SOEN 390

SOFTWARE ENGINEERING TEAM DESIGN PROJECT

Delivery Document

Instructor: Dr. Jinqiu Yang
Date: March 21 2024

Daniel Duguay 40202775
Nicolas Chelico 40156158
Walid Achlaf 40210355
Wadeh Hamati 40216893
Charles Eimer 26747310
Houssam Righi 40155074
Ceyhun Topcu 40159200
Monika Moanes 40188452
Mohammed Rahman 40203098
Khalil Garaali 40226310
Youssef Alsheghri 40108014

Chapter 1: Product vision statement

1.1 Introduction

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

1.2 Positioning

1.2.1 Problem Statement

The problem of	Inefficient and disjointed management of condominium properties and their facilities.
affects	Condo management companies, condo owners and rental users.
the impact of which is	A lack of streamlined communication, difficulty in managing financial and operational aspects of condo management, and challenges in accessing and sharing relevant information and resources.
a successful solution would be	A comprehensive, user-friendly condo management system that offers features such as easy profile creation, a central dashboard for condo owners, efficient document management, detailed unit and financial information, a reservation system for common facilities, role-based access for employees, and interactive features like forums and event organization.

1.2.2 Product Position Statement

For	condo owners, renters and management companies
Who	want a comprehensive user-friendly platform that facilitates the communication, collaboration of all parties and management of a condominium
The Condo Management System	is a software product.

That	connects condo owners, renters and various management companies alike
Unlike	AppFolio and Buildium
Our product	helps condo owners, renters and management companies alike by bridging the gap between both condo owners by connecting them through an integrated platform. In turn, enabling access key financial, management, and property information in a centralized location in order to facilitate the management of various individual condominium properties.

1.3 Stakeholder and User Descriptions

1.3.1 Stakeholder Summary

Name	Description	Responsibilities
Legal Advisors	Regulatory body responsible for enforcing regulations and standards related to property management, data privacy, and digital platforms.	● Ensure compliance with regulatory requirements and standards ● Provide guidance on legal and regulatory matters related to the app's development and operation
Investors	Entities providing financial support or investment for the development of the condo management system.	● Provide funding or investment for the project ● Ensure the availability of financial resources for the development and implementation of the system
IT Department	Internal department or external service provider responsible for managing the technical infrastructure supporting the condo management system.	● Manage and maintain technical infrastructure supporting the app ● Ensure system stability, security, and performance ● Provide technical support and expertise as needed

Maintenance Contractors	Contractors or service providers responsible for carrying out maintenance and repair work on condominium properties.	● Participate in system testing and provide feedback on maintenance-related functionalities ● Integrate maintenance-related processes with the app's functionality as necessary ● Ensure effective communication and collaboration with other stakeholders

1.3.2 User Summary

Name	Description	Responsibilities	Stakeholder
Public User	These are users not yet registered through a registration key generated by a condo management company	● Can create a personal user profile. ● Can enter a registration key to become a condo owner or renter user.	Themselves
Condo Management Company	Primary stakeholder and client of the system. Provided the requirements.	● Can generate reports ● Can generate registration keys ● Can manage properties ● Can manage employees ● Can delegate requests submitted by condo owners and condo renters to the appropriate employee(s).	Themselves
Condo Owner	Public users who have registered via a registration key. They own a condo unit in one of the properties managed by the condo management company.	● Can view generated condo fee reports ● Can view documents sent to me by management ● Can generate requests for management. ● Can view the status of generated requests. ● Can reserve facilities ● Can view the facilities' calendar	Themselves

Condo Renter	Public users who have registered via a registration key. They rent a condo unit from one of the condo owner users.	● Can view documents sent to me by management ● Can generate requests for management. ● Can view the status of generated requests. ● Can reserve facilities ● Can view the facilities' calendar	Themselves
Employee	Employee with a specific role (finance, maintenance, etc.) that works for a specific condo property. Managed by the condo management company	● Can view and manage requests delegated to them by condo management	Condo Management Company

1.3.3 User Environment

- Number of people involved in completing the task for a user to become a member of the property?
    - Two, the public user who is a condo owner or a condo renter and the condo management system who is responsible for the property.
- Is this changing?
    - For a condo management company, we would expect the number not to change because public users can only be approved by the condo management company and the condo management company will only receive access requests from the public users.
- How long is a task cycle?
    - The approval process time may vary depending on the volume of requests. It could be between 3 to 10 business days, or a couple of weeks depending on the condo.
- Amount of time spent in each activity?
    - Requesting access to the full interface of the condo management application that includes all the features for the condo owners and renters could take between a few minutes to an hour. After, it would take three to seven business days for the management company to receive and evaluate the request. The company would then have to prepare the necessary documents and give proper access codes to the user.
- Is this changing?

    - It could change because every company has a different way of approving and giving access to its users.

- Any unique environmental constraints?
    - The user would require internet access and a computing device.

- Which system platforms are in use today?
    - Multiple web browsers such as Safari and Chrome. Compatible with mobile phones. The user would also be required to have either one of the following platforms: Android, iOS, Windows, and MacOS.

- What other applications are in use?

    - Butler will utilize Google for sign-up and log-in operations.

1.3.4 Key Stakeholder or User Needs

Need	Priority	Concerns	Current Solution	Proposed Solutions
Broadcast messages	High priority	Difficulty in disseminating important information to all users and stakeholders efficiently.	Manual methods such as email announcements or physical notices.	Implement a notification/broadcasting system within the app for real-time updates and announcements, ensuring timely and widespread communication.
Property services	High priority	Difficulty in providing timely services.	Residents request services either in person or by telephone.	Implement a property service appointment booking system or a scheduling platform to facilitate the process of requesting a property service and serving it.
Effective and fair financial systems	High priority	- Concerns may arise about the fairness of service costs. - Stakeholders are concerned regarding the accessibility of financial	- Fixed prices are set for each service provided regardless of the effort required and unit space. - Financial information is communicated	- Implement a system to calculate a service cost based on certain user inputs. - Provide an accessible and user-friendly financial system within the app that automates all

		information, making it challenging for condo owners and renters to understand and monitor their financial contributions and expenses.	through monthly physical statements or emails and condo owners and renters may find it challenging to understand the details and keep track of them.	service fees and general costs for condo owners and renters.
User account and condo security	High priority	Condo owners and renters fear personal data leak.	This problem is currently solved by a traditional way of authentication (login username and password).	Provide one-time private tokens to users in addition to their login username and password, which may increase the level of security within the condo management app.

1.3.5 Alternatives and Competition

Competitors	**Strengths**	**Weaknesses**
AppFolio	● User-friendly interface. ● Robust financial management features. ● Good customer support.	● Not suitable for smaller-scale condo property management companies. ● Some users report occasional glitches in the system.
Condo Manager	● Specialized in condo and homeowner association management. ● Offers features like accounting, communication, and document storage	● User interface might not be as modern or intuitive as some competitors. ● Limited mobile app capabilities.

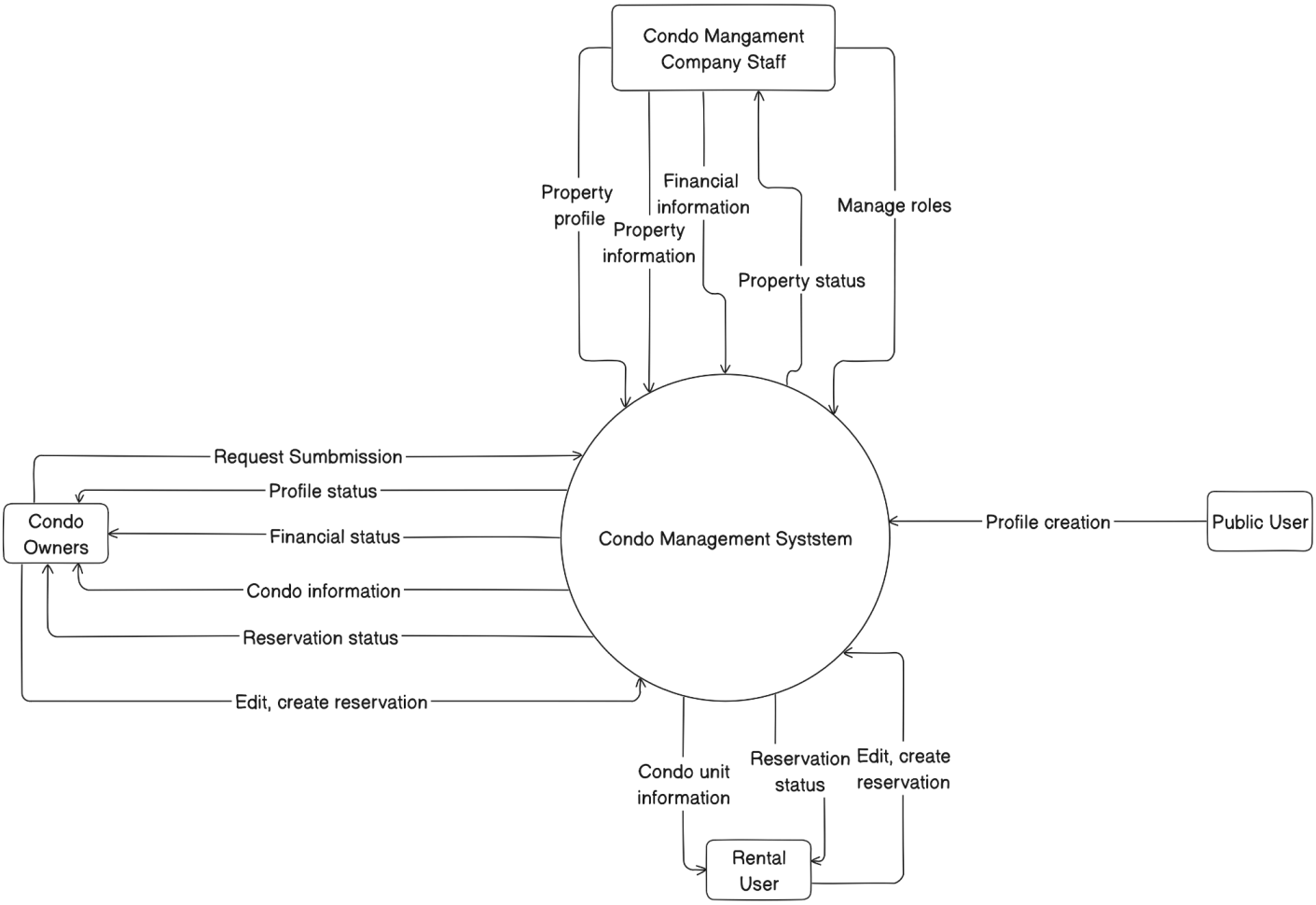
Yardi	● Extensive experience in property management solutions. ● Offers a wide range of functionalities.	● Can be expensive for smaller associations. ● Some users report a learning curve for more advanced features.
Buildium	● Well-established and widely used in the property management industry. ● Comprehensive features, including financial management, communication, and maintenance tracking. ● Good customer support	● Some users may find it complex for smaller-scale condo management needs. ● Pricing might be on the higher side for smaller associations.

1.4 Product Overview

1.4.1 Product Perspective

The Condo Management System is designed to be a comprehensive and self-contained product, but it also functions as a component within a broader ecosystem of property management and residential living. It interfaces with various external entities like condo management company staff, condo owners, rental users. The system may be viewed as follows.

Context Diagram



Context Diagram for the Condo Management System

1.4.2 Assumptions and Dependencies

Features	Assumptions	Factors
Profile Creation	This feature assumes that the condo management companies have a secure way of distributing registration keys.	User privacy laws and data protection regulations.
Owner’s Dashboard	Secure and reliable internet connection.	Integration capabilities with other systems (financial system, reservation system)

Property's Profile Creation	The management company owns the property.	Digital literacy of management staff and data storage infrastructure.
Simplified Financial System	We assume the condo management companies input accurate financial data.	Integration with banking systems.
Simplified Reservation System	Real-time scheduling system that can interface with the system's database.	
Role Distribution	A clear definition of roles and permissions.	
Request Submissions	An efficient ticketing system is available.	Staff training on the system. Response time.
Notification Page	Ability to update users about their requests in real time.	

1.5 Product Features

1.5.1 Profile Creation

Public users can create their unique profile in the system. A public user shall provide a registration key from their condo management company to become either a condo owner or a rental user.

1.5.2 Owner's Dashboard

Condo owners shall have a simplified view of their properties. Information like personal profile, condo information, status of submitted requests, as well as their financial status shall be displayed in the dashboard. The system shall restrict the accessibility of this information only to the condo owner.

1.5.3 Property's Profile Creation

The System allows management companies to create detailed profiles for properties they manage. It facilitates the uploading of condo files accessible to all owners. The system also enables the input of detailed information for each condo unit, parking spot, and locker.

Additionally, management companies can issue registration keys to unit owners or rental users, enabling the seamless association of units with their respective profiles. This feature set ensures efficient property management and communication within the condominium community.

1.5.4 Simplified Financial System

The System features a simplified financial system, enabling management companies to input condo fees and costs efficiently. Condo fees are automatically calculated and presented to unit owners. The system records operational budgets, including collected fees, and allows input of operation costs. Additionally, it generates annual reports ensuring transparent financial management within the condominium community

1.5.5 Simplified Reservation System

The System offers common facility reservations, allowing users to book amenities. The process, done through a calendar-like interface, operates on a first-come-first-serve basis. Once booked, a facility becomes temporarily unavailable for the reserved time, enhancing accessibility for condo owners and rental users.

1.5.6 Role Distribution

Condo management companies can assign roles to employees for efficient task management within a property.

1.5.7 Requests Submissions

Condo owners can request services like moving, intercom changes, or report issues. Requests are categorized and assigned to the relevant employee for efficient resolution.

1.5.8 Notification Page

Users have a notification page for real-time updates on their submitted or assigned requests, ensuring timely information about activities within the community.

1.5.9 Extra Features

The System enhances community interaction with a user forum and event organization. Management companies have the option to offer discounts that are visible to all unit owners or rental users of one property.

1.6 Other Product Requirements

The condo management app and its accompanying website must adhere to industry standards and best practices to ensure a seamless user experience. Platform compatibility is crucial, requiring the application to function smoothly across major web browsers like Chrome, Firefox, Safari, and Edge, as well as on popular mobile platforms such as iOS or Android. Compliance with data security standards, including GDPR and applicable regulations, is imperative to safeguard user information.

In terms of performance requirements, the system must exhibit swift response times, with pages and features loading within 2-3 seconds. Scalability is a key consideration, necessitating the ability to handle an increasing number of users and properties without compromising performance. A high level of reliability is expected, with a minimum uptime of 99.9% to ensure uninterrupted access for users.

Environmental requirements focus on energy efficiency, urging the design to minimize energy consumption, particularly on mobile devices. Quality ranges for the system to emphasize high performance, robustness against errors, fault tolerance mechanisms for recovery, and an intuitive user interface to enhance usability.

Design constraints involve strict adherence to relevant regulations and legal requirements, ensuring the system's compliance with established standards. Additionally, technology stack considerations serve as constraints, dictating the selection and usage of specific technologies and frameworks during development.

External constraints and dependencies include reliance on third-party APIs for functionalities such as payment processing and mapping services. Stable internet connectivity is essential for users to access the application seamlessly, marking an external dependency.

Documentation requirements encompass the creation of comprehensive user manuals, online help resources within the application, clear installation instructions for various devices, and guidelines for labeling and packaging if applicable. Prioritization should focus on stability, with a high priority assigned to ensuring the system operates reliably and efficiently. Additionally, the benefits derived from the features, efforts required for implementation and maintenance, and risk mitigation strategies should be considered to establish a well-balanced priority framework.

Chapter 2: Requirements and User Stories Backlog

2.1 Requirements

ID	Type	Description	User Story
RPU1	Must	Public users can create their own (unique) profile.	PU_2
RPU2	Must	A profile should at least include a profile picture, user name, contact email, phone number.	PU_6
RPU3	Must	Public users need to provide a registration key obtained from their condo management company to become condo owners in the system.	PU_4
RPU4	Must	Public users need to provide a registration key obtained from their condo management company to become rental users in the system.	PU_5
RCO1	Must	Condo owners can have a good view (dashboard) of their properties, including general information, e.g., personal profile, condo information, financial status (i.e., remaining balance in terms of monthly condo fee payments), status of the submitted request, etc.	CO_2
RCM1	Must	Condo management companies can create a profile for a property under their management	CMC_4
RCM2	Must	The property profile should have at least property name, unit count, parking count, locker count, address.	CMC_32
RCM3	Must	Condo management companies can upload condo files for each property. The condo files are accessible to all condo owners of that property. Examples of such condo files (in pdf) include condo declarations, annual budgets, board meeting minutes etc.	CMC_8
RCM4	Must	Condo management companies can enter detailed information for each condo unit, each parking spot, and each locker in a building. Basic information of a condo unit includes unit id, size, unit owner, occupant information (i.e., may be occupied by a rental user instead of the owner), as well condo fee associated with the unit. Basic information of a parking spot (or a locker) includes parking spot id, spot owner, occupant information, and condo fee associated with the parking spot.	CMC_9/ CMC_13/ CMC_27
RCM5	Must	Condo management companies can send registration keys to unit owners or rental users for their dedicated units. Such registration key will be used by unit owners or rental users to link a condo unit with their profiles.	CMC_10
RCM6	Must	Condo management companies can enter condo fee per square foot, per parking spot.	CMC_12
RMS1	Must	Condo fee of each unit will be calculated and presented to unit owners.	CMC_14/ CMC_15
RMS2	Must	The financial system records operational budget (i.e., the collected condo fee) and cost.	CMC_16

RCM7	Must	Condo management companies can enter the cost for each operation.	CMC_17
RMS2	Must	The financial system can generate an annual report. For example, all the condo fee collected for a given year.	CMC_20
RCM8	Must	Condo management companies set up the common facilities, which require reservations. Examples include a sky lounge, a spa fitness.	CMC_21
RCO2/ RCR1	Must	The reservation system allows condo owners and rental users to reserve common facilities in a calendar-like interface.	CO_10/ CR_10
RMS5	Must	The reservation system should show availability of common facilities.	CO_9/ CR_9/ CMC_22
RMS6	Must	The reservation is first-come-first-serve. Once a facility is booked, it will become unavailable for the reserved time.	CMC_23
RCM9	Must	Condo management companies can set up different roles for different employees, who are responsible for the same property. Such roles include manager, or someone who is responsible for daily operations, or someone who is responsible for finance.	CMC_28
RCO3	Must	Condo owners can submit requests.	CO_11
RCO4	Must	Examples of requests include 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.	CO_11
RMS8	Must	Each request will be assigned to a corresponding employee based on the type of the request.	CMC_25
RPU5/ RCO5/ RCR2	Must	All the users have a notification page, where they can see the latest activities in submitted or assigned requests.	CMC_26/ CO_12
RMS7	Should	A forum where users can post and reply.	CO_13
RMS8	Should	Events. Users can organize events and invite other occupants to attend.	CO_14
RMS9	Should	Discounts, offers. Condo management companies can choose to list coupons/offers that are visible to all unit owners or rental users of one property.	CMC_33
RMS10	Could	The app is accessible on Android, iOS, Linux, MacOS, and Windows.	AF_1
RMS11	Could	The app is available in English and, at least, one other language.	AF_2
RMS12	Must	The app must allow users to login using their Gmail account or other such account (Single Sign On), possibly offering multiple possibilities.	AF_3

2.2 User Stories

Template:

ID	**<TITLE>**		
As <user> I want <what> because <value>			
Feature\Epic name\			
MoSCoW	Business Value	Risk	Effort

2.2.1 Public User User Stories

PU_1	**Public User Login**		
As a public user, I want to be able to log in to the app, so that I can view my personal profile.			
Login Feature			
Must	8	Medium	5

PU_2	**Create Public User Profile**		
As a public user, I want to be able to create a unique profile, so that management knows how to reach me.			
User Profile Feature			
Must	5	Low	3

PU_3	Update Public User Profile		
As a public user, I want to be able to update my unique profile, so that I can keep the information up to date.			
User Profile Feature			
Must	5	Low	3

PU_4	Register Condo Owners		
As a public user, I want to be able to obtain a registration key, so that I can view the features reserved for Condo Owners.			
Registration Feature			
Must	8	Medium	5

PU_5	Register Condo Renters		
As a public user, I want to be able to obtain a registration key, so that I can view the features reserved for Condo Renters.			
Registration Feature			
Must	8	Medium	5

PU_6	Public User Profile Content		
As a public user, I want to be able to add a profile picture, username, contact email, and phone number to my profile, so that this information is accessible to my condo management company.			
User Profile Feature			
Must	8	Medium	3

2.2.2 Condo Owner User Stories

CO_1	Condo Owner Login		
As a condo owner, I want to login as a condo owner, so that I can access information reserved for condo owners.			
Login Feature			
Must	8	Medium	5

CO_2	View Condo Owner Dashboard		
As a condo owner, I want to view the condo owner dashboard, so that important information about my properties is easily accessible.			
Condo Owner Dashboard Feature			
Must	5	Low	5

CO_3	Create Condo Owner Profile		
As a condo owner, I want to create my personal profile on the dashboard, so that management knows how to reach me.			
User Profile Feature			
Must	5	Low	3

CO_4	Update Condo Owner Profile		
As a condo owner, I want to be able to update my unique profile, so that I can keep the information up to date.			
User Profile Feature			
Must	5	Low	3

CO_5	Display General Information on Dashboard		
As a condo owner, I want to be able to view general information on the dashboard, so that I can stay up to date on important details.			
Condo Owner Dashboard Feature			
Must	3	Low	5

CO_6	Display Condo Information on Dashboard		
As a condo owner, I want to be able to view information about my properties on the dashboard, so that I can make sure the information is correct.			
Condo Owner Dashboard Feature			
Must	3	Low	3

CO_7	Display Financial Information on Dashboard		
As a condo owner, I want to be able to view financial information about my properties on the dashboard, so that I keep track of payments due.			
Condo Owner Dashboard Feature			
Must	8	Medium	5

CO_8	Display Request Status		
As a condo owner, I want to be able to view the status of any requests I made on the dashboard, so that I keep track of these requests.			
Request Feature			
Must	5	Low	5

CO_9	Read Reservation Availability		
As a condo owner, I want to view the reservation system calendar, so that I can see what facilities are available or not.			
Facility Reservation System Feature			
Must	8	Low	3

CO_10	Reserve Facility		
As a condo owner, I want to be able to reserve available facilities using the reservation system calendar, so that I can plan activities involving these facilities.			
Facility Reservation System Feature			
Must	8	Medium	5

CO_11	Submit Requests		
As a condo owner, I want to be able to submit requests to management, so that I can inform them of any issues I might have related to the condo property.			
Request Feature			
Must	13	Medium	5

CO_12	Read Notification Page		
As a condo owner, I want to be able to view the status of any of my submitted requests on my notification page, so that I can keep track of these requests.			
Request Feature			
Must	8	Low	3

*This is a duplicate of CO_8, but this is reflected in the requirements. Clarification is required.

2.2.3 Condo Management Company User Stories

CMC_1	Create Condo Management Login		
As a condo management company, I want to login as a condo management company, so that I can access property information.			
Condo Management Property Profile Feature			
Must	13	Medium	3

CMC_2	Create Condo Management Profile		
As a condo management company, I want to create my company profile on the dashboard, so that I can keep the information up to date.			
Condo Management Property Profile Feature			
Must	13	Low	3

CMC_3	Update Condo Management Profile		
As a condo management company, I want to be able to update my company profile, so that I can keep the information up to date.			
Condo Management Property Profile Feature			
Must	13	Low	3

CMC_4	Create Property Profile		
As a condo management company, I want to be able to create a property profile on the app, so that I can keep track of these properties.			
Condo Management Property Profile Feature			
Must	13	Low	8

CMC_5	Read Property Profile		
As a condo management company, I want to be able to view property profiles that I have created, so that I can review these profiles.			
Condo Management Property Profile Feature			
Must	13	Low	3

CMC_6	Update Property Profile		
As a condo management company, I want to be able to update property profiles that I have created, so that I can keep the information up to date.			
Condo Management Property Profile Feature			
Must	13	Low	3

CMC_7	Delete Property Profile		
As a condo management company, I want to be able to delete property profiles that I have created, because I might sell a property.			
Condo Management Property Profile Feature			
Must	13	Medium	3

CMC_8	Upload Condo Files to Property		
As a condo management company, I want to be able to upload condo files to properties I manage, so that this information can be shared to all relevant condo owners.			
Condo Management Property Profile Feature			
Must	13	Medium	5

CMC_9	Enter Detailed Condo Unit Information		
As a condo management company, I want to be able to enter detailed information about each condo unit in a property, so that management can keep track of this important information.			
Condo Management Property Profile Feature			
Must	13	Low	5

CMC_10	Send Registration Keys		
As a condo management company, I want to be able to send registration keys to condo owners and condo renters, so that they may register to the condo management app.			
Registration Feature			
Must	13	High	5

CMC_11	Revoke Registration Keys		
As a condo management company, I want to be able to revoke registration keys from condo owners and condo renters if ever they sell/move, so that only residents of the condo units get access to the app.			
Registration Feature			
Must	13	High	5

CMC_12	Enter Condo Fee Per Square Foot		
As a condo management company, I want to be able to enter condo fee per square foot, so that this information can be given to condo owners.			
Condo Management Financial System Feature			
Must	13	Low	3

CMC_13	Enter Detailed Locker Information		
As a condo management company, I want to be able to enter detailed information about each locker in a property, so that management can keep track of this important information.			
Condo Management Property Profile Feature			
Must	13	Low	5

CMC_14	Calculate Total Condo Fees		
As a condo management company, I want to view the total condo fees per condo unit, so that this information can be given to condo owners.			
Condo Management Financial System Feature			
Must	13	Low	3

CMC_15	Send Total Condo Fees to Condo Owners		
As a condo management company, I want to send the total condo fees due to condo owners, so that they are made aware of what they owe.			
Condo Management Financial System Feature			
Must	13	Medium	3

CMC_16	Record Operational Budget		
As a condo management company, I want to see the amount of condo fees each property has generated for the year, so that management can know the operational budget.			
Condo Management Financial System Feature			
Must	13	Medium	5

CMC_17	Record Operational Costs		
As a condo management company, I want to be able to enter the cost of each operation, so that management can keep track of the total costs for the year.			
Condo Management Financial System Feature			
Must	13	Low	5

CMC_18	Read Operational Costs		
As a condo management company, I want to view the operation costs for the year so far, so that management can keep track of the total costs for the year.			
Condo Management Financial System Feature			
Must	13	Low	3

CMC_19	Update Operational Costs		
As a condo management company, I want to be able to update the costs for each operation, so that management can keep track of the total costs for the year.			
Condo Management Financial System Feature			
Must	13	Medium	3

CMC_20	Generate Annual Report		
As a condo management company, I want to see an annual report for each property, so that management can see the condo fees collected for a given year.			
Condo Management Financial System Feature			
Must	13	Low	5

CMC_21	Set up Reservation System		
As a condo management company, I want to add the facilities available to reserve into the system, so that condo owners/renters can view what facilities can be reserved.			
Facility Reservation System Feature			
Must	8	Medium	8

CMC_22	Read Facility Availabilities		
As a condo management company, I want to see the availabilities for each facility up for reservation, so that management can keep track of reservations.			
Facility Reservation System Feature			
Must	8	Low	5

CMC_23	Block Reserved Facilities		
As a condo management company, I want to be able to block off a facility in the reservation system, in case maintenance needs to be done.			
Facility Reservation System Feature			
Must	8	Low	3

CMC_24	Create Employee		
As a condo management company, I want to add employees, so that I can manage their roles and assign them requests.			
Employee Management Feature			
Must	8	Low	3

CMC_25	Assign Requests to Employees		
As a condo management company, I want to be able to assign requests from condo owners/renters to appropriate employees, so that the right employees can resolve these requests			
Request Feature			
Must	8	Low	3

CMC_26	Provide a Notification Page		
As a condo management company, I want to post notifications so condo owners and renters can see the latest activities in submitted or assigned requests.			
Notification Page Feature			
Must	8	Low	3

CMC_27	Enter Detailed Parking Spot Information		
As a condo management company, I want to be able to enter detailed information about each parking spot in a property, so that management can keep track of this important information.			
Condo Management Property Profile Feature			
Must	13	Low	5

CMC_28	Assign Employee Roles		
As a condo management company, I want to assign different employee roles, so that condo owners/renters know who is responsible for what.			
Employee Management Feature			
Must	8	Low	3

CMC_29	Read Employee		
As a condo management company, I want to see the employees that I have created, so that I can see if their information is correct.			
Employee Management Feature			
Must	8	Low	3

CMC_30	Update Employee		
As a condo management company, I want to update employee information, so that I can make sure their information is up to date.			
Employee Management Feature			
Must	8	Low	3

CMC_31	Delete Employee		
As a condo management company, I want to delete employees that I have added, so that the information reflects any changes in our team.			
Employee Management Feature			
Must	8	Low	3

CMC_32	Property Profile Content		
As a condo management company, I want my property profile to contain property name, unit count, parking count, locker count, and address, so that I can keep track of these properties.			
Condo Management Property Profile Feature			
Must	13	Low	8

2.2.4 Additional Features User Stories

AF_1	**Responsive App**		
As any user, I want to be able to access the app on Android, iOS, Linux, MacOS, and Windows, so that I can access this important information conveniently from anywhere.			
Responsive Feature			
Would	13	High	8

AF_2	**Multilingual App**		
As any user, I want the app to be available in more than one language, so that more people can experience the app in their native language.			
Responsive Feature			
Would	13	Medium	8

AF_3	**Single Sign On**		
As any user, I want to be able to login to the app using Gmail or any other popular account, so that logging in becomes more convenient			
Responsive Feature			
Must	5	High	8

2.2.5 Optional Features User Stories

CO_13	Create User Forum		
As a condo owner, I want to be able to interact with other users in a forum setting, so that I can get to know my neighbors.			
User Forum Feature			
Should	5	High	8

CO_14	Organize Events		
As a condo owner, I want to organize events and invite other occupants to attend, so that I can get to know my neighbors.			
User Events Feature			
Should	5	Medium	8

CMC_33	List Coupons/Offers		
As a condo management company, I want to list coupons/offers to all users or specific users, to encourage users to try new things.			
Coupon/Offers Feature			
Should	8	Low	5

2.2.6 Renter User Stories

CR_1	Condo Renter Login		
As a condo renter, I want to login as a condo renter, so that I can access information reserved for condo renters.			
Login Feature			
Must	8	Medium	5

CR_2	View Condo Owner Dashboard		
As a condo renter, I want to view the condo renter dashboard, so that important information about my condo is easily accessible.			
Condo Owner Dashboard Feature			
Must	5	Low	3

CR_3	Create Condo Owner Profile		
As a condo renter, I want to create my personal profile on the dashboard, so that management knows how to reach me.			
User Profile Feature			
Must	5	Low	3

CR_4	Update Condo Owner Profile		
As a condo renter, I want to be able to update my unique profile, so that I can keep the information up to date.			
User Profile Feature			
Must	5	Low	3

CR_5	Display General Information on Dashboard		
As a condo renter, I want to be able to view general information on the dashboard, so that I can stay up to date on important details.			
Condo Owner Dashboard Feature			
Must	3	Low	3

CR_6	Display Condo Information on Dashboard		
As a condo renter, I want to be able to view information about my condo on the dashboard, so that I can make sure the information is correct.			
Condo Owner Dashboard Feature			
Must	3	Low	3

CR_7	Display Financial Information on Dashboard		
As a condo renter, I want to be able to view financial information about my condo on the dashboard, so that I keep track of payments due.			
Condo Owner Dashboard Feature			
Must	8	Medium	5

CR_9	Read Reservation Availability		
As a condo renter, I want to view the reservation system calendar, so that I can see what facilities are available or not.			
Facility Reservation System Feature			
Must	8	Low	3

CR_10	Reserve Facility		
As a condo renter, I want to be able to reserve available facilities using the reservation system calendar, so that I can plan activities involving these facilities.			
Facility Reservation System Feature			
Must	8	Medium	5

2.3 Use Cases

Created as issues on GitHub to help with task tracking, link to them here.

Chapter 3: Software Architecture Document

3.1 Introduction

This chapter describes introductory information items of the AD, including identifying and supplementary information.

3.1.1 Identifying information

3.1.1.1 Architecture Name

Condo Management System (CMS) - Monolithic Architecture

3.1.1.2 System of Interest

The Condo Management System is designed as a centralized platform to manage condominium properties, facilitating interactions between condo management companies, owners, and tenants.

3.1.2 Supplementary Information

3.1.2.1 Date of Issue

Tuesday 27th of February

3.1.2.2 Status

Second Draft

3.1.2.3 Summary

This document provides a detailed description of the Condo Management System, employing a monolithic architecture approach to cater to various stakeholders involved in condo management.

3.1.2.4 Scope

This document covers the architectural design of the Condo Management System, including its features, functionalities and the rationale behind the chosen architecture.

3.1.2.5 Context

The Condo Management System is conceptualized to operate in a diverse technological environment, supporting various platforms including Android, iOS, MacOS, and Windows.

3.1.3 Other Information

3.1.3.1 Overview

The Condo Management System is designed to be a robust, all-encompassing platform addressing the unique needs of condo management. It allows for the creation of user profiles by public users, property profile management by condo management companies, financial tracking, a reservation system for common facilities, and the handling of various requests and notifications.

3.1.3.2 Architecture Evaluations

The Condo Management System is developed with a Monolithic Architecture framework. This section evaluates this decision to adopt this architecture, focusing on its alignment with the Condo Management System functional requirements, operational environment, and development context.

- **Simplicity in Design and Implementation**

  For the Condo Management System, a Monolithic architecture offers a straightforward, cohesive design, making implementation simpler compared to more complex architectures at the level at which we are operating. This simplicity is vital in ensuring a quick turnaround time for development and deployment.

- **Ease of Deployment**

  The Condo Management System benefits from the monolithic model's ease of deployment. Given That the system is a unified instance, deployment complexities are significantly reduced, which is crucial for ensuring reliable and consistent system availability.

- **Single Development Context**

  A monolithic approach allows the development team to work within a single, unified developmental environment. This facilitates better coordination and understanding across different system components, enhancing overall development efficiency.

- **Consistency and Performance**

  The Condo Management System, being a centralized platform for managing condominium properties, benefits from the consistent performance offered by a monolithic architecture, where all components run in a single process.

3.1.3.3 Rationale for Key Decisions

This section outlines and provides the rationale for key decisions made in the architectural design of the Condo Management System. These decisions are instrumental in shaping the architecture, functionality, and overall user experience of the system.

Decision 1: Adoption of Monolithic Architecture	
Decision	The Condo Management System will be developed using a Monolithic Architecture approach.
Rationale	This decision was driven by the need for a simplified and cohesive development process, especially critical in the early stages of the CMS. The Monolithic Architecture offers ease of deployment, testing, and maintenance, which is essential for a rapid development cycle and streamlined initial rollout. Additionally, considering the scope and scale of the system at inception, a monolithic model is more cost-effective and easier to manage.
Impact	This approach simplifies development and deployment processes. It also ensures that the system's performance and reliability are maintained in a controlled, single-process environment.

Decision 2: Centralized Financial Management System

Decision	Integration of a centralized financial management system within the Condo Management System.
Rationale	The inclusion of an integrated financial system is to ensure seamless management of condo fees, operational budgets, and financial reporting. This centralized approach enhances the efficiency of financial transactions and record-keeping for both condo management companies and owners.
Impact	This provides a one-stop solution for financial management within the Condo Management System, greatly enhancing user experience for condo management companies by streamlining financial processes.

Decision 3: Comprehensive User Profile and Property Management	
Decision	Implementation of detailed user profiles and property management features.
Rationale	To address the diverse needs of the system's stakeholders, including public users, condo owners, and condo management companies. The system provides detailed profiling and property management capabilities to ensure a tailored and efficient user experience.
Impact	Enhances user engagement and satisfaction by providing users with comprehensive control and visibility over their profiles and properties.

Decision 4: Integration of Reservation System for Common Facilities	
Decision	The Condo Management System will include a reservation system for common facilities.
Rationale	To add value to the user experience by providing a convenient, accessible means for condo owners and rental users to reserve common facilities. This system addressed a key need for efficient management of shared resources in condominium complexes.
Impact	This feature is expected to streamline the process of facility management.

Decision 5: Role-Based Access for Condo Management Companies	
Decision	Implementation of a role-based access control system for different

	employees of condo management companies.
Rationale	To provide a secure and organized framework for managing various operational aspects of condominium management. This decision ensures that employees have access only to the relevant sections of the Condo Management System, enhancing both security and operational efficiency.
Impact	Improves system security and operational effectiveness by ensuring that employees can access and manage information pertinent to their roles.

3.2 Stakeholders And Concerns

This chapter contains information items for stakeholders of the architecture, the stakeholders' concerns for that architecture, and the traceability of concerns to stakeholders.

3.2.1 Stakeholders

The stakeholders for the architecture are defined as follow:

3.2.1.1 Condo Owners

Condo owners are individuals who own the condo units. They are responsible for maintaining the property, complying with property rules and paying condo fees.

3.2.1.2 Rental Users

Individuals who are renting the condo units. They adhere to rental agreements and regulations.

3.2.1.3 Condo Management Company

Responsible for overall management of the condo complex. They handle the complex maintenance, finance management, and rule enforcement.

3.2.1.4 Developers

Group in charge of constructing and deploying the system. They develop, test, and deploy the software.

3.2.1.5 Software Architects

Responsible for the system's overall design and architecture. They ensure the software is secure, maintainable, and scalable.

3.2.2 Concerns

Each stakeholder has their concerns regarding the system architecture. The concerns are the following:

- Need for clear visibility of their property's status.
- Need for financial transparency.
- Efficient management of properties.
- Streamline administrative tasks.
- Secure role management.
- Ease of development, deployment, and maintainability.
- Security integrity.
- System's longevity.

3.2.3 Concern - Stakeholder Traceability

	Condo Owners	Rent Users	Condo Management Company	Developers	Software Architects
Clear visibility of property's status	X	X	X		
financial transparency	X	X	X		
Efficient management of properties	X		X		
Streamline administrative tasks			X		
Secure role management			X		

Ease of development, deployment, and maintainability			X	X	X
Security integrity			X		X
System’s longevity			X		X

3.3 Viewpoints+

3.3.1 Functional Viewpoint

3.3.1.1 Rationale

The functional viewpoint provides the stakeholders with an overview of the system’s features which helps them in understanding about the behavior of the application. For example, this viewpoint forces the system's functionality to be aligned with the need for efficient tools for managing properties that the condo owners would want to use.

3.3.1.2 Overview

The Functional Viewpoint abstracts the system’s behavior regarding scenarios, use cases, and functional requirements. It describes the system's features and the connections between its different components.

3.3.1.3 Concers and Stakeholders

Concerns

1. **Functionality:** How well does the system provide clear visibility of a property's status?
2. **Usability:** Does the system ensure financial transparency and efficient property management for condo owners and rent users?
3. **Interoperability:** How does the system streamline administrative tasks for the condo management company?

Typical Stakeholders

1. Condo Owners
2. Rent Users
3. Condo Management Company

3.3.2 Structural Viewpoint

3.3.2.1 Rationale

The Condo Management System is likely to evolve and handle increased loads over time. The structural viewpoint helps ensure that the system is scalable and easily maintainable.

3.3.2.2 Overview

The Structural Viewpoint gives a static representation of the system's architecture. It provides how different elements such as classes, modules, and components are organized and connected. This viewpoint handles the organization and structure of the system's components and their interrelationships.

3.3.2.3 Concerns and Stakeholders

Concerns

1. **Component Structure:** How is the system structured to ensure financial transparency and efficient property management?
2. **System's Longevity (Scalability):** Is the system designed to scale for potential growth in property management needs?
3. **Maintainability:** How easy is it to develop, deploy, and maintain the system?

Typical Stakeholders

1. Condo Management Company
2. Condo Owners
3. Rent Users
4. Software Architects
5. Developers

3.3.3 Behavioral Viewpoint

3.3.3.1 Rationale

Administrative tasks require dynamic behavior. This viewpoint ensures that the system responds effectively to events, providing a user-friendly experience for the condo management company.

3.3.3.2 Overview

The Behavioral Viewpoint gives a dynamic representation of how the systems react to different triggers. It focuses on describing the dynamic behavior of the system in terms of state transitions, interactions, and event-driven activities.

3.3.3.3 Concerns and Stakeholders

Concerns

1. **Event-Driven Processes:** How does the system respond to events related to property management concerns?

Typical Stakeholders

1. Condo Owners
2. Condo Management Company

3.3.4 Information Viewpoint

3.3.4.1 Rationale

Data integrity is crucial for security and overall system integrity. The information viewpoint ensures that data is accurately stored, retrieved, and protected, addressing concerns related to security and system integrity.

3.3.4.2 Overview

The information Viewpoint handles how data is input, processed, output, and stored within the system. This viewpoint addresses representations of data flow, storage, and processing.

3.3.4.3 Concerns and Stakeholders

Concerns

1. **Data Flow:** How is data flowing to ensure financial transparency and clear property status visibility?
2. **Storage and Retrieval:** How is data stored and retrieved for efficient property management?
3. **Data Integrity:** How is data integrity maintained for security and system integrity concerns?

Typical Stakeholders

1. Condo Owners
2. Rent Users
3. Condo Management Company
4. Software Architects

3.3.5 Usability Viewpoint

3.3.5.1 Rationale

A user-friendly interface is necessary for the daily users of the system. This viewpoint makes sure that the system is designed in a way that makes it intuitive, interactive, and easy to navigate. This is necessary for a clear visibility of the property's status which some of the stakeholders are concerned about.

3.3.5.2 Overview

The Usability Viewpoint is focused on the user interface and the overall user experience of the system. It considers aspects such as user interaction, accessibility, and user feedback.

3.3.5.3 Concerns and Stakeholders

Concerns

1. **User Interface Design:** How is the user interface designed for ease of property management tasks and for clear visibility of property's status?

Typical Stakeholders

1. Condo Management Company

2. Condo Owners
3. Rent Users

3.3.6 Security Viewpoint

3.3.6.1 Rationale

Secure role management is necessary for controlling access to sensitive information. The security viewpoint guarantees that access control measures are implemented to protect against unauthorized access. Additionally, Maintaining the system's integrity is crucial for building trust with the client. This viewpoint also makes sure that the system forbids unauthorized modifications.

3.3.6.2 Overview

The Security Viewpoint addresses the security matter of the system. It handles how features are implemented to protect against potential vulnerabilities. This viewpoint involves the identification of these threats, the establishment of security measures, and the implementation of countermeasures.

3.3.6.3 Concerns and Stakeholders

Concerns

1. **Access Control:** How is secure role management implemented to control access to sensitive information and system functionalities?
2. **System Integrity:** How is the system's integrity maintained to prevent unauthorized modifications?

Typical Stakeholders

1. Condo Management Company
2. Software Architects

3.4 Model Kinds

Under Functional Viewpoint:

- **Use Case Models**

Use Case Models conventions

Language: UML (Unified Modeling Language)

Notations: Use Case Diagrams

Modeling Techniques: Describing user interactions through actors and use cases

Analytical Methods: Use case analysis, scenario analysis

Use Case Models Correspondence Rules

Use case models should correspond to the functional requirements specified in the requirements documentation.

Each use case should correspond to one or more scenarios that illustrate the interaction between actors and the system.

Use case models should align with the system's business processes and goals as defined in the project scope.

- **Scenario Models**

  Scenario Models conventions

  Language: UML (Unified Modeling Language)

  Notations: Sequence Diagrams, Activity Diagrams

  Modeling Techniques: Describing the flow of events in a particular use case

  Analytical Methods: Identifying system behavior under different conditions

  Scenario Models Correspondence Rules

  Scenario models should correspond to specific use cases or user stories.

  Each scenario should illustrate a particular sequence of interactions between actors and the system to achieve a specific goal or outcome.

  Scenario models should accurately represent the various paths through which the system can respond to different inputs or events.

- **Functional Requirement Models**

Functional Requirement Models conventions

Language: Natural language

Notations: Use of formalized requirement statements

Modeling Techniques: Requirement elicitation, documentation

Analytical Methods: Analyzing requirements for completeness and clarity

Functional Requirement Models Correspondence Rules

Functional requirement models should correspond directly to the functional requirements documented in the requirements specification.

Each functional requirement should be represented clearly and unambiguously in the model.

Functional requirement models should align with the overall system architecture and design.

Under Structural Viewpoint:

- **Class Diagrams**

  Class Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: Class diagrams

  Modeling Techniques: Representing classes and their relationships

  Analytical Methods: Identifying potential structural issues

  Class Diagrams Correspondence Rules

  Class diagrams should correspond to the object-oriented design of the system.

  Each class in the diagram should correspond to a logical entity or component within the system.

  Class diagrams should accurately represent the relationships and associations between classes as defined in the system's design.

- **Module Diagrams**

  Module Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: Module diagrams

  Modeling Techniques: Illustrating the organization of system modules

  Analytical Methods: Assessing modularity and dependencies

  Module Diagrams Correspondence Rules

  Module diagrams should correspond to the modular structure of the system.

  Each module in the diagram should represent a cohesive unit of functionality or a logical grouping of related components.

  Module diagrams should accurately depict the dependencies and interactions between modules within the system.

- **Component Diagrams**

  Component Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: Component diagrams

  Modeling Techniques: Depicting components and their interactions

  Analytical Methods: Evaluating component interactions and dependencies

  Component Diagrams Correspondence Rules

  Component diagrams should correspond to the physical or runtime architecture of the system.

  Each component in the diagram should represent a deployable unit of the system, such as a module, library, or executable.

  Component diagrams should accurately represent the relationships and dependencies between components within the system.

Under Behavioral Viewpoint:

- **State Diagrams**

  State Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: State diagrams

  Modeling Techniques: Describing states and transitions in response to events

  Analytical Methods: Analyzing system behavior under different states

  State Diagrams Correspondence Rules

  State diagrams should correspond to the behavioral aspects of the system, particularly the lifecycle of objects or entities.

  Each state in the diagram should correspond to a distinct condition or mode of operation for an object or system component.

  State diagrams should accurately represent the transitions between states and the events that trigger those transitions.

- **Interaction Diagrams**

  Interaction Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: Sequence Diagrams, Communication Diagrams

  Modeling Techniques: Illustrating interactions between system components

  Analytical Methods: Examining system dynamics during interactions

  Interaction Diagrams Correspondence Rules

  Interaction diagrams should correspond to the dynamic behavior of the system during specific scenarios or use cases.

  Each interaction diagram should illustrate the sequence of messages exchanged between objects or components to accomplish a particular task or scenario.

  Interaction diagrams should accurately represent the flow of control and data between system elements.

Under Information Viewpoint:

- **Data Flow Diagrams**

  Data Flow Diagrams conventions

  Language: UML (Unified Modeling Language)

  Notations: Data Flow Diagrams

  Modeling Techniques: Representing data flow within the system

  Analytical Methods: Analyzing data movement and transformation

  Data Flow Diagrams Correspondence Rules

  Data flow diagrams should correspond to the flow of data through the system's processes and components.

  Each data flow should correspond to the movement of data between inputs, outputs, processes, and storage locations within the system.

  Data flow diagrams should accurately represent the transformations and processing performed on the data as it moves through the system.

- **Data Storage Models**

  Data Storage Models conventions

  Language: UML (Unified Modeling Language)

  Notations: Entity-Relationship Diagrams, Database Schema Diagrams

  Modeling Techniques: Describing how data is stored and related

  Analytical Methods: Assessing data storage efficiency and integrity

  Data Storage Models Correspondence Rules

  Data storage models should correspond to the organization and structure of data within the system.

  Each data entity in the model should correspond to a logical data element or entity within the system.

Data storage models should accurately represent the relationships and constraints between data entities, such as entities, attributes, and relationships.

Under Usability Viewpoint:

- **User Interface Models**

  User Interface Models conventions

  Language: Prototyping tools, graphical design tools

  Notations: Mockups

  Modeling Techniques: Visual representation of the user interface

  Analytical Methods: Usability testing, user feedback analysis

  User Interface Models Correspondence Rules

  User interface models should correspond to the design and layout of the system's user interface.

  Each interface element in the model should correspond to a user-visible component or interaction point within the system.

  User interface models should accurately represent the navigation paths, controls, and widgets available to users within the system.

- **Interaction Models**

  Interaction Models conventions

  Language: UML (Unified Modeling Language)

  Notations: Interaction Diagrams

  Modeling Techniques: Describing user-system interactions

  Analytical Methods: Assessing user experience, identifying usability issues

  Interaction Models Correspondence Rules

  Interaction models should correspond to the ways in which users interact with the system to accomplish tasks or goals.

Each interaction model should illustrate the flow of user actions and system responses during specific scenarios or use cases.

Interaction models should accurately represent the user's perspective and experience with the system.

Under Security Viewpoint:

- **Access Control Models**

  Access Control Models conventions

  Language: UML, Role-Based Access Control (RBAC) notations

  Notations: Access Control Diagrams

  Modeling Techniques: Defining roles and access permissions

  Analytical Methods: Evaluating access control effectiveness

  Access Control Models Correspondence Rules

  Access control models should correspond to the security requirements and policies defined for the system.

  Each access control rule or policy in the model should correspond to a specific authorization decision or constraint within the system.

  Access control models should accurately represent the permissions and restrictions applied to users or system components based on their roles or privileges.

3.3.7 Correspondence Rules

Functional Viewpoint

- **Across Models:** The team should make sure that all the different functional models such as Use Case Models, Functional Requirement Models, and Scenario Models are coherent with each other. Information presented across these different artifacts should align with one another.
- **Across view:** Ensure that the functional requirements of the systems correspond to the representations in models such as Activity Diagrams, Sequence Diagrams, etc.

Structural Viewpoint

- **Across Models:** The team should make sure that all the different structural models such as Class Diagrams, and Component Diagrams are coherent with each other. Information presented across these different artifacts should align with one another.
- **Across Views:** Ensure coherence in the organizational structure and relationships.

Behavioral Viewpoint

- **Across Models:** The team should make sure that all the different behavioral models such as State Diagrams, and Interaction Diagrams are coherent with each other. Ensure that the states and transitions in State Diagrams align with the events and interactions presented in Interaction Diagrams.
- **Across Views:** The team should make sure that the different states in the State Diagrams are coherent with the insights gained from the Interaction Diagrams.

Information Viewpoint

- **Across Models:** The team should verify the consistency between data-centric models such as Data Flow Diagrams, and Data Storage Models.
- **Across Views:** Ensure that data movement and transformation analysis in Data Flow Diagrams corresponds to the description of data storage and relationships in Data Storage Models.

Usability Viewpoint

- **Across Models:** The team should make sure of coherence between user-centric models such as User Interface Models and Interaction Models. A visual representation of the user interface should align with descriptions of user-system interactions.
- **Across Views:** The team should check for consistency in representing user experience and identifying usability issues.

Security Viewpoint

- **Across Models:** Verify coherence between security models such as Access Control Models and Component Diagrams. Role's distribution and access restrictions should be consistent with the interactions and dependencies identified in other models.
- **Across Views:** Ensure that the interactions and dependencies in Access Control Models correspond to the distribution of roles and access restrictions in other models.

3.3.8 Sources

[1]R. Hilliard, "Views and Viewpoints in Software Systems Architecture *." Accessed: Feb. 08, 2024. [Online]. Available: https://www.mit.edu/~richh/writings/hilliard99-ifip.pdf.

[2]"Software Systems Architecture," *www.viewpoints-and-perspectives.info*. https://www.viewpoints-and-perspectives.info/home/viewpoints/

[3]“The ‘4+1’ Model View of Software Architecture — PySD 3.13.3 documentation,” *pysd.readthedocs.io*. https://pysd.readthedocs.io/en/master/development/pysd_architecture_views/4+1view_model.html (accessed Feb. 08, 2024).

3.4 Views

3.4.1 View: User Profile and Access Management View

identifying and Supplementary Information:

Name: User Profile and Access Management View

Description: This view focuses on the management of user profiles and access control within the condo management system.

Organization/Project-Specific Information: [Specify any relevant details]

Viewpoint Governing this View:

Viewpoint: Functional Viewpoint (as identified in §3)

3.4.1.1 Models: User Profile Model

Version Identification: 1.0

Governing Model Kind: User Profile Model

Adheres to conventions from Data Flow Diagrams viewpoint.

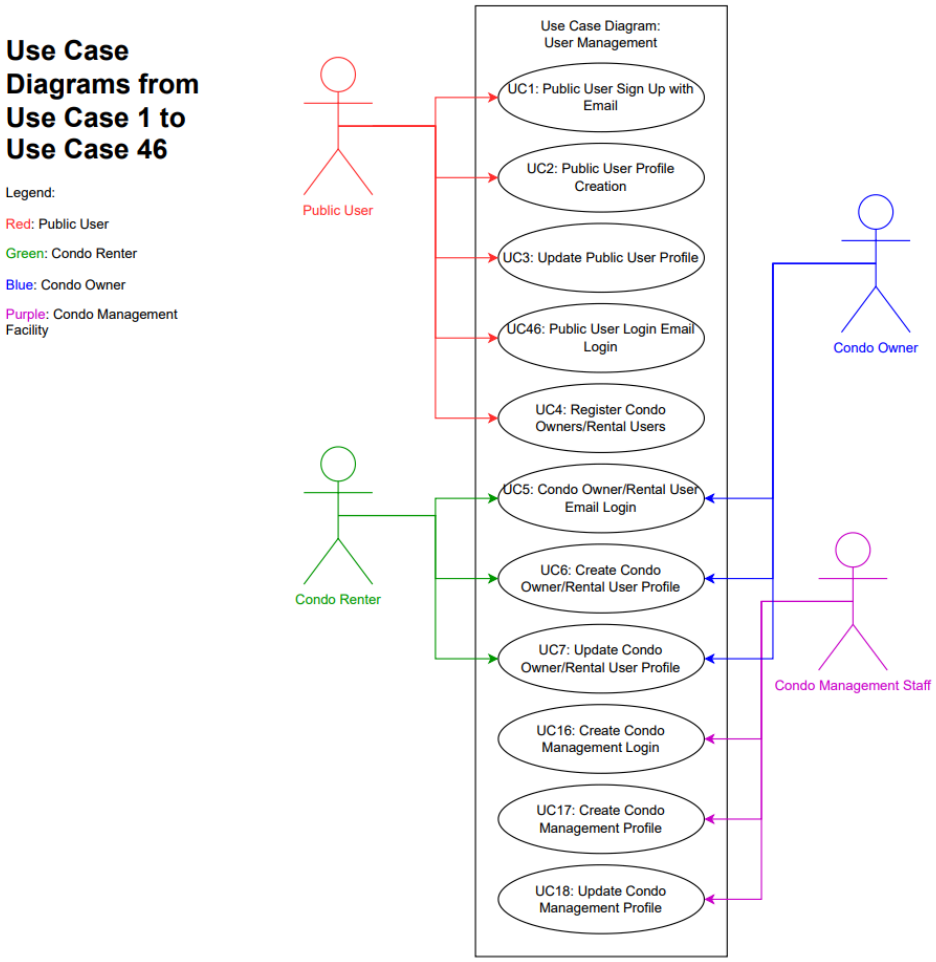
Language: UML (Unified Modeling Language)

Notations: Class Diagrams

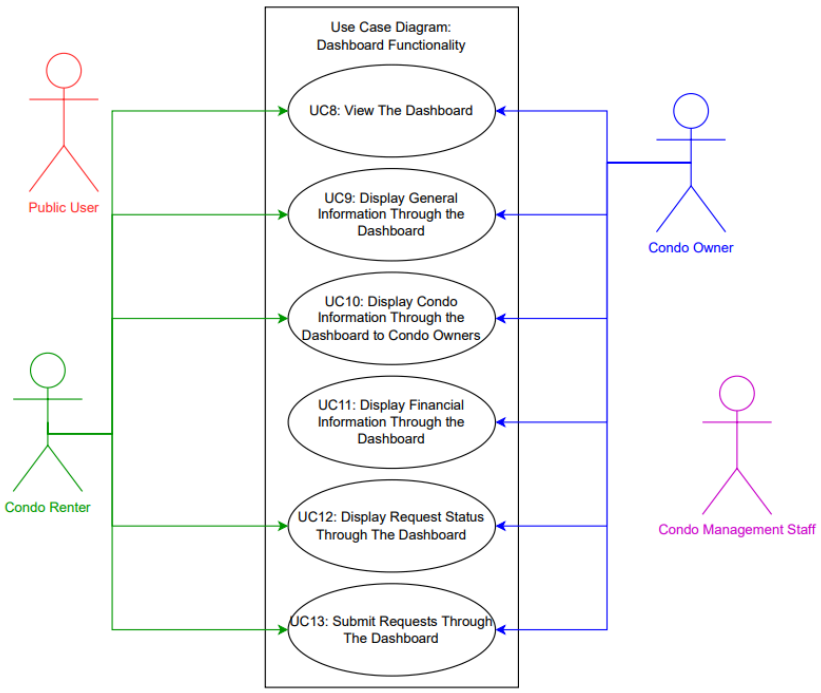
Modeling Techniques: Representing user profiles and their attributes.

Analytical Methods: Analyzing data flow within the user profile management system.

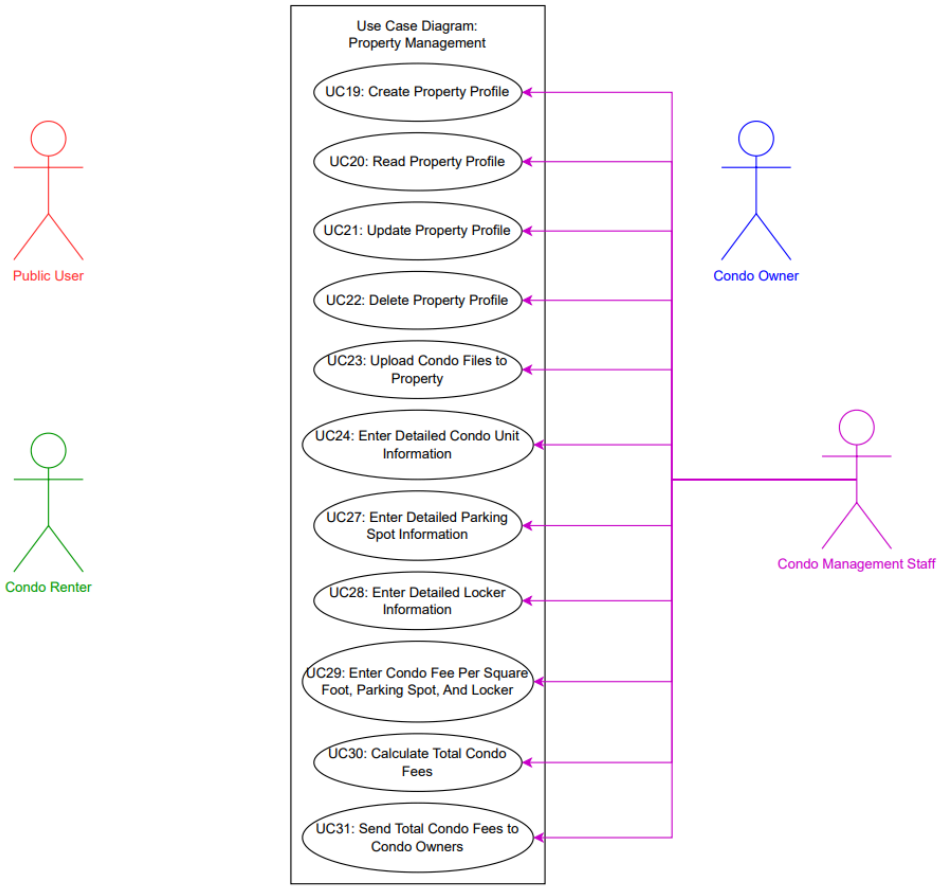
1. Use Case Diagram: User Management



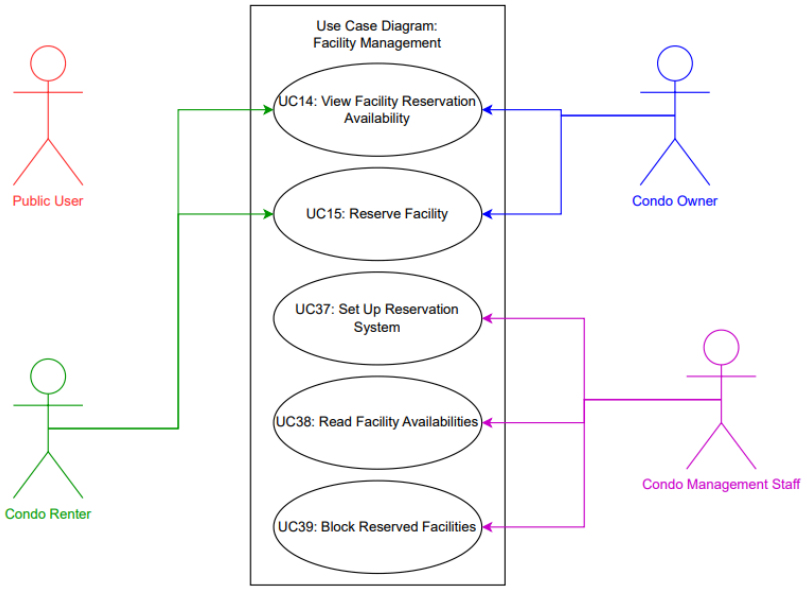
2. Use Case Diagram: Dashboard Functionality



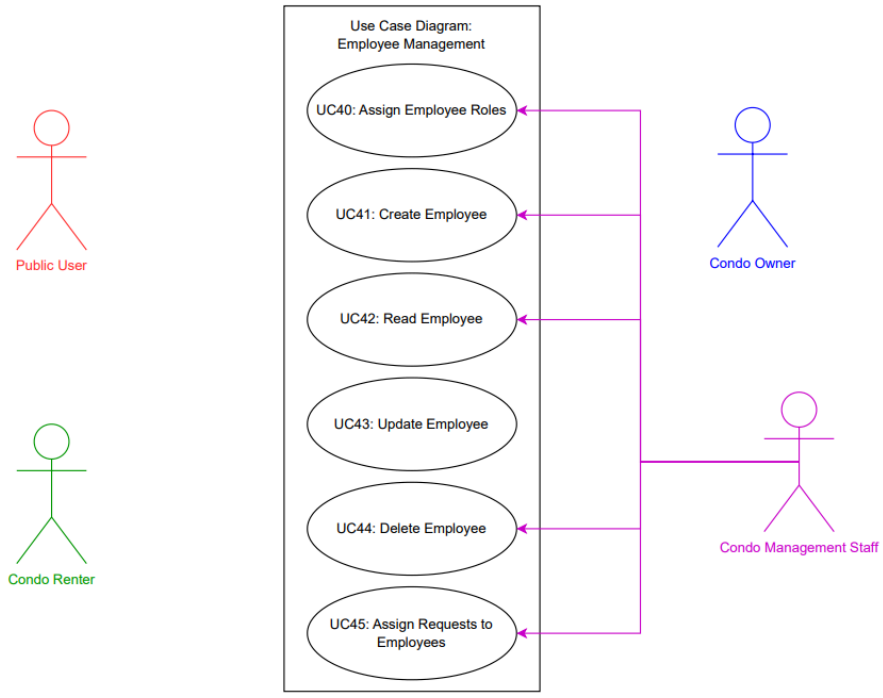
3. Use Case Diagram: Property Management



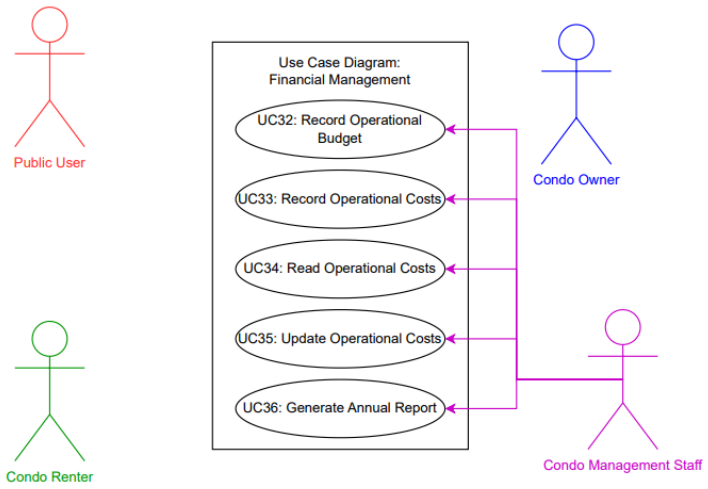
4. Use Case Diagram: Facility Management



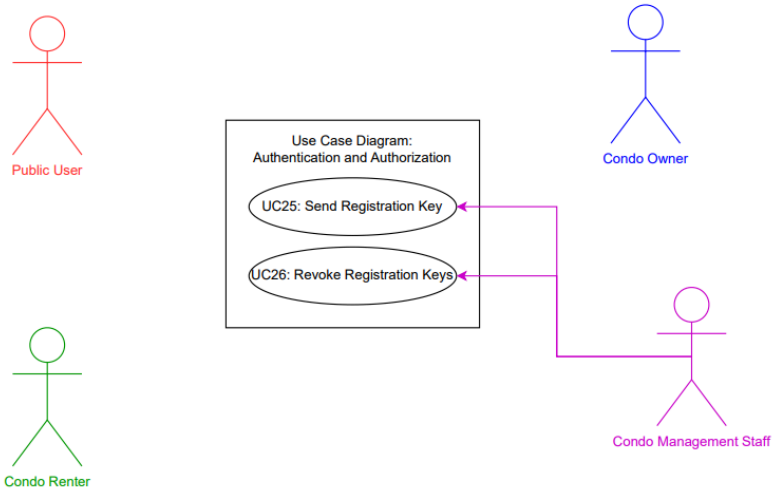
5. Use Case Diagram: Employee Management



6. Use Case Diagram: Financial Management



7. Use Case Diagram: Authentication and Authorization



3.4.1.2 Access Control Model

Version Identification: 1.0

Governing Model Kind: Access Control Model

Adheres to conventions from Data Storage Models viewpoint.

Language: UML (Unified Modeling Language)

Notations: Entity-Relationship Diagrams

Modeling Techniques: Describing how access control is implemented.

Analytical Methods: Assessing access control efficiency and integrity.

3.4.1.3 Known Issues with View

Inconsistencies in updating user profile information.

Open issue regarding the synchronization of access control data.

Decision outcome: A synchronization mechanism will be introduced to ensure consistency (Documented in decision outcomes and rationale).

3.4.2 View: Logical View

Identifying and Supplementary Information:

The Logical View provides an abstract representation of the system's logical structure, focusing on the organization of components and their interactions.

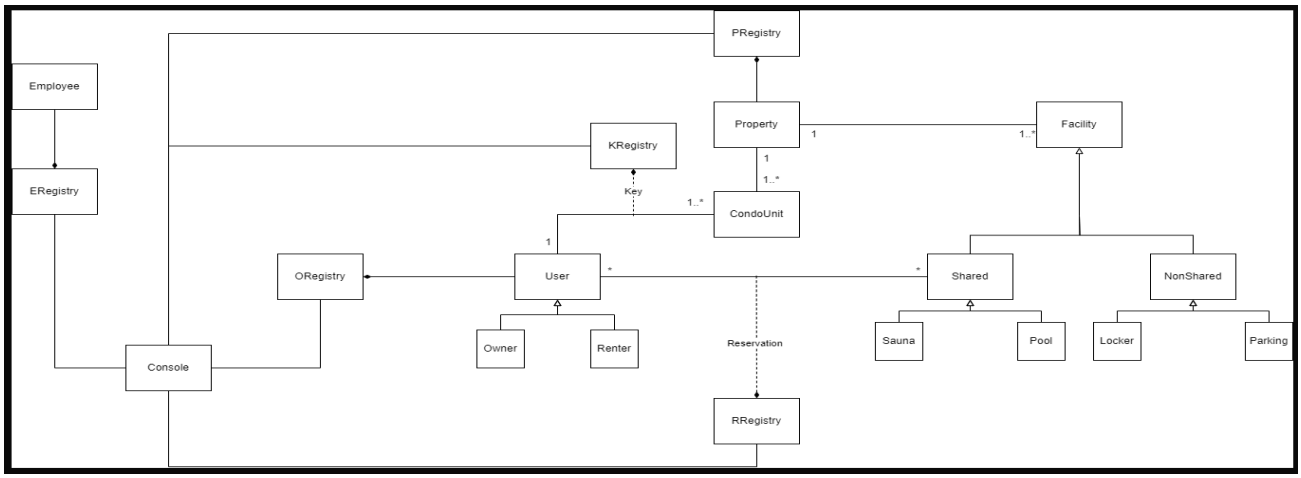
This view facilitates understanding of the system's functional aspects and the relationships between its constituent parts.

Viewpoint Governing this View:

The Structural Viewpoint (Section 3.2) governs the Logical View, focusing on the organization and structure of the system's components and their interrelationships.

3.4.2.1 Models+

Domain Model



3.4.2.2 Architecture Models:

The Logical View comprises several architecture models, each addressing specific concerns framed by the governing viewpoint.

These models collectively cover the entire system from a structural perspective.

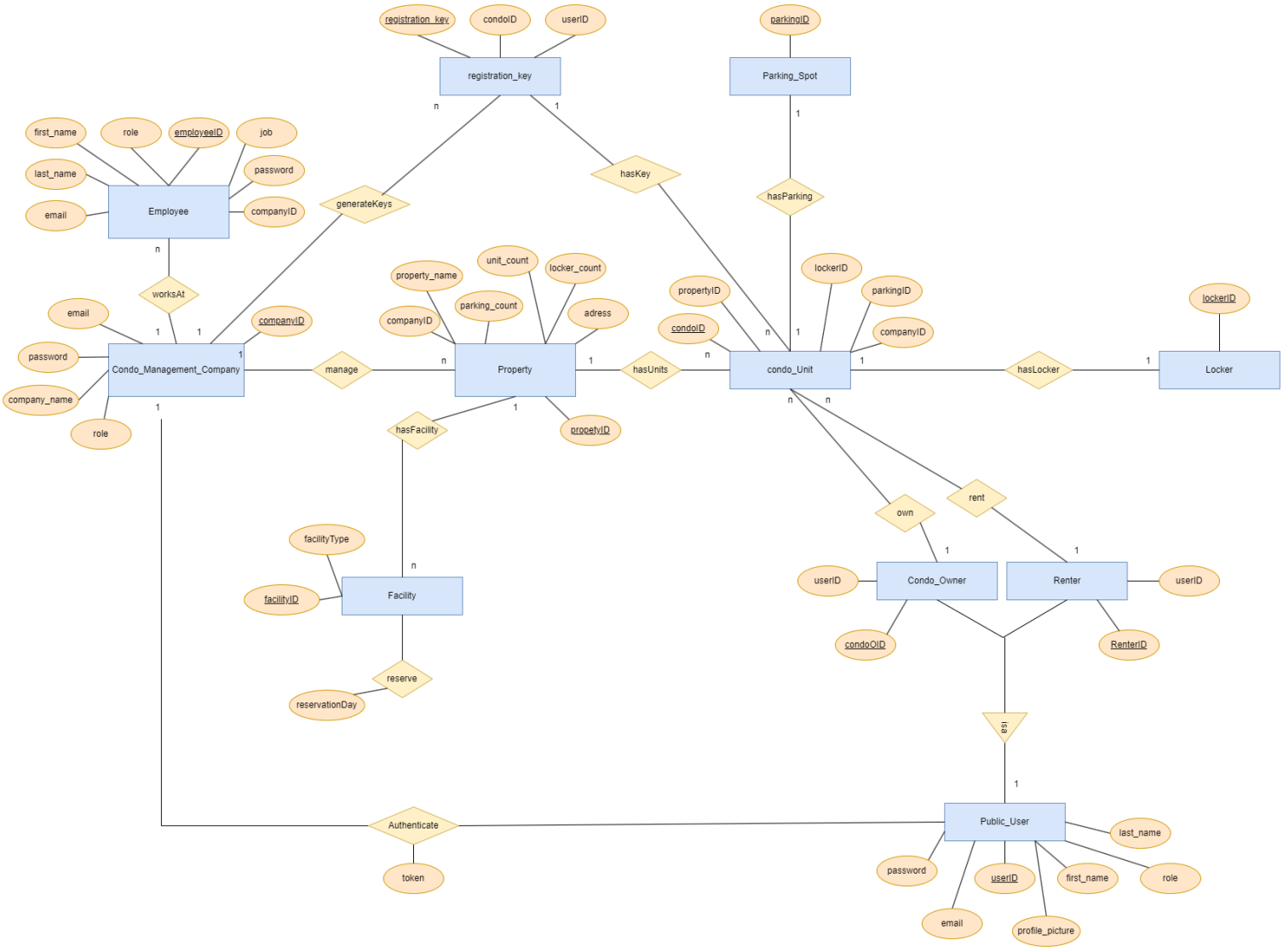
3.4.2.3 Logical Component Diagram:

Governing Model Kind: Component Diagram

Description: The Logical Component Diagram illustrates the high-level organization of system components and their interconnections. It depicts the logical structure of the system, including modules, classes, and interfaces, and highlights the dependencies and relationships among them.

3.4.2.4 Models+

Entity Relationship Diagram (dr)



3.4.2.5 Known Issues with View:

There are no known issues with adherence to the Structural Viewpoint conventions for the Logical View at this time.

Any discrepancies or deviations from the viewpoint conventions will be documented promptly and addressed to ensure consistency and alignment with architectural principles.

3.4.3 View: Process View

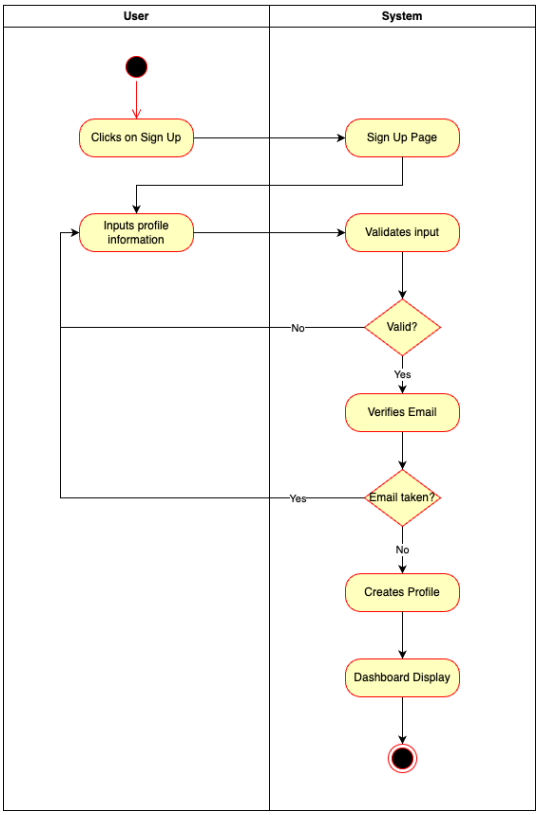
The dynamic behavior and interactions of the system's processes or activities are the main emphasis of the software architecture perspective known as the process viewpoint. It offers perceptions into the way the system works throughout time, including the data flow, control, and intermodular communication.

3.4.3.1 Models+

Activity Diagrams

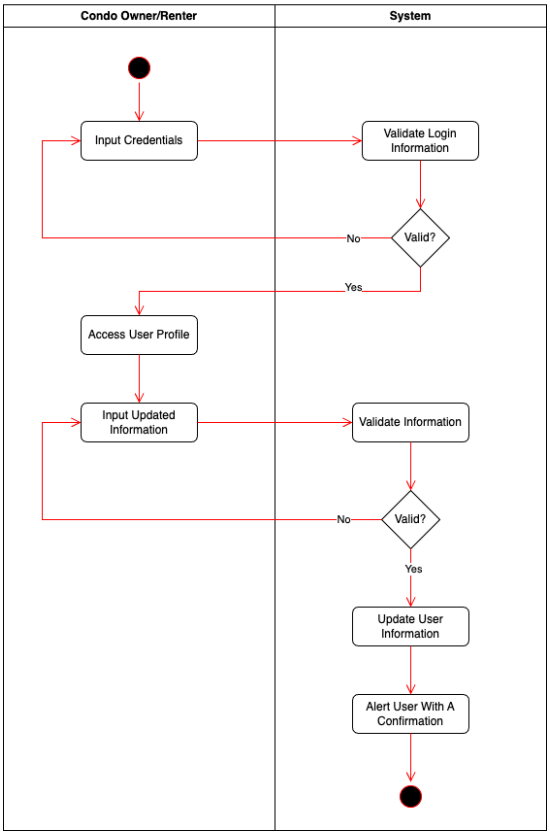
Activity Diagram: User Profile Creation

UC - 1.1: User Profile Creation



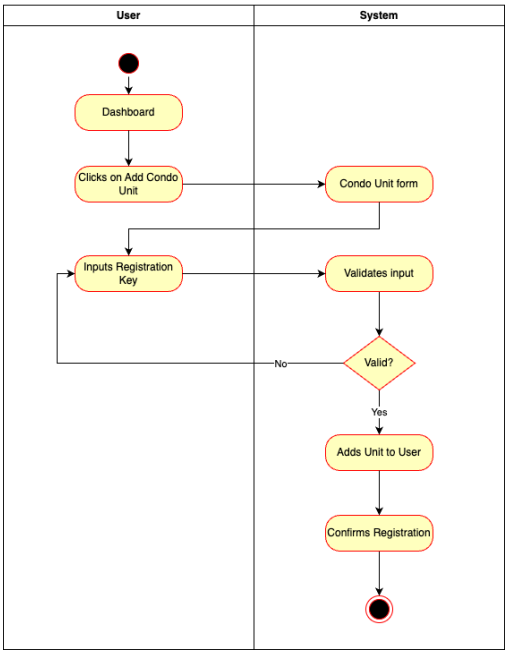
Activity Diagram: User Profile Update

UC - 1.2: User Profile Update



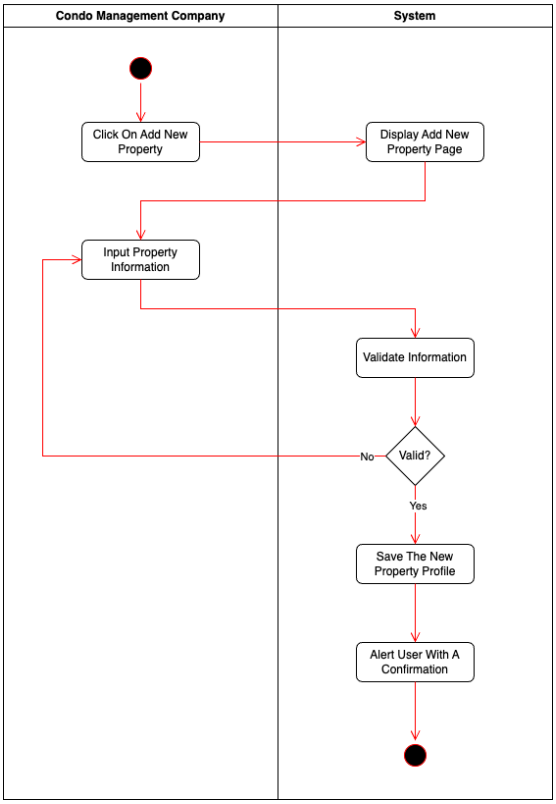
Activity Diagram: Condo Owner/Rental User Registration

UC - 1.3: Condo Owner/Rental User Registration

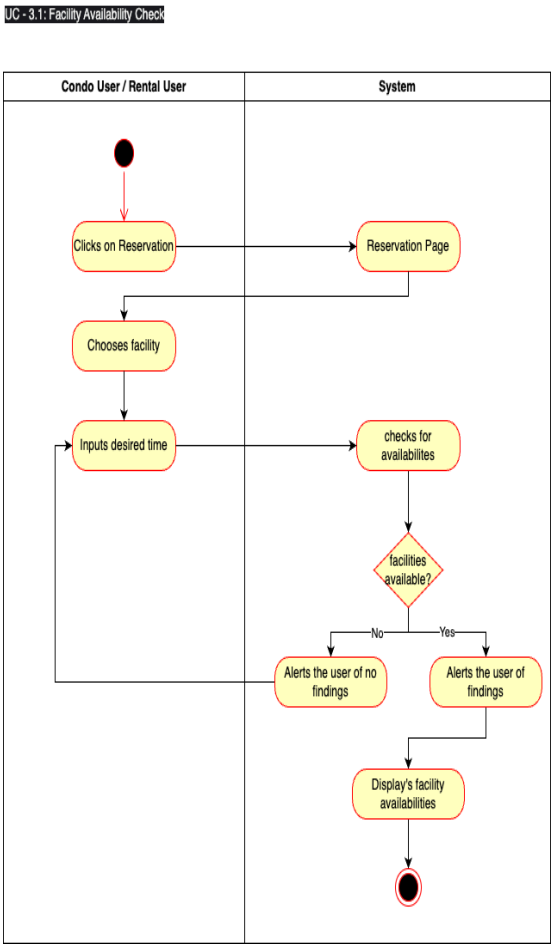


Activity Diagram: Create Property Profile

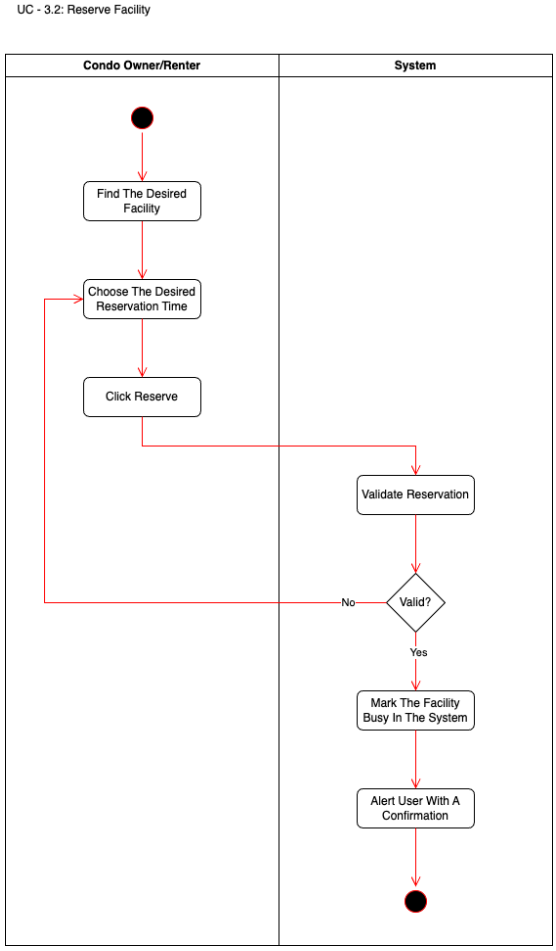
UC - 2.1: Create Property Profile



Activity Diagram: Facility Availability Check

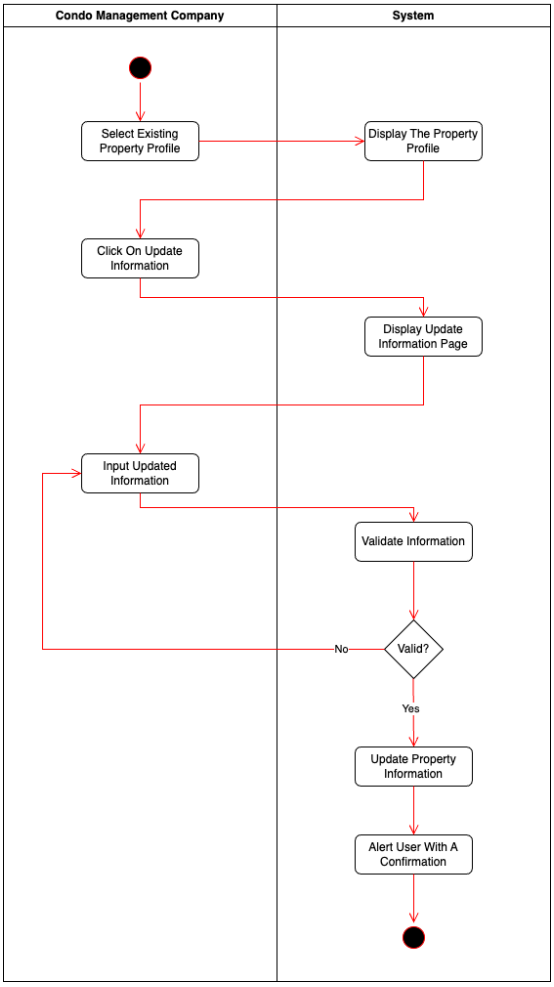


Activity Diagram: Reserve Facility



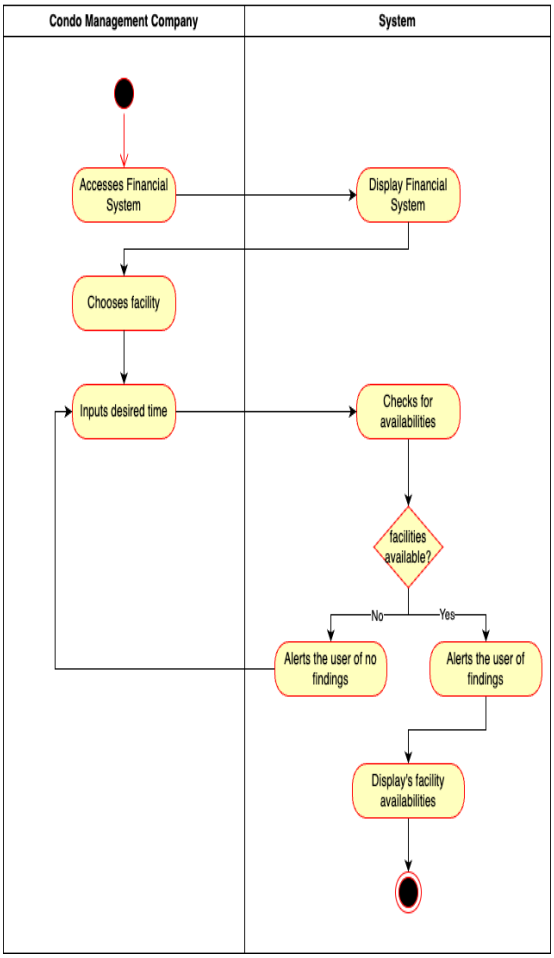
Activity Diagram: Update Property Profile

UC - 2.2: Update Property Profile



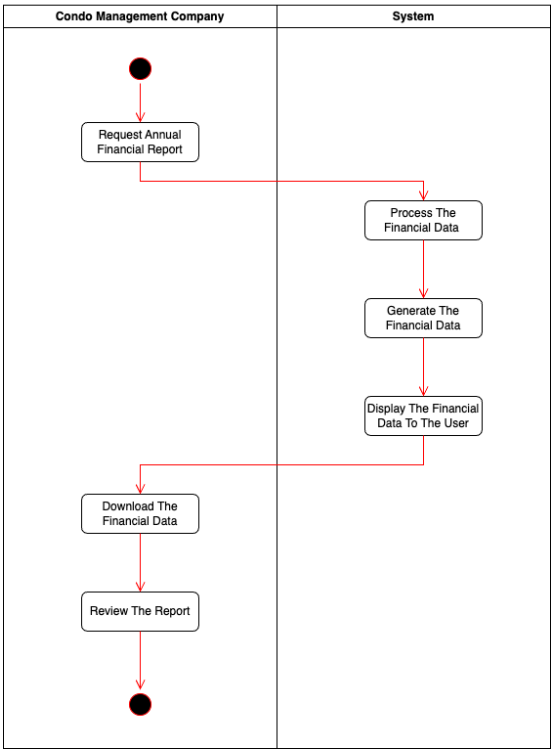
Activity Diagram: Enter Condo Fee Rates

UC - 4.1: Enter Condo Fee Rates



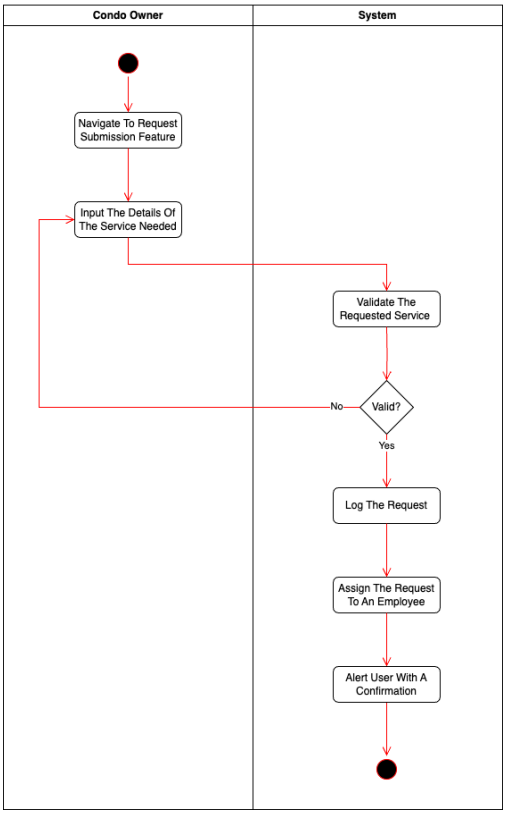
Activity Diagram: Generate Financial Report

UC - 4.2: Generate Financial Report



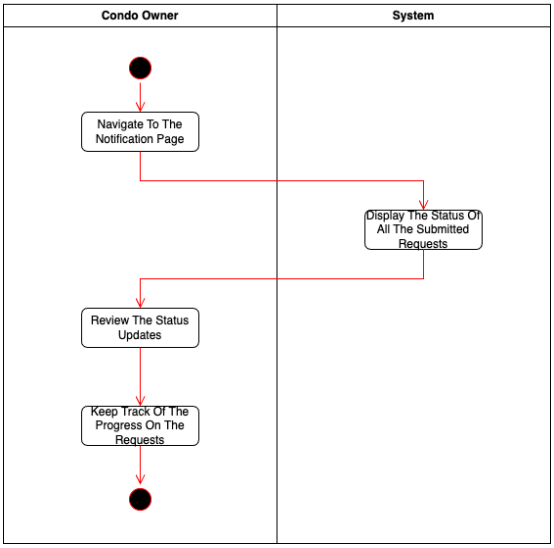
Activity Diagram: Submit Service Request

UC - 5.1: Submit Service Request



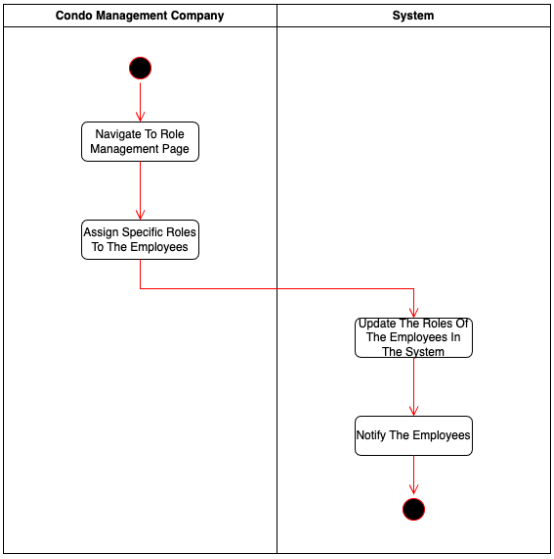
Activity Diagram: Track Request Status

UC - 5.2: Track Request Status



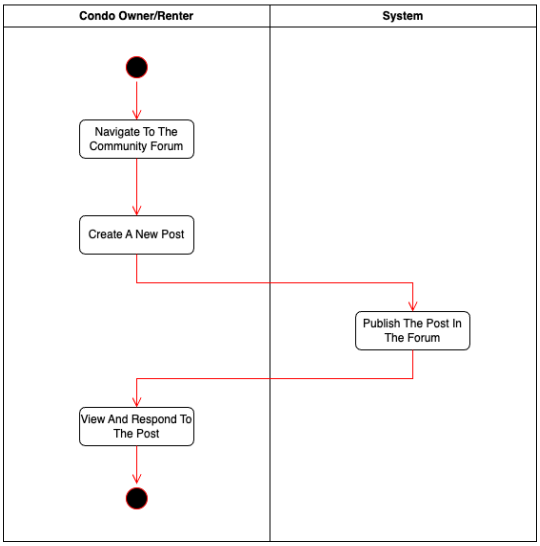
Activity Diagram: Role Assignment for Employees

UC - 6.1: Role Assignment for Employees



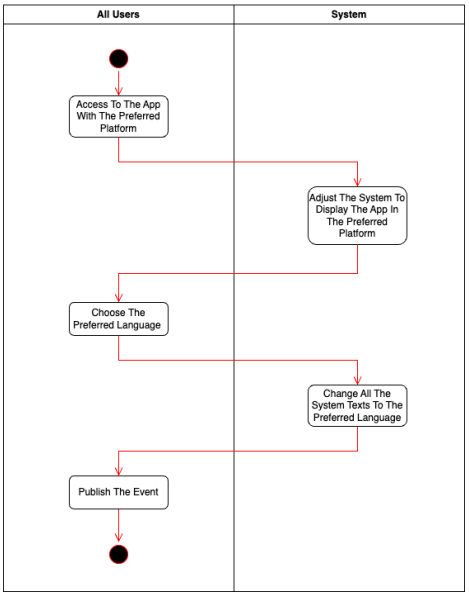
Activity Diagram: Create Community Forum Post

UC - 7.1: Create Community Forum Post

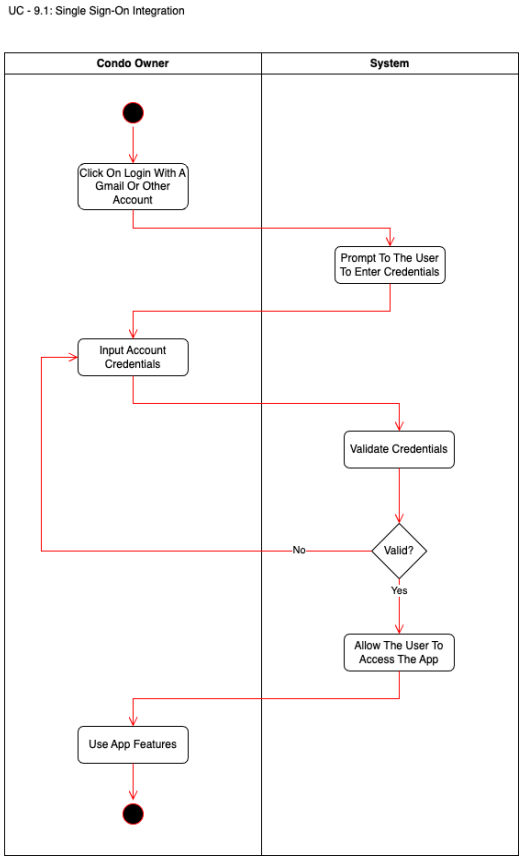


Activity Diagram: Cross-Platform and Multilingual Support

UC - 8.1: Cross-Platform and Multilingual Support



Activity Diagram: Single Sign-On Integration



3.4.3.2 Activity Diagrams

Activity diagrams can be looked at as both functional and behavioral diagrams. Activity diagrams represent behavioral diagrams due to the fact their main emphasis is on illustrating the dynamic behavior or the flow of events that take place inside the system in order to accomplish a specific objective or functionality. They show how the system performs different actions in response to outside stimuli, human input, or internal triggers. It's crucial to remember, though, that functional characteristics can also be indirectly represented by activity diagrams. Even though their main focus is on the sequence and flow of control, the activities themselves frequently match up with system functions or operations. Usually, each activity stands for a distinct job, function, or process that helps the system achieve a certain feature or functionality.

3.4.3.3 Known Issues with View

Activity diagrams may run into a number of recognized problems while following the rules regulating architecture views, which could compromise the accuracy and efficacy of the architectural depiction. These problems usually have to do with keeping things consistent,

handling unfinished or partial parts, handling exceptions, and recording departures from accepted practices.

Among the most common problems that could occur are inconsistent activity diagrams. These discrepancies might appear in a number of ways, including inconsistent activity sequences, illogical decision-making, and misaligned notation. Inconsistencies have the potential to undermine the overall correctness and clarity of the architectural representation and cause confusion among stakeholders.

Activity diagrams may have elements that are designated as unfinished or as stand-ins for later work. Although this can be a useful strategy for iterative or rapid development processes, it might make it difficult to comprehend the intended behavior of the system and make it more difficult to make decisions about the architecture.

Each architecture view must adhere to the conventions of its governing architecture viewpoint.

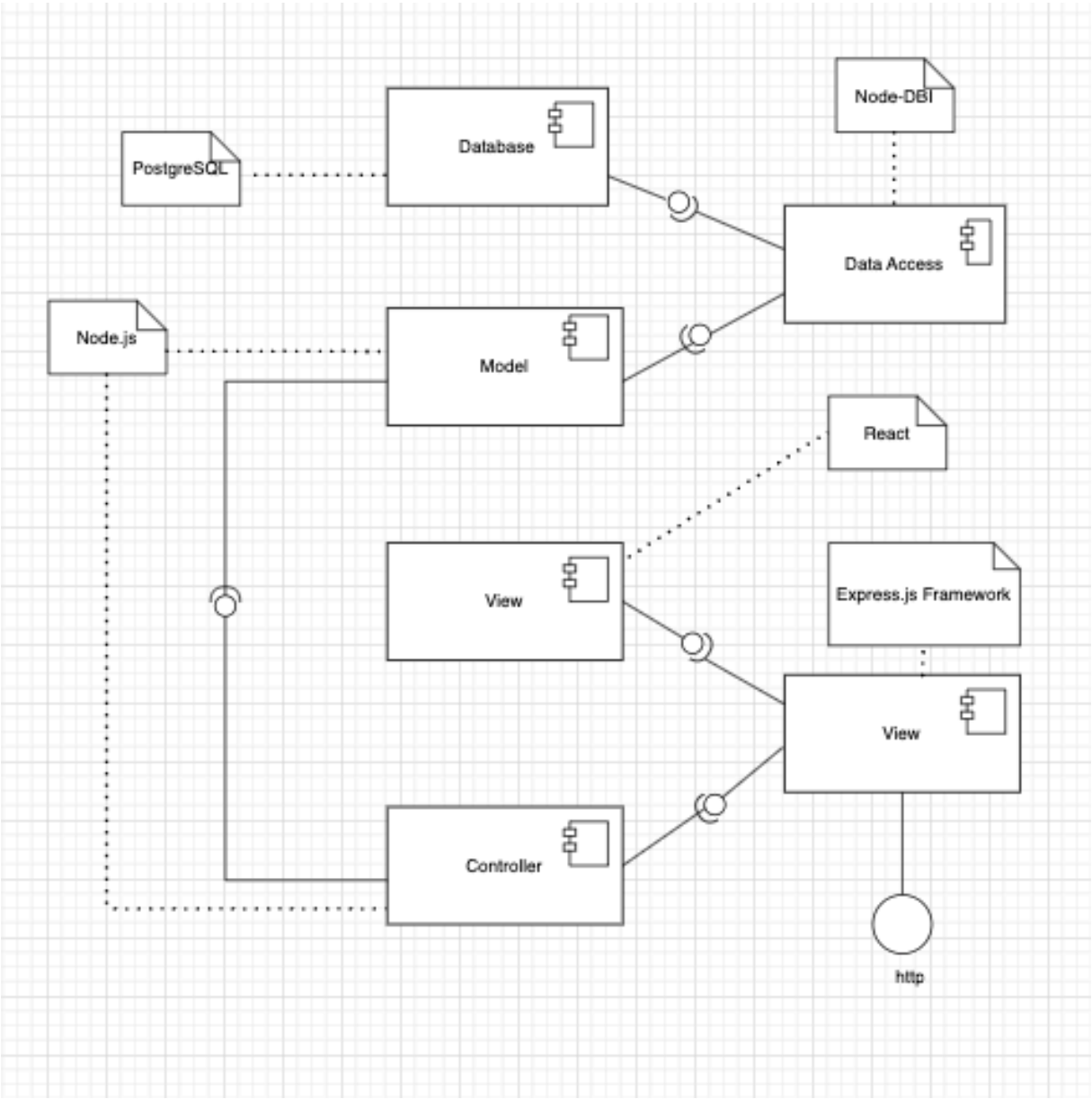
Known issues could include: inconsistencies, items to be completed, open or unresolved issues, exceptions and deviations from the conventions established by the viewpoint. Open issues can lead to decisions to be made. Exceptions and deviations can be documented as decision outcomes and rationale.

3.4.4 View: Implementation View

Within software architecture, the implementation viewpoint is a viewpoint that concentrates on the specifics of how the system will be built, executed, and arranged in terms of software modules, components, and code.

3.4.4.1 Models+

Component Diagram



3.4.4.2 Component Diagram

In software architecture, the component diagram is categorized under the structural viewpoint. The goal of structural viewpoints is to depict the system's connections, linkages, and physical or logical structure. Component diagrams are an essential component of the structural viewpoint because they explicitly show the arrangement and interdependence of the system's physical components, or modules.

3.4.4.3 Known Issues With View

Component diagrams may run into a number of recognized problems while following the rules of the governing architecture viewpoint, which could compromise the accuracy and efficacy of the architectural depiction. These problems usually have to do with keeping things consistent, handling unfinished or partial parts, handling exceptions, and recording departures from accepted practices.

One of the main problems that could occur is component diagram discrepancies. These contradictions can show up as disparate component naming conventions, incompatible interfaces, or contradictory component relationships, among other ways. Inconsistencies have the potential to undermine the overall correctness and clarity of the architectural representation and cause confusion among stakeholders.

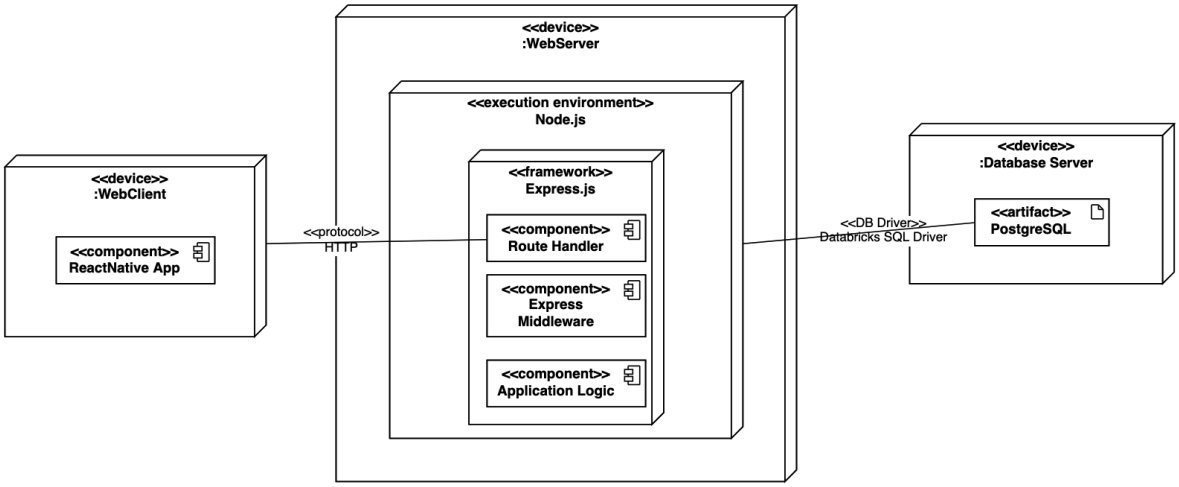
Also, elements that are designated as incomplete or placeholders for future development may be found in component diagrams. These elements can be incomplete interface specifications, unresolved dependencies, or continuing debates about component bounds. Although placeholders might act as a helpful reminder for more tasks, they can also cause confusion and postpone the completion of the component architecture.

3.4.5 View: Deployment View

Within a software system's architecture, the deployment viewpoint is a viewpoint that concentrates on the actual physical deployment of software artifacts onto hardware infrastructure. It offers a thorough grasp of how the system's parts, modules, and services are dispersed over environments, networks, and computer resources in order to meet its operating needs.

3.4.5.1 Models+

Deployment Diagram



3.4.5.2 Deployment Diagram

In software architecture, the deployment diagram is categorized under the structural viewpoint. The goal of structural viewpoints is to depict the system's connections, linkages, and physical or logical structure. Deployment diagrams are an essential component of the structural viewpoint because they explicitly show the physical deployment of software artifacts onto hardware nodes, networks, and middleware components.

3.4.5.3 Known Issues With Vlew

Deployment diagrams may run into a number of recognized problems while following the rules of the controlling architecture viewpoint, which could compromise the accuracy and efficacy of the architectural depiction. These problems usually have to do with keeping things consistent, handling unfinished or partial parts, handling exceptions, and recording departures from accepted practices.

One of the main problems that could occur is that deployment diagrams can be inconsistent. These disparities can show up as disparate deployment settings, mismatched component and node connections, or differences in how software artifacts are assigned to hardware resources, among other things. Inconsistencies can result in performance problems, deployment mistakes, or trouble making sure the system works as intended in various situations.

3.5 Consistency And Correspondences

This chapter describes consistency requirements, recording of known inconsistencies in an AD, and the use and documentation of correspondences and correspondence rules.

3.5.1 Known Inconsistencies

In the AD, inconsistencies may arise due to various factors such as time constraints, resource limitations, or incomplete information. Despite efforts to maintain consistency, it may be impractical to resolve all inconsistencies. An analysis of consistency within architecture models and views should be included in the AD.

Known Inconsistencies:

1. Financial data discrepancies due to recording or calculation issues.
2. Inaccurate reflection of common facilities availability for reservations.
3. Challenges in assigning requests to corresponding employees.
4. Potential limitations in language translations for the app.

3.5.2 Correspondences in the AD

Correspondences play a crucial role in expressing, recording, enforcing, and analyzing consistency within the AD. These correspondences establish relationships between various AD elements, including stakeholders, concerns, viewpoints, views, model kinds, models, decisions, and rationales. They are represented as n-ary mathematical relations and can be depicted through tables, links, or other forms of association.

Identified Correspondences:

1. **Profile Creation Correspondence:**
   - Participating AD Elements: Public Users, User Profile Model
   - Correspondence Rule: Each public user should have a unique profile with essential information.

2. **Property Profile Correspondence:**
   - Participating AD Elements: Condo Management Companies, Property Profile Model
   - Correspondence Rule: Each property under management should have a profile with essential information.

3. **Financial Data Correspondence:**
    - Participating AD Elements: Condo Management Companies, Financial System Model
    - Correspondence Rule: Financial data should be accurately recorded and updated.

4. **Reservation System Correspondence:**
    - Participating AD Elements: Condo Management Companies, Reservation System Model
    - Correspondence Rule: Reservation system should accurately reflect the availability of common facilities.

5. **Request Management Correspondence:**
    - Participating AD Elements: Condo Owners, Condo Management Companies, Request Management Model
    - Correspondence Rule: Requests should be assigned to corresponding employees based on request type.

3.5.3 Correspondence Rules

Correspondence rules govern the relationships established by correspondences within the AD. These rules can be introduced by the AD, viewpoints, or architecture frameworks/languages. It's important to determine whether these rules are satisfied or if there are any known violations.

Identified Correspondence Rules:

1. **Profile Completeness Rule:**
    - Holds: Each public user has provided the necessary information for profile creation.

2. **Financial Data Accuracy Rule:**
    - Holds: Financial data is accurately recorded and updated.

3. **Reservation Availability Rule:**
    - Holds: Reservation system accurately reflects the availability of common facilities.

4. **Request Assignment Rule:**
    - Holds: Requests are assigned to corresponding employees based on request type.

5. **Translation Quality Rule:**
    - Violation: Language translations may not perfectly convey the intended meaning due to nuances.

3.6 Bibliography

[1] Paul C. Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. Documenting Software Architectures: views and beyond. Addison Wesley, 2nd edition, 2010.

[2] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering, 2(1):31–57, March 1992.

[3] IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, October 2000.

[4] ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description, December 2011.

[5] Alexander Ran. Ares conceptual framework for software architecture. In M. Jazayeri, A. Ran, and F. van der Linden, editors, Software Architecture for Product Families Principles and Practice, pages 1–29. Addison-Wesley, 2000.

[6] Nick Rozanski and Eo´in Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison Wesley, 2nd edition, 2011.

[7] Uwe van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. The Journal of Systems & Software, 85(4):795–820, April 2012.

[9]R. Hilliard, “Views and Viewpoints in Software Systems Architecture *.” Accessed: Feb. 08, 2024. [Online]. Available: https://www.mit.edu/~richh/writings/hilliard99-ifip.pdf.

[10]“Software Systems Architecture,” *www.viewpoints-and-perspectives.info*. https://www.viewpoints-and-perspectives.info/home/viewpoints/

[11]“The ‘4+1’ Model View of Software Architecture — PySD 3.13.3 documentation,” *pysd.readthedocs.io*. https://pysd.readthedocs.io/en/master/development/pysd_architecture_views/4+1view_model.html (accessed Feb. 08, 2024).

Chapter 4: Risk Assessment and Risk Management Plan

4.1 Purpose

The primary objective of conducting a comprehensive risk assessment and management plan is to systematically identify, analyze, mitigate, and monitor potential risks that may confront the development team throughout the project lifecycle. This strategic approach aims to minimize schedule disruptions, enhance system performance, foster effective communication, and minimize adverse impacts.

4.2 Risk Identification

The process of identifying risks involved drawing upon past experiences, insights gleaned from analogous projects, and collaborative brainstorming sessions. By leveraging these resources, the team endeavored to proactively recognize and articulate potential threats to project success.

4.3 Conducting the Risk Assessment

The risk assessment commenced with the initial phase of risk identification. Subsequently, each identified risk underwent evaluation utilizing the framework delineated in the risk management chart. Additionally, the team utilized a risk analysis template, facilitated through an Excel spreadsheet format provided by the instructor. This systematic approach facilitated a thorough comprehension and analysis of the identified risks.

4.4 Criteria for Different Levels of Risks

4.4.1 Very Low Risk

These are risks that are highly unlikely to occur or have very minimal impact if they do occur.

Criteria:

- Probability of occurrence: Less than 10%.
- Impact: Negligible or no impact on project objectives.
- Mitigation: Easily manageable with existing controls or resources.

Example: Communication barriers among team members.

4.4.2 Low Risk

These risks have a relatively low probability of occurrence and a low impact on project objectives if they do occur.

Criteria:

- Probability of occurrence: 10% to 30%.

- Impact: Minor impact on project objectives; can be absorbed without significant disruption.
- Mitigation: Requires minimal resources or adjustments to manage.

Example: Technology risks.

4.4.3 Medium Risk

These risks have a moderate probability of occurrence and can have a noticeable impact on project objectives if they materialize.

Criteria:

- Probability of occurrence: 30% to 50%.
- Impact: Moderate impact on project objectives; may require some resources to address.
- Mitigation: Requires proactive monitoring and some resources for mitigation.

Example: Scheduling conflicts.

4.4.4 High Risk

These risks have a significant probability of occurrence and can have a substantial impact on project objectives if they materialize.

Criteria:

- Probability of occurrence: 50% to 70%.
- Impact: Significant impact on project objectives may lead to delays or budget overruns if not addressed promptly.
- Mitigation: Requires immediate attention and allocation of resources for mitigation.

Example: Team member burnout.

4.4.5 Critical Risk

These risks have a high probability of occurrence and can severely impact or jeopardize the success of the project if they materialize.

Criteria:

- Probability of occurrence: More than 70%.
- Impact: Severe impact on project objectives; may lead to project failure or significant damage.
- Mitigation: Requires immediate and intensive efforts, possibly involving all available resources.

Example: Unclear requirements.

4.5 Risk Management Chart

Risk Management Matrix

Impact / Probability	1- Very Low	2- Low	3- Medium	4- High	5- Critical
1- Very Low			R11: Communication barriers. R15: Conflicts between team members.		R16: Lack of review.
2- Low			R2: Technology risks. R4: Security vulnerabilities. R7: Lack of team cohesion.	R8: Skill gaps. R10: Lack of accountability.	R14: Low-quality deliverables.
3- Medium			R3: Integration issues. R5: Unrealistic timelines. R6: Scheduling conflicts. R12: Feature creep.	R9: Team member burnout.	R13: Lack of planning.
4- High					R1: Unclear requirements.
5- Critical					

4.6 List of Identified Risks

Risk Management Table

Risk ID	Risk Type and Description	Risk Score	Resolved in Sprint	Strategy and Effectiveness
R1: Unclear Requirements	Management	Critical	2	Mitigate and Accept
R2: Technology	Technical	Low	2	Mitigate and

Risks				Accept
R3: Integration Issues	Technical	Medium	2	Mitigate and Accept
R4: Security Vulnerabilities	Technical	Low	2	Mitigate and Accept
R5: Unrealistic Timelines	Schedule and Management	Medium	2	Mitigate and Accept
R6: Scheduling Conflicts	Schedule and Management	Medium	2	Mitigate and Accept
R7: Lack of Team Cohesion	Management	Low	2	Mitigate and Accept
R8: Skill Gaps	Management and Technical	Medium	3	Mitigate and Accept
R9: Team Member Burnout	Management	High	2	Mitigate and Accept
R10: Lack of Accountability	Management	Medium	2	Mitigate and Accept
R11: Communication Barriers	Management	Very Low	2	Mitigate and Accept
R12: Feature Creep	Management	Medium	1	Mitigate and Accept
R13: Lack of Planning	Management	High	2	Mitigate and Accept
R14: Low-Quality Deliverable	Technical	Medium	2	Mitigate and Accept
R15: Conflicts Between Team Members	Management		1	Mitigate and Accept
R16: Lack of Review	Management & Technical	Low	2	Mitigate and Accept

4.7 Risk Analysis Table

https://docs.google.com/spreadsheets/d/1HAuqeZkYPTOLLIWWbgLS1g6C5UPmLzDJ/edit#gid=1283469489

Chapter 5: Testing

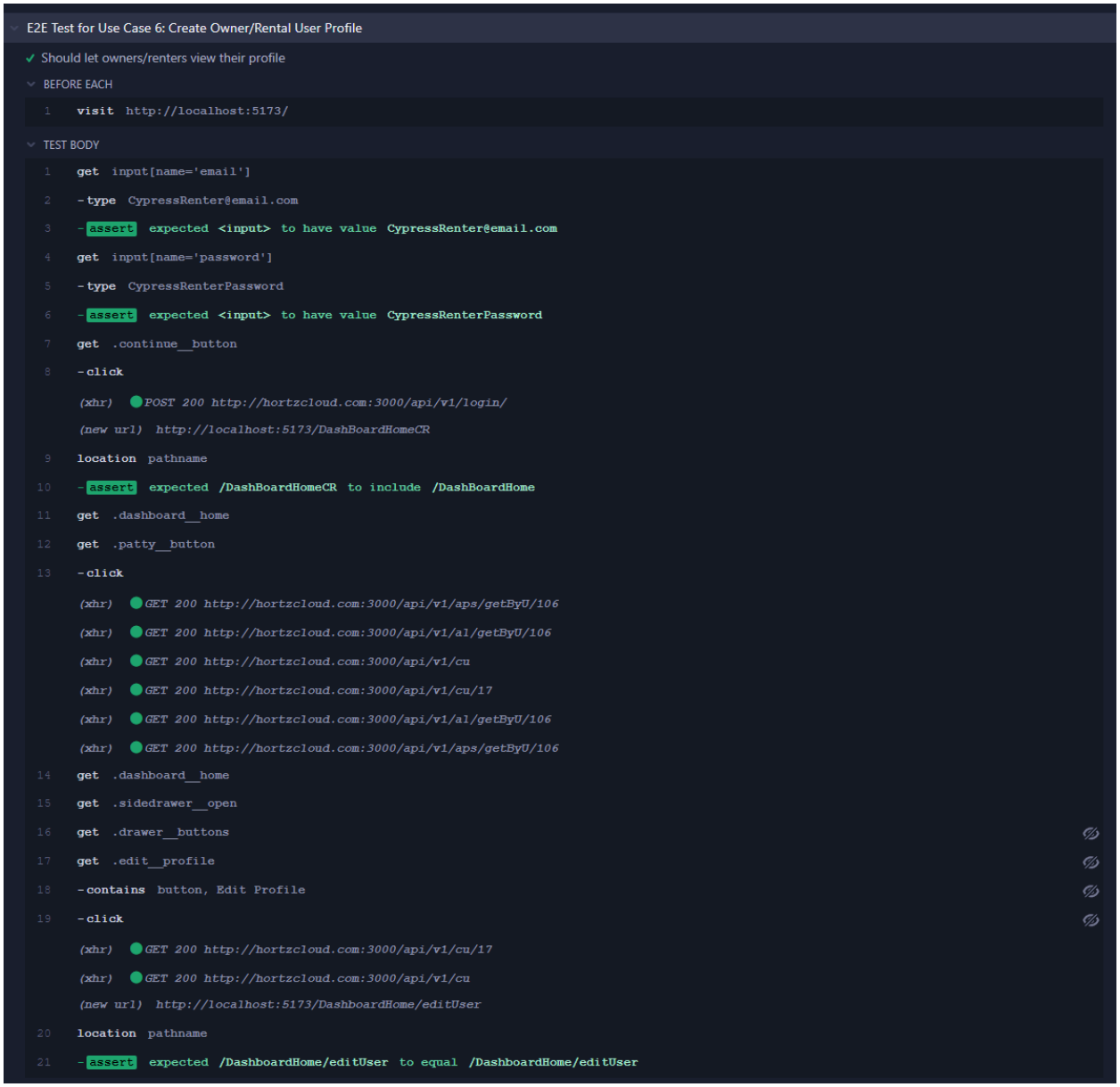
FRONTEND

E2E Testing

For the client side of the project, we used Cypress to perform End to End tests for each of the completed Use Cases. The goal of these test is to reach 100% Requirement Coverage by the end of sprint 5.

So far, we have passing E2E tests for all of the fully implemented use cases, excluding the ones just implemented in Sprint 3.

Here is an example of a E2E test:



BACKEND

Unit Testing

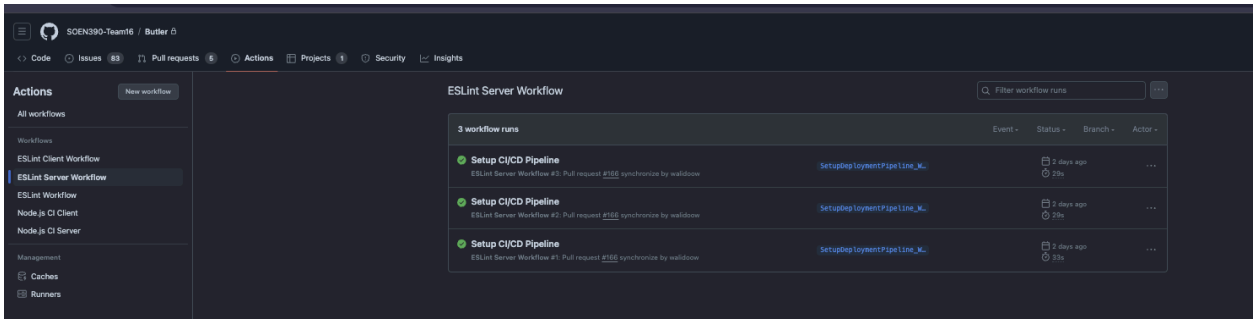
For the server side of the project, we used JEST to unit test all of our components implemented this sprint, in which we tested the functions for the controller.js files for financial systems including calculating condo fees, employee management, locker and parking unit management.

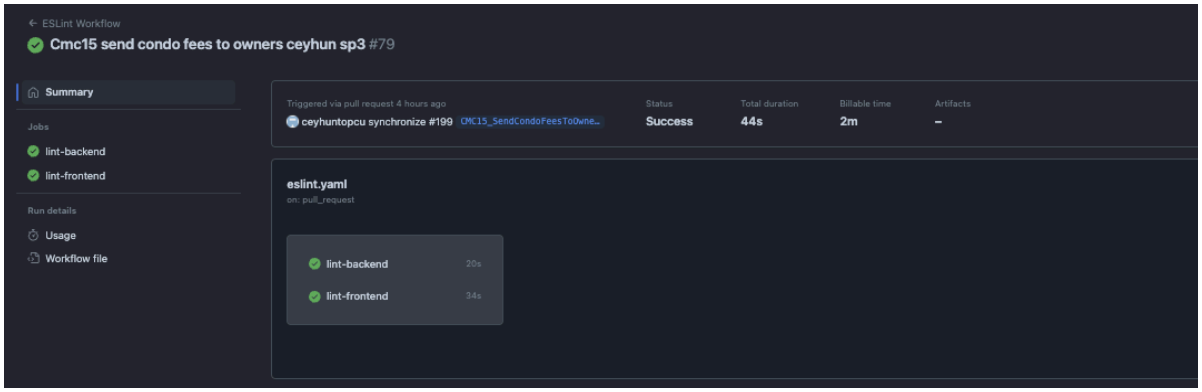
Test Coverage using JEST for testing components in the backend (Function Coverage: 80.5%)

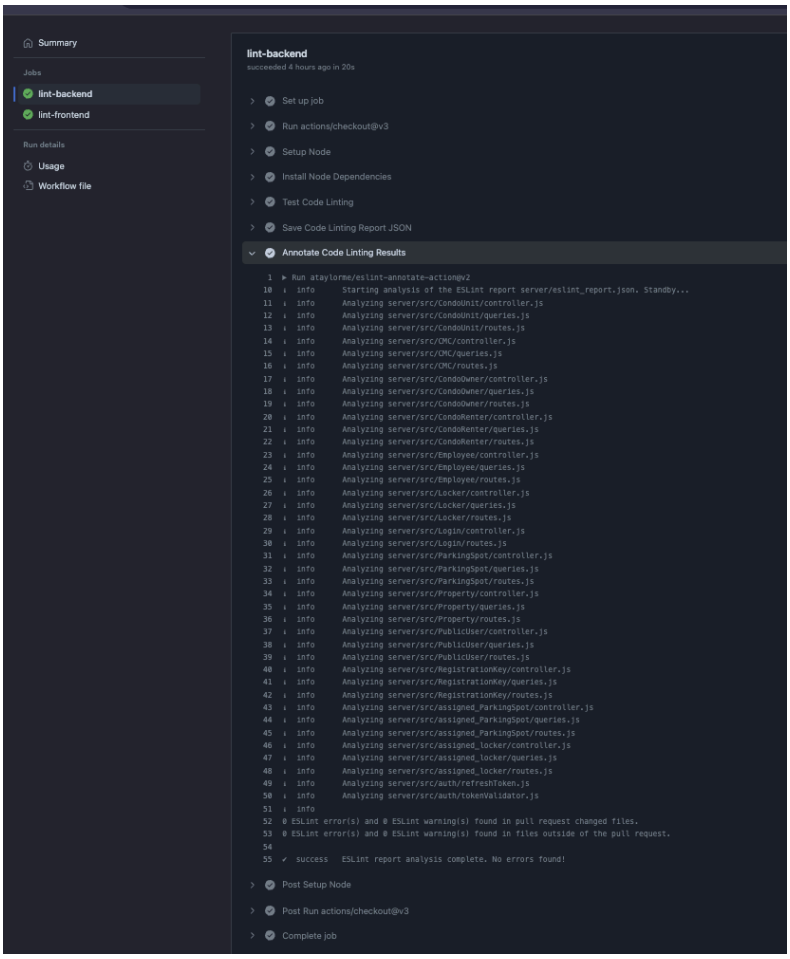
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	78.68	65.69	80.5	79.16	
server	100	100	100	100	
db.js	100	100	100	100	
server/src/CMC	83.33	74.35	92.3	84.26	
controller.js	82.14	74.35	92.3	83.13	46,61-62,74,105-118,135-136
queries.js	100	100	100	100	
server/src/CondoOwner	78.37	64.15	75	78.37	
controller.js	76.47	64.15	75	76.47	46-69,133-134,138,162-163,168-170
queries.js	100	100	100	100	
server/src/CondoRenter	79.81	66.03	81.25	79.81	
controller.js	78	66.03	81.25	78	47-69,132-133,137,161-162,168-169
queries.js	100	100	100	100	
server/src/CondoUnit	92.98	87.5	100	92.98	
controller.js	92	87.5	100	92	50-51,68-69
queries.js	100	100	100	100	
server/src/Employee	95.6	80	100	95.6	
controller.js	95.23	80	100	95.23	88,124,136,141
queries.js	100	100	100	100	
server/src/Login	79.16	57.14	57.14	80.85	
controller.js	79.16	57.14	57.14	80.85	47-59,68-69,83,88-89
server/src/Property	65.58	43.63	76.19	66.88	
controller.js	62.93	43.63	76.19	64.28	5-21,26,30-31,35-44,79,95,110,125,140-163,177,215-225,239-251
queries.js	100	100	100	100	
server/src/PublicUser	82.97	80	84.61	82.97	
controller.js	81.81	80	84.61	81.81	40-61,113-114,126
queries.js	100	100	100	100	
server/src/RegistrationKey	56.45	41.66	54.54	56.45	
controller.js	50	41.66	54.54	50	6,10-34,63-64,71-72,78-79
queries.js	100	100	100	100	

Github Actions/ESLint

In our development workflow, we've integrated ESLint with GitHub Actions to streamline automated testing and ensure code quality as part of our Continuous Integration (CI) and Continuous Deployment (CD) process. ESLint, a powerful static code analysis tool for identifying and fixing code inconsistencies, is configured within our GitHub Actions workflows to automatically check pull requests and commits against predefined coding standards and best practices. By leveraging GitHub Actions, each code change triggers ESLint checks, providing immediate feedback to developers, preventing potential issues, and maintaining code consistency across our projects. This integration not only enhances the reliability of our codebase but also accelerates the CI/CD pipeline by automating crucial code quality checks.



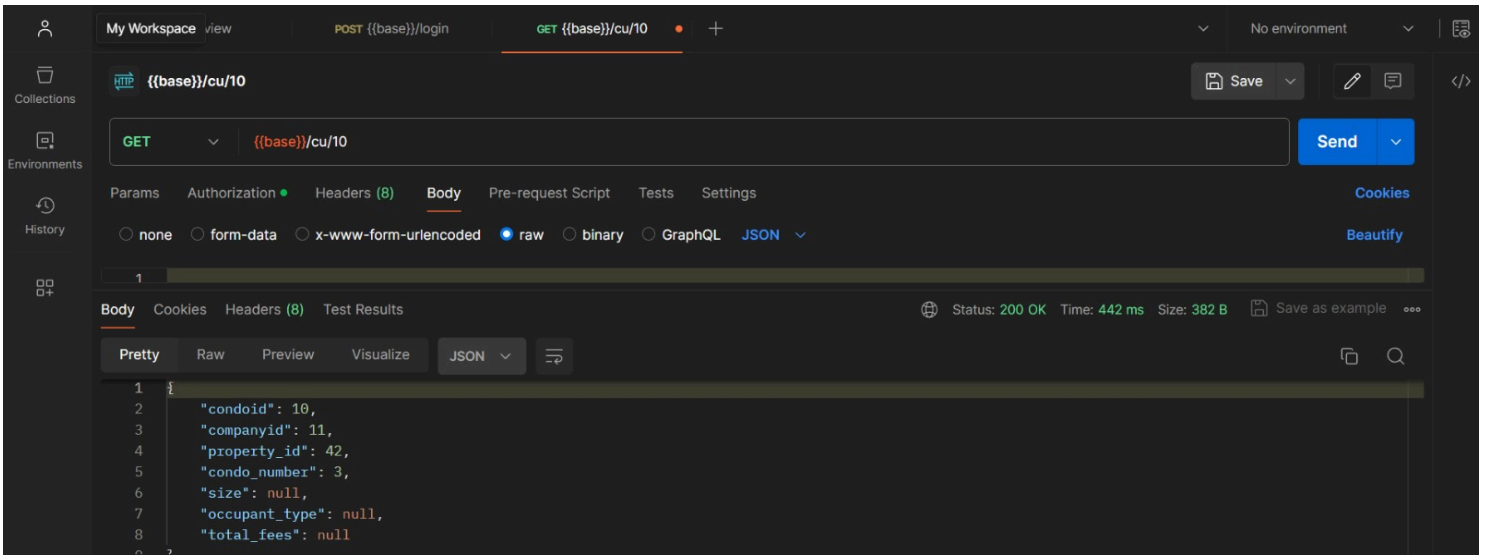




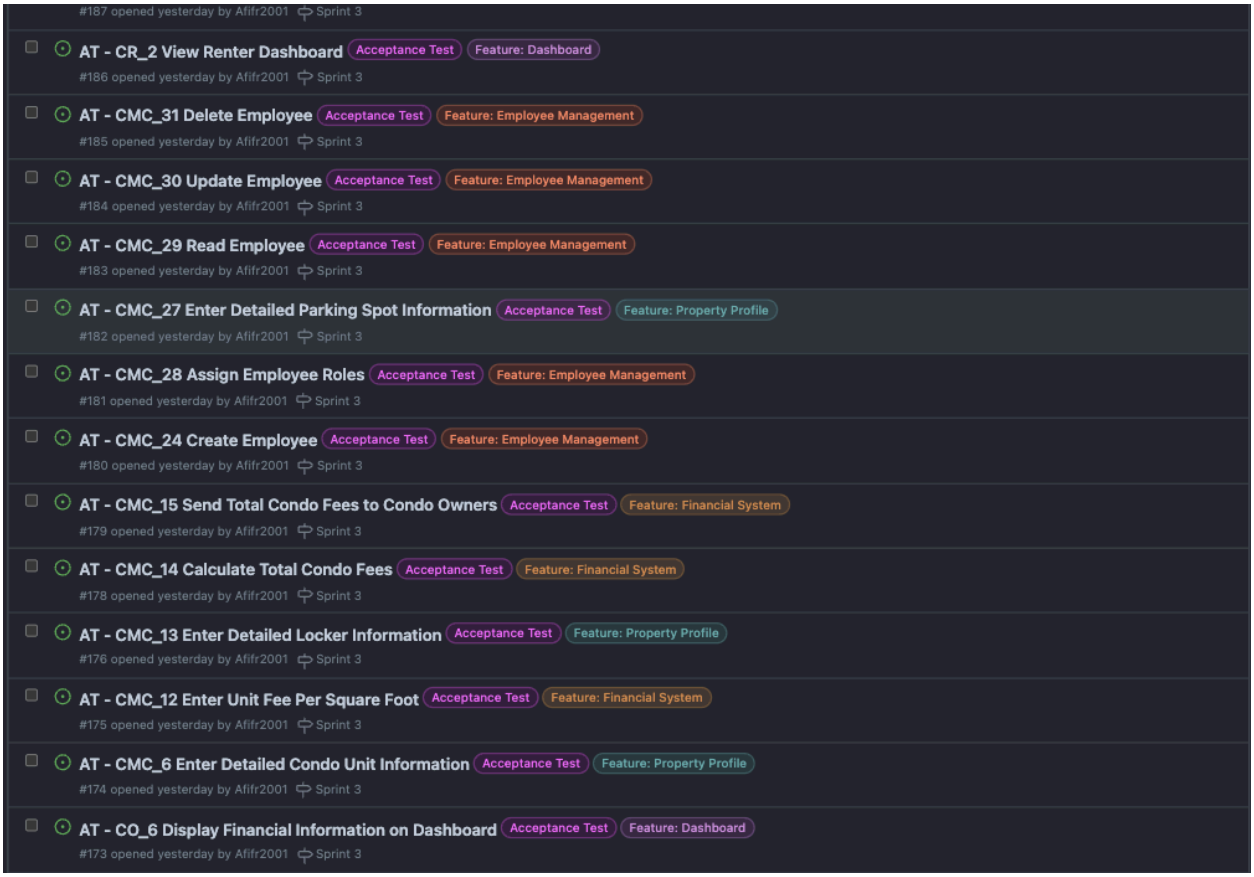
API Testing

Throughout the sprint, each developer who worked on their tasks individually did API testing for the components to test each endpoints and routes to work and is connected to the database in PgAdmin. The API Testing was done on Postman

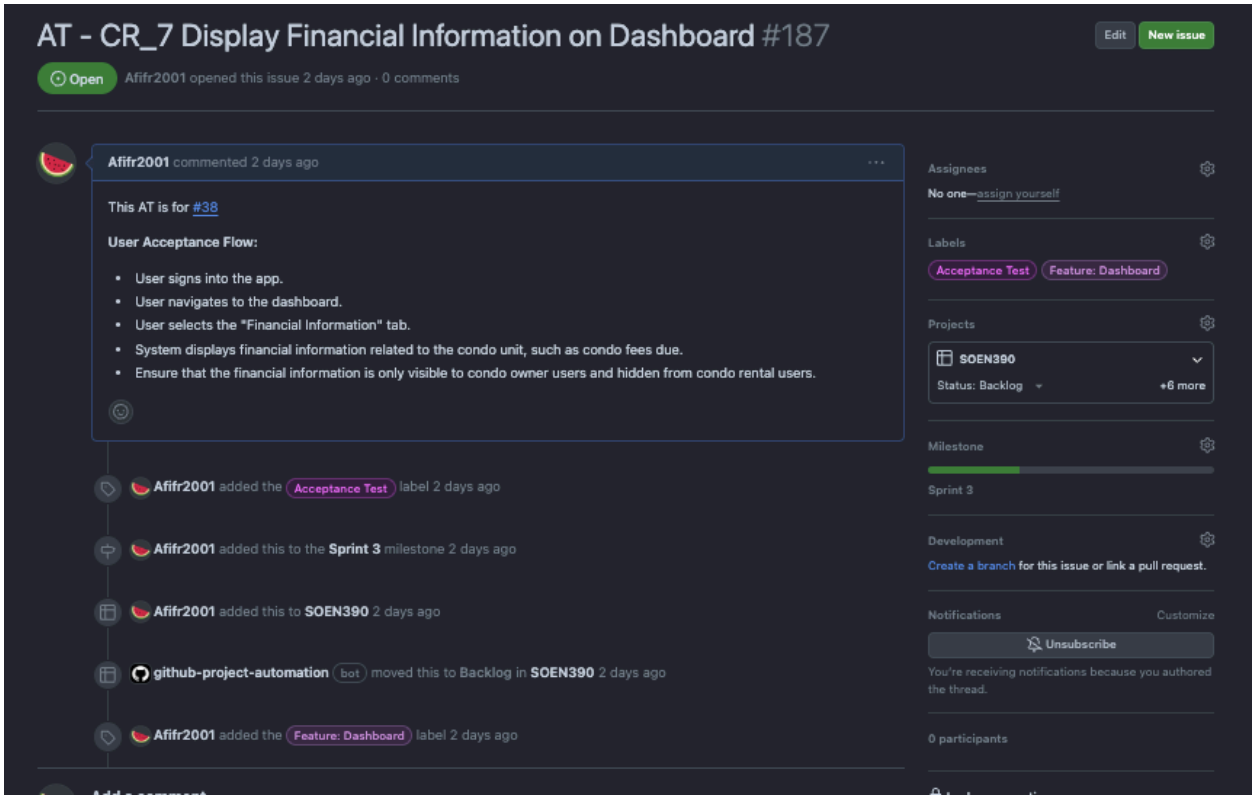
Here is an example testing the get endpoint for Condo Unit:



Acceptance Tests for Sprint 3



Here is an example of an Acceptance Test for this sprint



Chapter 6: Sprint Retrospectives

6.1 Sprint 1 Retrospective

From Blueprint to Balcony: Unveiling the Journey of our Condo Management System

6.1.1 Introduction

Our goal in creating a condo management system was to improve communication, simplify operations, and offer a strong platform for managers and rentals alike. The condo management system, an enterprise resource planning system (ERP), would consolidate duties like financial tracking, maintenance requests, and community involvement. There were high expectations because a user-friendly and effective system was to be delivered. This postmortem explores the successes and setbacks encountered during the development process, providing insight into the decisions made, resources employed, and overall architecture.

6.1.2 What Went Wrong?

6.1.2.1 Uncertain Initial Phase Needs:

What it means: Poor communication with the product manager and a disjointed approach to gathering project needs resulted in unclear requirements, which caused problems for the project early on.

Why did it happen? Insufficient methods for communication with the product manager and an unorganized approach to gathering requirements.

Impact: Extended development schedules, savings because of false presumptions, and possibly additional work.

How was it managed? As part of an improved communication strategy, regular check-ins and clarification meetings with the product manager were put into place.

What could have been made better? Create a requirements phase of own creation and make sure it is well documented and updated often.

6.1.2.2 Designs Over Ambiguities In Architecture:

What it means: The project ran into difficulties because team members had different ideas about how to interpret the architectural specifications, which resulted in disagreements over design choices.

Why did it occur? Divergent perspectives among team members regarding the architectural requirements of the project resulted in incompatible design decisions.

Impact: A less cohesive product, more debugging work, and a delayed development period.

In what way was it handled? conducted targeted architecture meetings, keeping precise documentation, and including every important team member in decision-making.

What could have been improved upon? At the beginning of the project, create a comprehensive architecture plan that includes explicit guidelines for making decisions.

6.1.2.3 Insufficient Risk Handling:

What it means: The project encountered difficulties because possible risks were not identified and addressed, which led to unanticipated issues during implementation.

Why did it happen? Unexpected problems during project execution could arise from a failure to identify and manage potential risks.

Impact: Unexpected challenges may cause project delays, increased expenses, and poor project results.

How was it managed? implemented a thorough risk management plan with proactive mitigation strategies and recurring risk assessments.

What could have been made better? During project planning, put in place a comprehensive risk identification process, and keep an eye on and update your risk mitigation plans on a regular basis.

6.1.2.4 Inconsistency In Documentation:

What it means: Misunderstandings brought on by team members' separate documentation styles restricted the project's advancement.

What caused it to occur? Team members misunderstood each other's interpretations and gaps because of inconsistent documentation practices.

Impact: Greater possibility of miscommunications that cause the project to diverge off course.

In what way was it handled? established a uniform documentation procedure and highlighted its value in team building exercises.

What could have been improved upon? To guarantee consistency, enforce stringent documentation requirements and carry out frequent comments.

6.1.2.5 Inadequate Testing Procedures:

What it means: Insufficient testing during the first sprint suggests that important problems were overlooked at an early stage, which would have increased workload and possibly caused user dissatisfaction. Although prompt remedial action was taken, many problems could have been avoided with careful planning and devoted testing resources.

What caused it to occur? The lack of thorough testing protocols resulted in problems and bugs going unnoticed in the later phases of development.

Impact: A weakened user experience, more post-release bug fixes, and possible user discontent.

In what way was it handled? a more comprehensive testing protocol was put into place, which included user acceptability, integration, and unit testing.

What could have been improved upon? At every stage of development, give testing top priority. For maximum effectiveness, incorporate automated testing tools.

6.1.3 What Went Right?

6.1.3.1 Successful In Person Meetings:

What led to it? Acknowledged the importance of in-person contacts in promoting cooperation, creativity, and a common knowledge of project objectives.

Impact: Better team cohesion, innovative problem-solving techniques, and a unified project vision.

In what way was it handled? arranged frequent face-to-face meetings and, when needed, used collaborative tools for online brainstorming.

What could have been improved upon? Strike a balance between face-to-face and remote collaboration to suit different types of individuals.

6.1.3.2 Task Division And Strategic Time Management:

Why did it occur? adopted agile approaches, dividing the work into doable sections and establishing reasonable deadlines.

Impact: Steady progress, avoidance of delays, and effective use of resources.

In what way was it handled? employed project management software to monitor assignments, due dates, and team members' workloads.

What could have been improved upon? Work division and time management techniques should be continuously improved to accommodate changing project requirements.

6.1.3.3 Detailed Project Requirement Visualization:

What caused it to occur? Developed diagrams of the project specifications to guarantee that all team members agree.

Result: reduced miscommunications and inconsistencies while offering a clear development roadmap.

In what way was it handled? used diagrams and flowcharts as visual aids when discussing requirements.

What could have been improved upon? Throughout the project, use feedback loops to iterate on visual representations and guarantee continued clarity.

6.1.3.4 Iterative Development Method:

Why did it happen? Because the project used an iterative development approach, it was possible to continuously improve it in response to user feedback and evolving requirements.

Impact: the program changed in response to user input, creating a final version that is more full of features and easy to use.

How was it dealt with? Iterative cycles were incorporated into the development process, and regular feedback loops were established to facilitate frequent reviews and modifications.
What might have been done better? The iterative process was efficient, but it could have been made even more efficient by using an automated feedback analysis tool and a more reliable feedback collection mechanism.

6.1.3.5 Usability And User Experience:

why did this happen? The group recognized that a user-friendly condo management system is essential to the success of the project, so they focused strongly on making it.

Impact: Positive user experience and higher user adoption were the outcomes of the final product's excellent usability.

How was it dealt with? Usability tests, user feedback, and iterative interface refinement were all part of the active involvement of user experience designers in the development process.

What might have been done better? Although successful, conducting more thorough beta testing with real users prior to the official release could have produced more insights and improved the user experience even more.

6.1.4 Conclusion

To sum up, the entire team learned a great deal during the development of our condo management system. The necessity of having precise requirements and unambiguous communication from the start of the project was one of the primary lessons learned. The difficulties faced during the project's unclear beginning highlighted the necessity of a thorough requirements gathering procedure to lay a strong foundation for the project.

Moreover, the successful elements such as face-to-face meetings, agile approaches, and ongoing skill development emphasized the value of cooperation, flexible project management, and

continual skill enhancement. These constructive lessons underscore the importance of cultivating a unified team culture, adopting agile principles, and allocating resources towards the ongoing development of team members.

Essentially, this postmortem functions as a guide for upcoming projects as well as a reflection on previous efforts. It reaffirms the group's dedication to streamlining procedures, enhancing intercommunication, and fostering a flexible and dynamic development environment. With all its successes and failures, the condo management system project has shaped our collective knowledge and helped us to embark on even more fruitful efforts in the rapidly changing field of software development.

6.2 Sprint 2 Retrospective

6.2.1 Introduction

During sprint 3, we moved our attention from the original plan to presenting the upgraded Condo Management System (CMS) journey. Expanding upon the insights gained from the first sprint, our goals were to tackle obstacles, improve correspondence, and make advances toward producing a superior CMS. This retrospective explores the main areas in which sprint 3 went well and those that require improvement.

6.2.2 What Went Wrong?

6.2.2.1 Integration Challenges:

What it means:During the integration phase, challenges emerged because of differences in the ways that components interacted, leading to unexpected issues.
Why did it happen? Various interpretations of the integration requirements resulted in inconsistent solutions that were put into practice.
Impact: Possible effects on system performance as a whole, extra debugging work, and integration delays.
Handling: Performed focused integration testing, recorded integration procedures, and convened with the team members to make decisions.
What could have been better? Create a complete integration plan in the beginning to anticipate and resolve any conflicts.

6.2.2.2 Changing Requirements Difficulties

What does it mean? The dynamic character of the requirements presented difficulties and prevented the team's capacity for quick adaptation.
What caused it to occur? an inability to adapt and challenges keeping up with the dynamic changes in the project.

Impact: Difficulties in aligning with changing goals may result in delays and increased workload.
Handling: Agile principles were incorporated and frequent requirement refinement sessions were held.
What could have been better? To improve visibility and increase adaptability, automated requirement tracking tools could be implemented.

6.2.2.3 Activating Prototype (User Profile, Sign up, Login)

What caused it to occur? The presentation and demo were impacted by the unfinished Sprint 1 functionalities.
Impact: Possible misalignment with product owner expectations and incomplete demonstration.
How was it managed? Prior to presentations, the team should make sure that all Sprint 1 functionalities are finished and take care of any remaining tasks.
What might have been improved upon? a more complete planning procedure to guarantee that all assigned functionalities are finished.

6.2.2.4 Testing

What caused it to occur? There was no separate testing file with detailed test cases and code coverage measurements.
Impact: A lack of clarity regarding the efficacy and completeness of testing.
How was it managed? For every sprint, the team should produce a specific testing file with comprehensive test cases and at least 80% code coverage.
What might have been improved upon? a better-organized procedure for documenting test cases and code coverage metrics in the documentation.

6.2.2.5 Risk Management Plan

What caused it to occur? There was unclear communication because the risk management plan and risk chart were not presented according to the template.
Impact: Potentially overlooked major risks and inefficient risk mitigation techniques.
How was it managed? The team needs to follow the template for the risk management plan and make sure that the risk chart has criteria for different risk levels.
What might have been improved upon? a complete risk management strategy that includes a well organized risk chart and precise standards for risk levels.

6.2.3 What Went Right?

6.2.3.1 Enhanced Frontend Usability:

What caused this to occur? With the addition of new features, the frontend went through significant improvements that become user friendly.

Impact: An interface that is easier to use and more intuitive, improving the user experience.

How was it resolved? User interface improvements, user feedback integration, and iterative frontend testing.

What might have been done better? The frontend's positive evolution promises accelerated development timelines by positioning it well for the quick integration of new features.

6.2.3.2 Ongoing Improvement of Documentation:

What caused its occurrence? Continuous improvement of documentation procedures to guarantee consistency and clarity.

As a result, there are fewer misunderstandings, better communication, and more seamless project progress.

How was it resolved? Regularly reviewed documentation and enforced strict documentation requirements.

What might have been done better?: Introduce automated consistency checks for documentation.

6.2.3.3 Proactively Identifying and Reducing Risk:

Why did it happen? application of a proactive strategy for risk assessment and reduction.

Impact: Potential risks are identified early and mitigated to reduce their influence on the project.

How was it resolved? Provided resources for proactive risk management, carried out frequent risk assessments, and kept an up-to-date risk register.

Improvement :aim to build upon the existing proactive risk management strategy, ensuring that the team stays ahead of potential challenges and continues to strengthen the project's resilience.

6.2.3.4 Strengthened Backend Functionality:

What caused this to occur? The team overcame obstacles to successfully support the database and backend functionalities, guaranteeing a strong base for the condo management system.

Impact: A more dependable and efficient system that opens the way for the addition of new features in the future.

How was it resolved? The team utilized agile methodologies, comprehensive testing, and cooperative problem-solving techniques to adjust to changing backend requirements.

What might have been done better?The knowledge acquired from achieving backend difficulties can be an important asset for upcoming sprints, facilitating more precise resource allocation and planning.

6.2.3.5 Comprehensive User Story Analysis:

What caused this to occur? Unexpected technical complexities resulted in some user stories being abandoned unfinished.

Impact: Potential gaps in user expectations and incomplete functionalities.

How was it resolved? During the sprint planning phase, conduct a more thorough analysis of user stories to better identify potential challenges and allocate resources.

What might have done better? During the sprint planning phase, analyze user stories in greater detail to better identify potential challenges and allocate resources more accurately.

6.2.4 Conclusion

sprint 3 highlighted the iterative nature of the development process by presenting both obstacles and achievements. The group views obstacles as chances for development and is dedicated to ongoing improvement. The team's subsequent sprints will be guided by the lessons learned from Sprint 1, which will emphasize the value of flexibility, clear communication, and proactive risk management in attaining project success. The team's collective knowledge is still being shaped by the condo management system project, which promotes a resilient and dynamic development environment. Moreover, the project is in a position to develop more quickly and integrate new features easily in the upcoming sprints due to the implementation of the backend and the improvement of frontend usability.

6.3 Sprint 3 Retrospective

6.3.1 Introduction

We continued to work on improving the Condo Management System (CMS) in Sprint 3 using the knowledge we had gained from earlier sprints. The purpose of this retrospective is to assess the

sprint's accomplishments and difficulties, with an emphasis on ongoing development and efficient CMS delivery.

6.3.2 What Went Wrong?

6.3.2.1 Incomplete User Stories

What does it mean? Some user stories remained unfinished, impacting the completeness of the sprint's deliverables.
Cause: Insufficient allocation of resources or unforeseen technical complexities.
Impact: Potential delays and gaps in functionality.
Handling: Conduct a detailed analysis of user stories during sprint planning to accurately estimate resources and identify potential challenges.
What might have been done better? Improve estimation techniques by involving team members with relevant expertise, and implement a buffer for unexpected technical challenges.

6.3.2.2 Inadequate Frontend Testing

Description: Frontend testing may have been insufficiently prioritized, leading to potential usability issues.
Cause: Lack of emphasis on frontend testing procedures.
Impact: Reduced user satisfaction and increased post-release bug fixes.
Handling: Prioritize frontend testing alongside other testing protocols to ensure comprehensive coverage.
What might have been done better? Implement automated frontend testing tools to streamline the testing process and increase coverage.

6.3.2.3 Challenges in Code Management

Description: Issues related to code management practices, such as quality of source code reviews and adherence to design patterns.
Cause: Inconsistent code review standards and deviations from design principles.
Impact: Reduced code quality and maintainability.
Handling: Document and enforce robust code review processes and design patterns to improve code quality and consistency.
What might have been done better? Provide training sessions on code review best practices and design patterns to ensure alignment across the team.

6.3.2.4 Suboptimal Testing Plan

Description: The testing plan may have lacked clarity regarding testing tools, metrics, and coverage.
Cause: Inadequate planning and documentation of testing procedures.
Impact: Uncertainty about testing efficacy and completeness.

Handling: Develop a comprehensive testing plan outlining testing tools, metrics, coverage, and acceptance tests for thorough testing.

6.3.3 What Went Right?

6.3.3.1 Improved UI Prototypes

Description: Enhanced UI prototypes for Sprint 3 user stories.
Impact: Increased usability and user satisfaction.
Handling: Developed a variety of prototypes, focusing on usability and comprehensive coverage of user stories.
What might have been done better? Conduct user feedback sessions earlier in the sprint to iterate on UI prototypes and incorporate user preferences more effectively.

6.3.3.2 Effective Release Planning

Description: Strategic release planning for Sprint 3.
Impact: Streamlined development process and clear roadmap for delivering features.
Handling: Breakdown sub-user stories, estimate user stories, and prioritize tasks for efficient sprint execution.
What might have been done better? Improve estimation accuracy by analyzing historical data from previous sprints and adjusting estimates accordingly.

6.3.3.3 Continued Refinement of Documentation

Description: Ongoing enhancement of documentation procedures for clarity and consistency.
Impact: Improved communication and seamless project progress.
Handling: Regularly review and enforce strict documentation requirements to maintain consistency.

6.3.4 Conclusion

Sprint 3 represented a new phase in the CMS's iterative development, complete with obstacles as well as achievements. We are dedicated to resolving the issues that have been found to need improvement going forward and using the knowledge gained to strengthen our development procedures even more. Our goal is to offer a robust and user-friendly CMS that satisfies the objectives of our stakeholders by placing a high priority on careful planning, extensive testing, and effective communication.

Chapter 7: Release Plan

7.1 Sprint 3 Release Plan

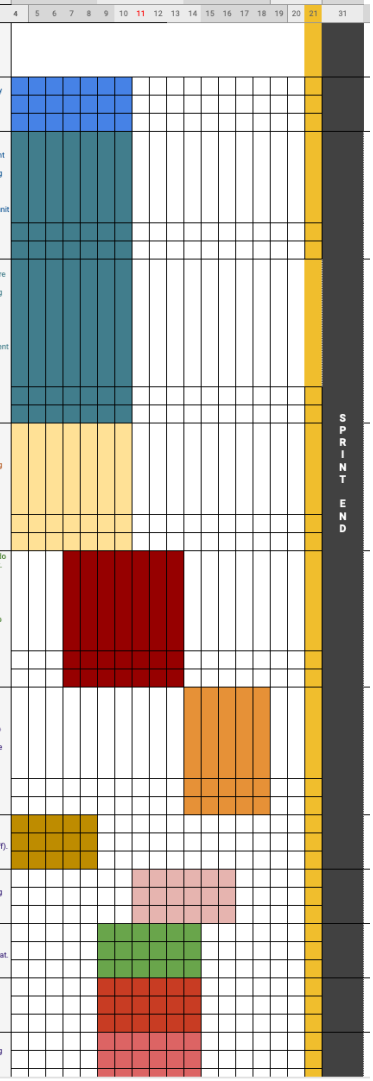
Our Sprint 3 release plan can be viewed here Copy of Sprint3_Project_timeline_Butler

SPRINT 3

PROJECT TITLE	Butler	DATE CREATED	2/10/24
SPRINT LENGTH	21 days 17 Days Left	DUE DATE	3/21/24

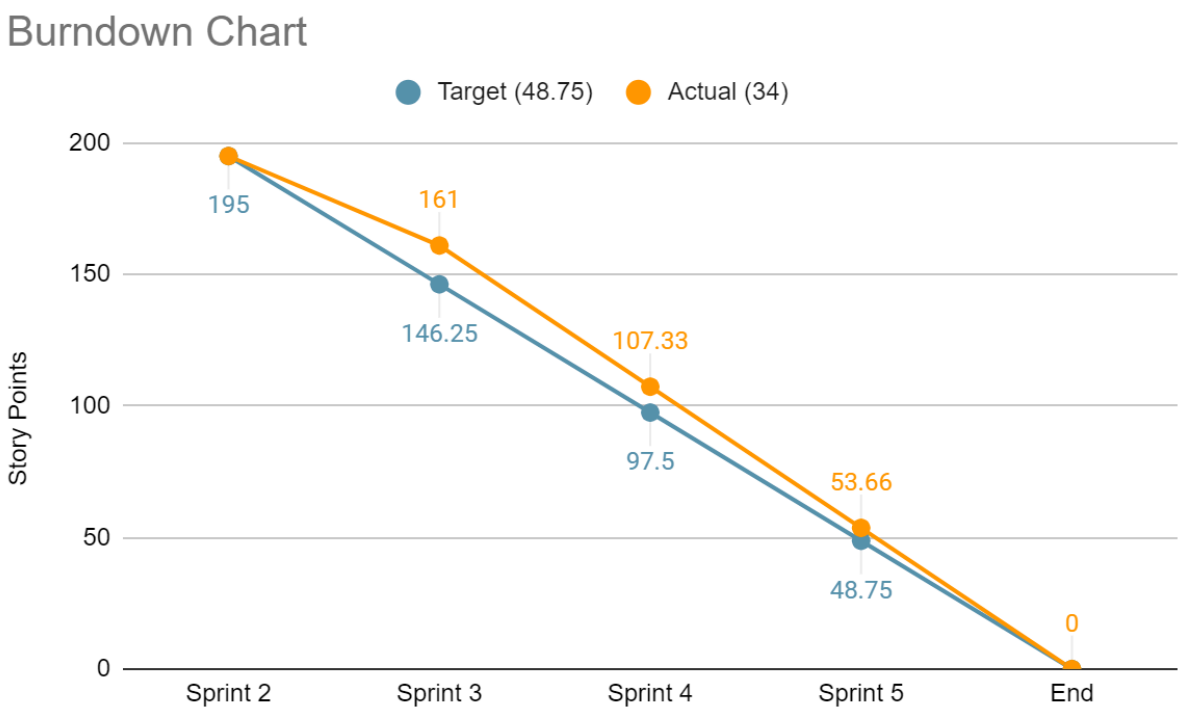
STORIES LINK:
https://docs.google.com/docume

		RESPONSIBLE		DETAILS
	PROJECT WEEK:	TO DO	Assignee	Job Description
	Branch Creation convention **storyName_Number_SprintNumber -> PU_1_Sprint1** **Link to User Stories:**			
	CO_6 Display Financial Information on Dashboard	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	YOussef Nicolas Wadeh	Check Updates: The dashboard should provide updates on any changes or important events related to the user's properties, such as maintenance schedules or property tax updates. Verify Information Accuracy: The user should be able to verify
1	**CMC_6 Enter Detailed Condo Unit Information**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef Nicolas Wadeh	Create a form or interface for entering detailed information about condo units. Include fields for key details such as unit number, owner/tenant information, size, layout, and any special features. Ensure the form/interface allows for easy editing and updating of condo unit information. Implement a database or storage system to store the entered condo unit information securely. Provide a way to view and search through the entered condo unit information for management purposes.
2	**CMC_12 Enter Unit Fee Per Square FootUnit**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Monica Nicolas Daniel	Create a form or interface for entering the condo fee per square foot. Ensure the form/interface allows for easy editing and updating of the fee. Implement a storage system to store the entered fee information securely. Provide a way to retrieve and display the fee information for condo owners. Consider integrating the fee information with other management systems or documents for easy access and communication.
3	**CMC_13 Enter Detailed Locker Information**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Charles Nicolas Afif	Create a form or interface for entering detailed information about lockers. Include fields for key details such as locker number, location, size, and any special features. Ensure the form/interface allows for easy editing and updating of locker information. Implement a storage system to store the entered locker information securely. Provide a way to view and search through the entered locker information for management purposes.
4	**CMC_14 Calculate Total Condo Fees**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	YOussef Charles Daniel	Implement a feature to calculate the total condo fees per condo unit based on the fee per square foot and the size of each unit. Ensure the calculation is accurate and takes into account any special circumstances or adjustments. Provide a way to view the total fees for each unit, possibly in a report or summary format. Consider integrating this feature with other management systems or documents for easy access and communication to condo owners. Ensure the privacy and security of fee information when displaying it to condo owners.
5	**CMC_15 Send Total Condo Fees to Condo Owners**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Monica Nicolas Khalil	Generate a report or statement showing the total condo fees due for each condo unit. Ensure the report includes detailed breakdowns if applicable (e.g., base fee, additional charges). Implement a method to securely distribute the report to condo owners (e.g., email, online portal). Provide clear instructions on how and when the fees should be paid. Ensure the accuracy of the information and provide a way for condo owners to verify their fees if needed.
6	**CMC_24 Create Employee**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Nicolas Nicolas Khalil	Include fields for key employee information such as name, contact details, and role. Implement a role management system to define and assign different roles to employees (e.g., administrator, manager, staff). Provide functionality to assign requests or tasks to specific employees based on their roles.
7	**CMC_27 Enter Detailed Parking Spot Information**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Houssam Nicolas Cey	Include fields for key details such as parking spot number, location, size, and any special features. Ensure the form/interface allows for easy editing and updating of parking spot information. Implement a storage system to store the entered parking spot information securely.
8	**CMC_28 Assign Employee Roles**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Houssam Nicolas Afif	As a condo management company, I want to assign different employee roles, so that condo owners/renters know who is responsible for what.
9	**CMC_29 Read Employee**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Houssam Cey	Ensure the list includes key employee information such as name, contact details, and role. Provide functionality to view and edit employee information if necessary. Consider implementing filters or search functionality to easily find specific employees.
9	**CMC_30 Update Employee**	FRONTEND: TODO BACKEND: TODO	Houssam Nicolas	Include fields for key employee details such as name, contact details, and role. Ensure the form/interface allows for easy editing and updating of employee information. Implement validation to ensure the accuracy of the updated



7.2 Cumulative Burndown Chart

Cumulative Burndown Chart



For Sprint 3, our target velocity was 48.75 Story Points. Our actual velocity was 34. This gives us an adjusted velocity of 53.66 for Sprint 3 through 5 to bring us back on track.

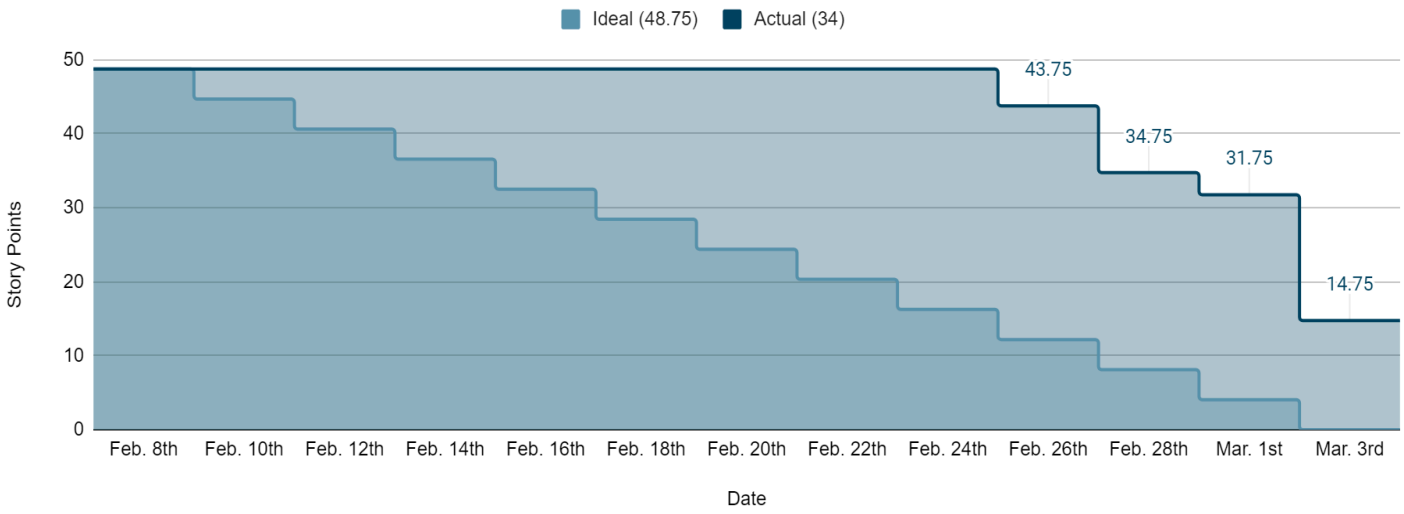
7.3 Sprint 3 Burndown Chart

Sprint 3 Burndown Chart

7.3.1 Implemented Use Cases

For Sprint 3, the following Use Cases were implemented:

- Displaying financial information on dashboard for condo owner
- Condo management companies can enter detailed information for a condo unit



- Condo management company can enter unit fee per square foot for a condo
- Condo management company can enter information regarding lockers
- Condo management company can calculate total condo fees
- Condo management company can send the total condo fees to their respective owner
- Condo management company can perform all operations on an employee (Create, Read, Update, Delete)
- Condo management company can enter detailed information for a parking spot
- Condo renters can view their dashboard
- Condo renters can see their financial information

7.4 Sprint 4 Release Plan

Our Sprint 4 Release Plan can be viewed Sprint4_Project_timeline_Butler.

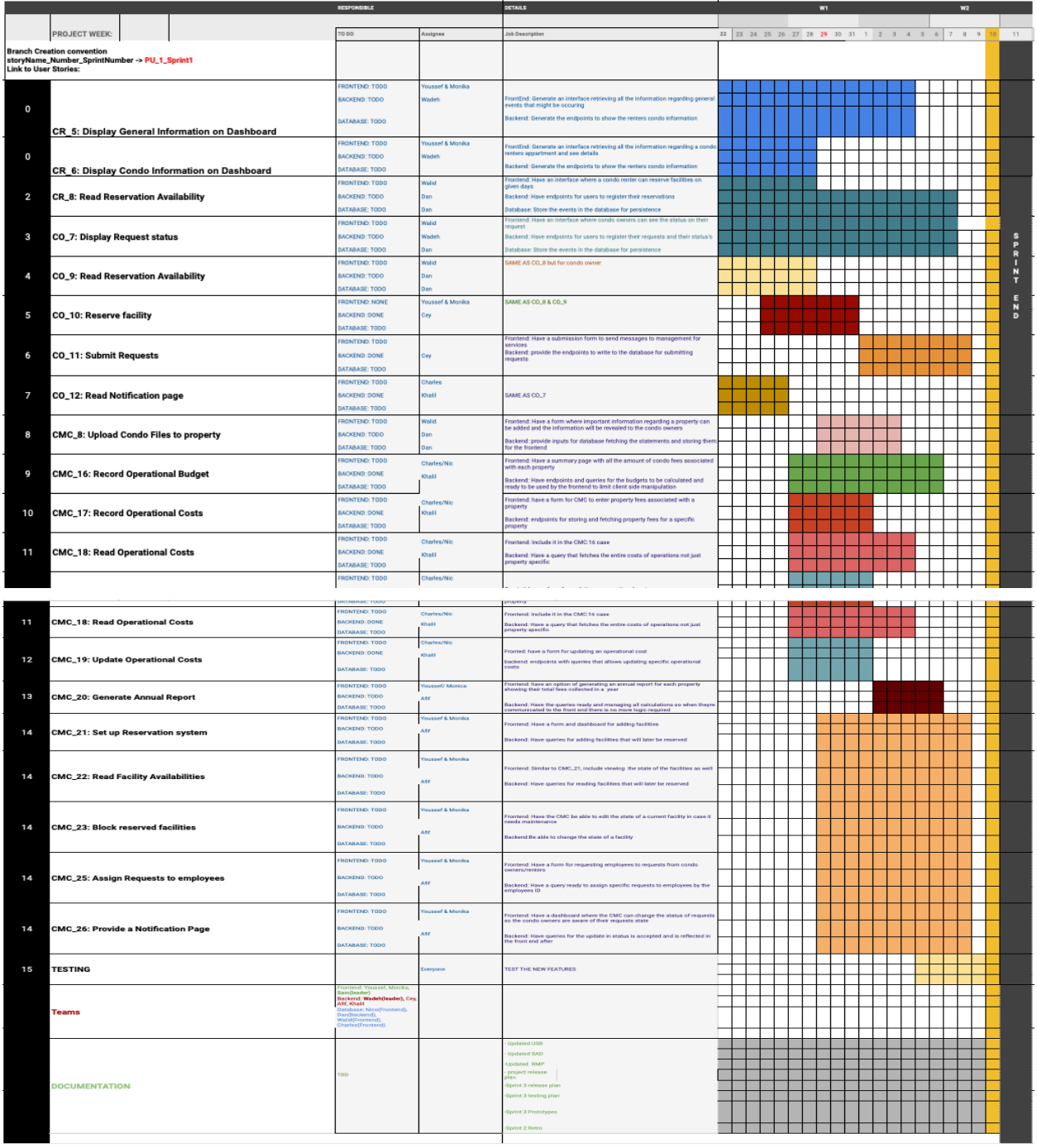
Sprint 4 Release Plan Sample

PRINT 4

PROJECT TITLE	Butler
SPRINT LENGTH	21 days — 21 Days Left
DATE CREATED	3/22/24
DUE DATE	4/11/24

STORIES LINK:
https://docs.google.com/docume

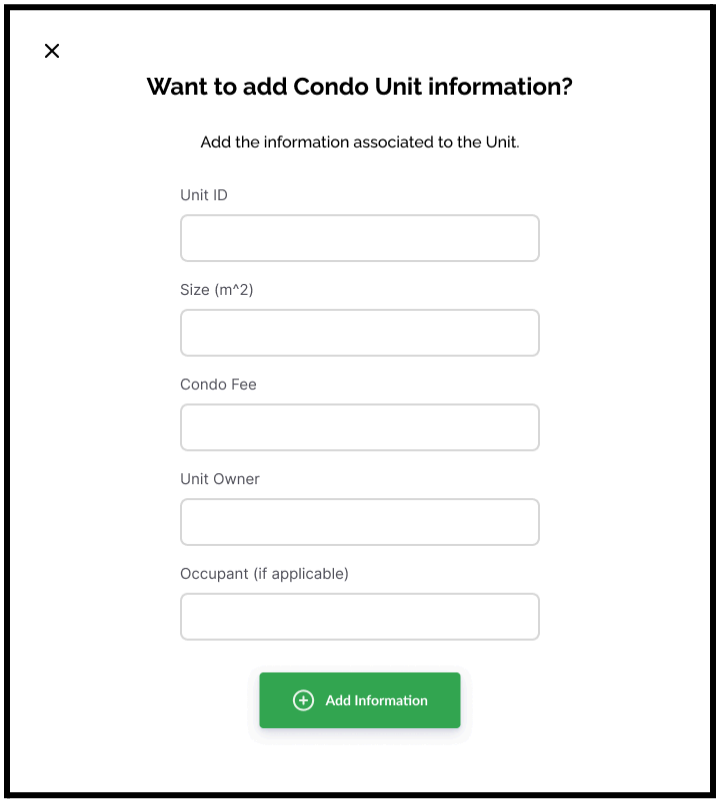
	PROJECT WEEK:	RESPONSIBLE: TO DO	RESPONSIBLE: Assignee	DETAILS: Job Description
	Branch Creation convention **storyName_Number_SprintNumber -> PU_1_Sprint1** **Link to User Stories:**			
0	**CR_5: Display General Information on Dashboard**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Wadeh	FrontEnd: Generate an interface retrieving all the information regarding general events that might be occuring Backend: Generate the endpoints to show the renters condo information
0	**CR_6: Display Condo Information on Dashboard**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Wadeh	FrontEnd: Generate an interface retrieving all the information regarding a condo renters appartment and see details Backend: Generate the endpoints to show the renters condo information
2	**CR_8: Read Reservation Availability**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Walid Dan Dan	Frontend: Have an interface where a condo renter can reserve facilities on given days Backend: Have endpoints for users to register their reservations Database: Store the events in the database for persistence
3	**CO_7: Display Request status**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Walid Wadeh Dan	Frontend: Have an interface where condo owners can see the status on their request Backend: Have endpoints for users to register their requests and their status's Database: Store the events in the database for persistence
4	**CO_9: Read Reservation Availability**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Walid Dan Dan	SAME AS CO_8 but for condo owner
5	**CO_10: Reserve facility**	FRONTEND: NONE BACKEND: DONE DATABASE: TODO	Youssef & Monika Cey	SAME AS CO_8 & CO_9
6	**CO_11: Submit Requests**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Cey	Frontend: Have a submission form to send messages to management for services Backend: provide the endpoints to write to the database for submitting requests
7	**CO_12: Read Notification page**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles Khalil	SAME AS CO_7
8	**CMC_8: Upload Condo Files to property**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Walid Dan Dan	Frontend: Have a form where important information regarding a property can be added and the information will be revealed to the condo owners Backend: provide inputs for database fetching the statements and storing them for the frontend
9	**CMC_16: Record Operational Budget**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles/Nic Khalil	Frontend: Have a summary page with all the amount of condo fees associated with each property Backend: Have endpoints and queries for the budgets to be calculated and ready to be used by the frontend to limit client side manipulation
10	**CMC_17: Record Operational Costs**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles/Nic Khalil	Frontend: have a form for CMC to enter property fees associated with a property Backend: endpoints for storing and fetching property fees for a specific property
11	**CMC_18: Read Operational Costs**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles/Nic Khalil	Frontend: Include it in the CMC16 case Backend: Have a query that fetches the entire costs of operations not just property specific
		FRONTEND: TODO	Charles/Nic	
11	**CMC_18: Read Operational Costs**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles/Nic Khalil	Frontend: Include it in the CMC16 case Backend: Have a query that fetches the entire costs of operations not just property specific
12	**CMC_19: Update Operational Costs**	FRONTEND: TODO BACKEND: DONE DATABASE: TODO	Charles/Nic Khalil	Frontend: have a form for updating an operational cost backend: endpoints with queries that allowe updating specific operational costs
13	**CMC_20: Generate Annual Report**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef/ Monica Atif	Frontend: have an option of generating an annual report for each property showing their total fees collected in a year Backend: Have the queries ready and managing all calculations so when theyre communicated to the front end there is no more logic required
14	**CMC_21: Set up Reservation system**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Atif	Frontend: Have a form and dashboard for adding facilities Backend: Have queries for adding facilities that will later be reserved
14	**CMC_22: Read Facility Availabilities**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Atif	Frontend: Similar to CMC_21, include viewing the state of the facilities as well Backend: Have queries for reading facilities that will later be reserved
14	**CMC_23: Block reserved facilities**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Atif	Frontend: Have the CMC be able to edit the state of a current facility in case it needs maintenance Backend:Be able to change the state of a facility
14	**CMC_25: Assign Requests to employees**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Atif	Frontend: Have a form for requesting employees to requests from condo owners/renters Backend: Have a query ready to assign specific requests to employees by the employees ID
14	**CMC_26: Provide a Notification Page**	FRONTEND: TODO BACKEND: TODO DATABASE: TODO	Youssef & Monika Atif	Frontend: Have a dashboard where the CMC can change the status of requests so the condo owners are aware of their requests state Backend: Have queries for the update in status is accepted and is reflected in the front end after
15	**TESTING**		Everyone	TEST THE NEW FEATURES
	Teams	Frontend: Youssef, Monika, Sam(leader) Backend: **Wadeh(leader)**, Cey, Atif, Khalil Database: Nic(Frontend), Dan(Backend), Walid(Backend), Charles(Frontend)		
	DOCUMENTATION	TBD		- Updated USB - Updated SAD -Updated RMP - project release plan -Sprint 3 release plan -Sprint 3 testing plan -Sprint 3 Prototypes -Sprint 2 Retro



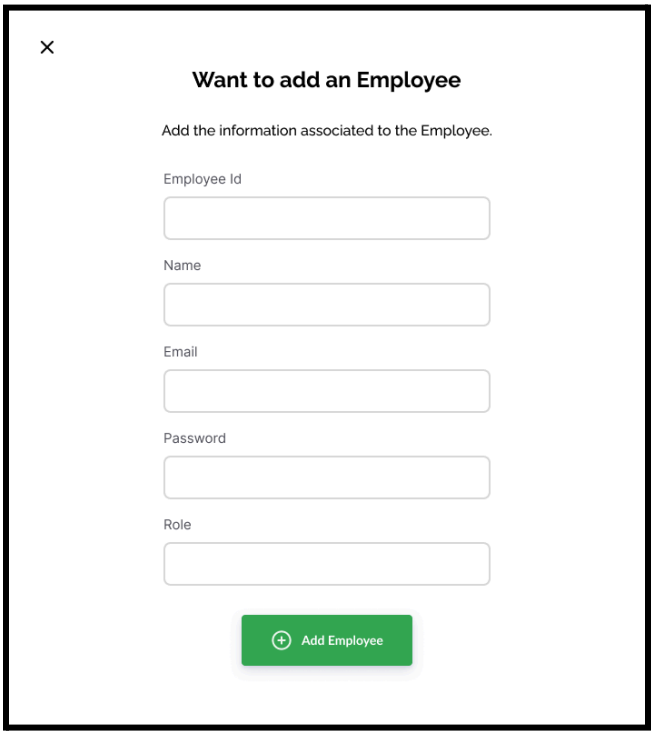
Chapter 8: UI Prototypes

8.1 Sprint 3 UI Prototypes

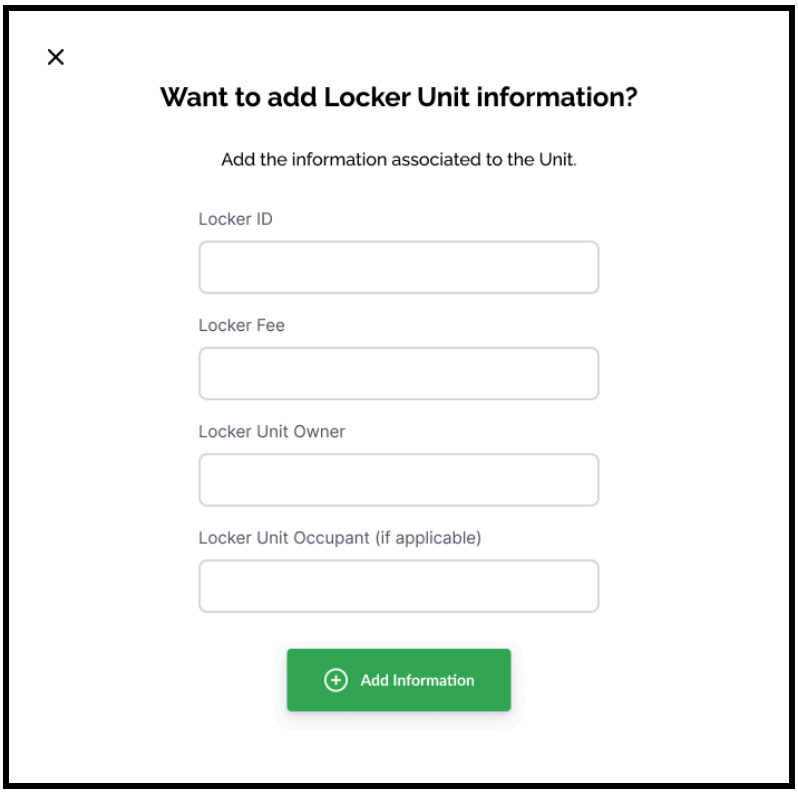
Add - Condo Unit Form Modal



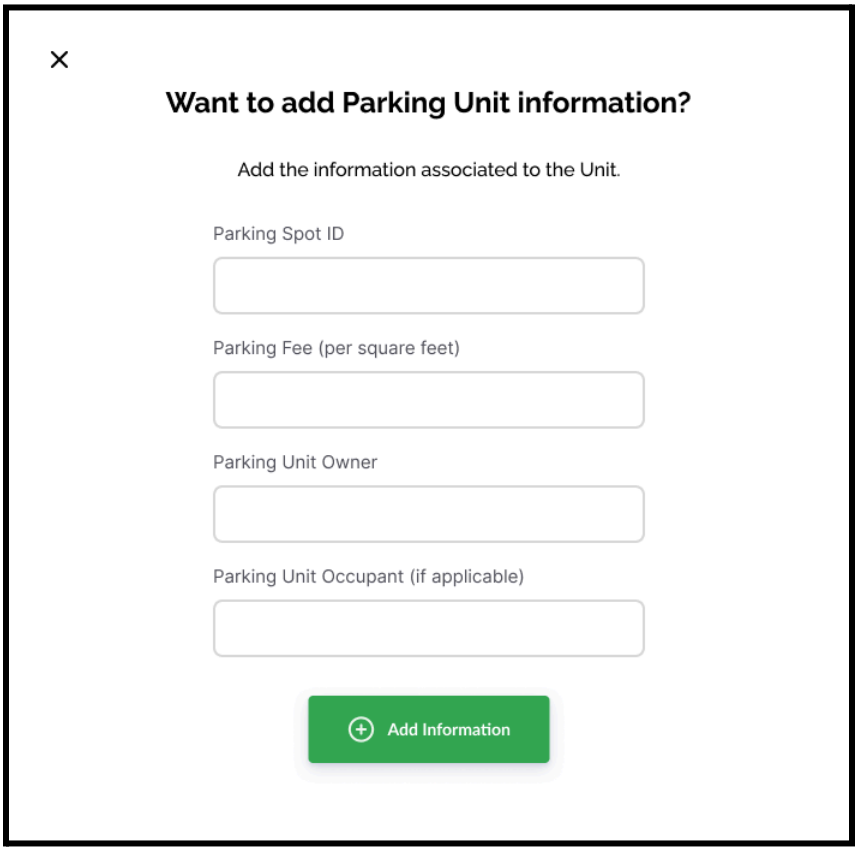
Add - Employee Unit Form Modal



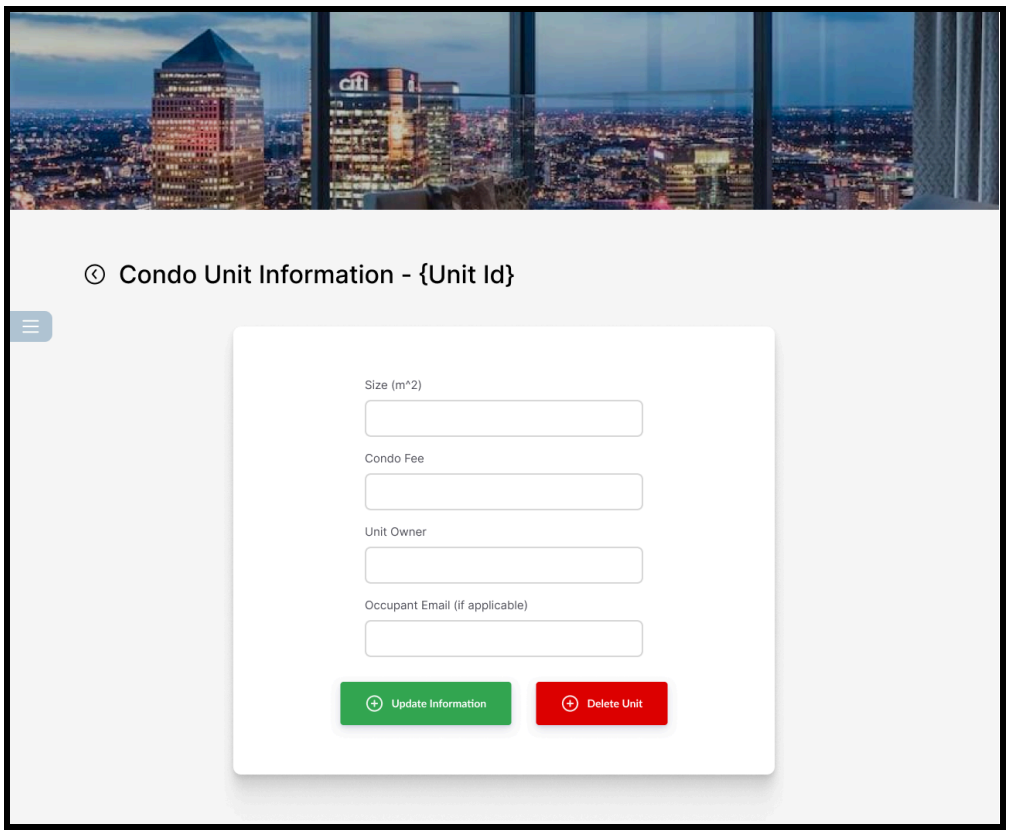
Add - Locker Unit Form Modal



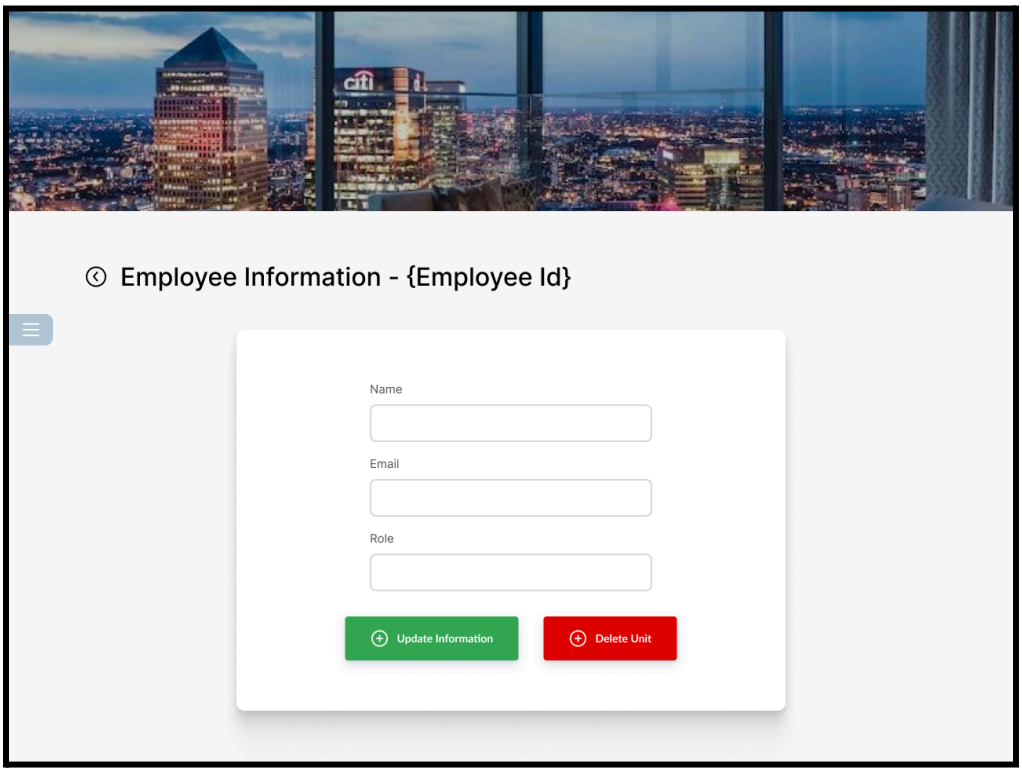
Add - Parking Unit Form Modal



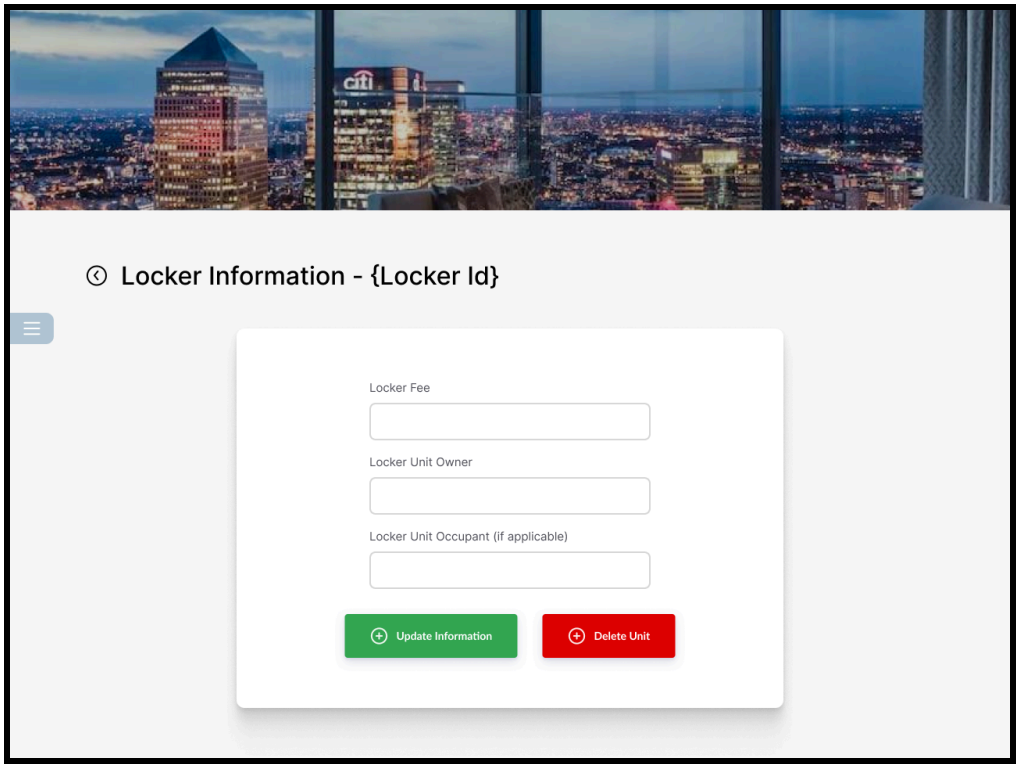
Desktop - CMC - Condo Unit Information



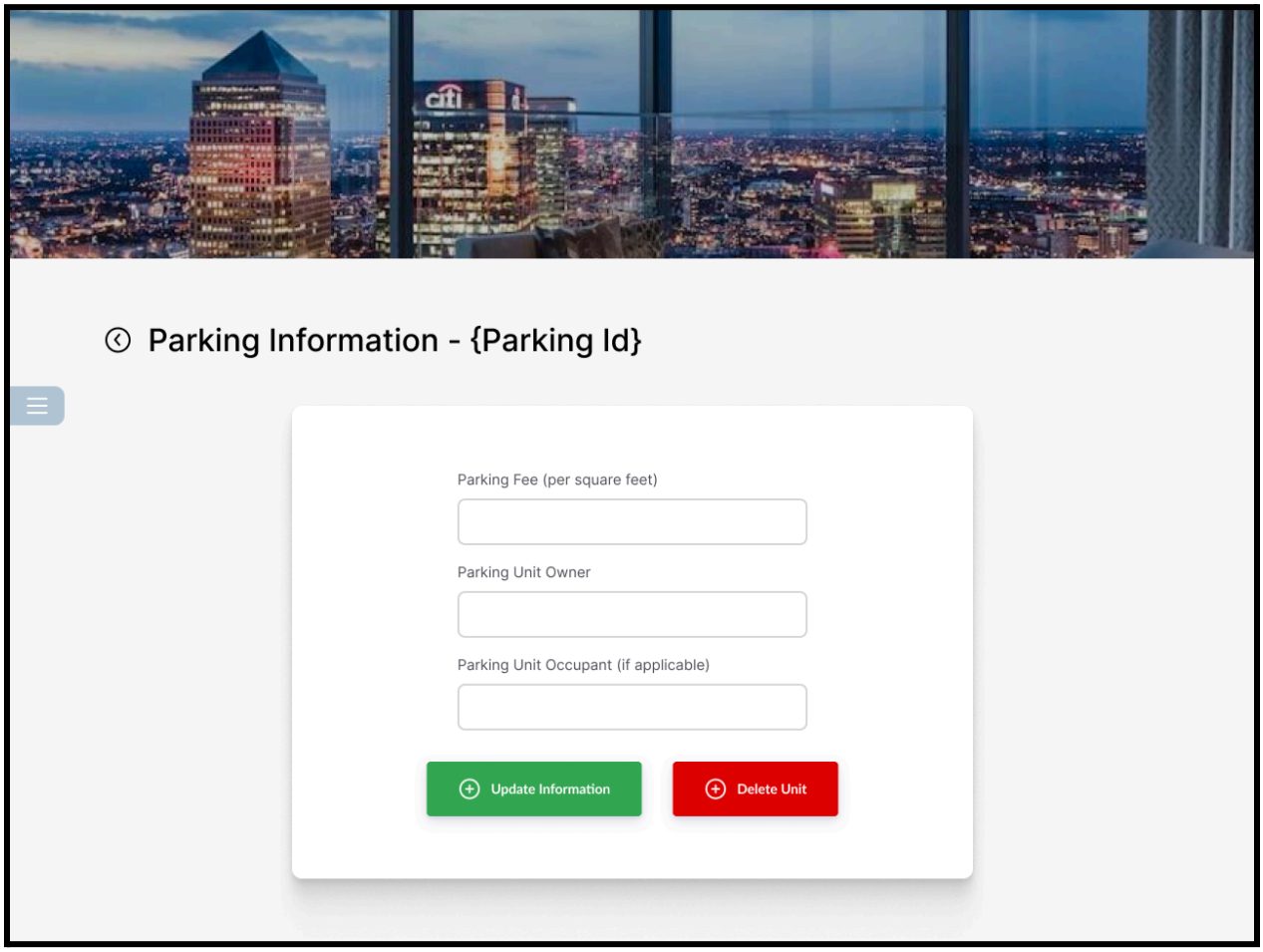
Desktop - CMC - Employee Information



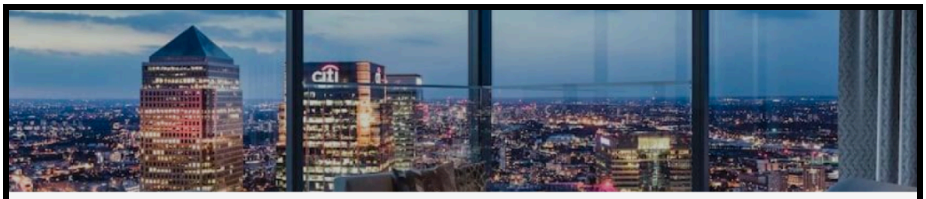
Desktop - CMC - Locker Unit Information



Desktop - CMC - Parking Unit Information



Desktop - CMC - Property Profile



Property Details - {Property_name}

Property Information

Property Name:	{Property_name}
Property Address:	{Property_address}
Number of Condo Units:	{Property_condo_total}
Number of Parking Units:	{Property_parking_total}
Number of Locker Units:	{Property_locker_total}

My Condo Units

See more | Add Condo Unit

Unit ID	Size	Unit Fee	Unit Owner	Unit Occupant
202	300	1200	John Cena	John Cena
111	450	1800	Blue Yeti	Blue Yeti
122	450	1800	Mister Clean	Mister Clean

My Parking Units

See more | Add Parking Unit

Unit Id	Unit Fee	Unit Owner	Unit Occupant
0023	112	John Cena	John Cena
0042	131	Blue Yeti	Blue Yeti
0112	131	Mister Clean	Mister Clean

My Locker Units

See more | Add Locker Unit

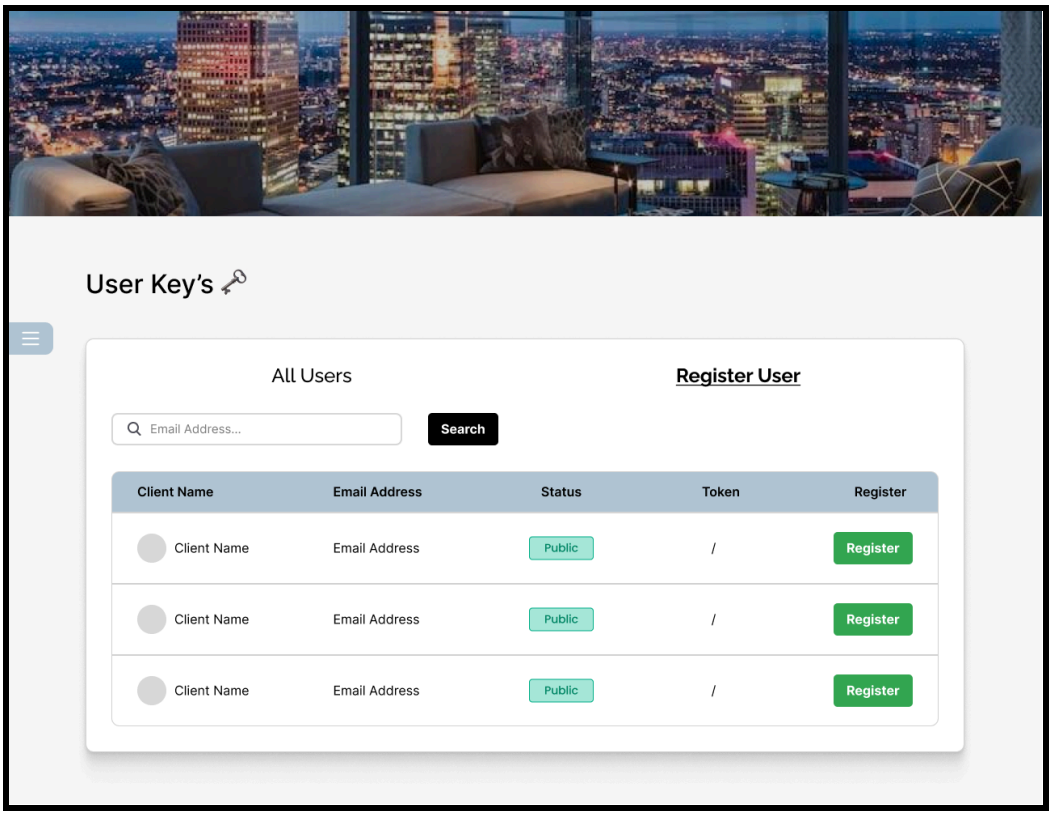
Unit Id	Unit Fee	Unit Owner	Unit Occupant
1233	80	Cristiano Ronaldo	Cristiano Ronaldo
0122	90	Eren Yaeger	Mikasa Akerman
7702	80	Mohammed Salah	Mohammed Salah

My Employees

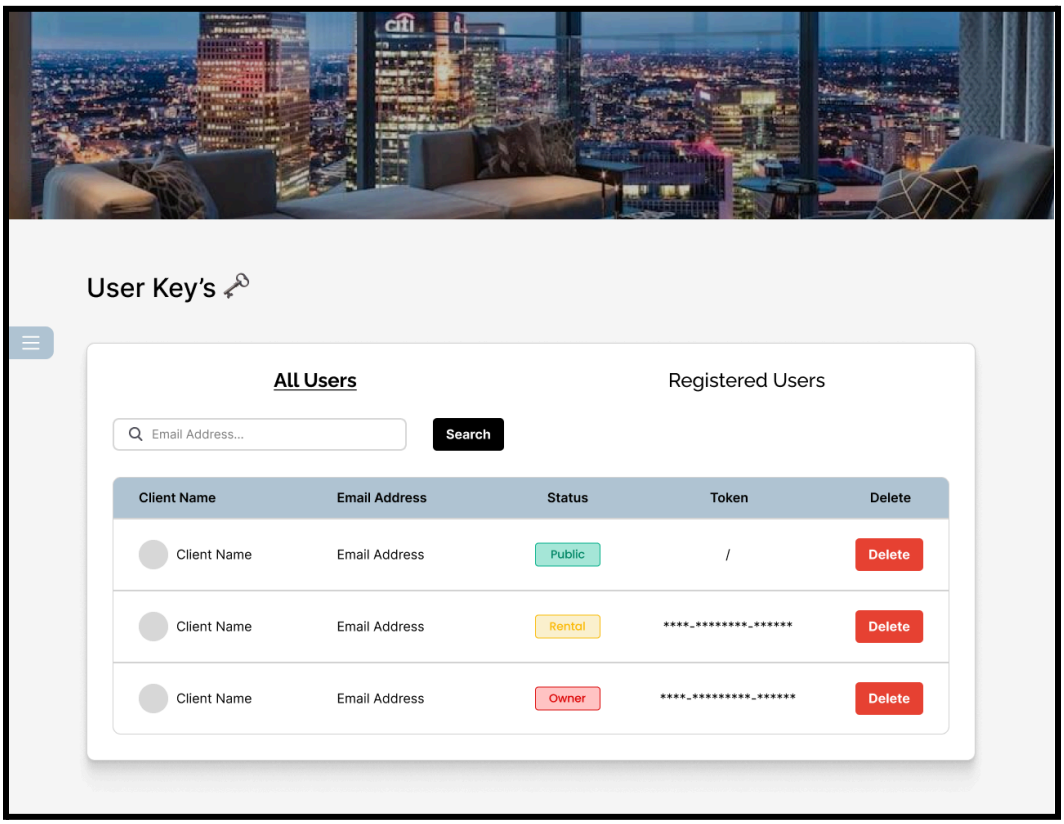
See more | Add Employee

Employee Id	Name	Email	Role
7563765	Baghdad Boujah	b.bounjah@hotmail.dz	Finance Director
4312542	Hakim Ziyech	hakim@gmail.com	Operation Director
0987634	Yacine Bono	yacine@yahoo.fr	Manager

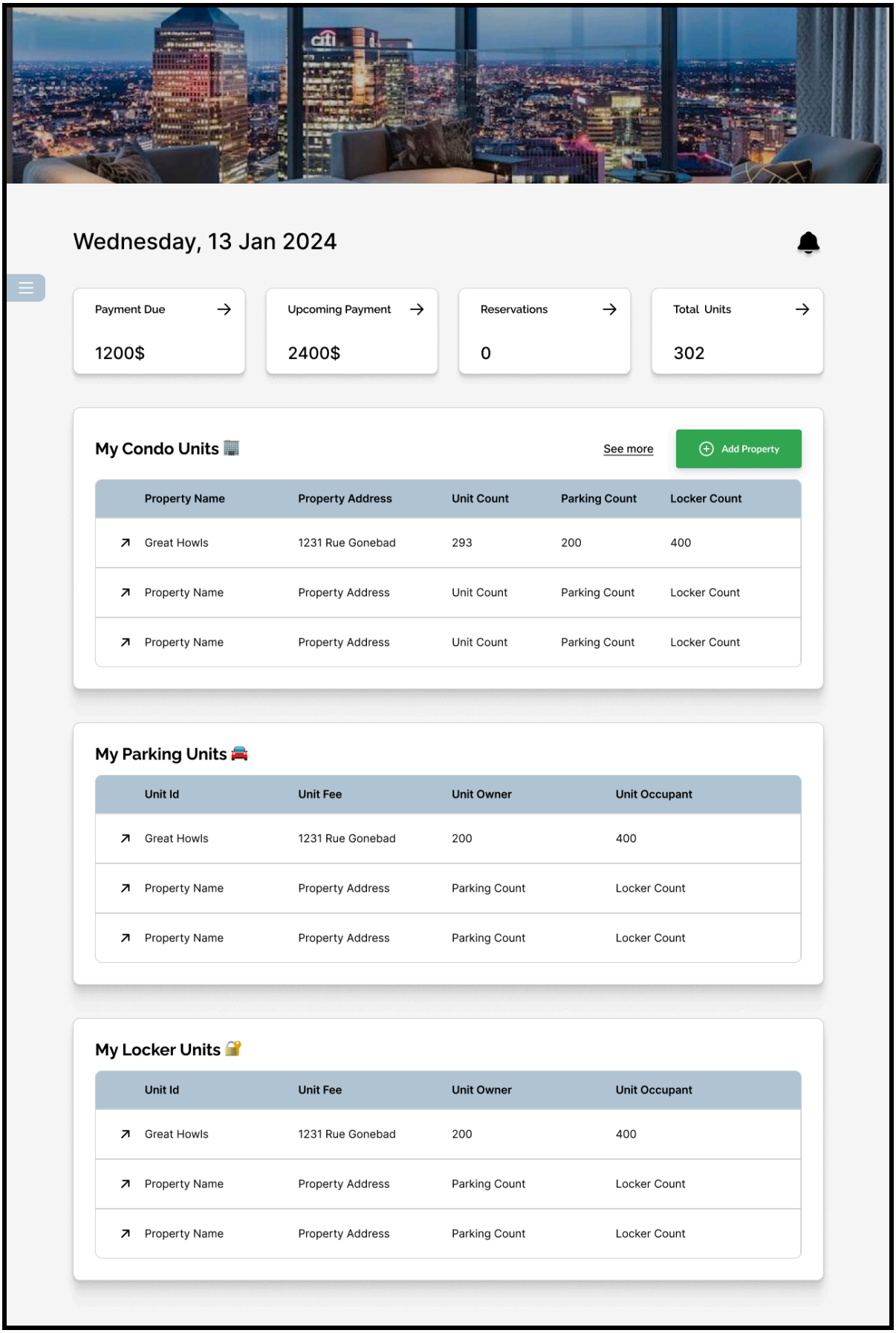
Desktop - User Key's - Registered Users



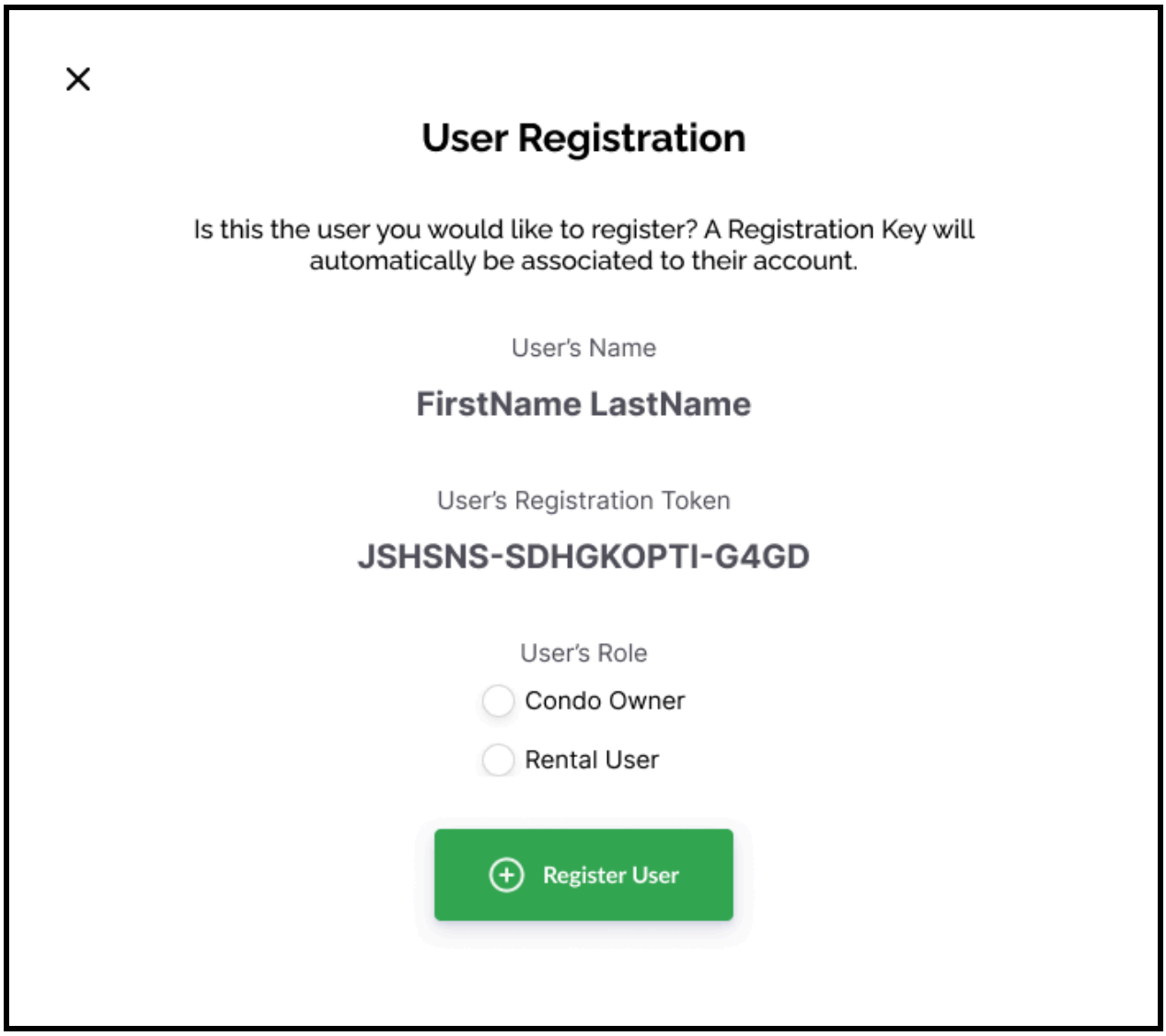
Desktop - User Key's - All Users



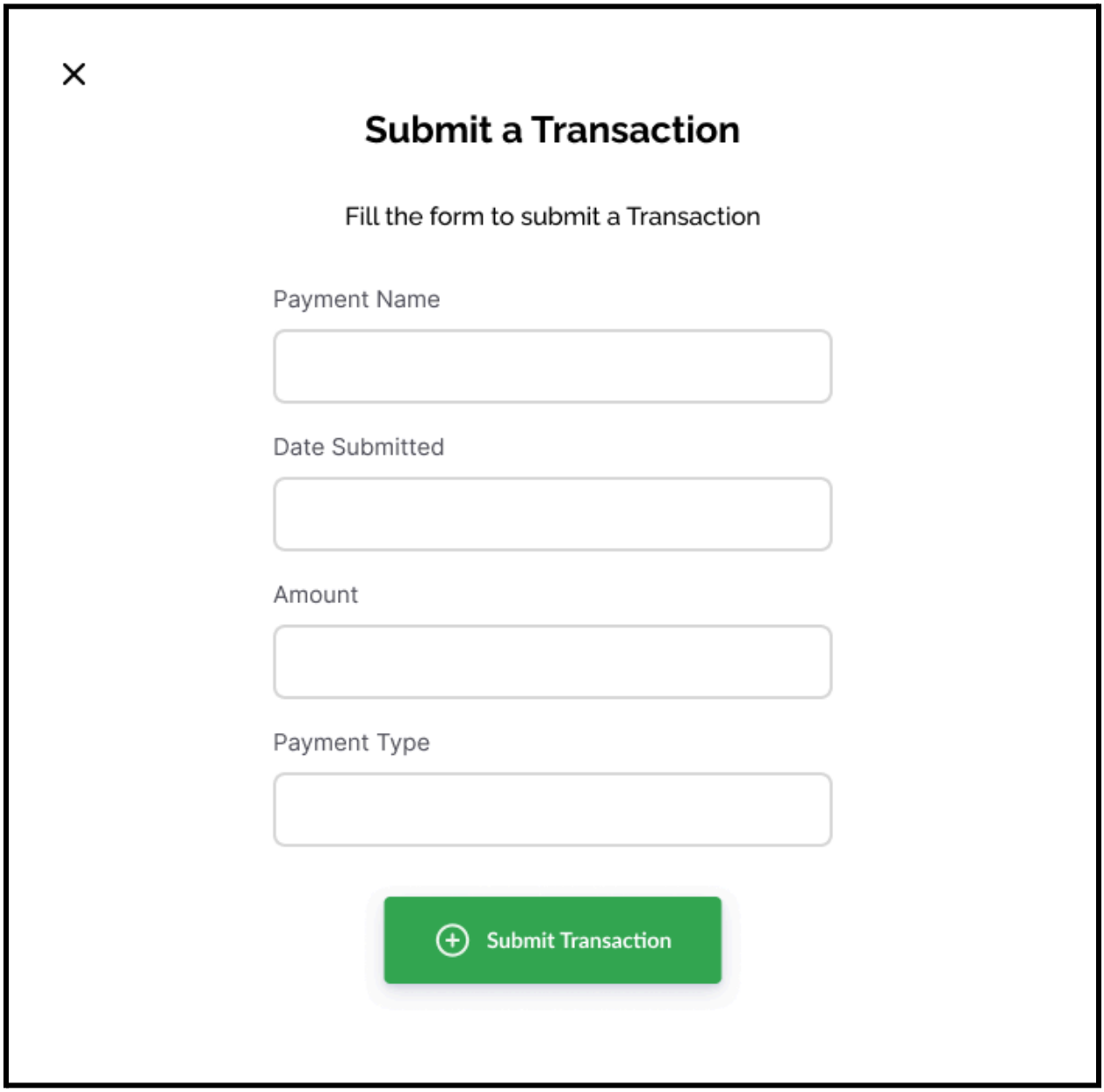
Desktop - Owner/Rental - Dashboard

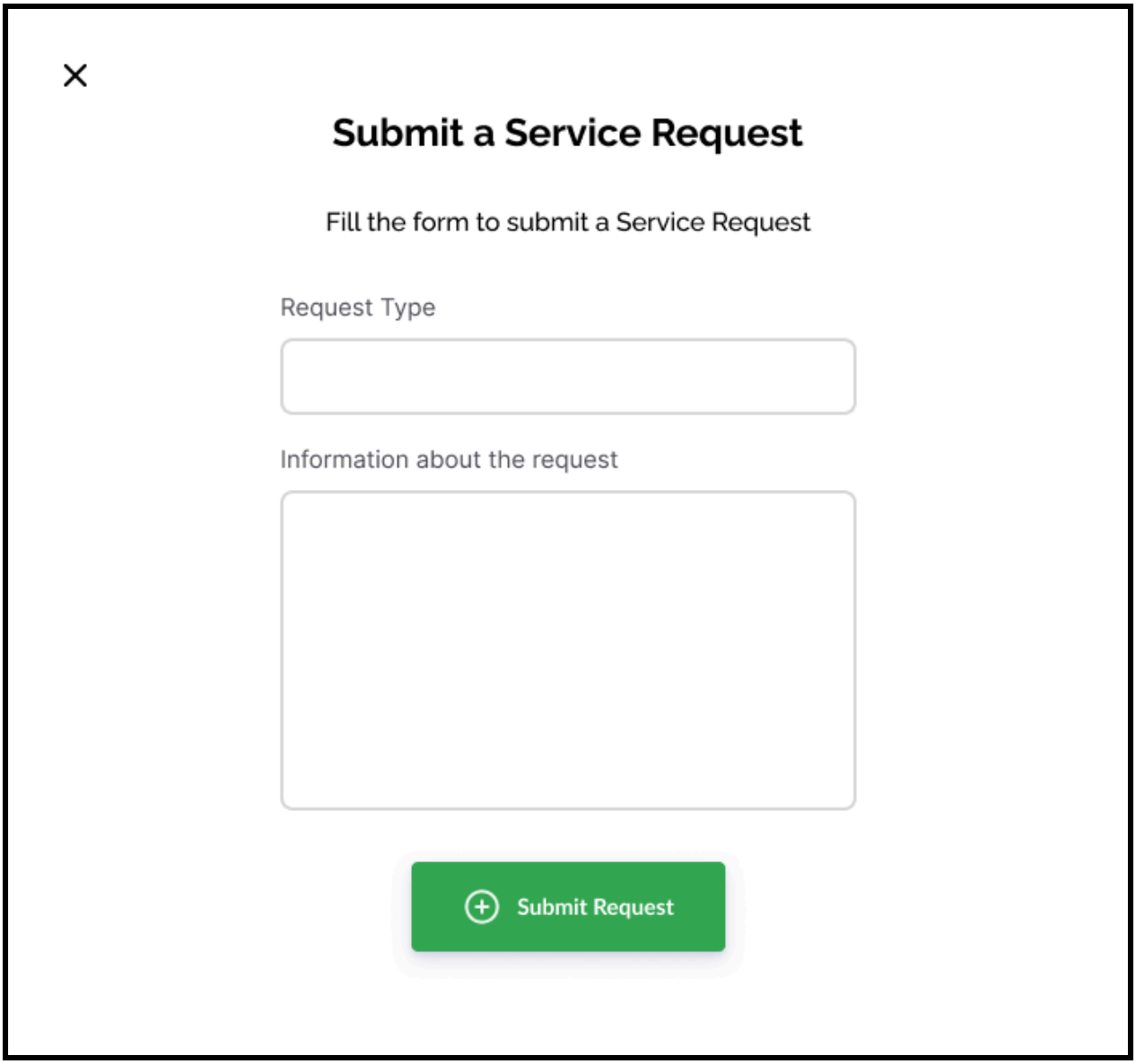


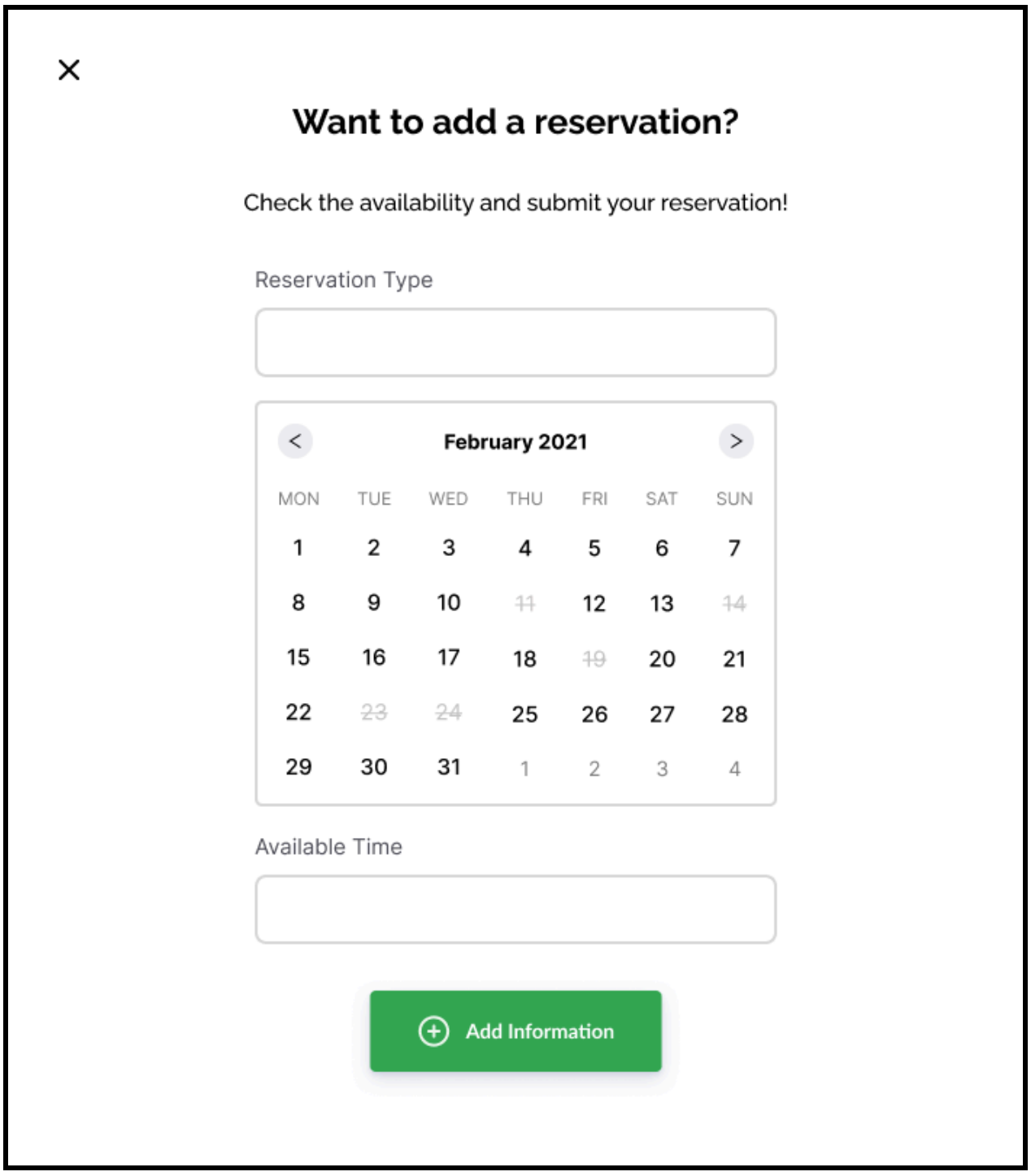
User Registration Modal

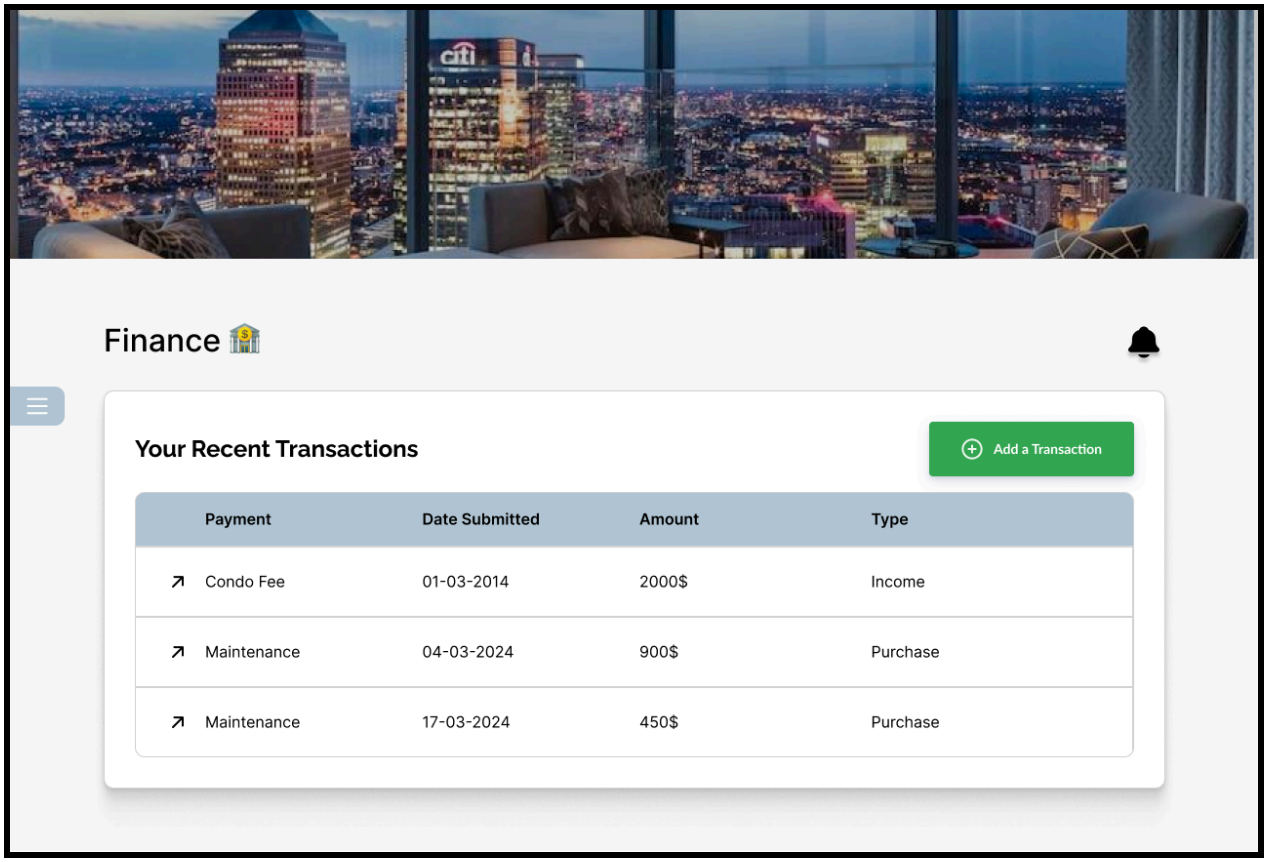


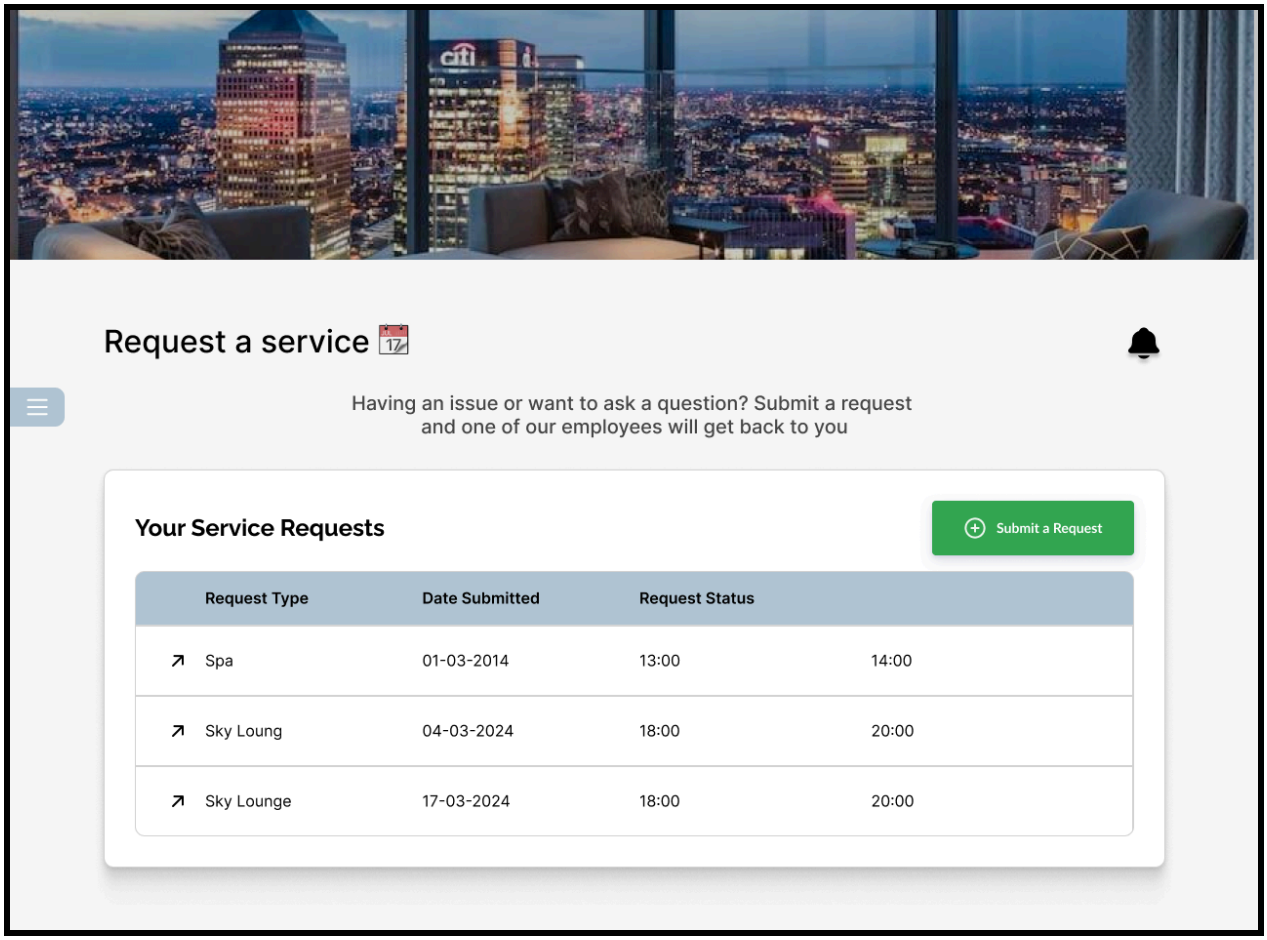
8.2 Sprint 4 UI Prototype




Submit a Service Request
Fill the form to submit a Service Request
Request Type
Information about the request
Submit Request


Want to add a reservation?
Check the availability and submit your reservation!
Reservation Type
February 2021
MON TUE WED THU FRI SAT SUN
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31 1 2 3 4
Available Time
Add Information


citi
Finance
Your Recent Transactions
Add a Transaction
Payment
Date Submitted
Amount
Type
Condo Fee
01-03-2014
2000$
Income
Maintenance
04-03-2024
900$
Purchase
Maintenance
17-03-2024
450$
Purchase


citi
Request a service
Having an issue or want to ask a question? Submit a request and one of our employees will get back to you
Your Service Requests
Submit a Request
Request Type
Date Submitted
Request Status
Spa
01-03-2014
13:00
14:00
Sky Loung
04-03-2024
18:00
20:00
Sky Lounge
17-03-2024
18:00
20:00

Chapter 9: Testing Plan

FOCUS IN SPRINT 4

In Sprint 4, our testing efforts will prioritize key features crucial for enhancing user experience and operational efficiency. We'll focus on implementing tests for assigning requests to employees, facilitating facility reservations, and ensuring accuracy in financial operations recording and updates. As we approach the final stages of development, we'll continue to augment our test suite with additional end-to-end and system tests. Moreover, we'll rigorously test every new database schema and API route introduced to maintain the robustness of our application.

1. Assign Requests to Employees:

- Inputs: Admin credentials, employee selection, request selection
- Rule: Verify that the admin user can successfully assign a request to an employee. Check that the request is valid and not already assigned.

Case	Admin Credentials	Employee Selection	Request Selection	Expected Output
1	Valid	Valid	Valid	Successful assignment, Request status updated, Confirmation sent
2	Invalid	-	-	Failure: Access Denied
3	Valid	Invalid	Valid	Failure: Invalid Employee
4	Valid	Valid	Invalid	Failure: Invalid Request
5	Valid	Valid	Valid	Successful assignment, Request status updated, Confirmation sent

2. View Facility Reservation Availability:

- Inputs: Day selection
- Rule: Display available facilities for the selected day. Check that the day selected is valid and within the reservation window.

Case	Day Selection	Expected Output
1	Valid	Display available facilities
2	Invalid	Display error message: Invalid Selection

3. Reserve Facility:

- Inputs: Selected day (date), Selected facility (facility ID, name), Selected time slot (start time, end time)
- Rule: Reserve the facility for the selected time slot. Ensure that the facility, time slot, and day are available for reservation.

Case	Day Selection	Facility Selection	Time Slot Selection	Expected Output
1	Valid	Valid	Valid	Successful reservation, Facility blocked, Confirmation sent
2	Invalid	-	-	Failure: Invalid Selection

4. Record Operational Budget:

- Inputs: Admin credentials (username, password)
- Rule: Calculate and display the operational budget. Ensure that the admin user has access to the financial section.

Case	Admin Credentials	Expected Output
1	Valid	Display operational budget calculations, Confirmation sent
2	Invalid	Display error message: Access Denied

5. Record Operational Costs:

- Inputs: Admin credentials (username, password), Operation details (date, description, cost)
- Rule: Record the operational cost report. Check that the required information is provided and valid.

Case	Operation Details	Expected Output
1	Valid	Successful submission, Confirmation sent
2	Invalid	Failure: Incomplete Information

6. Read Operational Costs:

- Inputs: Admin credentials (username, password)
- Rule: Display previously entered operational costs. Ensure that the admin user has access to the financial section.

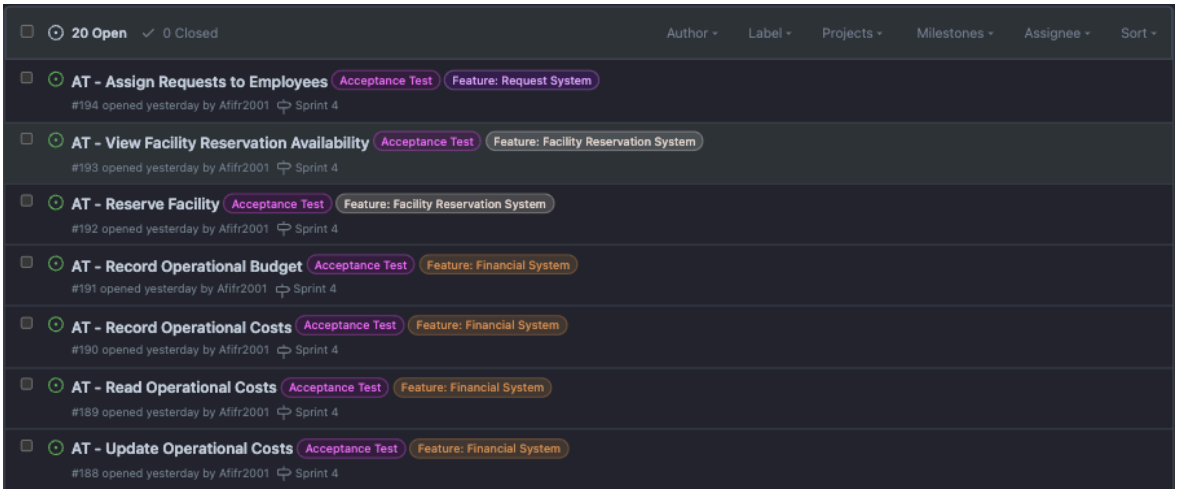
Case	Admin Credentials	Expected Output
1	Valid	Display previous costs
2	Invalid	Display error message

7. Update Operational Costs:

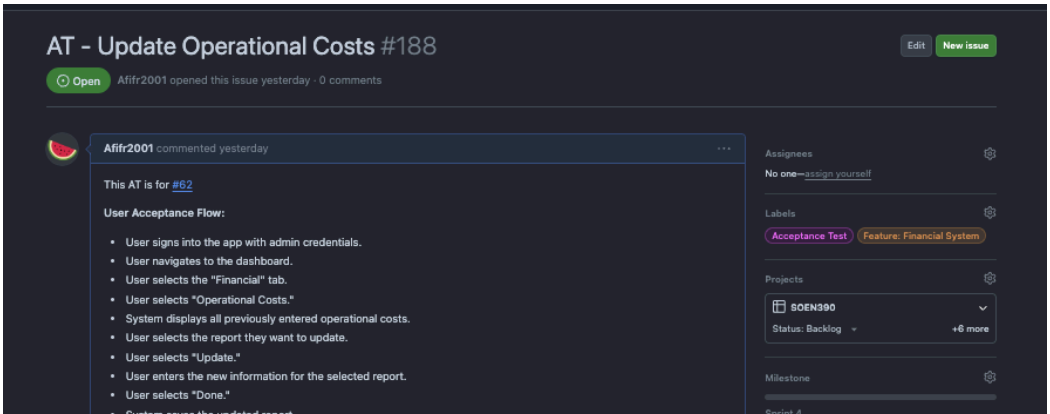
- Inputs: Admin credentials (username, password), Updated operation details (date, description, cost)
- Rule: Update the specified operational cost report. Check that the admin user can successfully update the report with valid information.

Case	Updated Operation Details	Expected Output
1	Valid	Successful update, Confirmation sent
2	Invalid	Failure: Incomplete Information

***Here are also the **Acceptance Tests** for Sprint 4 :

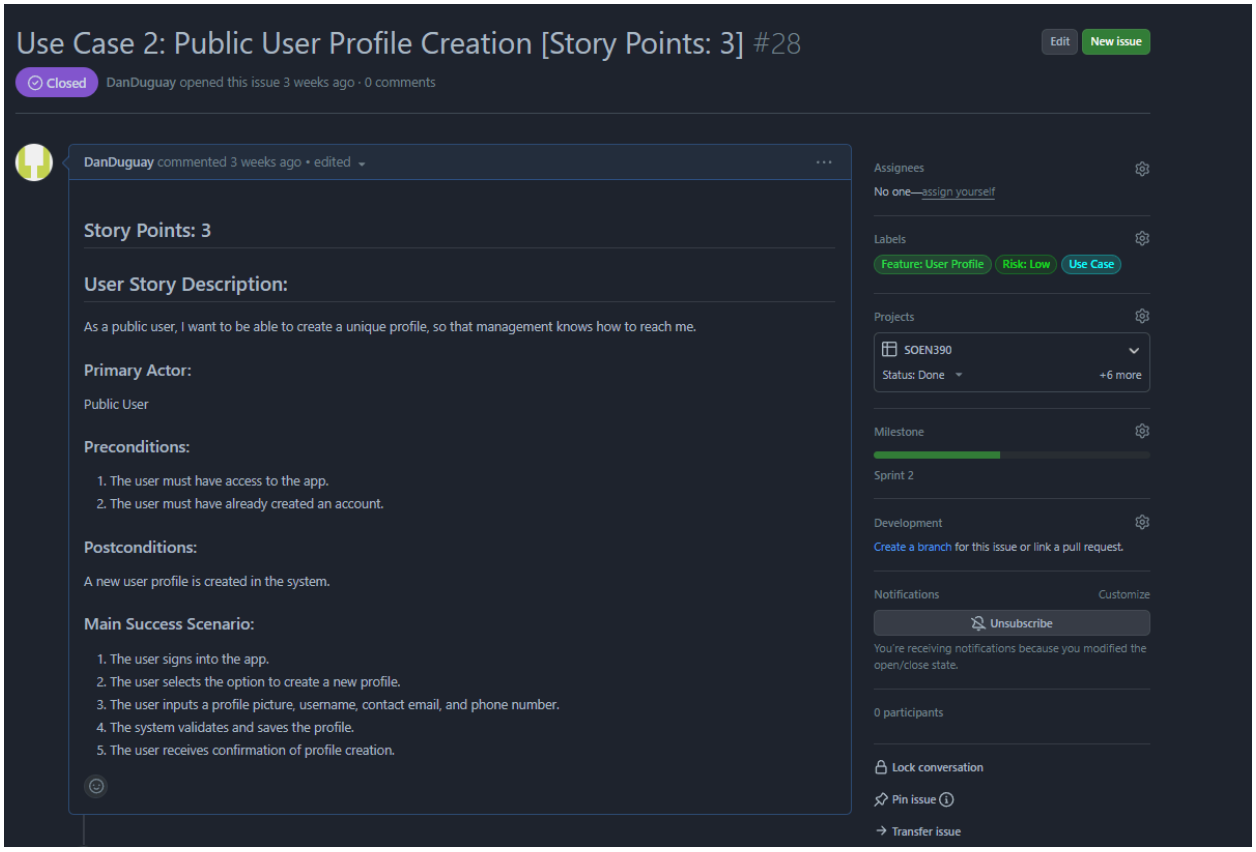


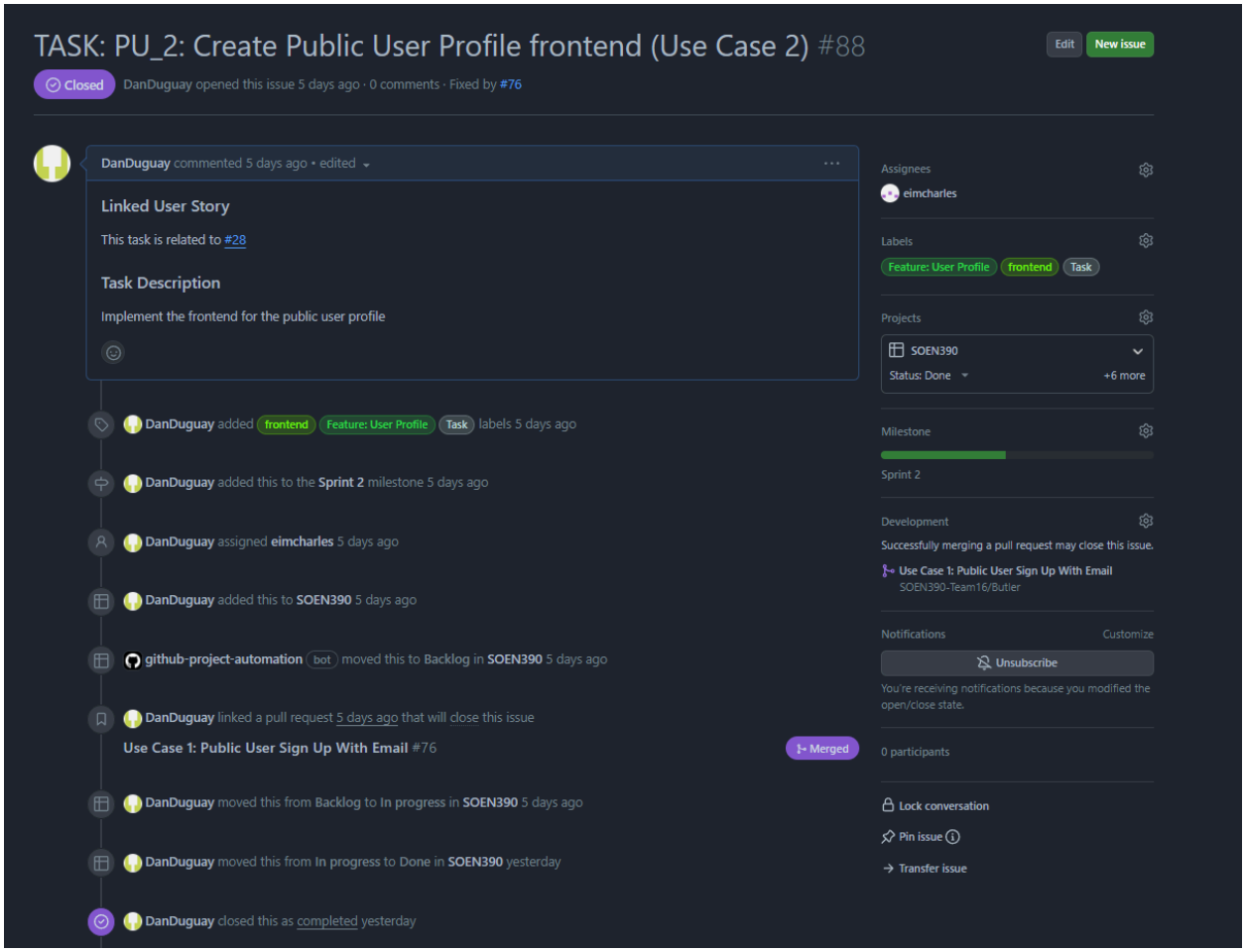
Example of an AT for an user story on Github, in Issues:



Chapter 10: Software Code Quality

10.1 Software Implementation of Planned User Stories





From user stories we created use cases on github. From these use cases we created smaller tasks that we assigned to people and linked to the use case. In this way, when all tasks related to a use case are completed, we know that the use case is ready for an e2e test (acceptance test). Once this is done, we close the use case

10.2 Bug Reports/Fixing

Bug Report Template #130

Edit New issue

Open DanDuguay opened this issue yesterday · 0 comments

DanDuguay commented yesterday • edited

Describe the bug
A clear and concise description of what the bug is.

Priority
Critical/High/Moderate/Low (pick one)

Delete this next section after choosing

Critical: Total website failure, affects all users

High: failure of an important section of the website, affects most users

Moderate: Failure of a small section of the website, affects some users

Low: More of an enhancement, no real failure

To Reproduce
Steps to reproduce the behavior:

1. Go to '...'
2. Click on '....'
3. Scroll down to '....'
4. See error

Expected behavior
A clear and concise description of what you expected to happen.

Screenshots
If applicable, add screenshots to help explain your problem.

Desktop (please complete the following information):

- OS: [e.g. iOS]
- Browser [e.g. chrome, safari]
- Version [e.g. 22]

Smartphone (please complete the following information):

- Device: [e.g. iPhone6]
- OS: [e.g. iOS8.1]
- Browser [e.g. stock browser, safari]
- Version [e.g. 22]

Additional context
Add any other context about the problem here.

Delete the following sections before submitting the bug report

Assignees
Leave blank when creating, assign to yourself if you're working on solving the bug

Labels
On the right, add a bug label and the feature label related to your problem.

Projects
On the right, select SOEN390 as the project

Milestone
On the right, select the current sprint as the Milestone

Working on a bug
If you fixed a bug, link to it in your commit's message when creating a pull request
example: fixed bug #130, then the person who reviews your pull request can close the bug after merging

Assignees
No one—assign yourself

Labels
bug

Projects
SOEN390
Status: Backlog +6 more

Milestone
Sprint 2

Development
Create a branch for this issue or link a pull request.

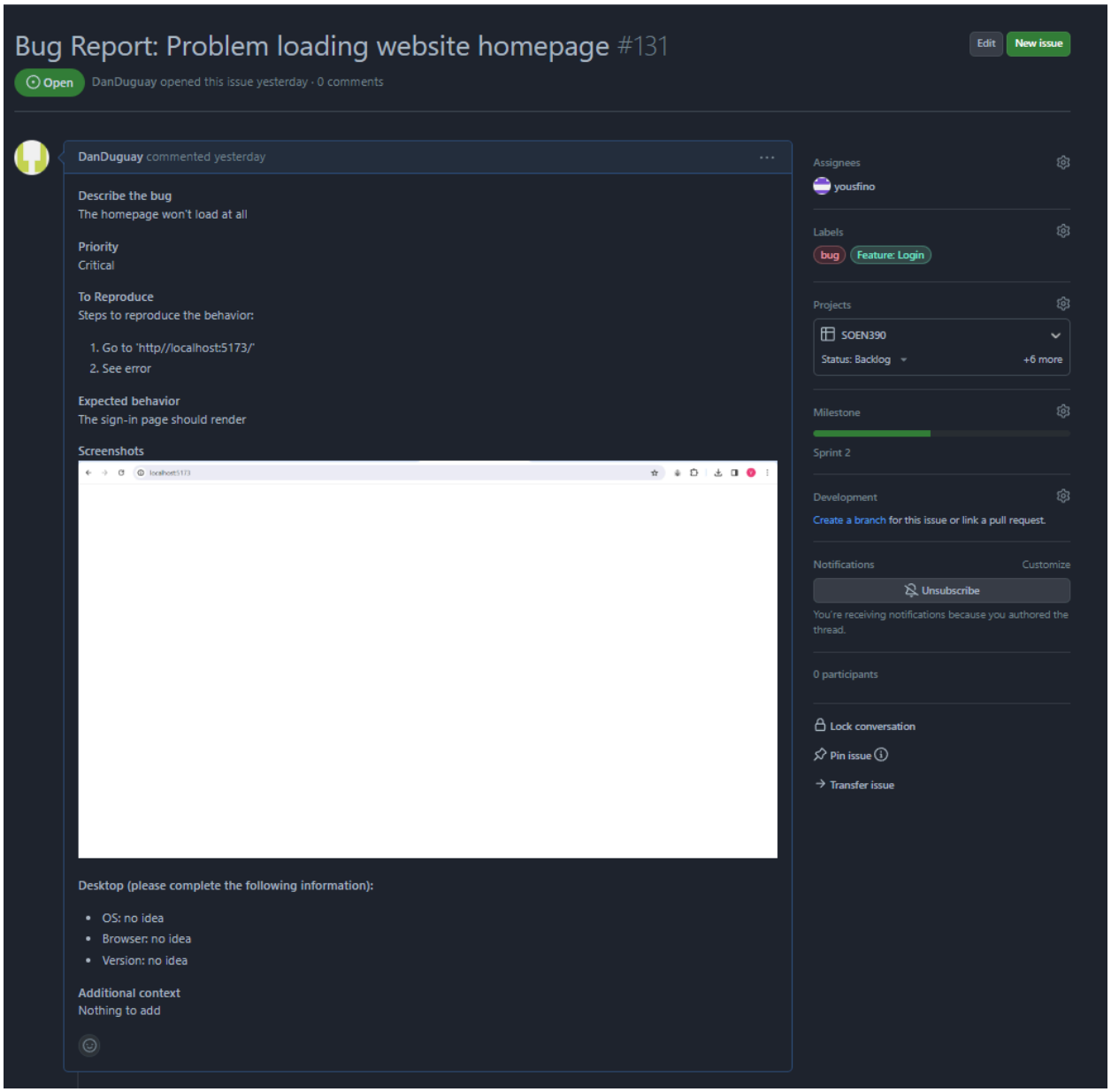
Notifications Customize
Unsubscribe
You're receiving notifications because you authored the thread.

0 participants

Lock conversation
Pin issue
Transfer issue

a bug report template was created as an issue on github, detailing the bug report system in place and also how to go about fixing reported bugs

10.2.1 Bug Reports Fixing Techniques



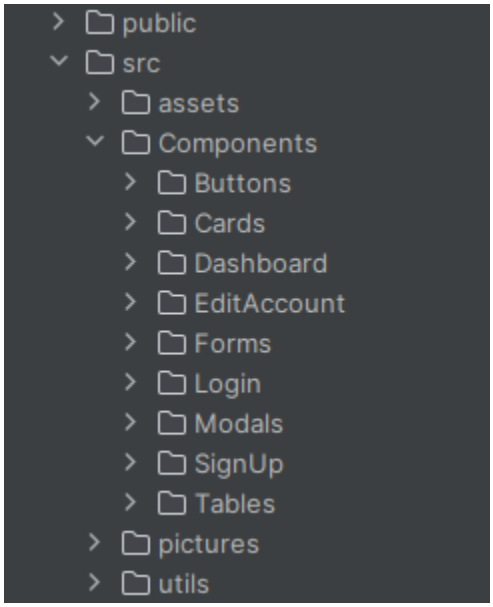
Once a bug report is created, the one who will work on it assigns themselves to the bug report. Once they fix the bug, they link their Pull Request to the bug report issue number, which will close the bug report issue when merged.

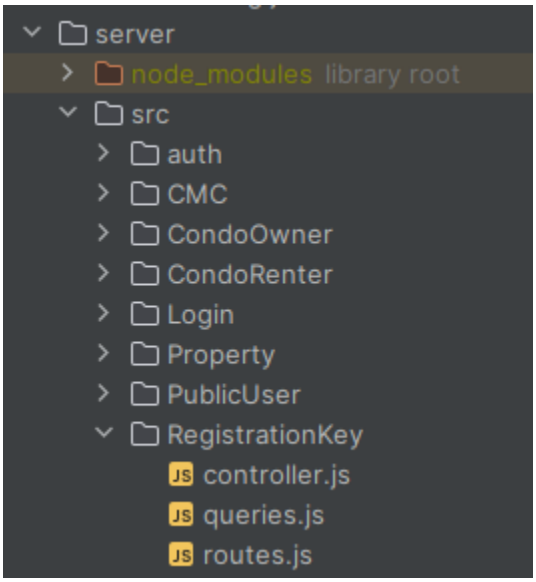
10.3 Code Coverage

Test Coverage using JEST for testing components in the backend, achieved above 80% code coverage:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	78.68	65.69	80.5	79.16	
server	100	100	100	100	
db.js	100	100	100	100	
server/src/CMC	83.33	74.35	92.3	84.26	
controller.js	82.14	74.35	92.3	83.13	46,61-62,74,105-118,135-136
queries.js	100	100	100	100	
server/src/CondoOwner	78.37	64.15	75	78.37	
controller.js	76.47	64.15	75	76.47	46-69,133-134,138,162-163,168-170
queries.js	100	100	100	100	
server/src/CondoRenter	79.81	66.03	81.25	79.81	
controller.js	78	66.03	81.25	78	47-69,132-133,137,161-162,168-169
queries.js	100	100	100	100	
server/src/CondoUnit	92.98	87.5	100	92.98	
controller.js	92	87.5	100	92	50-51,68-69
queries.js	100	100	100	100	
server/src/Employee	95.6	80	100	95.6	
controller.js	95.23	80	100	95.23	88,124,136,141
queries.js	100	100	100	100	
server/src/Login	79.16	57.14	57.14	80.85	
controller.js	79.16	57.14	57.14	80.85	47-59,68-69,83,88-89
server/src/Property	65.58	43.63	76.19	66.88	
controller.js	62.93	43.63	76.19	64.28	5-21,26,30-31,35-44,79,95,110,125,140-163,177,215-225,239-251
queries.js	100	100	100	100	
server/src/PublicUser	82.97	80	84.61	82.97	
controller.js	81.81	80	84.61	81.81	40-61,113-114,126
queries.js	100	100	100	100	
server/src/RegistrationKey	56.45	41.66	54.54	56.45	
controller.js	50	41.66	54.54	50	6,10-34,63-64,71-72,78-79
queries.js	100	100	100	100	

10.4 Class/Function/Packages By Size

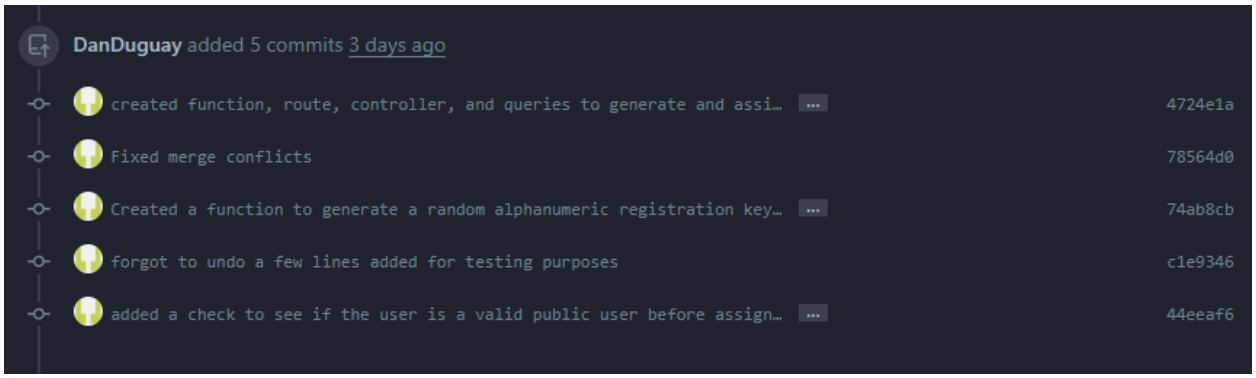




Our file structure ensures separation of concerns

10.5 Quality Of Source Code Documentation

10.6 Refactoring Activity Documented In Commit Messages

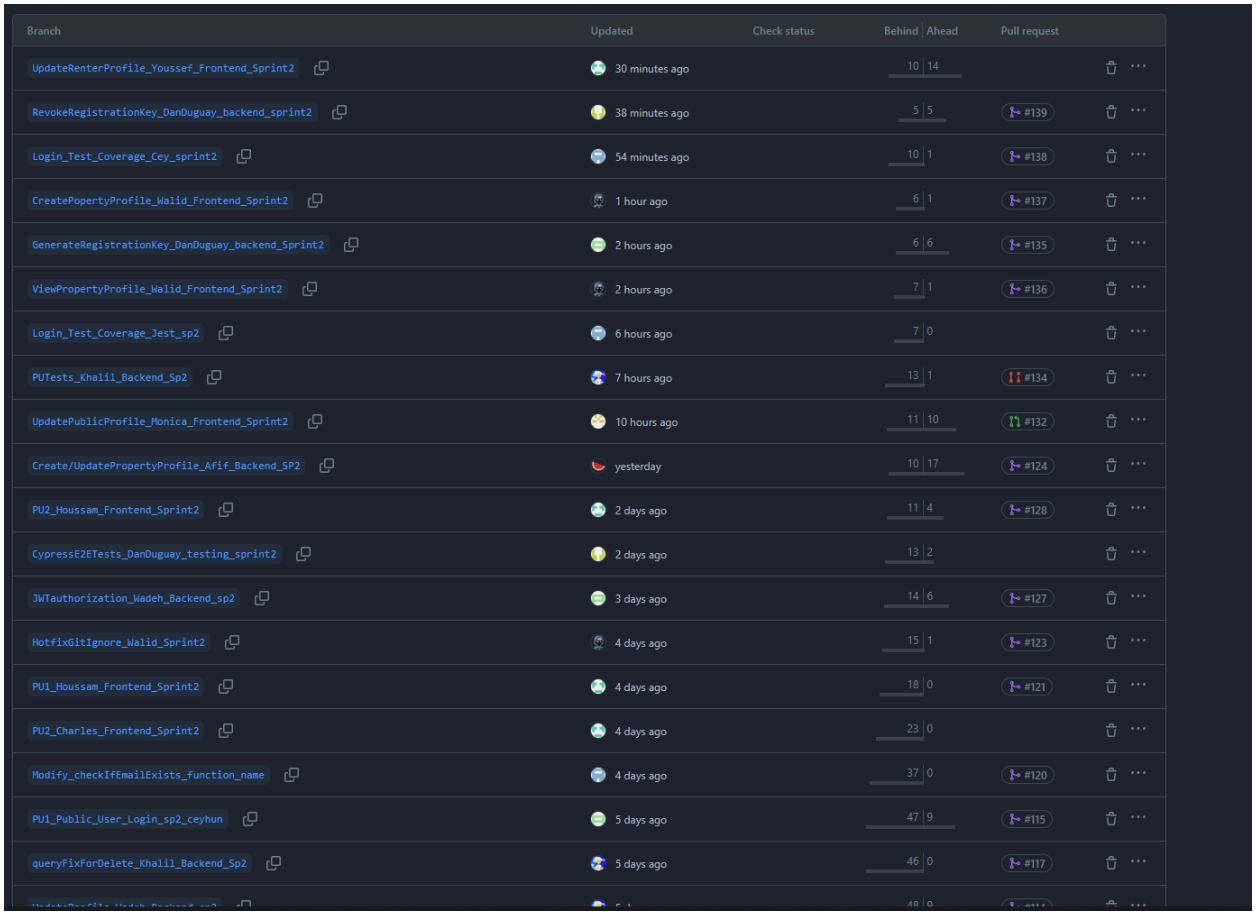


Here is a sample of a commit chain detailing refactoring the Generate Registration Key Feature

10.7 Quality/Detail Of Commit Messages

we try to give as much detail about the feature that was implemented in our commit messages.

10.8 Use Of Feature Branches



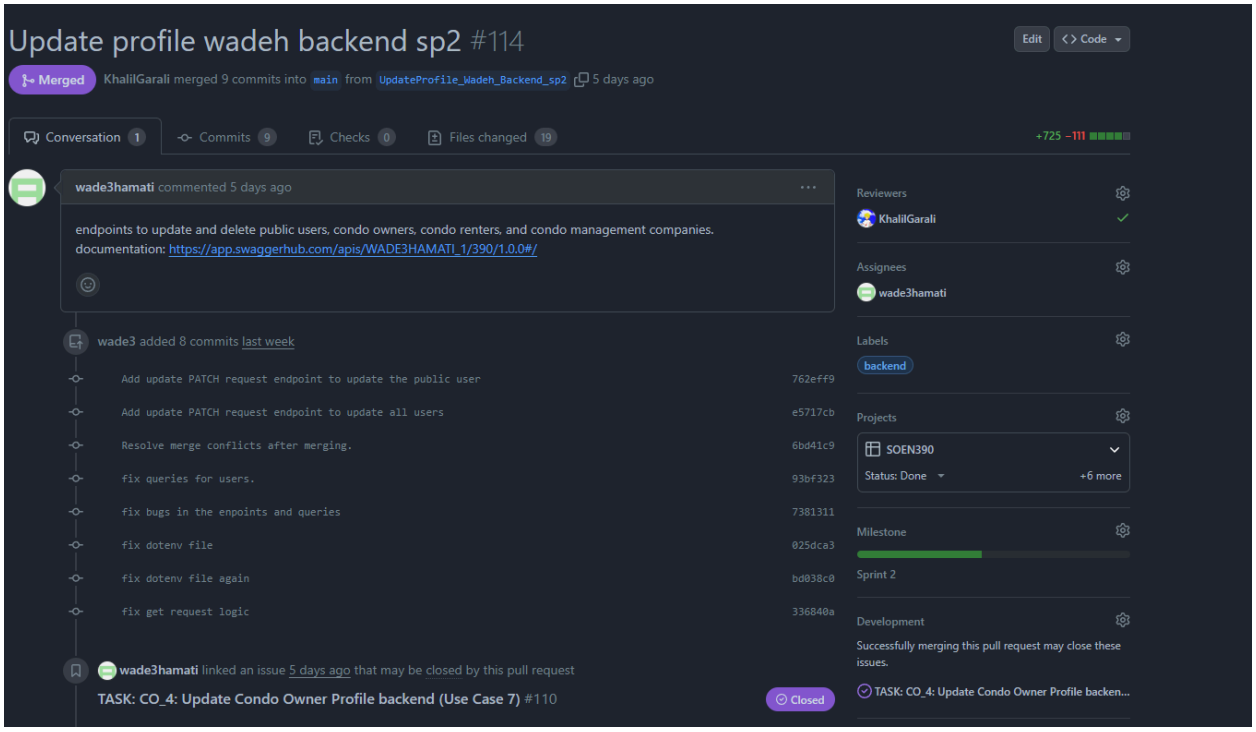
Branch	Updated	Check status	Behind	Ahead	Pull request
UpdateRenterProfile_Youssef_Frontend_Sprint2	30 minutes ago		10	14	
RevokeRegistrationKey_DanDuguay_backend_sprint2	38 minutes ago		5	5	#139
Login_Test_Coverage_Cey_sprint2	54 minutes ago		10	1	#138
CreatePopertyProfile_Walid_Frontend_Sprint2	1 hour ago		6	1	#137
GenerateRegistrationKey_DanDuguay_backend_Sprint2	2 hours ago		6	6	#135
ViewPropertyProfile_Walid_Frontend_Sprint2	2 hours ago		7	1	#136
Login_Test_Coverage_Jest_sp2	6 hours ago		7	0	
PUTests_Khalil_Backend_Sp2	7 hours ago		13	1	#134
UpdatePublicProfile_Monica_Frontend_Sprint2	10 hours ago		11	10	#132
Create/UpdatePropertyProfile_Afif_Backend_SP2	yesterday		10	17	#124
PU2_Houssam_Frontend_Sprint2	2 days ago		11	4	#128
CypressE2ETests_DanDuguay_testing_sprint2	2 days ago		13	2	
JWTauthorization_Wadeh_Backend_sp2	3 days ago		14	6	#127
HotfixGitIgnore_Walid_Sprint2	4 days ago		15	1	#123
PU1_Houssam_Frontend_Sprint2	4 days ago		18	0	#121
PU2_Charles_Frontend_Sprint2	4 days ago		23	0	
Modify_checkIfEmailExists_function_name	4 days ago		37	0	#120
PU1_Public_User_Login_sp2_ceyhun	5 days ago		47	9	#115
queryFixForDelete_Khalil_Backend_Sp2	5 days ago		46	0	#117

Our branch conventions can be found on the “wiki” portion of the Butler repo on github. Every branch is dedicated to one small feature, to keep commits atomic.

10.9 Atomic Commits

As seen above, given the small scope of each branch/commit, it is easy to roll back the main branch if ever a bad commit gets through

10.10 Linking Of Commits To Bug Reports/Features



Here is an example of a linked commit. This particular commit was linked to Task: CO_4