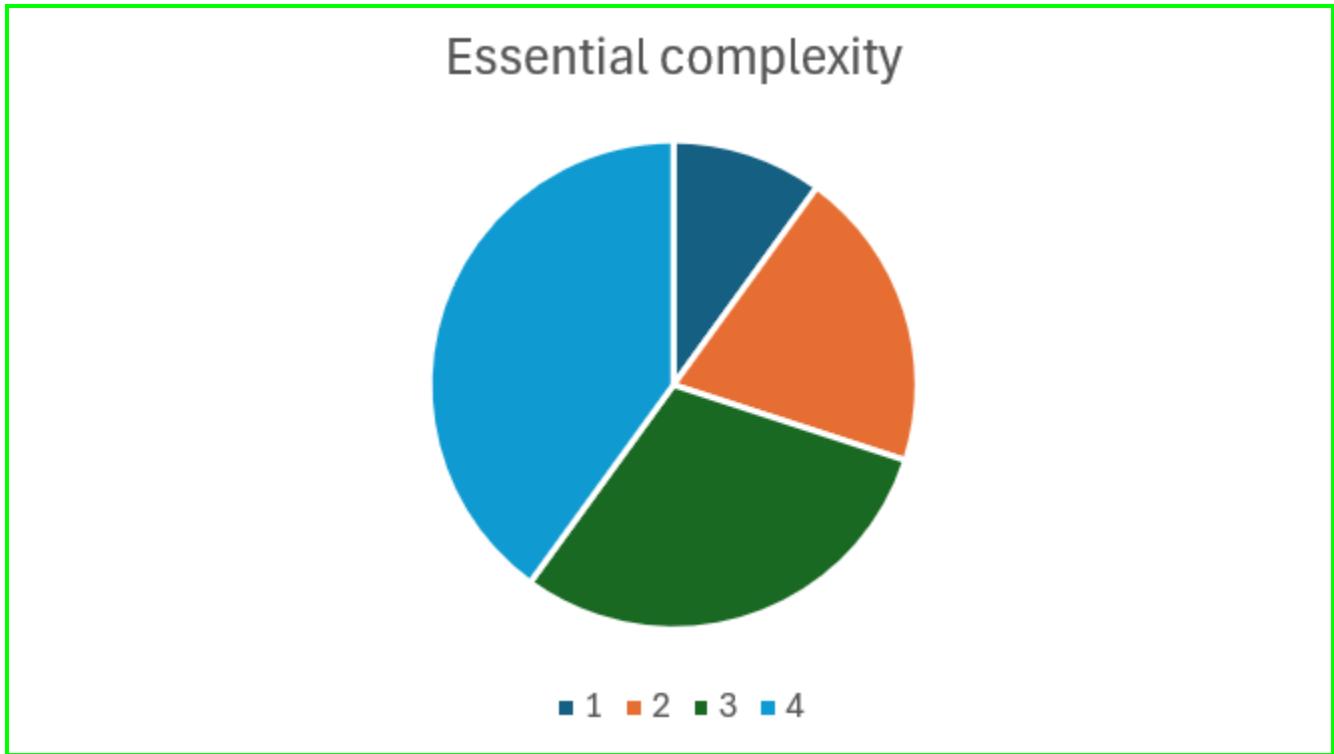
## Cyclomatic Complexity



**Figure 30**

As we can see in the above graph, the majority of our code's cyclomatic complexity is of 1, however, the cyclomatic complexity of some functions has increased. This indicates that the number of independent paths in a function has augmented, but is still 0 for most of them. Therefore, the code complexity has augmented but is still easy to maintain.

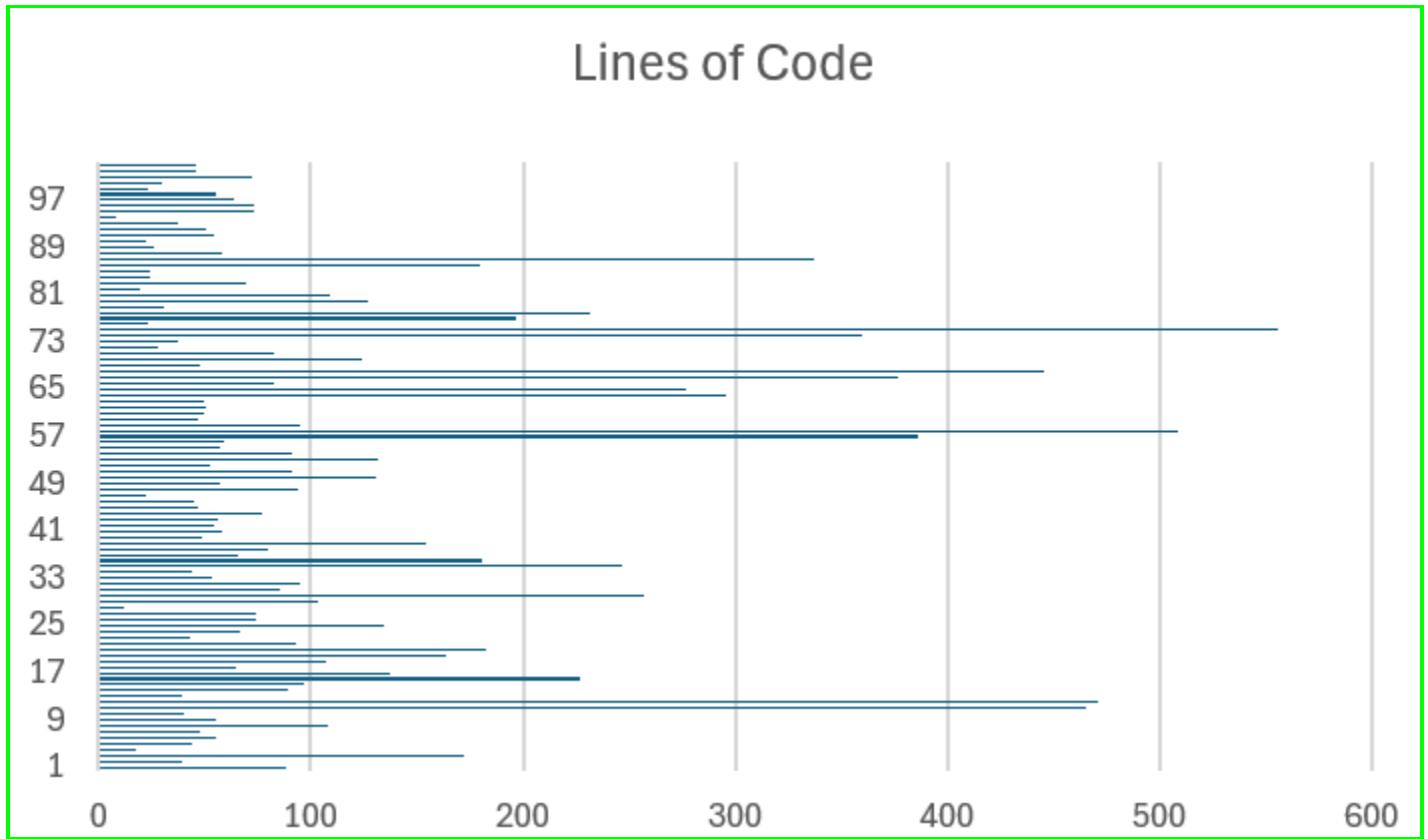
Essential complexity is a metric aimed at assessing the structural quality of a program. It evaluates the count of entry points, termination points, and non-deductible nodes. A value closer to 1 indicates a better structured program.



**Figure 31**

As we can see in the above graph, the essential complexity of the code is mostly 4. This demonstrates that our project's complexity has increased, but is still pretty manageable.

Lines of code is a simple and commonly used metric in software development that quantifies the size of a program by counting the number of lines in the source code. This metric includes all lines of code, including comments and blank lines, and provides a rough estimate of the overall complexity and scale of a software project.



**Figure 32**

As we can see in the above graph, most of LOCs in our project files have augmented, meaning that the complexity of the project has also augmented. This also means that the project is harder to maintain and to review.

### *Source Code Documentation*

Comprehensive documentation accompanied our source code to provide insights into its functionality, usage, and implementation details. Most of the code documentation we used is in the form of inline comments next to our code. This documentation served as a valuable resource for understanding the codebase, facilitating future development efforts, and onboarding new team members. By documenting key aspects of our code, we aimed to promote clarity, reduce ambiguity, and enhance maintainability throughout the project lifecycle.

```
// Generate annual report
setAnnualReport(sampleData);

// Generate overview data
const monthData = Array.from({ length: 12 }, () => 0);
sampleData.forEach((item) => {
  const month = new Date(item.date).getMonth();
  monthData[month] += item.amountPaid;
});
```

**Figure 33**

We also followed code documentation guidelines by adding documentation at the beginning of code files to explain the purpose of the files, such as the following figure.

```
// This custom hook, useAxiosPrivate, is designed to provide an Axios instance with built-in functionality for handling authentication tokens and automatic token refreshing.  
// It utilizes the useRefreshToken and useAuth hooks to access authentication-related state and functionality.  
// The Axios instance created here includes request and response interceptors to automatically attach the authentication token to outgoing requests and handle token refreshing in case of a 403 (Forbidden) response.  
// By encapsulating this logic within a custom hook, components can easily access a pre-configured Axios instance for making authenticated requests while abstracting away the complexities of token management and refreshing.  
  
import axios from 'axios';  
import { useEffect } from "react";  
import useRefreshToken from "./useRefreshToken";  
import useAuth from "./useAuth";  
  
const BASE_URL = 'http://localhost:5000';  
  
const axiosInstance = axios.create({  
  baseURL: BASE_URL,  
  headers: { 'Content-Type': 'application/json' },  
  ...  
});  
  
export default useAxiosPrivate;
```

Figure 34

Link: <https://github.com/Tanya-STY/UrbanKey/blob/main/urbankey/src/CustomeHooks/axiosPrivate.js>

### Refactoring Activity

Refactoring activities were systematically documented in commit messages to track code improvements and optimizations. Whenever code refactoring was performed to enhance readability, performance, or maintainability, detailed explanations were provided in commit messages. By documenting refactoring activities, we aimed to maintain transparency, communicate rationale behind code changes, and ensure continuous improvement of the codebase.

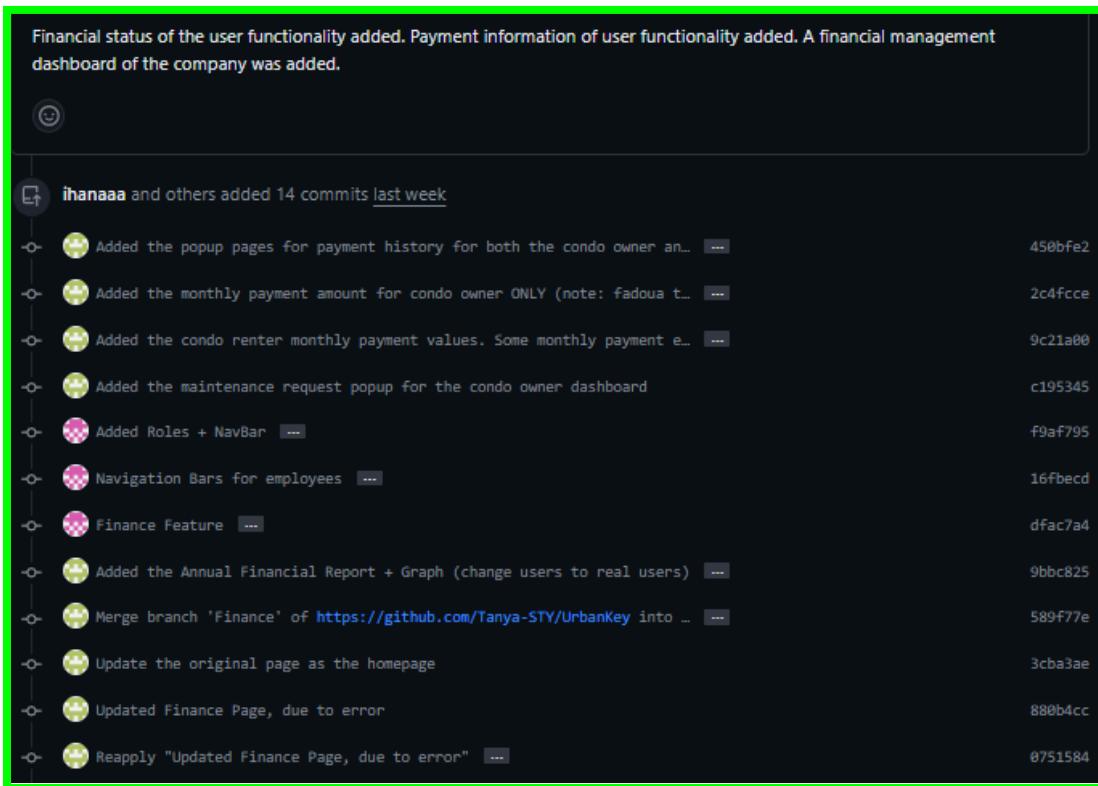


Figure 35

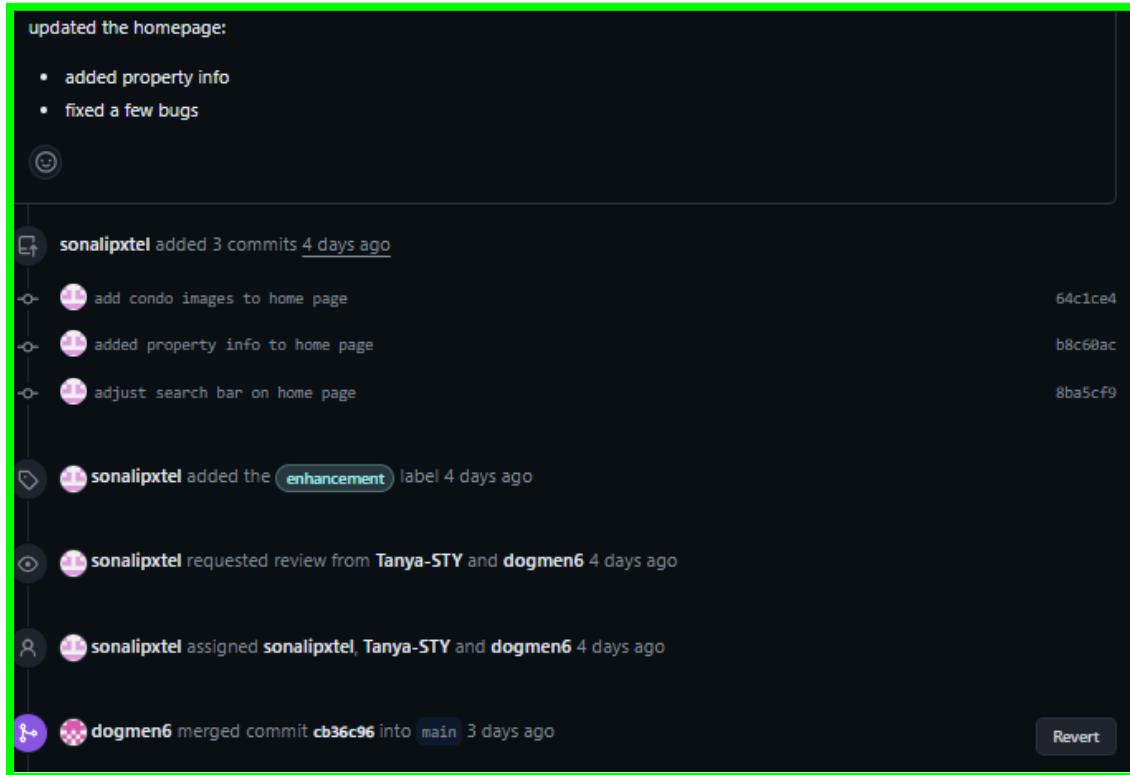


Figure 36

## Links: Update Homepage & Financial Dashboard Completion

### Commit Message Quality

Commit messages were crafted to be detailed and informative, providing context and rationale for each code change. Clear and concise commit messages helped team members understand the purpose and impact of code modifications, facilitating smoother code review processes and enhancing overall collaboration. By adhering to high standards of commit message quality, we aimed to improve traceability, code comprehension, and collaboration efficiency.

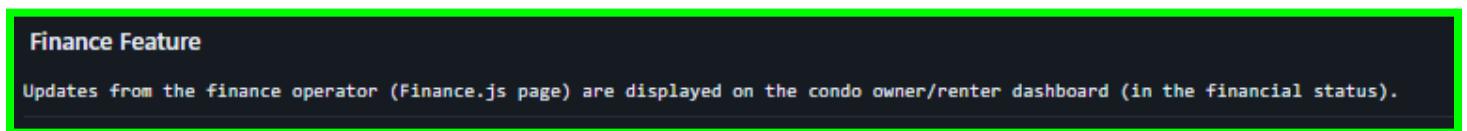


Figure 37

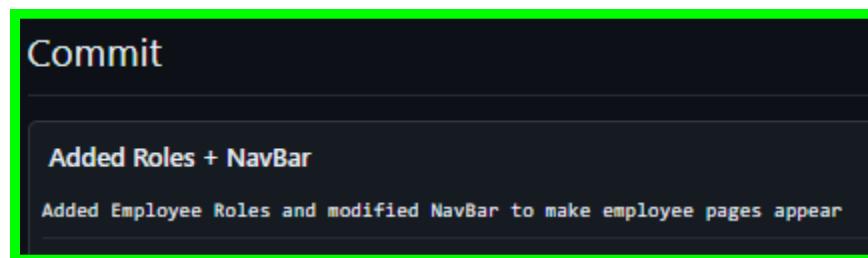
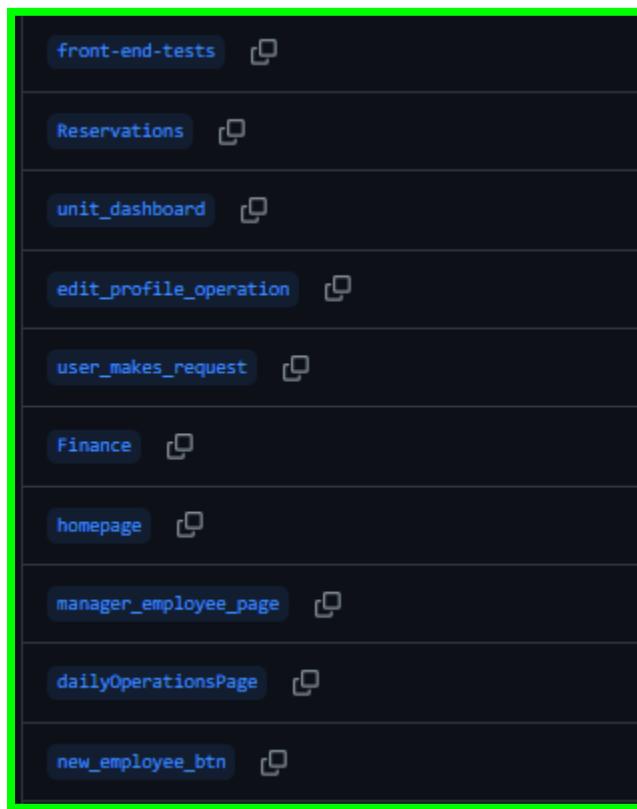


Figure 38

## Links: Added Roles + Navbar & Finance Feature

### *Use of Feature Branches*

Feature branches are essential for our development workflow, providing dedicated spaces for implementing specific features or functionalities on our website. Instead of using personal branches in the repository, we switched to enforcing a feature-branch workflow where each new feature, bug fix, or task has its own branch derived from and merged back into the main branch upon completion. For each element, such as frontend testing, reservations, and finance, we create separate branches. This approach allows us to work independently, minimizing conflicts and streamlining collaboration. By isolating changes, feature branches ensure that new features are thoroughly tested and integrated smoothly into the main codebase. Overall, feature branches play a crucial role in our development process, enabling organized and efficient development across all aspects of our website.



**Figure 39**

### *Atomic Commits*

Commits were kept atomic, focusing on a single logical change to enhance code review and facilitate rollback if necessary. By breaking down changes into smaller, self-contained units, we aimed to improve code traceability, reviewability, and maintainability. Atomic commits also facilitated easier identification and resolution of issues, contributing to overall code quality and stability.

## Commit

### Updated NavBars

I added the Reservation option for the User and added an import to make the profile page appear

Figure 40

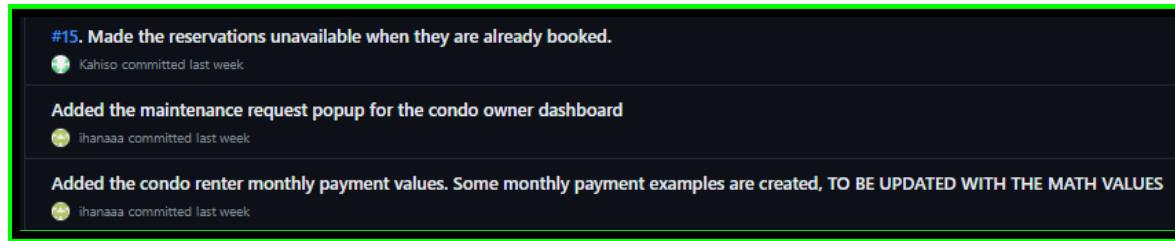


Figure 41

## Bug Reporting

If we encounter any issues or bugs, we systematically reported and tracked them using GitHub's issue-tracking system. By utilizing GitHub's issue-tracking system, we aim to ensure timely resolution of bugs, improve software reliability, and enhance user satisfaction.

<u>ID</u>	<u>User Story</u>	<u>Label</u>	<u>Description</u>	<u>Link to Bug</u>	<u>Status</u>
#17	#5	bug	The issue with the Property Profile Management page is that the sidebar cannot be placed in the correct position	<a href="https://github.com/Tanya-STY/UserKey/issues/17">https://github.com/Tanya-STY/UserKey/issues/17</a>	Completed
#18	#4	bug	Zooming in and out messes up with the CSS	<a href="https://github.com/Tanya-STY/UserKey/issues/18">https://github.com/Tanya-STY/UserKey/issues/18</a>	In Progress
#30	#5	bug & enhancement	CSS bug in Profile Management	<a href="https://github.com/Tanya-STY/UserKey/issues/30">https://github.com/Tanya-STY/UserKey/issues/30</a>	In Progress
#35	#1	bug, enhancement & help wanted	Edit Profile Operation	<a href="https://github.com/Tanya-STY/UserKey/pull/35">https://github.com/Tanya-STY/UserKey/pull/35</a>	Completed

Table 2: Bug Report

## Use of Labels for Issues/Pull Requests

In Sprint 4, we utilized issue labels for tracking and filtering tasks. By categorizing tasks and pull requests using appropriate labels such as bug, enhancement, feature, or priority level, we aim to prioritize effectively, allocate resources efficiently, and provide clear visibility into the status of ongoing development efforts. It also makes our Github repository more organized for all team members. The labels used for Sprint 4 included ‘bug’, ‘enhancement’, ‘help wanted’ and ‘documentation’.



Figure 42

A screenshot of a GitHub pull request details page for 'Edit profile operation #35'. The pull request is merged. The conversation tab shows a comment from 'mzmddi' about fixing a bug related to photo uploads. The commit section shows 10 commits from 'dogmen6'. The checks and files changed sections are empty. On the right, there are sections for Reviewers (Tanya-STY, dogmen6), Assignees (Tanya-STY, dogmen6), Labels (bug, enhancement, help wanted), and Projects (None yet). A code review interface is visible on the far right.

Figure 43

[Link: Edit Profile Operation](#)

## Links Between Commits and Issues

By systematically linking commits to relevant bug reports or features, we aim to enhance traceability, accountability, and collaboration within the development team. This approach facilitates easier navigation, contextual understanding, and efficient management of code changes.

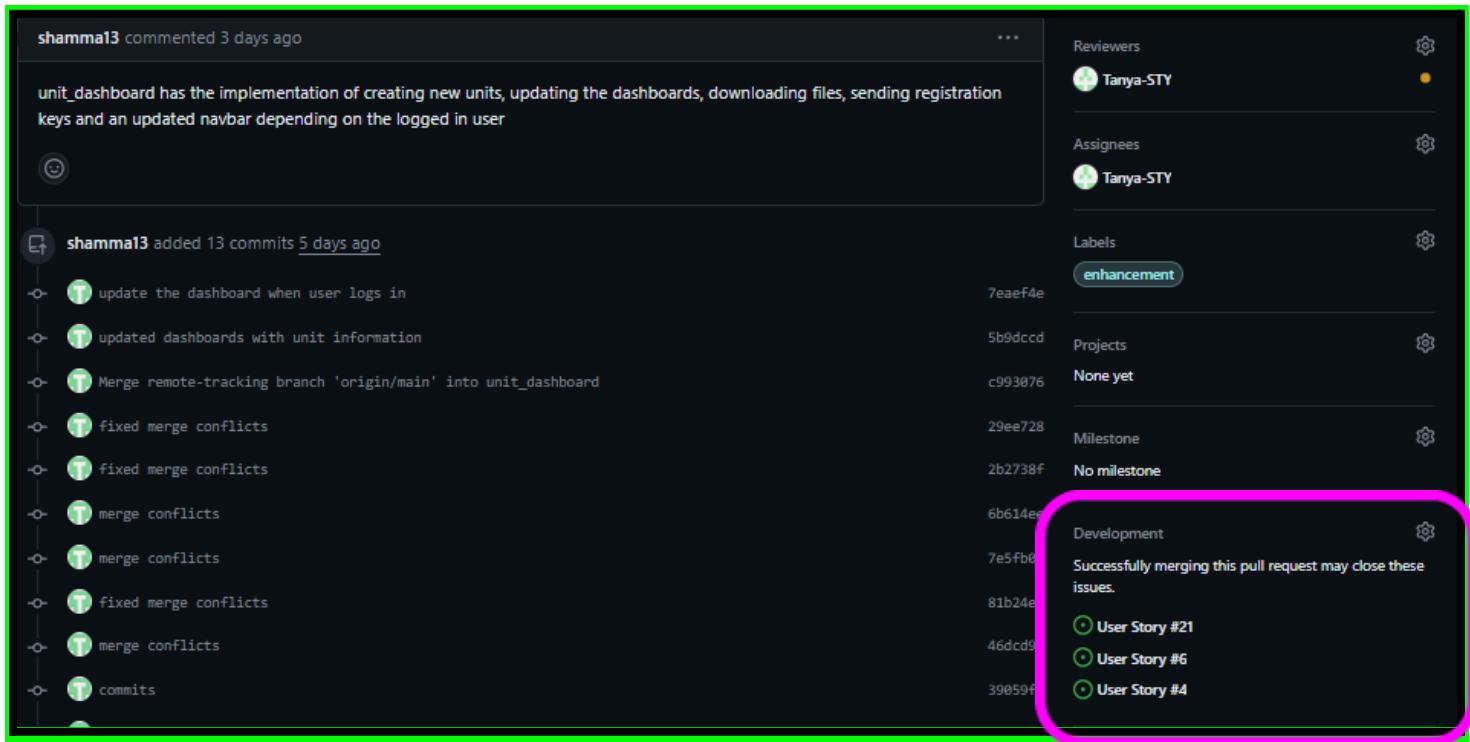


Figure 44

Link: <https://github.com/Tanya-STY/UrbanKey/pull/41>

### Conclusion:

The code management process employed during Sprint 4 ensured effective collaboration, maintained code quality, and facilitated the timely delivery of project milestones. By adhering to established practices and leveraging collaborative tools, our team successfully navigated the complexities of development, paving the way for future iterations and enhancements of the condominium management website.

# Deployment

## Front-End Deployment

For the deployment of our condo management React website, we opted for AWS S3 for frontend deployment, ensuring seamless access and scalability. Our frontend components, including the user interface and interactive features, are securely hosted on S3, allowing users to access the website effortlessly. With AWS S3, we leverage its robust infrastructure and global content delivery network (CDN) to ensure high availability and fast performance for our users. You can access our deployed frontend at the provided link, which offers a comprehensive view of the condo management system's frontend functionalities.

**Link to the Deployed Front-End:** <http://urbankey.s3-website.us-east-2.amazonaws.com/>

## Back-End Deployment

On the backend deployment front, we encountered challenges with AWS Lambda, leading to unresolved issues. As a result, we have deferred backend deployment to sprint 5 and are exploring alternative solutions such as AWS EC2 and Docker. By transitioning to AWS EC2, we aim to deploy our backend services within scalable virtual machines, providing greater control and flexibility over resource allocation. Additionally, Docker containers offer a portable and consistent deployment environment, facilitating smoother deployment and management of our backend services. With these adjustments, we aim to resolve deployment issues and ensure the seamless operation of both frontend and backend components in our condo management system.