SOEN 390

Software Engineering Team Design Project

Instructor:

Jinqiu Yang

Compiled Report

Name	Student ID
Karyenne Vuong	40157011
Cristian Gasparesc	40209208
Amirhossein Tavakkoly	40203604
William Nazarian	40213100
Vanisha Patel	40242554
Racha Kara	40210865
Ryan Guzelian	40211846
Fadi Nimer	40183225
Imane Madda	40208741
Daniel Bondar	40213095

Due Date: Wednesday, May 1st, 2024

Table of Contents

1. PRODUCT VISION STATEMENT

SOFTWARE PRODUCT VISION

1. Introduction

The purpose of this document is to collect, analyze, and define high-level needs and features of Condo360. It focuses on the capabilities needed by the stakeholders, and the target users, and why these needs exist. The details of how Condo360 fulfills these needs are detailed in the use-case and supplementary specifications. This document serves as an introduction to our vision, as well as an insight into the needs of stakeholders and users.

2. Positioning

2.1 Problem Statement

The problem of	Inefficient condo management system
affects	condo owners, condo management companies
the impact of which is	Disorganized property management, financial inconsistencies, restricted access to property details, communication problems between resident and owners
a successful solution would be	An accessible, easy-to-use website with its phone application, to effectively manage condos for the resident and owners.

2.2 Product Position Statement

For	Condo owners and condo management companies
Who	Look for a complete and efficient solution to manage condo properties
The (product name)	Condo360
That	Provides accessibility and effective property management by having various features like financial tracking and request handling.
Unlike	Manual (standard) condo management methods
Our product	Provides a user-friendly interface making interaction and navigation easy. It also improves productivity by centralizing all condo management activities within a software platform making it easy to

	track all activities and requests by condo owners and condo management companies.

3. Stakeholder and User Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibilities
Triet Pham	Team 7's TA	The stakeholder must ensure that all software requirements and interests are communicated with the team, while also validating the market demands for the system's features. In addition, the stakeholder must also monitor and evaluate the progress of the project. The stakeholder must provide constructive feedback for the team to improve the software and meet stakeholder's expectations. The stakeholder must also provide the necessary funds for the realization of this project.
Developer Team	Team 7's developers	The developers are responsible to communicate with the client and ensure that the system being developed meets the requirements. They need to ensure that the system is maintainable and monitor the project's progress. The team will be working in an Agile framework, and they will need to deliver an incremented running system each sprint.
Condo management companies	Organization managing many condo properties and using the system to monitor and supervise their properties	The condo management companies are responsible to manage the condo files and their owners' profiles. They need to enter detailed information about their services (parking, lockers, etc.), and manage the finances of their properties. They also manage reservations and bookings made by the condo owners, and assign roles to their employees.

3.2 User Summary

Name	Description	Responsibilities	Stakeholder
Condo owners	Individuals who own condo units and use the system to manage their properties and access important information	- Access and manage property information - Monitor financial status - Submit requests for services and maintenance - Communicate with condo management companies for issues	Condo management companies
Public users	Individuals who are potential condo owners who use the system to use condo services based on their ownership status	- Create their own profile - View property information - Submit requests regarding condo units or rental issues	Condo management companies
Employees of condo management companies	Individuals who are assigned specific roles from the condo management companies. They use the system to handle daily operations and manage property-related activities.	- Daily operations - Handle financial tasks - Respond requests by condo owners - Communicate with condo owners	Condo management companies

3.3 User Environment

The four most important users of this system are the condo management companies, the condo owners, the public users, and the employees of condo management companies. They have different responsibilities and tasks to complete, and it will not be changing. Task cycle durations are different depending on the complexity of activity. The users can interact with the product both on PC and mobile devices. The task cycle has no set time limit; a user is free to browse the website for as long as they desire.

3.4 Key User Needs

Need	Priority	Concerns	Current Solution	Proposed Solutions
Efficient profile creation	High	Complex profile setup	Manual profile creation with paper forms	Simplified profile creation process with easy interface and reduced steps
Efficient request handling	High	Delayed responses	Email or paper form submission	Implement an online ticket system
Make a responsive design for the software to be accessible across many devices	Medium	Flexibility	None	Have proper CSS organization that allows for a flexible UI
Simplified condo file management	High	Difficulty organizing and accessing files	Manual file organization	Develop digital form repository for easy organization and access

3.5 Alternatives and Competition

When considering alternatives and competition, stakeholders perceive multiple options such as opting for a competitive software, creating their own customized solution in-house, or continuing with the current manual process. While competitors might have larger user bases, they might lack modern features or user-friendly interfaces. To stand out, Condo360 prioritizes user-friendliness, comprehensive features, and seamless device integration.

4. Product Overview

4.1 Product Perspective

Condo360 is designed as an independent and self-contained condo management solution. It centralizes all condo management features, providing a comprehensive and seamless user

experience. There are no external systems, interfaces or components required. The system operates as a standalone platform, catering to the needs of condo owners, condo management companies, and any other user who will interact with the software.

4.2 Assumptions and Dependencies

There are many factors affecting the features stated in the Vision document. The stakeholder feedback will influence the development of features to meet the user requirements. The goal is to develop a product tailored for the stakeholder's needs. In addition, insufficient time and budget will impact the implementation of features. For these reasons, the Vision Document will be updated each sprint.

5. Product Features

There are many features to be developed for the condo management system. The Vision Document will state only the most important ones.

5.1 Condo File Management

The system shall let the condo management companies upload and organize condo files. This will centralize all file storage making it easy for access and management.
The documents uploaded are displayed in a table format displaying the document type, title, belonging type (property (Address) or corporate), the date it was uploaded, the name of the individual or department that uploaded the document, and its status (Pending Verification or Verified).

5.2 Financial Tracking

The system shall have financial tracking allowing the condo management companies and condo owners to record and monitor condo fees, operational budget, and expenses accurately.
The finance page is separated into two main sections. The first listing all the properties with their respective property address, unit number, type of transaction (expense or income), the transaction date, the type of transaction performed (employee salary, maintenance, etc.), description of the transaction, the amount of the transaction and its status (paid or pending).

The second section contains all the budget-related information (total budget, total income, total expenses, and remaining budget.

5.3 Reservation System

The system shall let condo management companies configure facility reservations. The condo owners will be able to book amenities like the gym or the pool.

The reservation page contains two main sections. The first displays all the owner's current reservations in a table format with the option to cancel.

The second section displays all the available facilities with the option to reserve them. Once the reserve option is clicked a popup allows the user to choose the date on a calendar system and place their reservation.

5.4 Request Management

The system shall let users submit different requests related to property management like maintenance issues to condo owners. This will make request handling and solving problems easier. The condo owners can see a list of the requests submitted by users. The requests are very detailed and display crucial information to help the condo owners decide. They can update the request status by approving or denying it. The users can see the status of all their requests.

2. USER STORIES BACKLOG

Effort Value Scale

1	Requires very little time, effort, or resources to implement, often just a matter of minutes or a few simple steps.
2	Requires a small amount of time and effort, typically straightforward and easily achievable.
3	Requires a modest amount of time and effort, with some complexity but manageable without significant challenge.
4	Requires a moderate level of time and effort, involving some complexity or dependencies but still manageable within a reasonable timeframe.
5	Represents an average level of time and effort, with moderate complexity or dependencies requiring focused attention and resources.
6	Requires a significant amount of time and effort, involving notable complexity or dependencies that may require additional planning or resources.
7	Requires a high level of time and effort, with considerable complexity or dependencies that may pose challenges and require careful coordination.
8	Requires intensive time and effort, involving substantial complexity or dependencies that may require dedicated focus and resources over an extended period.
9	Requires extensive time and effort, with significant complexity or dependencies that may necessitate specialized skills, extensive planning, and coordination.
10	Represents the highest level of time and effort, requiring maximum resources and attention due to exceptional complexity or dependencies, often requiring significant investment and careful management.

Business Value Scale

S	Incremental changes or additions that provide immediate but modest value to the business.
M	Enhancements or features that offer tangible benefits and contribute meaningfully to business objectives.
L	Substantial improvements or features that significantly advance business goals and have a notable impact on performance or competitiveness.
XL	Strategic initiatives or major developments that fundamentally transform the business landscape and drive substantial growth or innovation.

Card Template Used

ID	Title		
User Story			
Main Feature			
MoSCoW	Business Value	Risk	Effort

Color Code

Green → New
Yellow → Updated
Red → Deleted (non-mandatory/extra features removed, not a user story, or duplicate)
~~Crossed~~ → Completed

Public Users

~~40~~	~~Create a profile~~		
~~As a public user, I want to be able to create a unique profile by providing a profile picture, username, contact email, and phone number.~~			
~~Profile Feature~~			
~~Must~~	~~L~~	~~High~~	~~8~~

~~41~~	~~Provide a registration key~~		
~~As a public user, I want to provide a registration key obtained from my condo management company to become a condo owner or renter in the system.~~			
~~Property Feature~~			
~~Must~~	~~L~~	~~Low~~	~~3~~

61	Access my remaining payments from the mobile version		
As a public user, I want to access my remaining payments from the mobile version in order to have a quicker way to find the information.			
Financial System Feature			
Could	M	Low	4

~~70~~	~~Landing page~~		
~~As a user, I want a landing page that will show what the application is about and allows for redirection to specific parts of the website in order to access the needed information without having to type links.~~			
~~Must~~	~~L~~	~~Low~~	~~4~~

Condo Owners

42	Dashboard of my properties		
As a condo owner, I want to have a dashboard that displays general information about my properties, including personal profile, condo information, and financial status.			
Dashboard Feature			
Must	L	High	9

43	View the status of submitted requests		
As a condo owner, I want to view the status of submitted requests, such as moving in/out requests, intercom changes, or reporting violations.			
Requests Feature			
Should	L	Medium	8

44	Notifications of latest activities in requests		
As a condo owner, I want to receive notifications about the latest activities in submitted or assigned requests.			
Requests Feature			
Should	S	Low	5

56	See availability of common facilities		
As a condo owner, I want the reservation system to show the availability of common facilities.			
Reservation System Feature			
Must	L	Low	6

57	FIFO Reservation system		
As a condo owner, I want the reservation system to follow a first-come-first-serve approach.			
Reservation System Feature			
Should	M	Medium	5

~~63~~	~~Reserve common facilities~~		
~~As a condo owner or rental user, I want to reserve common facilities in a calendar-like interface.~~			
~~Reservation System Feature~~			
~~Must~~	~~M~~	~~Medium~~	~~7~~

58	Access forum		
As a condo owner, I want access to a forum where I can post and reply to discussions.			
Extra Feature			
Could	S	Medium	7

59	Organize events		
As a condo owner, I want the ability to organize events and invite other occupants to attend.			
Extra Feature			
Could	S	Medium	7

Condo Management Companies

~~62~~	~~Overview of all my owned properties~~		
~~As a condo management company owner, I want to have an overview of all my owned properties along with their essential information on the mobile version in order to have quicker access to the data.~~			
~~Dashboard Feature~~			
~~Must~~	~~L~~	~~High~~	~~8~~

~~53~~	~~Financial system to generate an annual report~~		
~~As a condo management company, I want the financial system to generate an annual report showing all condo fees collected for a given year.~~			
~~Financial System Feature~~			
~~Must~~	~~L~~	~~High~~	~~7~~

60	Option to list coupons/offers visible		
As a condo management company, I want the option to list coupons/offers visible to all unit owners or rental users of one property.			
Dashboard Feature			
Could	M	High	6

~~45~~	~~Create profiles for properties~~		
~~As a condo management company, I want to be able to create profiles for properties under my management, providing essential information like property name, unit count, parking count, locker count, and address.~~			
~~Profile Feature~~			
~~Must~~	~~L~~	~~High~~	~~8~~

~~52~~	~~Enter costs for each operation~~		
~~As a condo management company, I want to enter costs for each operation.~~			
~~Financial System Feature~~			
~~Must~~	~~M~~	~~Medium~~	~~7~~

~~47~~	~~Enter information for each condo unit, parking spot, and locker~~		
~~As a condo management company, I want to enter detailed information for each condo unit, parking spot, and locker, including unit size, owner information, and associated condo fees.~~			
~~Dashboard Feature~~			
~~Must~~	~~L~~	~~Medium~~	~~10~~

~~51~~	~~Record operational budgets and costs~~		
~~As a condo management company, I want to record operational budgets (collected condo fees) and costs.~~			
~~Financial System Feature~~			
~~Must~~	~~L~~	~~Medium~~	~~6~~

48	Send registration keys to unit owners or rental users		
As a condo management company, I want to send registration keys to unit owners or rental users for their dedicated units.			
Property Feature			
Must	L	Low	4

54	Set up common facilities requiring reservations		
As a condo management company, I want to set up common facilities requiring reservations, such as a sky lounge or spa fitness.			
Reservation System Feature			
Must	L	Medium	6

49	Set up different roles for employees		
As a condo management company, I want to set up different roles for employees responsible for daily operations, finance, etc.			
Financial System Feature			
Should	M	Medium	9

50	Set up the condo fee		
As a condo management company, I want to set up the condo fee per square foot and per parking spot and calculate condo fees for each unit.			
Property Feature			
Must	M	Low	6

46	Upload condo files for each property		
As a condo management company, I want to upload condo files for each property, accessible to all condo owners, including declarations, annual budgets, and board meeting minutes.			
Property Feature			
Should	M	Medium	8

Employees

~~217~~	~~Accept and Deny Requests~~		
~~As an employee, I want to accept or deny requests from condo owners living in properties that are under my responsibility.~~			
~~Requests Feature~~			
~~Should~~	~~S~~	~~Low~~	~~6~~

Mobile app

157	Access public user account		
As a public user, I want to access my account on the mobile version of the platform.			
Mobile App			
Should	L	Low	7

214	Access condo owner account		
As a condo owner, I want to access my account on the mobile version of the platform.			
Mobile App			
Should	L	Low	7

163	Access condo management company account		
As a condo management company, I want to access the details of my property from the mobile app.			
Mobile App			
Should	L	Low	7

215	Access employee account		
As an employee, I want to access my account on the mobile version of the platform.			
Mobile App			
Should	L	Low	7

3. SOFTWARE ARCHITECTURE DESCRIPTION

SOEN390 - Team Design Project

Architecture Description of Condo360 Architecture for Condo360

Version 3

By

Name	Student ID
Karyenne Vuong	40157011
Cristian Gasparesc	40209208
Amirhossein Tavakkoly	40203604
William Nazarian	40213100
Vanisha Patel	40242554
Racha Kara	40210865
Ryan Guzelian	40211846
Fadi Nimer	40183225
Imane Madda	40208741
Daniel Bondar	40213095

Department of Computer Science and Software Engineering
Gina Cody School of Engineering and Computer Science
Concordia University

Contents

1 Introduction

1.1.1 Summary:

This Software Architecture Document (SAD) outlines the architecture and design of the Condo Management Systems, detailing its main features, additional features, and bonus features as specified in the project requirements. It provides a comprehensive overview of the system's functionalities and the underlying architecture required to support them.

1.1.2 Vision

The Condo360 System project envisions revolutionizing how condo properties are managed by integrating digital solutions. This platform aims to simplify and automate tasks for property managers and enhance the living experience for residents, promoting a harmonious, efficient, and connected community environment.

1.1.3 Scope

The project is scoped to cater to medium-sized condo associations within urban settings, focusing on features such as online payments, service requests, resident communication, and management reporting. Initial deployment will target a single region with plans for future expansion.

1.2 Supplementary information

Date of Issue: 19/01/2024

Status: Draft

Authors: Karyenne Vuong, Cristian Gasparesc, Amirhossein Tavakkoly, William Nazarian,Daniel Bondar, Ryan Guzelian, Imane Madda, Fadi Nimer, Vanisha Patel, Racha Kara.

Reviewers: Triet Pham - Teaching Assistant

Approving Authority: Triet Pham - Teaching Assistant

Issuing Organization: Concordia University

1.2.1 Context:

The Condo Management Systems project is developed to address the needs of modern condominium management, providing a centralized platform for communication, administration, and coordination among stakeholders. It is intended to enhance efficiency, transparency, and convenience in managing condo properties.

1.2.2 Glossary:

Condo: Condominium, a type of housing where individuals own their units but jointly own common areas.

Condo Owner: Individual who owns a unit within a condominium complex.

Rental User: Individual who rents a unit within a condominium complex.

Condo Management Company: Entity responsible for managing and maintaining condominium properties.

Dashboard: Interface providing an overview of relevant information and functionalities.

Financial System: Component responsible for managing financial aspects such as condo fees and operational budgets.

Reservation System: Component facilitating the booking of common facilities within condominium complexes.

Request: Formal submission for various services or actions within the condominium complex.

References:

ISO/IEC/IEEE 42010: Systems and Software Engineering - Architecture Description

1.3 Other information

1.3.1 System or Architecture Overview:

The Condo Management Systems serve as a comprehensive platform for managing condominium properties, catering to the needs of condo owners, rental users, and condo management companies. It offers a user-friendly interface accessible through both mobile and web platforms, ensuring convenience and accessibility for all stakeholders. The system incorporates essential features such as user profile management, property administration, financial tracking, reservation handling, request submission, and notification management. Additionally, it provides supplementary features like forums, event organization, and discounts/offers, enhancing community engagement and satisfaction.

1.3.2 Reader's Guide to this Architecture Description (AD):

This Architecture Description (AD) provides a detailed breakdown of the Condo Management Systems, elucidating its core functionalities, additional features, and bonus features as specified in

the project requirements. It is organized into sections covering system overview, architectural views and models, supplementary information, and other pertinent details. Readers can navigate through the document to gain a comprehensive understanding of the system's architecture, design decisions, and rationale.

1.3.3 Results of Architecture Evaluations:

The architecture of the Condo Management Systems has undergone rigorous evaluations to ensure alignment with project requirements, scalability, security, and performance. These evaluations have resulted in the refinement of architectural components, identification of potential risks, and validation of design choices. The system's architecture has been validated through various means such as architectural reviews, simulations, and prototyping, ensuring its suitability for deployment in real-world scenarios.

1.3.4 Rationale for key decisions

Throughout the development of the Condo Management Systems, key architectural decisions have been documented to provide insight into the reasoning behind design choices. These decisions encompass aspects such as system architecture, technology selection, integration patterns, and trade-offs made to balance conflicting requirements. The rationale for key decisions serves as a reference point for stakeholders, facilitating a deeper understanding of the system's design principles and guiding future development efforts.

Trade-offs in Architectural Decisions:

a) HTML vs. NEXT:

Trade-off: Learn a better language in a short amount of time or stick to something reliable

Decision: Settled with HTML due to lack of knowledge and time constraint

b) Complex UI or Design vs. simple familiar UI:

Trade-off: A UI with flair that is unstable versus a boring looking UI that doesn’t break

Decision: We adopted a simple Ui design to save time, which allowed us to focus more on the essentials, such as backend and testing

2 Stakeholders and concerns

For the Condo360 Systems project, stakeholders span across various groups directly or indirectly interacting with the system. These include residents, property management companies, regulatory authorities, investors, and customer support teams. This section categorizes stakeholders, identifies their primary concerns, and links these concerns to specific quality attributes of the system.

2.1: Stakeholders

2.1.1User

Property Managers: Professionals responsible for the day-to-day operations of a condo.

- **Concerns**: Efficiency in handling service requests, accuracy of financial transactions.
- **Quality Attributes**: Performance, reliability.

Residents (Owners & Tenants): Individuals living within the condo community.

- **Concerns**: Ease of access to information, privacy of personal data.
- **Quality Attributes**: Usability, security.

Service Providers: External companies or individuals providing maintenance and other services.

- **Concerns**: Streamlined request handling, timely payments.
- **Quality Attributes**: Interoperability, efficiency.

2.1.2 Software Provider Organization

Developers & Architects: Those who build and maintain the system's architecture.

- **Concerns**: Scalability, maintainability, and adaptability of the system.
- **Quality Attributes**: Scalability, maintainability.

Project Managers: Individuals overseeing the project development.

- **Concerns**: Delivery within time and budget, resource allocation.
- **Quality Attributes**: Efficiency, feasibility.

2.2: Concerns

System Purpose(s): The system aims to streamline condo management by facilitating communication, financial transactions, and service requests between residents and property managers.

Architecture Suitability: The chosen architecture must support modularity, scalability, and security to meet the system's purposes effectively, accommodating future enhancements without significant rework.

Feasibility: Assessing technical, financial, and timeline feasibility is crucial to ensure the system can be developed and deployed as planned, considering available resources and constraints.

Potential Risks and Impacts: Identifying risks such as data breaches, system downtime, and scalability challenges, and planning for their mitigation to protect stakeholder interests throughout the system's lifecycle.

Maintenance and Evolution: Establishing a clear plan for system updates, bug fixes, and the addition of new features to adapt to changing user needs and technological advancements.

3 Viewpoints+

3.1 Condo Information Management Viewpoint

Rationale: This viewpoint is designed to address the concerns related to the effective management and retrieval of information within the condominium system. It specifically frames concerns from stakeholders such as property managers, residents, and administrative staff who need access to various types of information.

3.2 Overview

The "Condo Information Management Viewpoint" is a key architectural perspective designed to address the critical concerns surrounding the effective management and accessibility of information within the condo management web application. With a focus on stakeholders such as property managers, residents, and administrative staff, this viewpoint aims to ensure seamless access to vital condo-related data. By employing Entity Relationship Diagrams (ERD) and Information Flow Diagrams, it visualizes the relationships between entities, such as residents, units, and common areas, and illustrates the flow of information within the system. The viewpoint emphasizes standardized notations and access control specifications to maintain data integrity and security. Compliance with ISO/IEC/IEEE 42010, 7 ensures a well-documented and standardized approach to addressing the intricate concerns associated with condo information management in the architectural design.

3.3 Concerns and stakeholders

The Condo Information Management Viewpoint addresses critical concerns related to the effective and secure management of information within the condominium system. Foremost among these concerns is the need for seamless.

- **Information Accessibility:** ensuring that property managers, residents, and administrative staff can effortlessly retrieve relevant condo information.
- **Data Integrity:** guaranteeing the accuracy and consistency of the information stored in the system.
- **Security and Privacy concerns:** acknowledging the sensitive nature of condo-related data and implementing measures to safeguard against unauthorized access or data breaches. By focusing on these concerns, the viewpoint strives to create a robust foundation for the information management aspects of the condo management web application, fostering a system that is not only efficient but also trustworthy and compliant with privacy and security standards.

3.3.2 Typical stakeholders

Stakeholders:

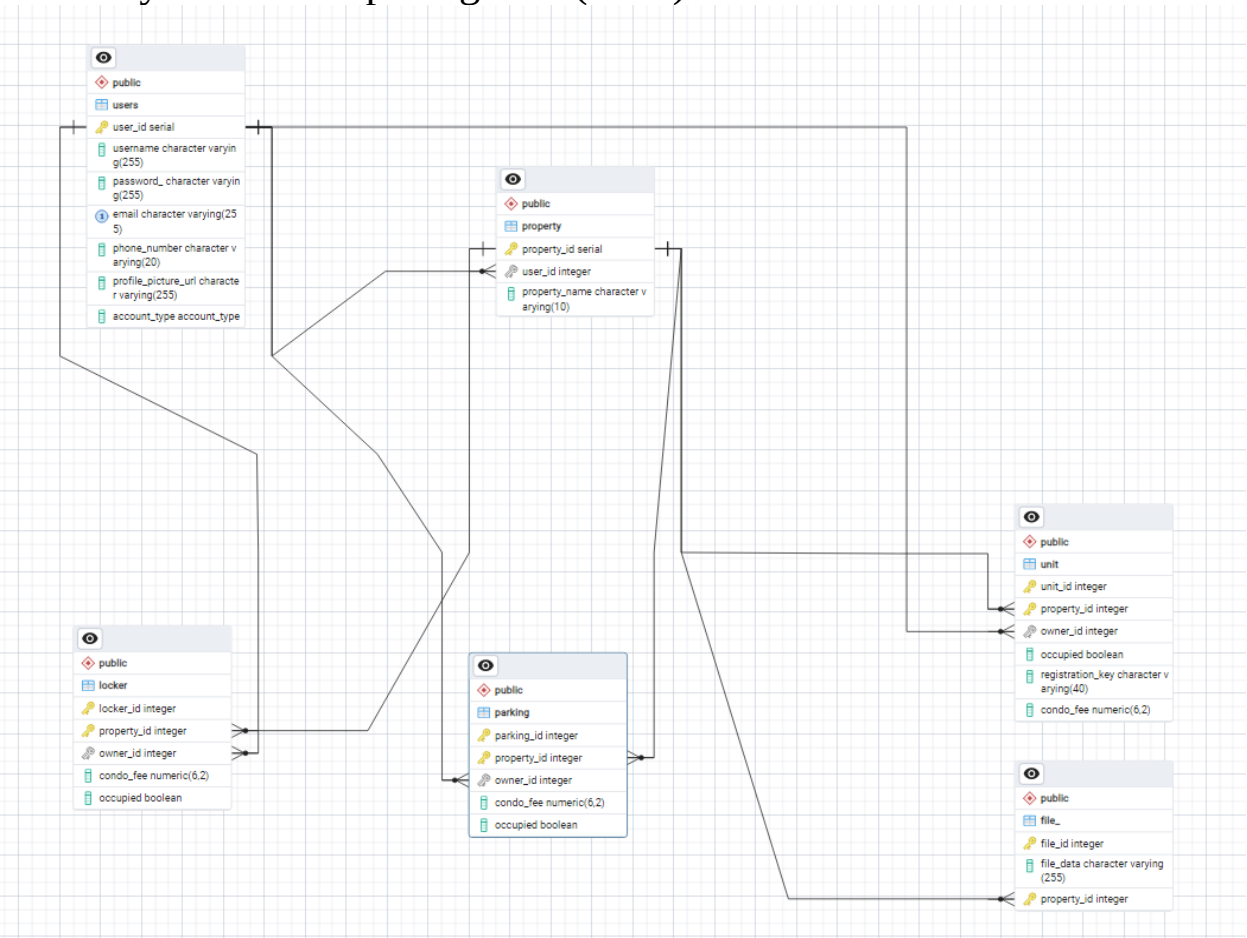
- Property Managers

- Residents

- Administrative Staff

3.4 Model kinds+

3.5 Entity Relationship Diagrams (ERD)



3.5.1 ERD conventions

When creating ERD, it is vital to follow notation conventions for consistency and clarity. Some standard notation conventions include:

Primary Key: In an ERD, the primary key of an entity is underlined. The primary key uniquely identifies each instance of an entity and is crucial for maintaining data integrity.

Foreign Key: A dashed underline in an ERD denotes a foreign key. It signifies a reference to the primary key of another entity, establishing a relationship between the two entities.

3.6 Operations on views

Operations define the methods to be applied to views and their models. Types of operations include:

- **construction methods**:
- List entities (e.g., Residents, Units)

  Define relationships

Specify attributes for each entity (e.g., Resident: Name, Contact)

Determine Relationships and Cardinalities

Create draft

- **interpretation methods**:

  Overview:

  The Entity Relationship Diagram (ERD) visually represents the structure of the Condo Information Management System. This guide provides a quick reference for interpreting key elements and relationships.

  Entities are denoted by rectangles

  Attributes are listed within entities, detailing specific characteristics. Example: Resident entity includes Name, Contact, and Unit Number as attributes.

  One-to-One: Indicated by a straight line between entities.

  One-to-Many: Represented by a line with a crow's foot symbol pointing to the 'Many' side.

  Many-to-Many: Shown with crow's foot symbols on both ends.

- **analysis methods** are used to check, reason about, transform, predict, and evaluate architectural results from this view, including operations which refer to model correspondence rules.

  Ensure that entities, relationships, attributes in the ERD align with corresponding representations in other views, like classes in Class Diagrams.

  Ensure Cardinality Alignment

  Validate Constraints

  Review Notation Consistency

- **implementation methods** are the means by which to design and build systems using this view.

  Map entities from the ERD to classes in the Class Diagram. Each entity becomes a class, capturing both data and behavior in the object-oriented model.

  Transform ERD attributes into class attributes, preserving data characteristics.

  Convert ERD relationships into associations between classes in the Class Diagram. Maintain consistency with cardinalities and multiplicity.

  If the ERD depicts many-to-many relationships, introduce association classes in the Class Diagram to represent these relationships explicitly.

  Determine methods (functions or procedures) associated with classes based on the operations implied by ERD relationships.

Another approach to categorizing operations is from Finkelstein et al. [2]. The work plan for a viewpoint defines 4 kinds of actions (on the view representations): *assembly actions* which contains the actions available to the developer to build a specification; *check actions* which contains the actions available to the developer to check the consistency of the specification; *viewpoint actions* which create new viewpoints as development proceeds; *guide actions* which provide the developer with guidance on what to do and when.

3.7 Correspondence rules

Correspondence with Use Case Diagrams:

Rule: Entities and relationships in the ERD should be associated with relevant use cases in Use Case Diagrams, clarifying the functionalities related to these entities.

Rationale: Establishes a connection between the data model and the system's functional requirements, providing a comprehensive understanding of the system's behavior.

Correspondence with Class Diagrams:

Rule: Entities in the ERD should be related to corresponding classes in Class Diagrams, providing a broader perspective on how these entities are encapsulated into classes in the object-oriented paradigm.

Rationale: Establishes a connection between the data model and the object-oriented design, supporting a seamless transition from conceptual models to implementation.

Correspondence with Sequence Diagrams:

Rule: Entities and relationships in the ERD should be correlated with interactions in Sequence Diagrams, depicting the chronological order of events related to these entities.

Rationale: Provides a time-based perspective on how entities interact within the system, supporting the understanding of system dynamics.

3.9 Sources

Chen, Peter (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems*. **1** (1): 9–36. CiteSeerX 10.1.1.523.6679. doi:10.1145/320434.320440. S2CID 52801746.

Entity–relationship model - Wikipedia

ERD Tool — pgAdmin 4 8.3 documentation

4 Views+

An Architecture Description (AD) integrates multiple architecture views, each conforming to the conventions of a defined architecture viewpoint. This chapter delineates the structuring of viewpoints within the AD, addressing the essential concerns identified in section 2.2 through specified viewpoints. Each viewpoint provides a rationale linked with its stakeholder relevance, framed concerns, and the model kinds utilized.

4.1 Logical Viewpoint

Overview: The Logical Viewpoint of the Condo360 System (CMS) provides a comprehensive view of the software's core conceptual foundation and operational functionality. It defines the system's essential structure, portraying how key entities such as condo owners, properties, and management companies interact within the CMS environment. This viewpoint is instrumental in illustrating the relationships between various system components, including users, properties, financial transactions, and service requests, thus ensuring the system's functionalities align with business objectives and user needs.

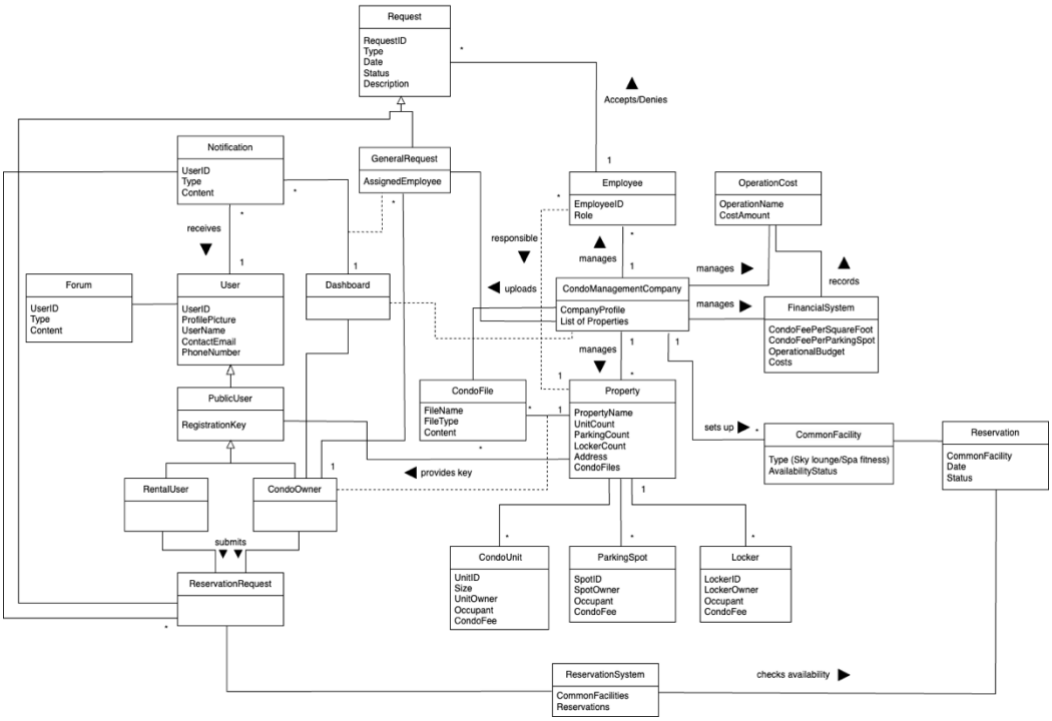
Concerns and stakeholders

- **Concerns**:
    - **System Interaction**: Understanding how different user roles (condo owners, tenants, management staff) interact with the CMS to perform their tasks is crucial. This includes navigating the interface, accessing property information, making payments, and submitting service requests.
    - **Core Functionalities**: Identifying the CMS's primary capabilities, such as profile management, property listing, financial management, service request handling, and facility booking. Ensuring these functionalities are well-defined and meet the users' and business' needs.
    - **Entity Relationships**: Clarifying the relationships between key entities in the system, such as the association between condo units and their owners, the allocation of parking spots, and the management of common areas.
- **Typical stakeholders**:
    - **Business Analysts**: Responsible for ensuring the system's functionalities align with the business requirements and user needs. They are concerned with the system's usability and the efficient representation of business processes within the CMS.
    - **Software Developers**: Focus on implementing the logical model into a functioning software system. They require a clear understanding of the system's entities, their attributes, and interactions to develop a robust and maintainable codebase.
    - **System Users**: Include condo owners, tenants, and management company staff who interact with the system. Their primary concern is the ease of use, access to necessary functionalities, and the efficiency of the CMS in managing their properties and related activities.
- **Known Issues with View**:
    - Difficulty in modeling complex relationships between condos and their amenities.
    - Challenges in ensuring the class diagram accurately reflects all user interactions without becoming overly complex.
    - Potential discrepancies in representing real-world business rules within the constraints of the chosen development framework.

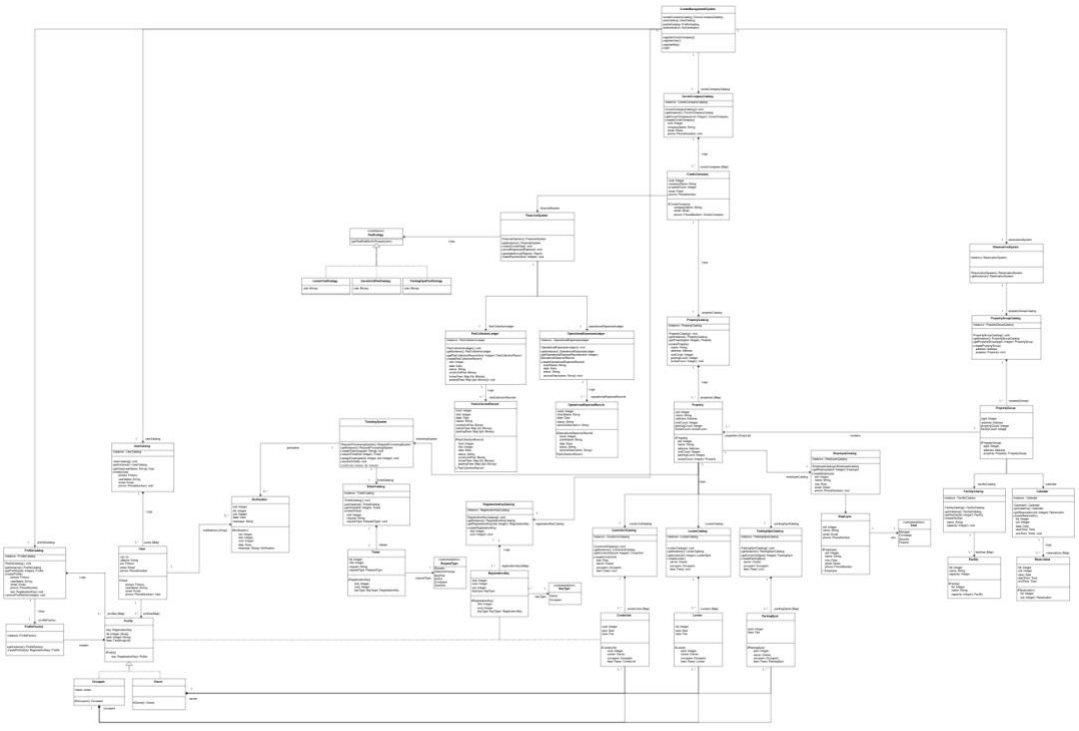
Model kinds: UML Class Diagrams, Domain Models.

- **Domain Models**

Domain Model



- **UML Class Diagrams**



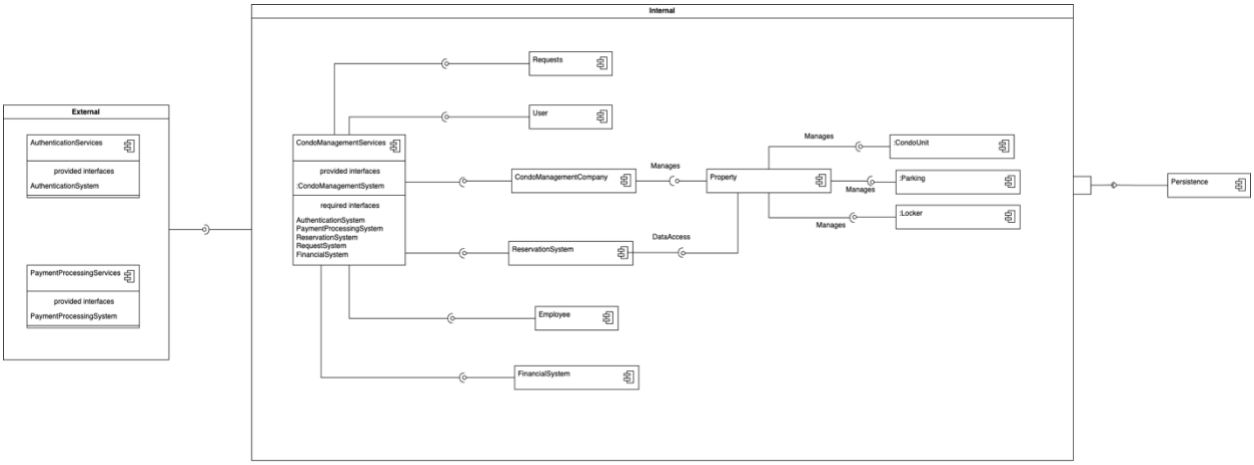
4.2 Development Viewpoint

Overview:
The Development Viewpoint of the Condo360 System (CMS) zeroes in on the system's architecture from a software development standpoint. It intricately details the structuring of software modules, their interdependencies, and the setup required for an efficient and effective development environment. This viewpoint is essential for understanding how the CMS's architecture supports modular development, facilitates maintenance, and enables scalability. It underscores the logical partitioning of the system into components, each responsible for a distinct piece of functionality, and how these components interact to form a cohesive system.

The focus on software modules and their dependencies is crucial for identifying potential areas for code reuse, optimizing system performance, and simplifying the integration of new features or services. Additionally, the Development Viewpoint addresses the setup of the development environment, including version control systems, development frameworks, and deployment pipelines, ensuring that the environment promotes productivity and supports the project's technical requirements.

Model Kinds: ComponentDiagram

- **Component Diagram**



Concerns and Stakeholders

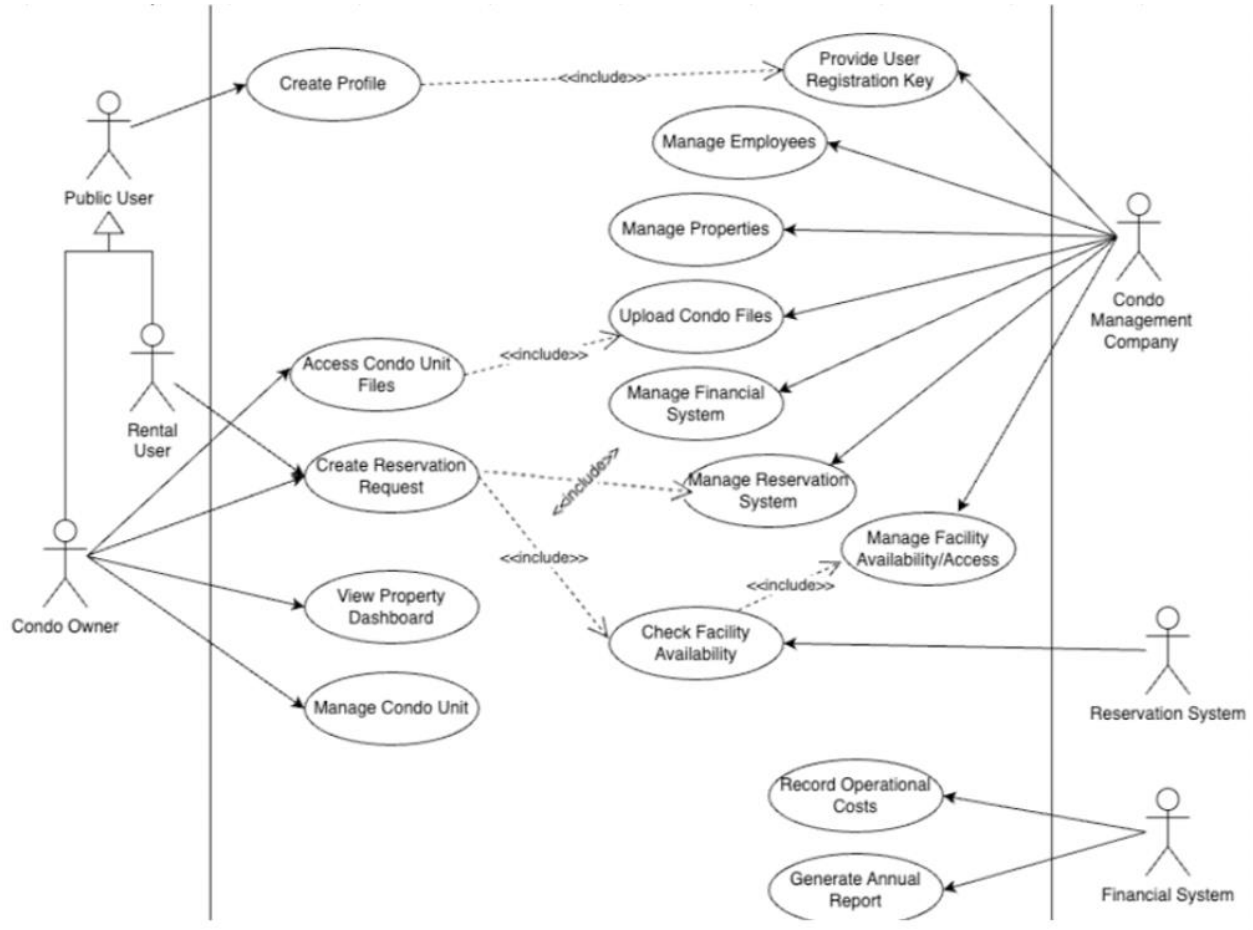
- **Concerns**:
  - **Modularity**: Ensuring the system is divided into well-defined, loosely coupled components that can be developed, tested, and deployed independently.
  - **Dependencies**: Identifying and managing dependencies between components to minimize coupling and facilitate easier integration and testing.
  - **Development Environment**: Establishing a development environment that supports continuous integration/continuous deployment (CI/CD) practices, automated testing, and other DevOps principles to enhance development efficiency and reduce time-to-market.
- **Typical Stakeholders**:
  - **Software Architects and Developers**: Interested in the structural organization of the CMS to ensure code quality, maintainability, and scalability. They rely on the Development Viewpoint to guide the system's architectural design and implementation.
  - **DevOps Engineers**: Focus on the deployment pipelines and development practices. They use the Development Viewpoint to streamline development operations and automate the build, test, and deployment processes.
  - **Project Managers**: Concerned with the overall project timeline and resource allocation. Understanding the system's modularity and dependencies helps in planning sprints and allocating tasks efficiently.
- **Known Issues with View**:

- Challenges in defining clear interfaces between components to ensure loose coupling and high cohesion.
- Potential issues in integrating third-party services for payment processing and notification services.
- Difficulty in ensuring that the component architecture supports scalability and maintainability as new features are added or as the user base grows.

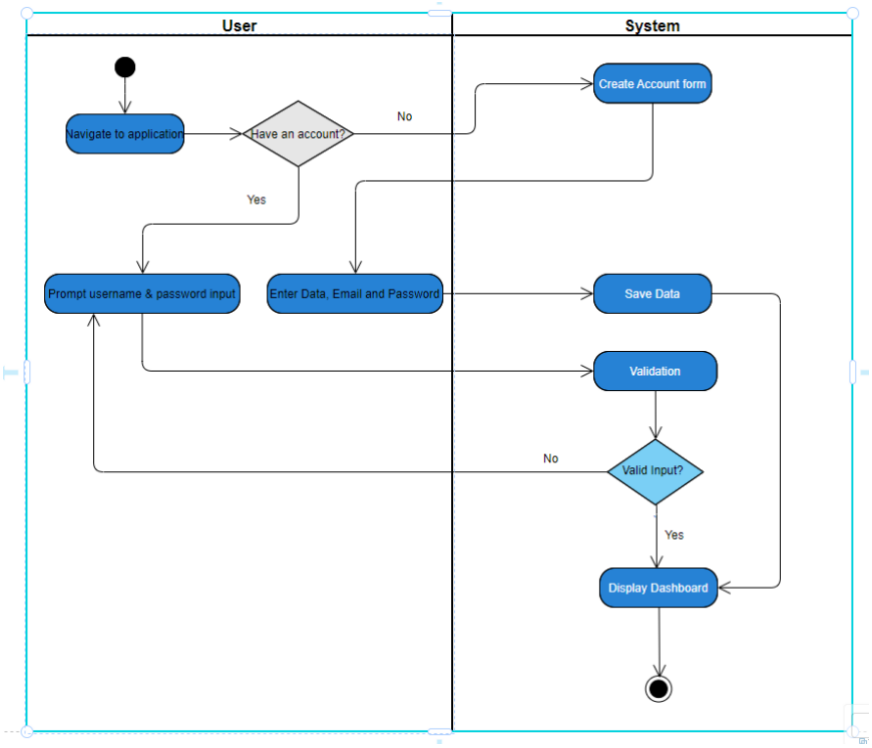
4.3 Process Viewpoint

Overview:
The Process Viewpoint for the Condo360 Systems project is centered on illustrating how the system operates in real-time, detailing the interactions between users and the system, and among the system's own components. It aims to provide a clear understanding of the workflow, user actions leading to system responses, and how data flows through the system during operation. This viewpoint is crucial for identifying potential bottlenecks, optimizing user experience, and ensuring seamless operation across different functionalities of the Condo360 System.

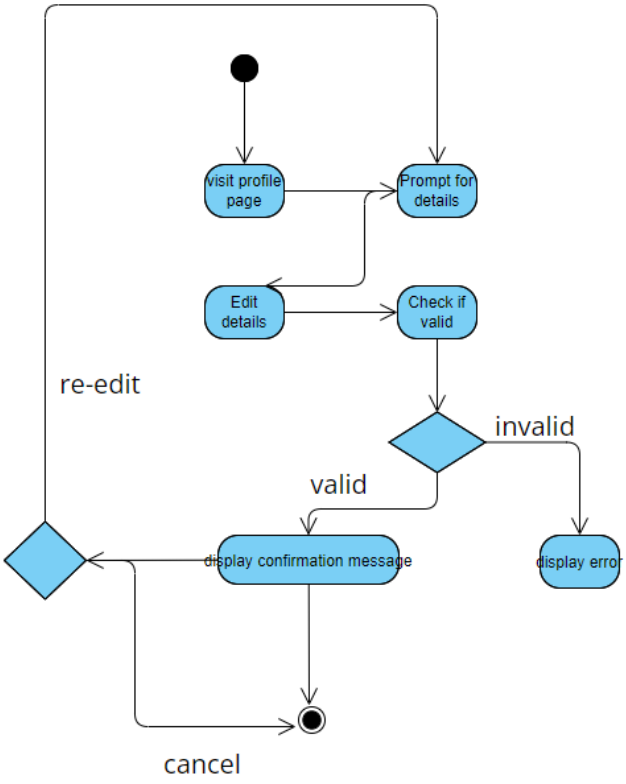
Model Kinds: UseCase Diagram, Activity Diagrams



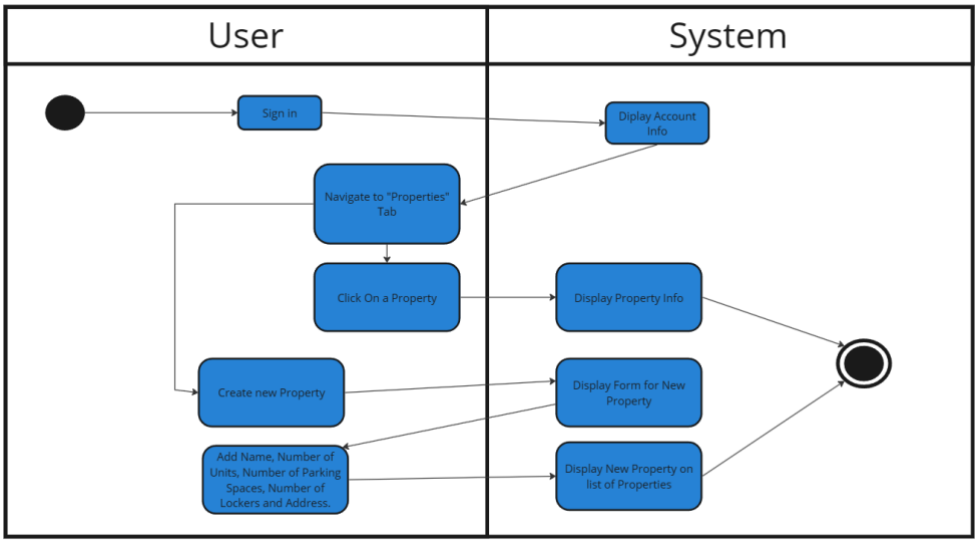
Sign Up/Login Activity Diagram:



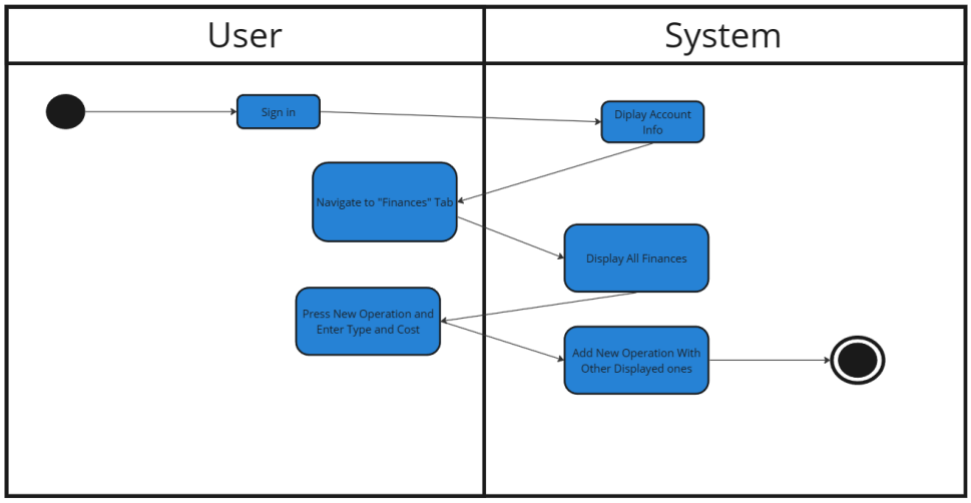
Profile Activity Diagram:



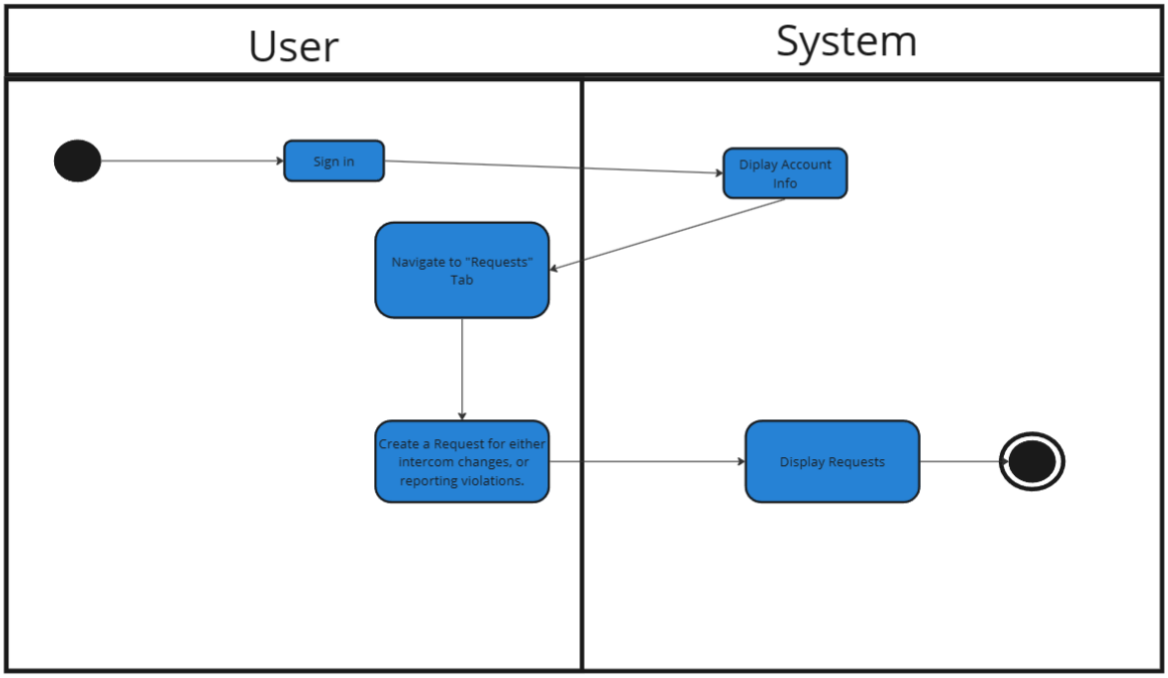
Property Activity Diagram:



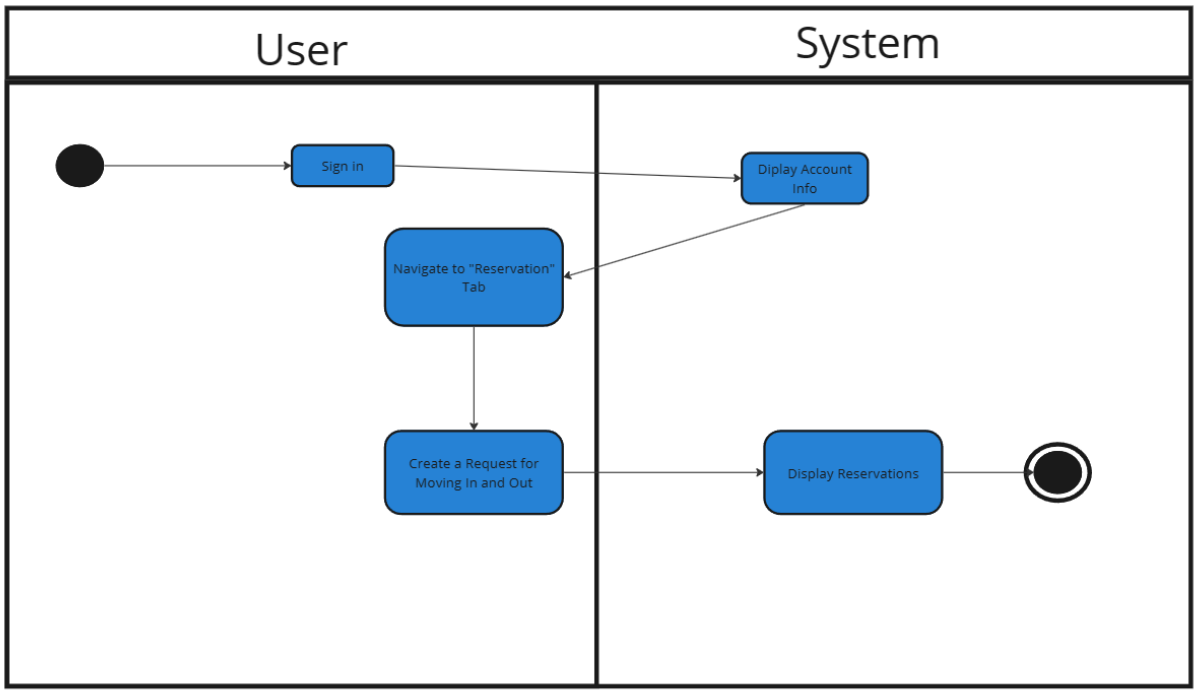
Finances Activity Diagram:



Requests Activity Diagram:



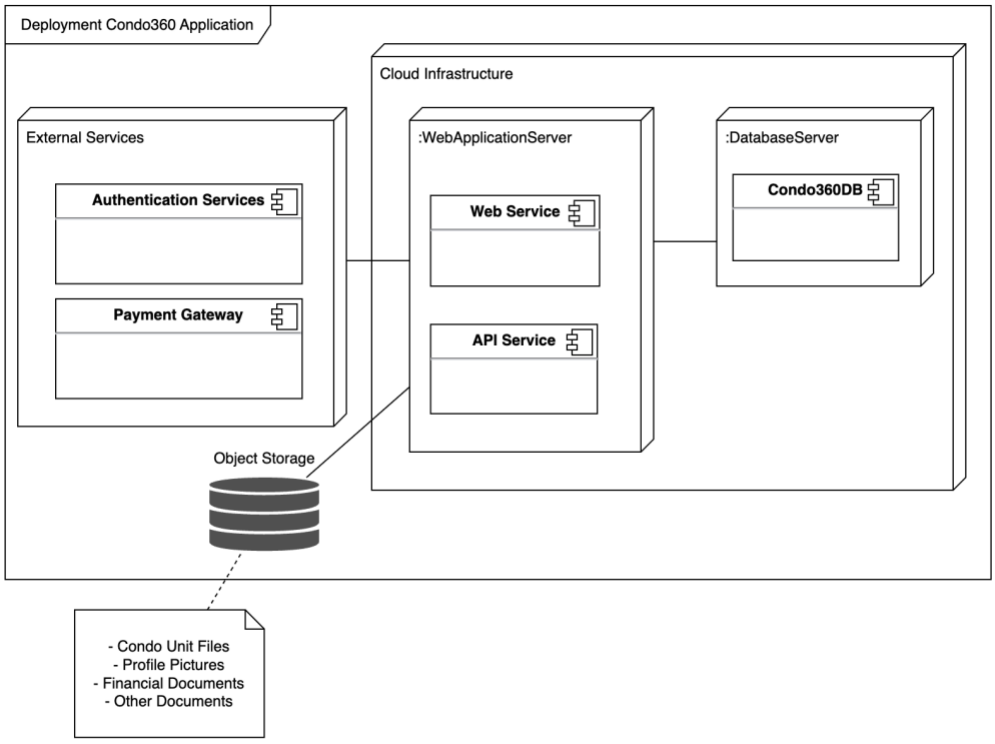
Reservation Activity Diagram:



4.4 Physical Viewpoint (Deployment Viewpoint)

Overview:
The Physical Viewpoint of the Condo360 Systems project provides a depiction of how the software components are allocated across the system's infrastructure, encompassing both physical servers and virtual environments. This viewpoint is crucial for understanding the deployment strategies, scaling options, and how different components communicate over the network. It covers the deployment of databases, application servers, web servers, and other services that make up the system, specifying their runtime environments and configurations for optimal performance and reliability.

Model Kinds: Deployment Diagram



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.

5.2 Correspondences in the AD

To maintain coherence in the Condo Management Systems project, correspondences will be identified across various architecture description (AD) elements such as stakeholders' concerns, viewpoints, views, and model kinds. A systematic approach will be used to map these elements, ensuring that every architectural decision, model, and viewpoint directly addresses identified concerns and stakeholder requirements. This alignment will be documented through UML diagrams and tables, facilitating traceability and consistency across the architecture.

5.3 Correspondence rules

Stakeholder Concern Alignment: Every architectural model or decision must trace back to at least one stakeholder concern.

Model Integrity: Models within views must remain consistent with each other, avoiding conflicting information or duplication.

Accurate labeling of nodes: Model nodes must have a corresponding description.

Documentation of correspondences: All correspondences between stakeholders' concerns and views must be documented.

Consistency across views and models: These correspondence rules will ensure consistency across views and models. Which will help to avoid contradictions or ambiguities in the AD.

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.

4. RISK ASSESSMENT & RISK MANAGEMENT PLAN

Condo Management System

RISK ASSESSMENT AND MANAGEMENT PLAN

Team 7

Version *4.0*

11/04/2024

Table of Contents

1 INTRODUCTION

1.1 PURPOSE OF RISK ASSESSMENT AND MANAGEMENT PLAN

A risk is a potential problem that can occur. It can have harmful consequences on the project's objectives in the future. Risk management is to identify and evaluate potential problems before they occur. The objective is to develop a plan to effectively address and mitigate risks.

The Risk Assessment and Management Plan (RMP) is an important document. It provides the team important information on possible problems. This helps the team to make informed decisions and prioritize tasks and risks during the project. The RMP will be monitored, documented and updated throughout the entire project lifecycle.

2 RISK MANAGEMENT PROCEDURE

2.1 PROCESS

All team members working on the project will be identifying, analyzing and managing risks throughout the project lifecycle. To reduce project delays and impacts, risks will be identified and accessed in the early stages of the project. To ensure that the documentation follows the RMP guidelines, one team member will be the Risk Manager (Vanisha Patel).

2.2 RISK IDENTIFICATION

To identify, document and manage risks during the project's lifecycle, there are going to be weekly team meetings to discuss and review each team member's progress on their assigned tasks to encourage team members to share any issues or challenges that they face because they could become risks. A risk analysis log file is accessible to all team members. The risk manager will take note of the concerns and analyze the risk identified.

2.3 RISK ANALYSIS

The team is using qualification analysis and is reporting the level of risk in the Risk Reporting Matrix to prioritize what risks should be addressed and which can be ignored. All risks identified are analyzed in a risk analysis table that provides information on ranking, threats, vulnerabilities, contextual factors, risks, likelihood, impact, reduction of threats, reduction of vulnerabilities and level of residual risk. These fields are analyzed by the risk manager.

2.4 RISK RESPONSE PLANNING

In the Risk Reporting Matrix, the risks will either be in the green, yellow or red zone. The risks can be addressed by avoiding, mitigating, accepting or transferring. Team members will be assigned to monitor risks falling in the red and yellow zone.

- **Avoid:** Avoiding the risk by avoiding the cause of this risk.

- **Mitigate:** Mitigating risk by minimizing its impacts on the system.
- **Accept:** Accepting the risk - nothing to be done.

To mitigate risks, the team members will find solutions to prevent risk from happening and reduce its impact and probability. The solution will be proposed to the Risk Manager and the manager will be verifying the solution's feasibility and will add and assign additional risk prevention tasks in the project schedule. This will minimize risk impacts.

2.5 RISK MONITORING, CONTROLLING, AND REPORTING

Important risks will be assigned to team members for monitoring, controlling and reporting during the project. The risk manager will be notified of project changes resulting from risk monitoring and controlling. Risks will be reported in the risk analysis table.

3 TOOLS AND PRACTICES

The risk analysis table serves as a log for risks for all team members. It will be maintained by the risk manager to ensure that it follows formatting guidelines. The Risk Reporting Matrix will be used for qualification analysis.

4 RISK ANALYSIS AND MANAGEMENT

4.1 RISK ANALYSIS TABLE

The risk analysis table helps to understand the severity of the risk by ranking them. The required information in the table are ranking, threats, vulnerabilities, contextual factors, risks, likelihood, impact, reduction of threats, reduction of vulnerabilities and level of residual risk. It helps the team to decide on how to address the risk.
The detailed risk analysis table can be found in the file Updated Risk Analysis.xlsx. Examples of two risks related to the condo management system project are shown in Figure 1.

Risk ID	Ranking	Threats (1)	Vulnerabilities (2)	Contextual factors	Risks (3)	Likelihood from 1 to 5	Impact from 1 to 5	Reduction of Threats (Acceptance Strategy)	Reduction of Vulnerabilities (Protection Strategy)	Level of residual Risk
1	9 - Medium	Unauthorized access to User's Profile	- User has a weak password - Sign up does not enforce complex password format - The passwords stored in the database are not encrypted - No multi-factor authentication	- Cyber attacks are common in in software that handles properties and finances - User's use unsecured public network	To be subjected to unauthorized access to user's sensitive data	2	5	- Monitor unusual user activities - Warn users of potential cyber attacks when creating an account	- Implement strong password requirement when user signs up - Implemented strong encryption for passwords - Implement multi-factor authentication	6 - Low
2	16 - High	Insufficient system testing	- The test coverage is low which causes undetected bugs and system failures - Inadequate compatibility testing - Incomplete exeption and error handling - Incomplete security testing	- Insufficient testing caused by tight sprint deadlines - Testing inefficient because of frequent changes in the code	To be subjected to software crashes and security vulnerabilities	5	4	- Allow users to report bugs on the application/website - Train the entire team on software testing and bug reporting	- Write unit, integration and system tests - Reserve five days before deadline for system testing and fixing bugs and error handling - Attain a test coverage of 80%	8 - Medium

Figure 1: Examples of risks from Risk Analysis.xlsx file

4.2 RISK REPORTING MATRIX

The risks identified in the risk analysis table are presented in the risk reporting matrix to visualize its severity. The list of identified risks shown in table 1 is a summary of the content of the risk analysis file. The risk reporting matrix is shown in figure 2. The risks are represented by their risk ID.
The risk manager analyzes the probability and impact of each identified risk.
Risks Probability:

- **High:** Greater than 70% probability of occurrence
- **Medium:** Between 30% and 70% probability of occurrence
- **Low:** Below 30% probability of occurrence

Risk Impact and Assessment:

- **High:** Risk that has the potential to greatly impact project cost, project schedule or performance
- **Medium:** Risk that has the potential to slightly impact project cost, project schedule or performance
- **Low:** Risk that has relatively little impact on cost, schedule or performance

Risk ID	Risk Type and Description	Risk Score	Resolved in Sprint	Strategy and Effectiveness	Delays Caused by Risk
R-1	**Security:** Unauthorized access to user's profile	Medium	4	Mitigate	This risk would cause up to 2 days of delay, because it would require us to identify and investigate the data breach.We will then assess the impact, implement security measures to prevent further unauthorized access, and potentially restore any compromised data.

R-2	**Technical:** Insufficient system testing	High	2	Mitigate	This risk would result in delays ranging from days to weeks, depending on the severity of the issues discovered and the complexity of the system.
R-3	**Security:** Data Breach	Medium	4	Accept	A data breach can cause delays ranging from hours to days or even longer, depending on the scale of the breach and the sensitivity of the compromised data.
R-4	**Management:** Poor communication within team	Medium	2	Mitigate	This could lead to delays ranging from hours to days. At best, it would slightly slow down the team's progress. At worst, it would lead to a regression if the team lacks collaboration and the project has many conflicts. Misunderstandings, redundant work, could also lead to delays that could cause missed deadlines.
R-5	**Management:**	High	4	Accept	This could lead to delays ranging

	Unqualified teammates for mobile app development				from hours to days, as team members will need more time to learn the technologies necessary for the development of the app.
R-6	**Requirements:** Poor requirements definition	Medium	3	Mitigate	This can lead to delays ranging from days to weeks, as unclear or ambiguous requirements may require significant rework and adjustments to the project scope. In the worst case, the progress made can even be scrapped if the implementation does not align with the requirements.
R-7	**Data:** Condo owner dashboard displaying incomplete or inaccurate information	Low	2	Mitigate	This could result in a few hours of delay, as resolving data inaccuracies requires identifying the root cause, correcting the data, and ensuring that the dashboard displays accurate information.

R-8	**Security:** User forgets password and is not able to access their account	High	4	Mitigate	This could result in short delays of a few seconds to a few minutes, depending on the efficiency of the password recovery process.
R-9	**Data:** Managing large numbers of documents	Low	2	Accept	This could lead to delays ranging from hours to days, as managing a large volume of documents requires time to organize, categorize, and retrieve relevant information efficiently.
R-10	**Data:** Data access violation for condo management companies information	Low	4	Accept	This could result in delays depending on the severity of the violation and the legal implications involved The delays could range from days to weeks.
R-11	**Management:** Poor time management and organization	High	3	Accept	This could lead to delays ranging from hours to days, as poor time management and organization can result in missed deadlines, inefficient use of resources.
R-12	**Tools:**	Low	3	Mitigate	This could result in delays ranging

	Unable to connect to database for developing system				from hours to days, as resolving connectivity issues requires identifying the cause, troubleshooting the problem, and implementing fixes or workarounds.
R-13	**Code Management:** Confusing repository structure	High	3	Mitigate	This could lead to hours of delay, as a confusing repository structure hinders code collaboration, version control, and code review processes.
R-14	**Deployment:** Data Migration Errors Leading to Data Loss or Corruption	High	5	Mitigate	The impact of data migration errors can range from minor data discrepancies or inconsistencies that require manual reconciliation, to significant data loss or corruption that necessitates data recovery processes. The duration could range from minutes to days.
R-15	**Deployment Risk:** Inadequate Scalability Planning	Medium	5	Mitigate	The delays caused by this risk depends on how quickly the issue is identified

	Leading to Performance Issues				and addressed, ranging from hours or longer. This risk can cause a decrease in performance and an increased response time. In the worst case, it would even cause a service outage. These problems will persist until appropriate scalability measures are implemented.

Table 1: List of identified risks

Impact	H	**R-3**	**R-14**	**R-2, R-4, R-11, R-13**
	M	**R-7, R-10, R-15**	**R-1, R-6**	**R-5, R-8**
	L	**R-9**	**R-12**	
		L	M	H
	Probability			

Figure 2: Risk Reporting Matrix

5. SPRINT 1 DELIVERABLES

5.1. SPRINT RETROSPECTIVE

Behind the Scenes of Sprint 1

Introduction

As we complete our project's first sprint, it is essential to look back and analyze our progress in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Entreprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system, and a financial system.

Upon the concept's announcement, we had high expectations and were excited by the opportunity that this project represented. We believed that by being 10 people in a team, we would easily manage any type of requirements the teacher would define. However, we quickly realized the magnitude of the task when going over this first sprint's deliverables. Even after overcoming ambiguity issues, we faced obstacles regarding task management, dependency, and schedule flexibility. Furthermore, the sprint's duration was reduced by a week. This pushed us to adapt more quickly and set up our work environment as soon as possible. We also scheduled a meeting with our TA to make sure that our understanding of the requirements was accurate.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement towards the lack of communication amongst team members and the issues it led to.

What went wrong

1 – Scope Ambiguity

During this first sprint, we noticed that most of the challenges we encountered stemmed from an unclear understanding of the project scope and the requirements. The issue started when we first received the grading schemes per sprint and, due to the short and vague description of the first sprint's deliverables, some of us misunderstood the teacher and TA's expectations. This led to confusion about what needs to be achieved during this sprint and to some delay in the progress.

To address this difficulty, we scheduled a meeting with the TA as soon as possible in order to clarify the project's and the sprint's scopes. In addition, we held a team meeting to ensure that all members had the same understanding of the deliverables and knew their responsibilities. These measures definitely dissipated some of the confusion and improved our ability to plan a detailed strategy, decreasing the risk of potential scope-related issues by the same occasion.

2 – Dependency

Dependency challenges were one of the main causes of workflow disruption and delay during this first sprint. First, receiving the project requirements late decreased the sprint duration from three weeks to two weeks, forcing us to adapt and compromise in terms of time management during this iteration. Furthermore, some deliverables were dependent on other parts that were still under development, causing the responsible team members to have to wait before beginning. The impact of these challenged started showing towards the end of the sprint.

To reduce the issues created by this challenge, we each made sure to update the rest of our team members when working on our parts or when facing an issue. In addition, we improved communication between team members by having two teammates exchange in private instead of disturbing everyone with a question that concerns only one's part. By helping each other out when needed and aiming for better internal coordination, we can affirm that the overall workflow was smoother by the end of the sprint.

3 – Meeting Scheduling

In order to keep track of our progress, we needed to schedule a weekly online meeting with as many team members present as possible. However, because of the diverse school and work schedules of each member, we could not find a suitable time slot that accommodated all 10 individuals. This caused us to divide the first sprint's deliverables in a way that minimized collaboration and exchanges. The decision-making process was also affected since we had to set up polls to vote in the discord server instead of discussing and deciding synchronously.

To solve this issue, we used the when2meet scheduling tool to conduct a survey and identify the time slots where most of the team members would be available. Then, we held a meeting during that time and made sure to update the members that could not attend it. In addition, we organized smaller subgroup meetings when needed. However, we still need to establish a more structured and regular meeting schedule in order to avoid the headache of debating on this subject each week.

What went right

1 – Effective Communication Channels

As mentioned in the section above, establishing effective and convenient communication channels quickly significantly contributed to overcoming some challenges and increasing collaboration. Indeed, we set up a discord server as soon as the team was formed and defined separate channels for each sprint in addition to the channel for links (GitHub repository and Google Drive folder). Furthermore, we frequently used our Slack private channel communicate and meet with our TA and, ultimately, ensure that everyone was on the same page.

These measures created a transparent and clear work environment, which led to greater coordination and understanding between team members. It also allowed us to voice any concern we had quickly and openly. However, communication could still be improved by, for example, hosting in-person meetings occasionally and encourage all members to provide their inputs.

2 – Positive Team Dynamics

Since we formed this team from separate friend groups, it was essential to establish a pleasant team dynamic in order to facilitate collaboration and productivity. Fortunately, all members understood this project's magnitude and expressed commitment to its success. In addition, we talked comfortably early and, being students, we could easily relate to each other.

This created a safe space of inclusivity where debates and feedback were welcomed. In addition to overcoming obstacles more efficiently, we maintained a lighthearted dynamic that kept us from feeling overwhelmed by the amount of work that this project represented.

3 – Adaptability to Requirements

During this first sprint where the project scope and requirements were still ambiguous, all team members demonstrated remarkable adaptability through a proactive mindset and clear communication. This was possible thanks to the common goals and commitment we shared for this project.

The team's flexibility also allowed the project's progress to remain on track despite starting the first sprint a week later than expected. By quickly accommodating to the time left, we operated systematically and prioritized momentum. However, we can still improve ourselves by approaching the next sprints in a structured manner in order to reduce the likelihood of misunderstanding instructions of missing an important mistake.

Conclusion

While reflecting on the first sprint, we realized that we encountered multiple challenges that were rooted in scope ambiguity, dependency, and difficulties with scheduling meetings. Although these obstacles were to be expected, we had to adapt quickly and come up with solutions that were convenient for all 10 members of the team. By leveraging platforms like Slack and Discord, we set up effective communications channels that enabled us to update each other and collaborate without necessarily working at the same time. In addition, the positive team dynamics that we created from the start significantly helped each one of us to share ideas freely.

As for the project itself, this first sprint gave us an insight on the amount of work that comes with a system of this magnitude. For our progress to go smoothly and as we plan it, we will need to follow a structured plan and divide tasks according to each member's skills. This does not mean that we must stay rigid since the Agile approach actually embraces flexibility. Furthermore, we recognize the need for continuous refinement of all artifacts because this sprint showed us that it is feasible to develop a running prototype and test it within two weeks. By applying the lessons we learned, we will tackle future challenges being more informed and committed to developing the best product possible.

5.2. RELEASE PLAN

Table 1. User story points and priority of user stories that will be implemented in Sprint 2

User Story ID	User Story Points (USP)	Priority	Status
#40	21	High	TO DO
#42	17	High	TO DO
#45	13	High	TO DO
#46	10	Medium	TO DO
#47	12	Medium	TO DO
Total USP	70		

This user stories (see below) only concern the web application. We decided to focus on the main website for next sprint and leave the development of the companion app to the third sprint.

Sub-User Stories:

#40 - As a public user, I want to be able to create a unique profile by providing a profile picture, username, contact email, and phone number.

- Create a signup form with fields for the username, email, and phone number. (3 points)
- Implement validation for each field to ensure the integrity of the data. (1 point)
- Allow users to upload a profile picture from their device. (2 points)
- Implement an authentication system to secure the users' profiles. (5 points)
- Store all the users' information in a database. (3 points)
- Create a user profile page for each user type. (5 points)
- Make the user interface appealing and user-friendly. (2 points)

#42 - As a condo owner, I want to have a dashboard that displays general information about my properties, including personal profile, condo information, and financial status.

- Create a user profile page for this user type. (2 points)
- Design a dashboard with sections for personal information, condo information and financial status. (2 points)
- Allow the user to select and display personal information from the dashboard. (3 points)
- Allow the user to select and display condo information from the dashboard. (3points)
- Allow the user to select and display information about the financial status from the dashboard. (3 points)
- Make the navigation easily understandable to the user (from the main dashboard to and back from each section. (2 point)
- Make the user interface appealing and user-friendly. (2 points)

#45 - As a condo management company, I want to be able to create profiles for properties under my management, providing essential information like property name, unit count, parking count, locker count, and address.

- Create a form to input information about a new property (name, unit count, parking count, locker count and address). (3 points)
- Implement validation for each field to ensure the integrity of the data. (1 point)
- Store the new condo's information in the database. (3 points)
- Create a table to view and manage all properties. (2 points)

- Add the new properties to the table. (2 points)
- Make the user interface appealing and user-friendly. (2 points)

#46 - As a condo management company, I want to upload condo files for each property, accessible to all condo owners, including declarations, annual budgets, and board meeting minutes.

- Allow this user type to upload condo-related documents through a specific button (within a condo info page). (2 points)
- Develop the user interface for this feature within the page of a condo information. (1 point)
- Stores these files in the database. (3 points)
- Create a small table or list of all uploaded files. (2 points)
- Allow condo owners to download these files and viewing a condo information from their respective property dashboards. (2 points)

#47 - As a condo management company, I want to enter detailed information for each condo unit, parking spot, and locker, including unit size, owner information, and associated condo fees.

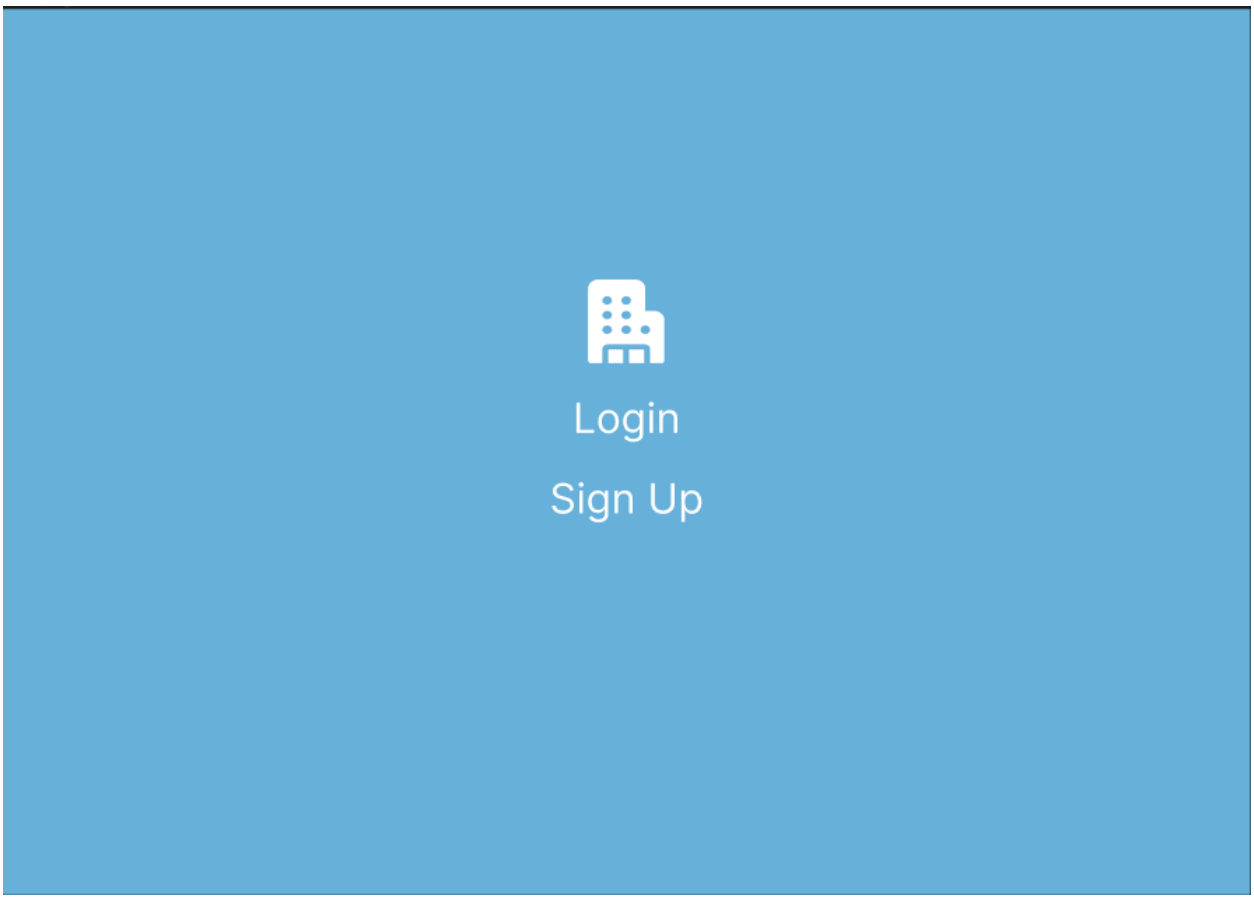
- Create a user interface button to allow this user type to modify information about one of their properties. (2 points)
- Create a form layout where fields are modifiable and can be saved. (3 points)
- Implement validation for each field to ensure the integrity of the data. (1 point)
- Save the new information in the database. (3 points)
- Make the information displayed on the dashboard updated automatically. (1 point)
- Make the user interface appealing and user-friendly. (2 points)

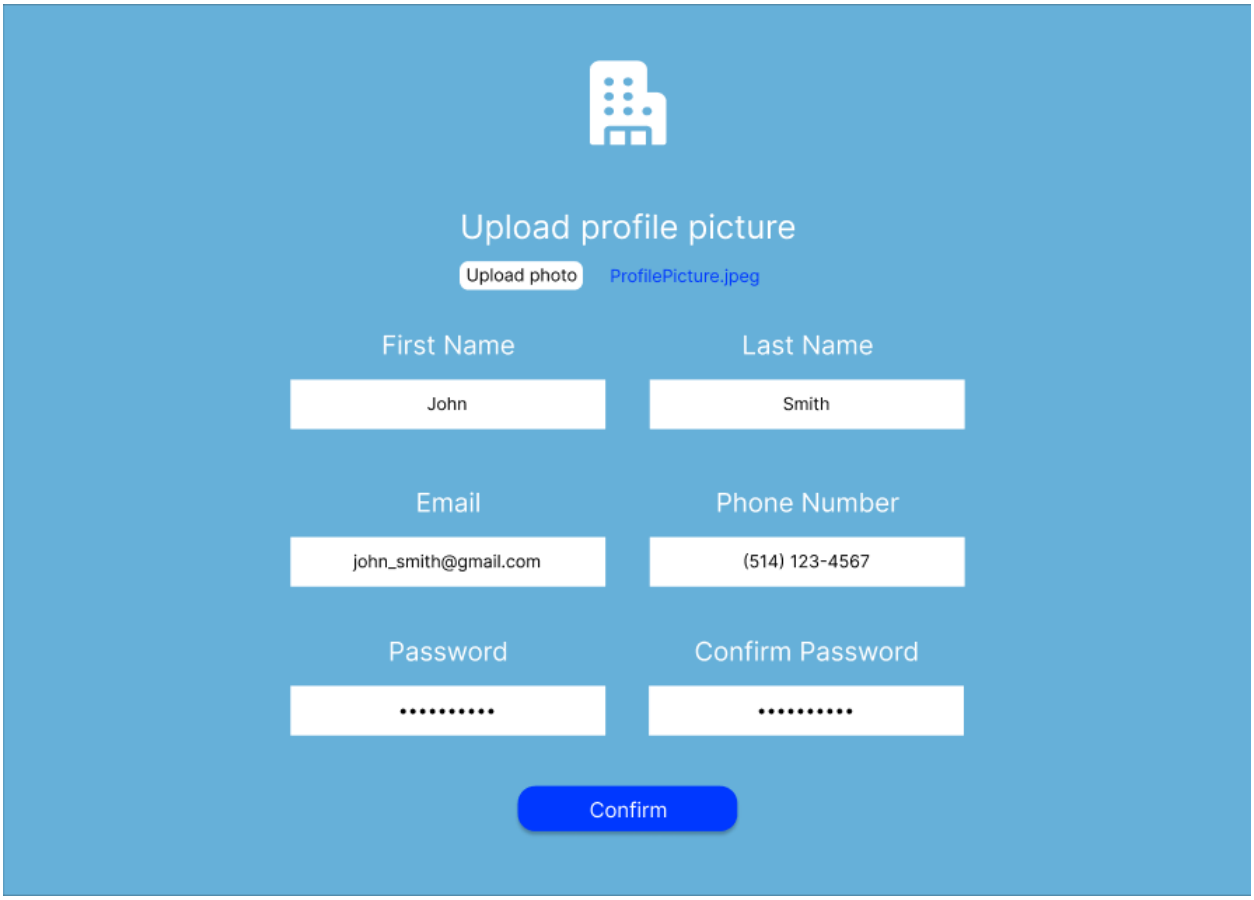
Future Deployment on Cloud Services or Devices

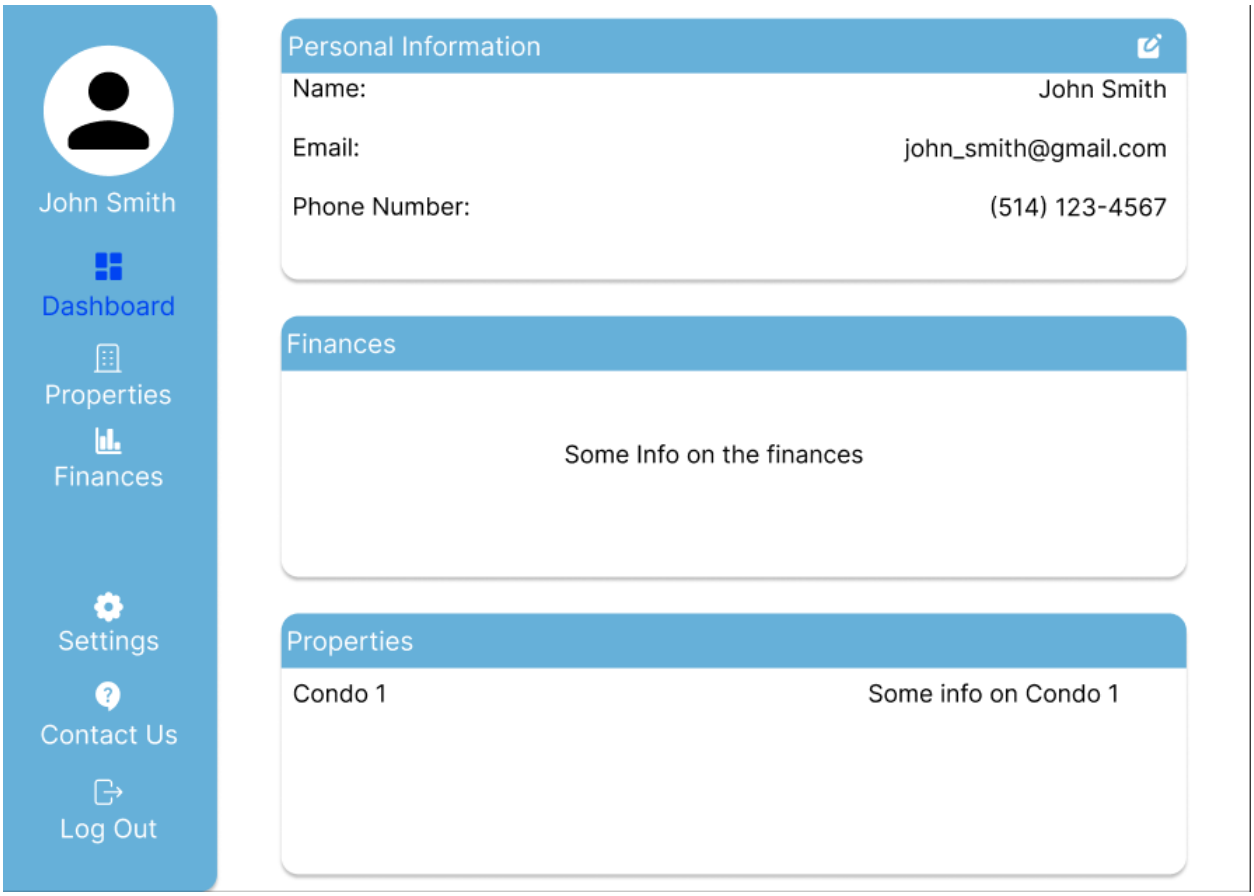
Considering that this condo management application will be developed with care and intent to make it as optimal as possible, deploying it on cloud services after its completion would make our project more professional and impressive. Moreover, cloud deployment would allow us to accommodate a larger customer base and increasing demand if the need arises. This would be possible since access to the platform would be enabled from any device connected to the internet, and our sensitive data would be better protected. In fact, utilizing cloud services to our advantage would reduce maintenance effort, which we could allocate towards improving our application. Overall, there are great benefits that would ultimately enhance user experience, making cloud deployment a potential possibility.

5.3. UI PROTOTYPE

US-40: As a public user, I want to be able to create a unique profile by providing a profile picture, user name, contact email, and phone number.



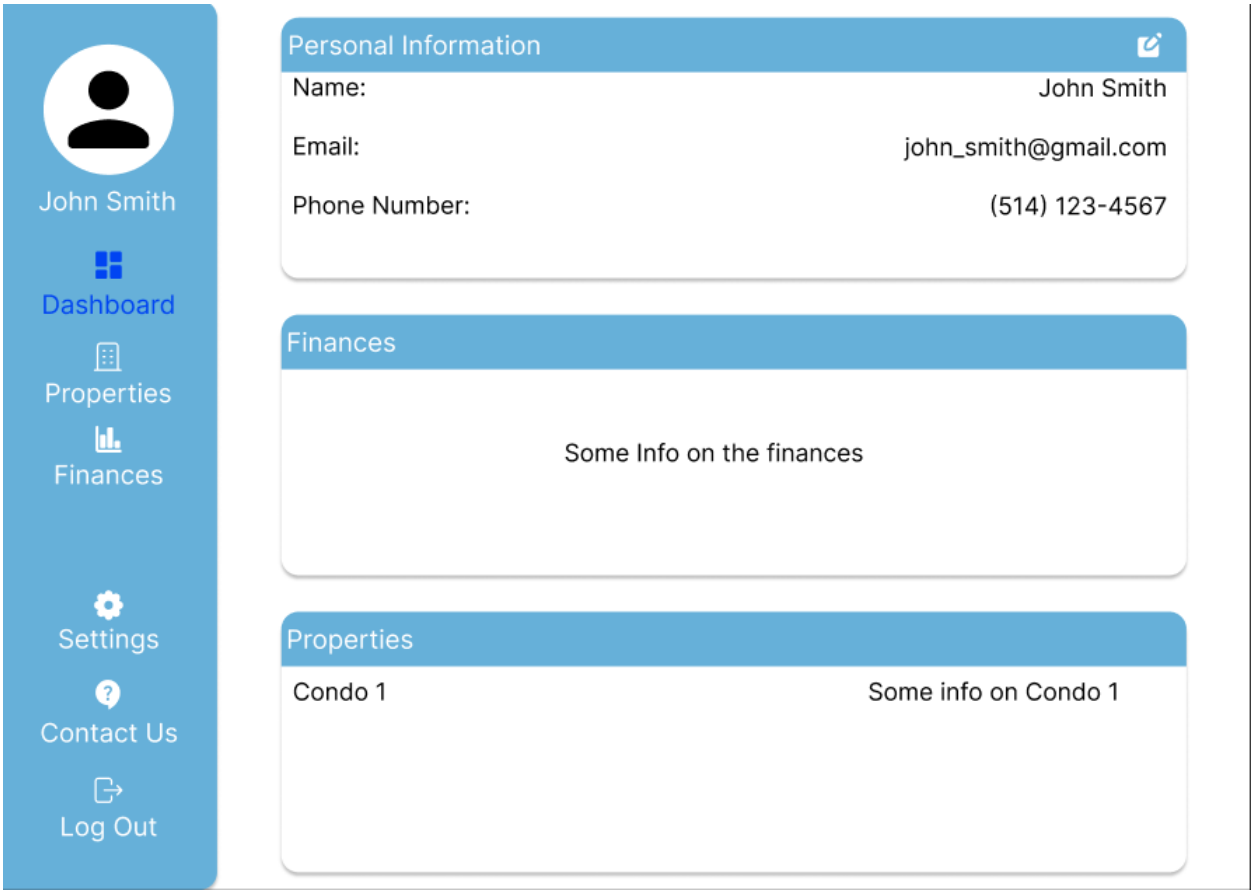

Upload profile picture
Upload photo
ProfilePicture.jpeg
First Name
John
Last Name
Smith
Email
john_smith@gmail.com
Phone Number
(514) 123-4567
Password
Confirm Password
Confirm



Steps:

1. The user accesses the website and clicks on Sign Up.
2. The user uploads their profile picture, first name, last name, email and phone number.
3. The user enters their password and enters a second time to confirm.
4. The user clicks on the Confirm button
5. The user is redirected to the Dashboard page

US-42: As a condo owner, I want to have a dashboard that displays general information about my properties, including personal profile, condo information, and financial status.



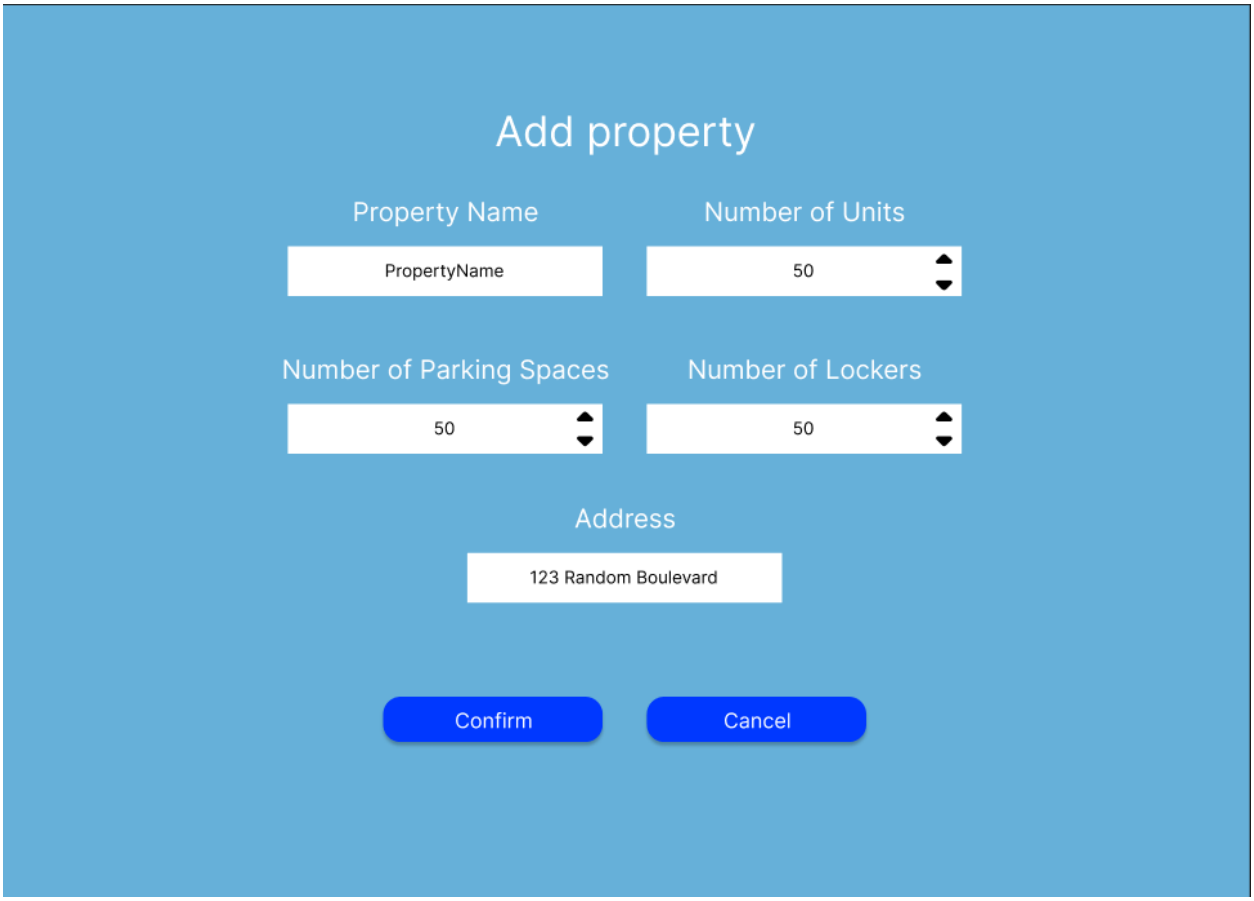
Dashboard page

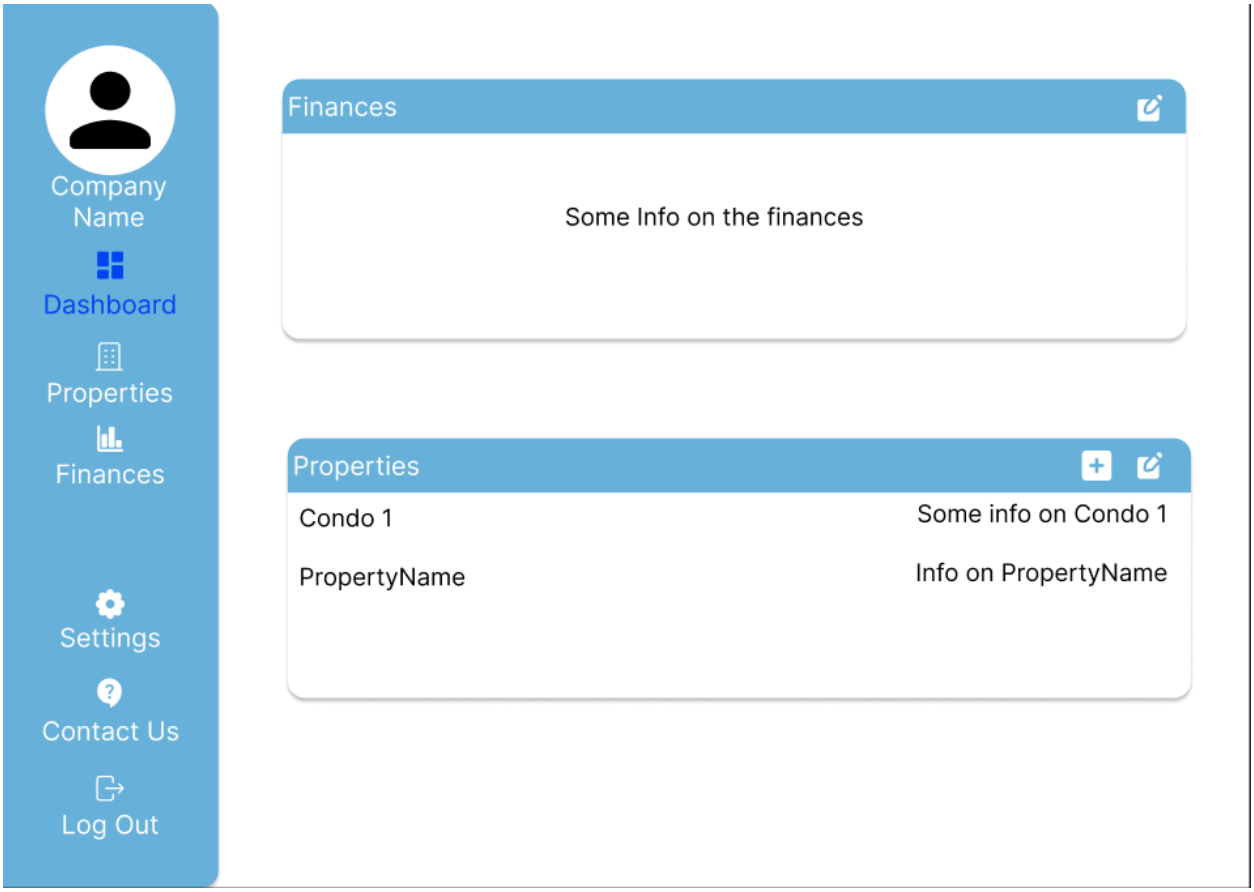
Steps:

1. After a successful login or sign up, the user will be redirected to this dashboard page.
2. The user will be able to see their personal information, properties and condo information and their financial information.

US-45: As a condo management company, I want to be able to create profiles for properties under my management, providing essential information like property name, unit count, parking count, locker count, and address.


Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
Finances
Some Info on the finances
Properties
Condo 1
Some info on Condo 1

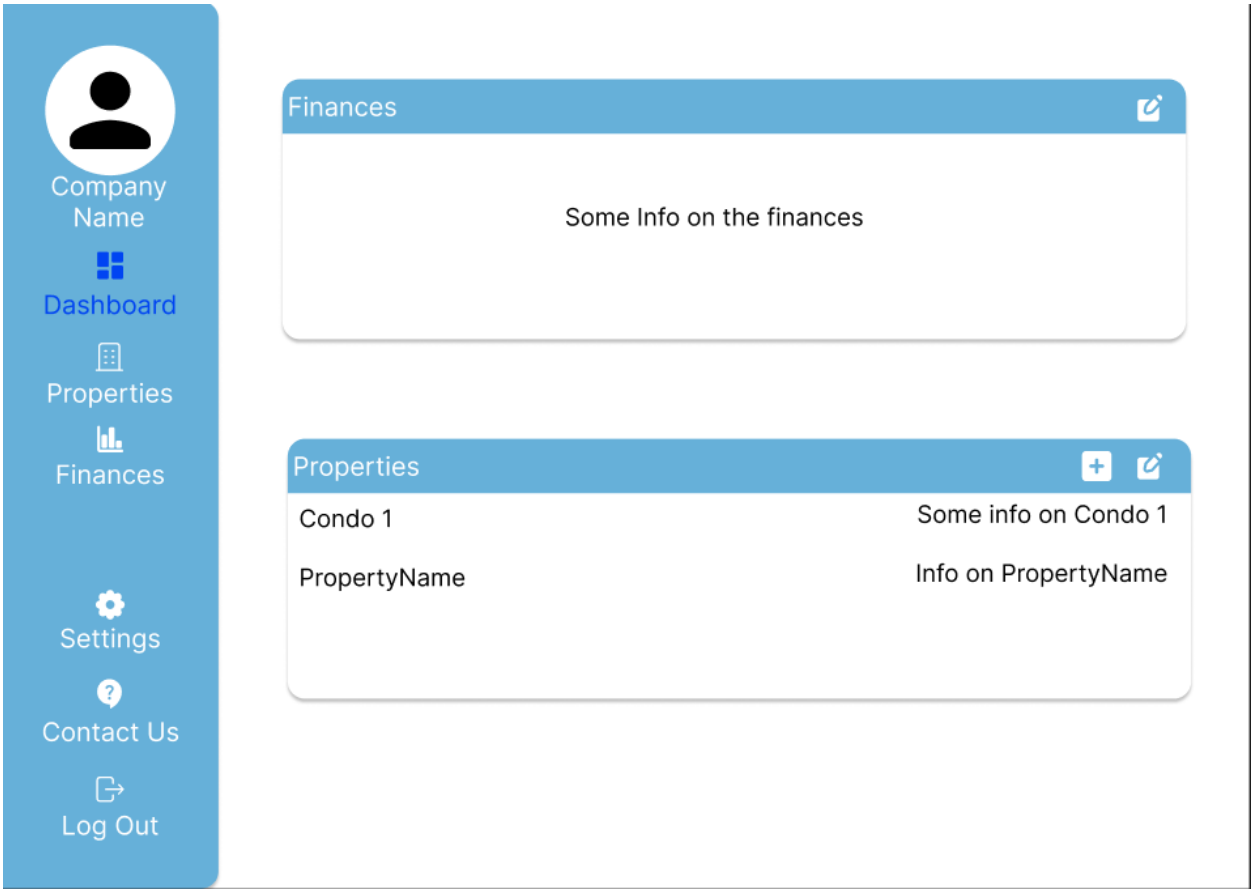

Add property
Property Name
PropertyName
Number of Units
50
Number of Parking Spaces
50
Number of Lockers
50
Address
123 Random Boulevard
Confirm
Cancel

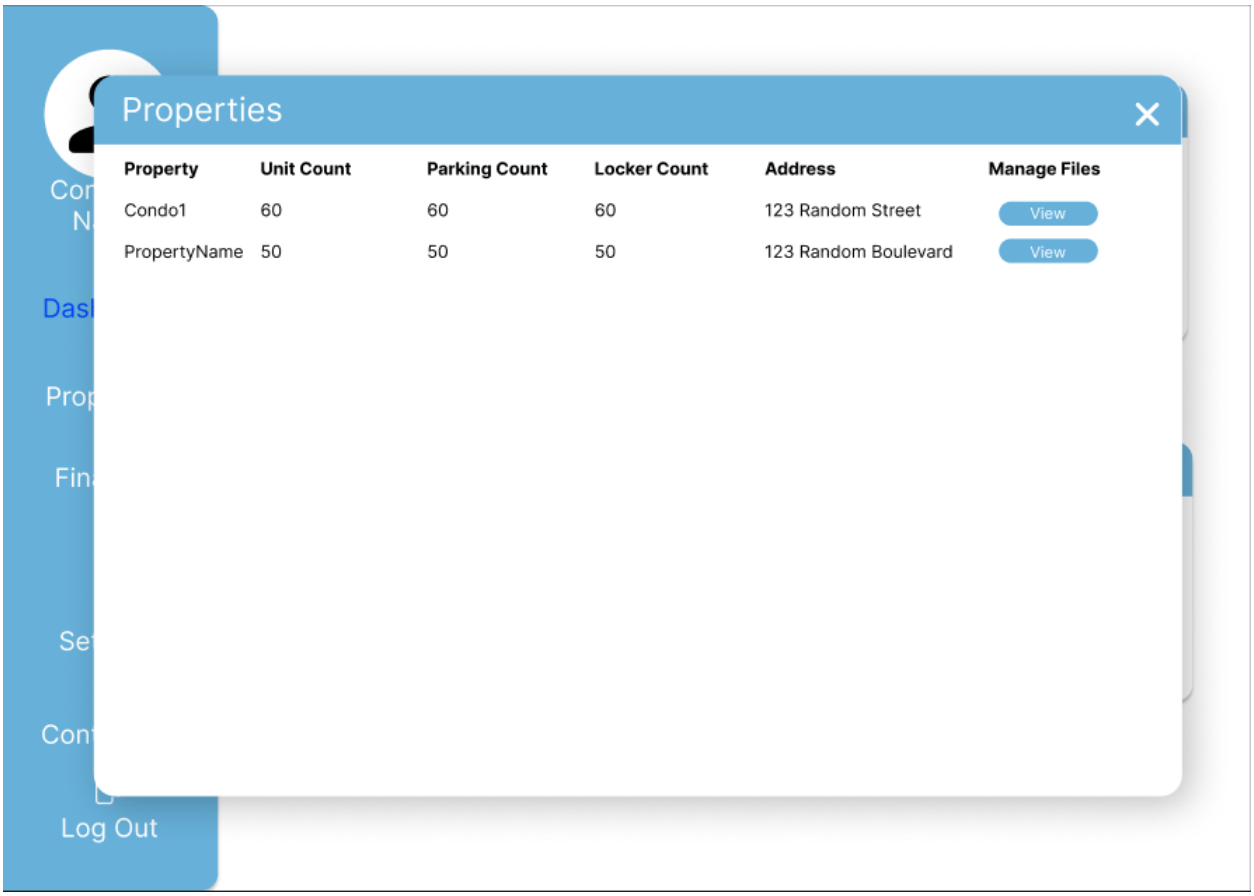


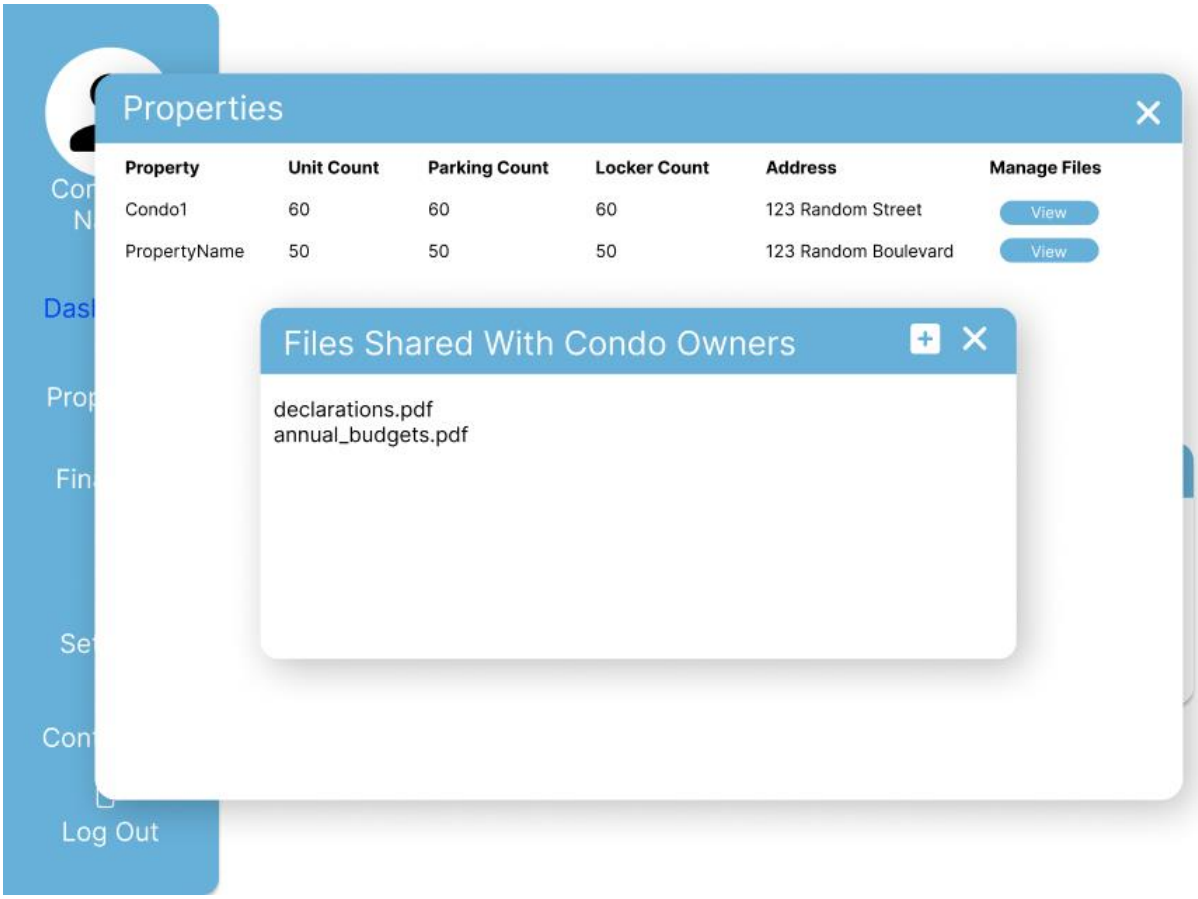
Steps:

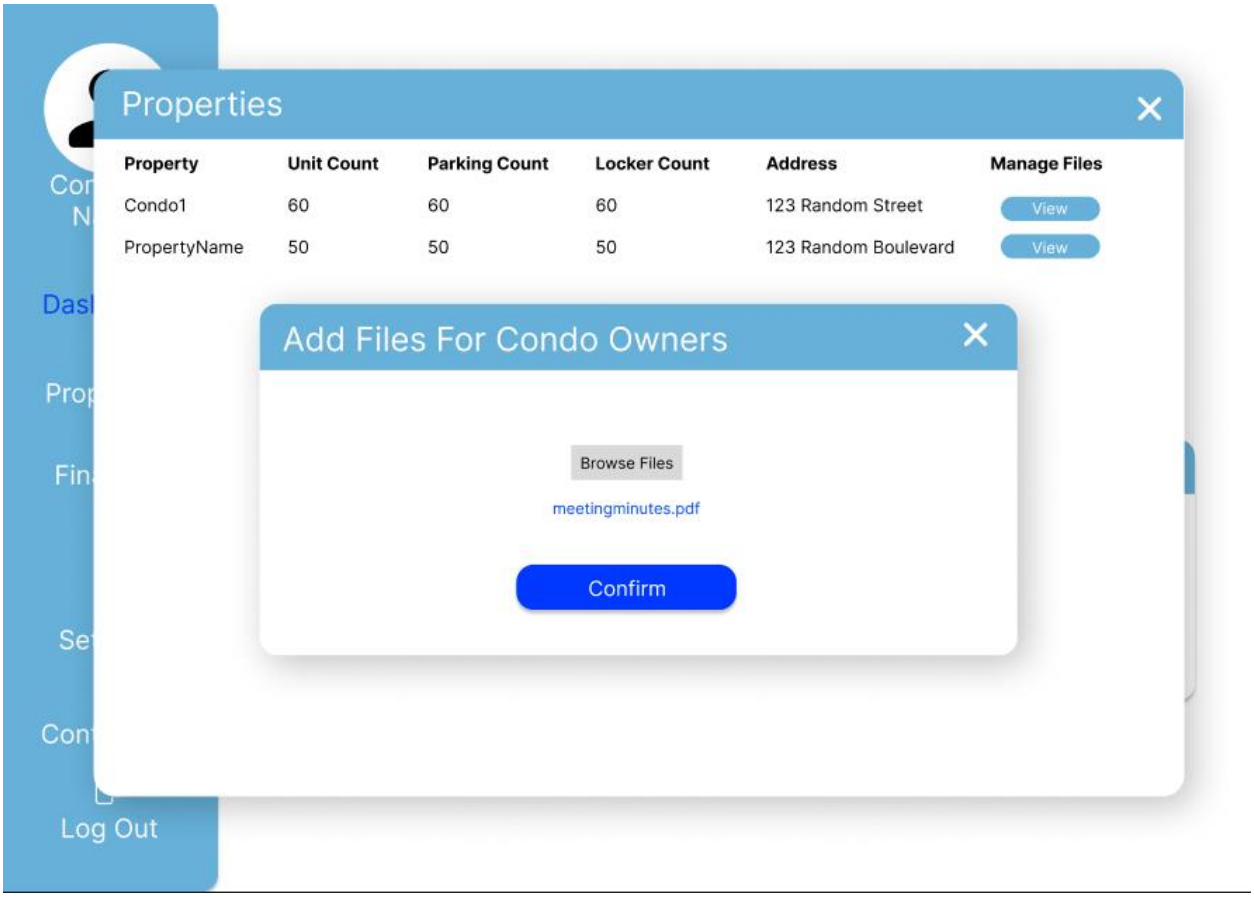
1. On the condo management company dashboard, the user clicks on the add icon in the Properties box.
2. The user enters the name of the property, the number of units, parking spaces and lockers and the address of the new property.
3. The user clicks on Confirm.
4. The user is redirected to an updated Dashboard page with the new property and its info added in the Properties box.

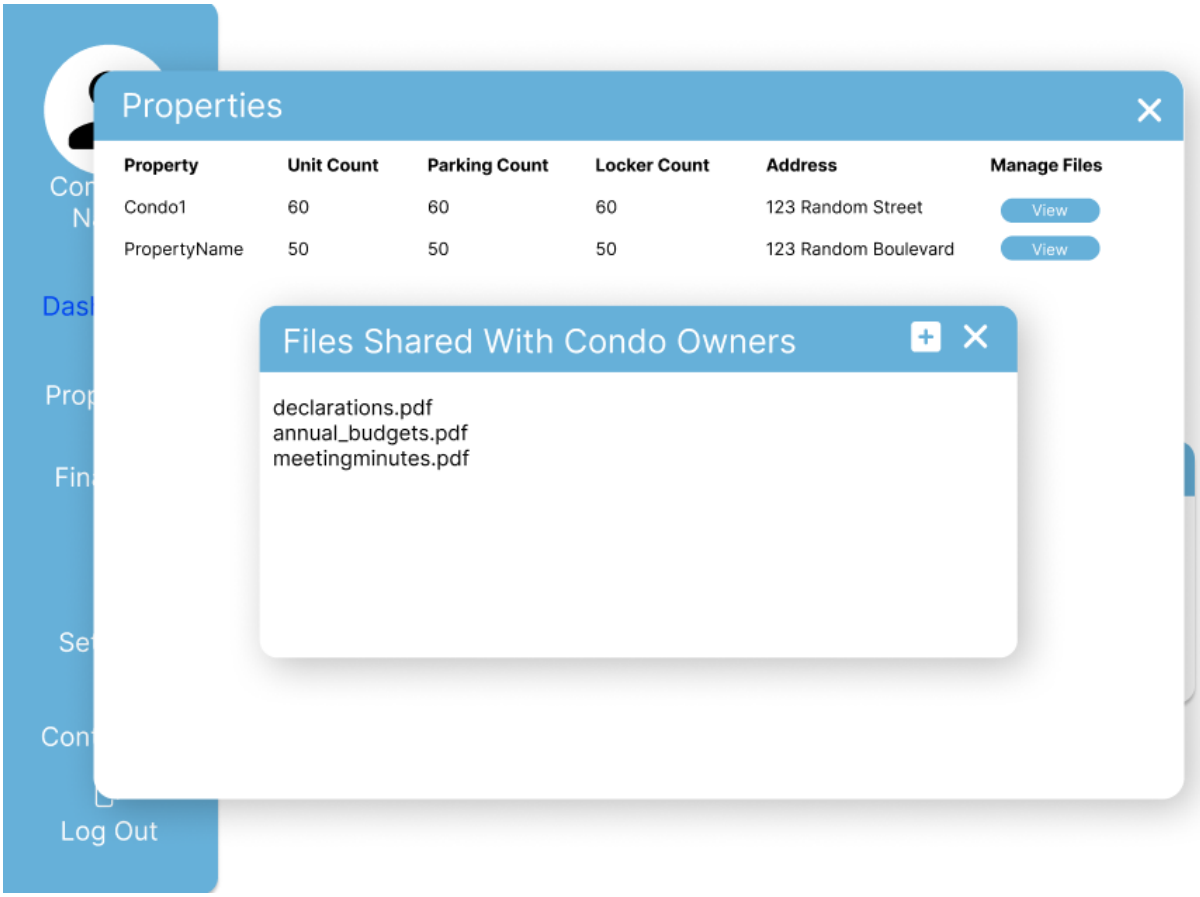
US-46: As a condo management company, I want to upload condo files for each property, accessible to all condo owners, including declarations, annual budgets, and board meeting minutes.


Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
Finances
Some Info on the finances
Properties
Condo 1
Some info on Condo 1
PropertyName
Info on PropertyName


Properties
Property
Unit Count
Parking Count
Locker Count
Address
Manage Files
Condo1
60
60
60
123 Random Street
View
PropertyName
50
50
50
123 Random Boulevard
View
Log Out


Properties
Property
Unit Count
Parking Count
Locker Count
Address
Manage Files
Condo1
60
60
60
123 Random Street
View
PropertyName
50
50
50
123 Random Boulevard
View
Files Shared With Condo Owners
declarations.pdf
annual_budgets.pdf
Log Out

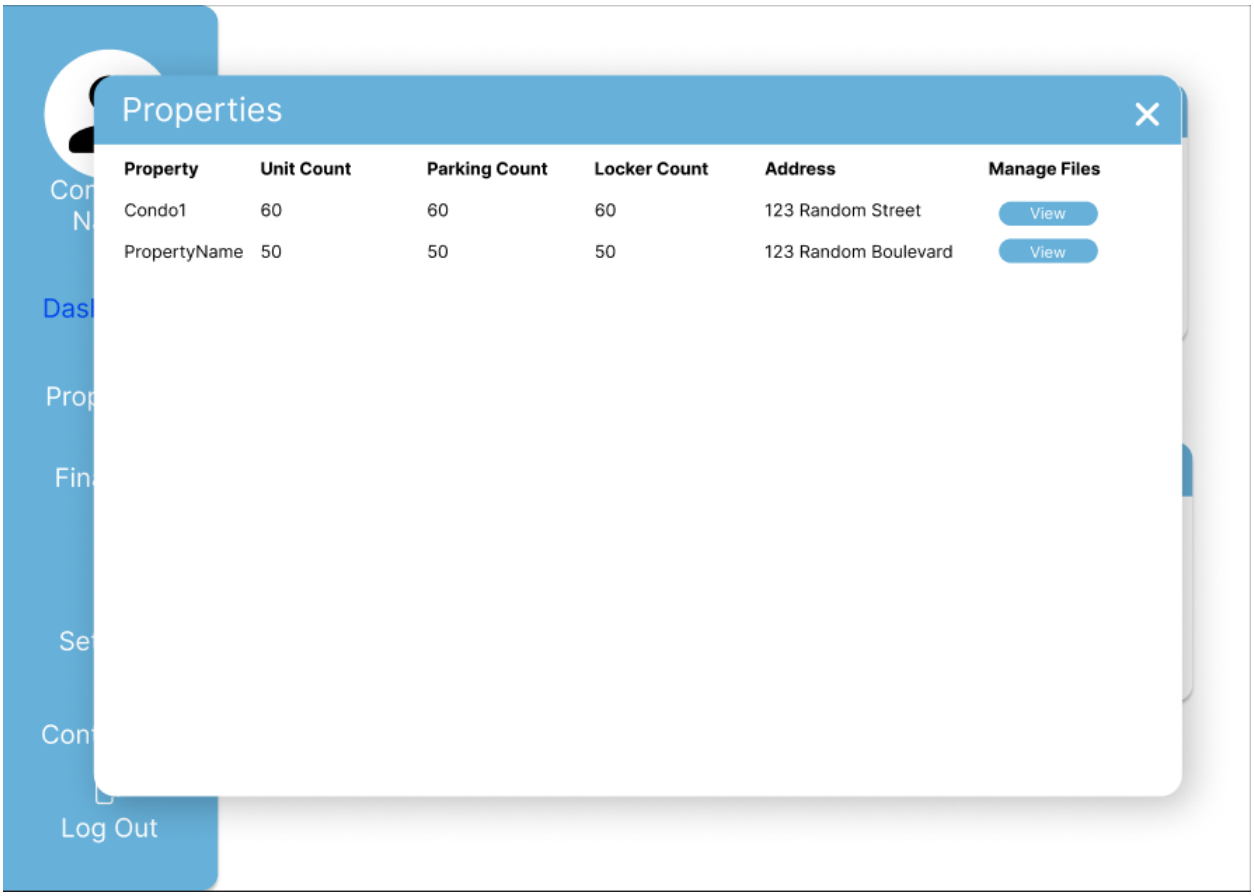

Properties
Property
Unit Count
Parking Count
Locker Count
Address
Manage Files
Condo1
60
60
60
123 Random Street
View
PropertyName
50
50
50
123 Random Boulevard
View
Add Files For Condo Owners
Browse Files
meetingminutes.pdf
Confirm
Log Out

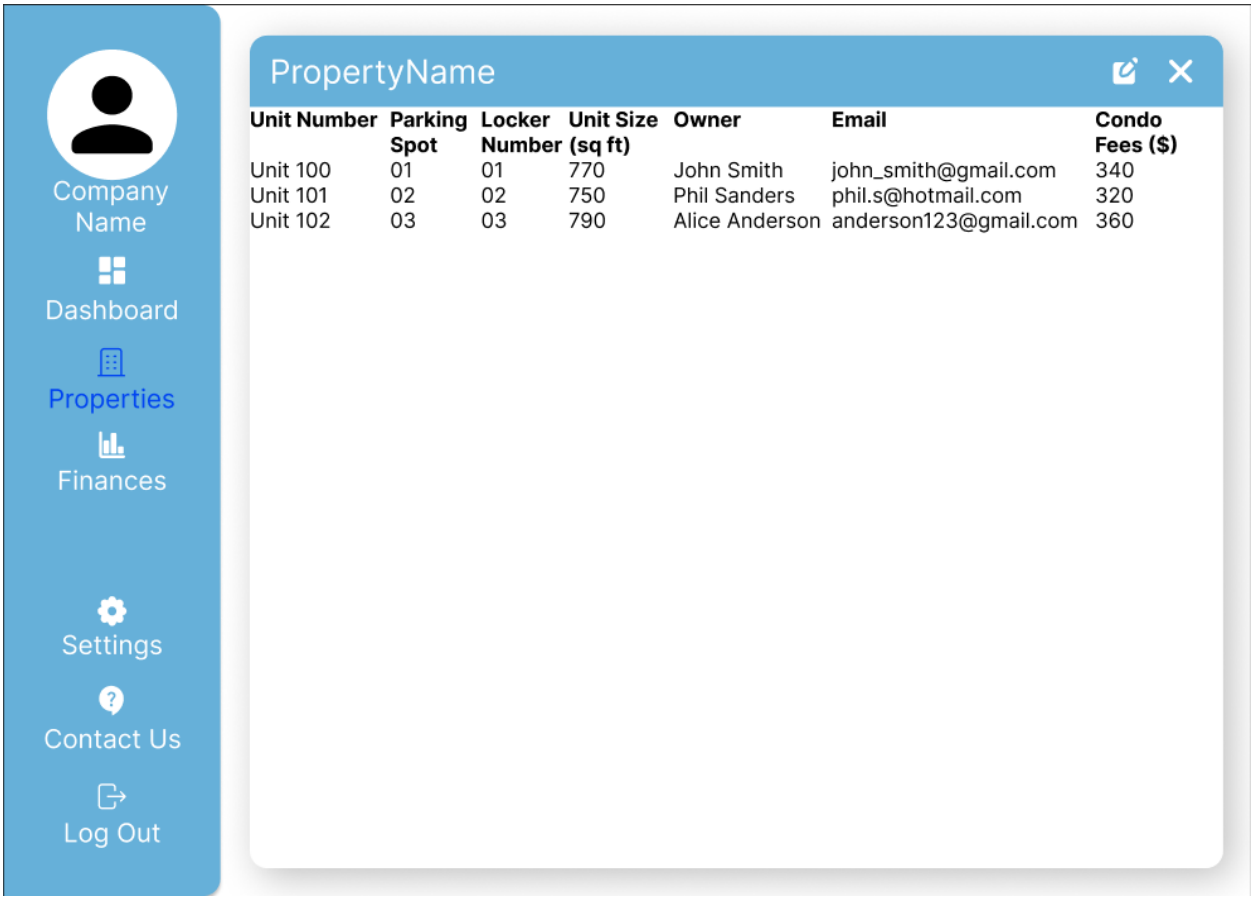


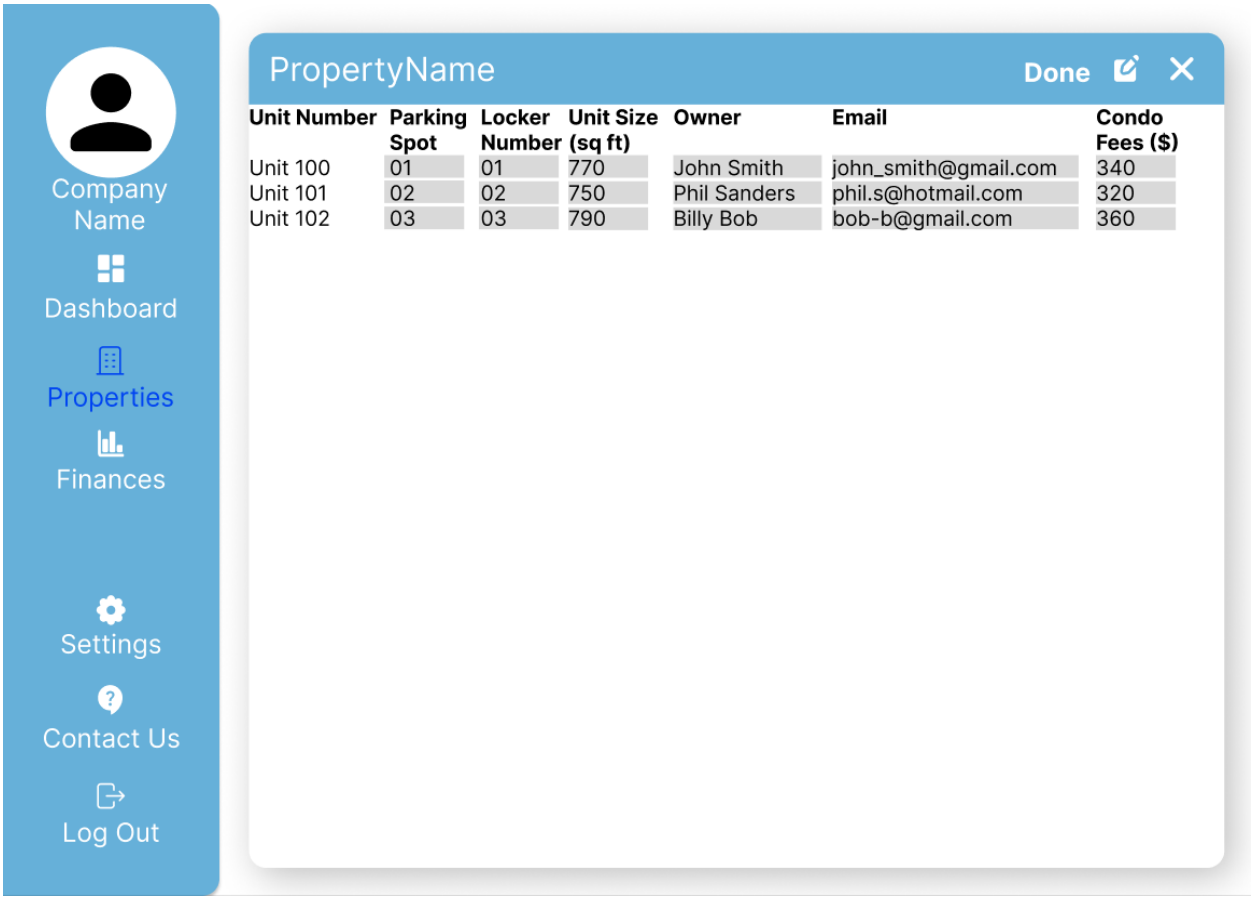
Steps:

1. From the dashboard, the user clicks on the edit properties button in the properties box.
2. The user clicks on the view button for the corresponding property for which they want to manage its files.
3. The user clicks on the add files button to upload new files.
4. The user browses files to upload the ones that they want to share with the condo owners.
5. The user clicks on Confirm.
6. The user is redirected to the Files Shared With Condo Owners window with the newly uploaded file in the view.

US-47: As a condo management company, I want to enter detailed information for each condo unit, parking spot, and locker, including unit size, owner information, and associated condo fees.


Properties
Property
Unit Count
Parking Count
Locker Count
Address
Manage Files
Condo1
60
60
60
123 Random Street
View
PropertyName
50
50
50
123 Random Boulevard
View
Log Out


Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
PropertyName
Unit Number | Parking Spot | Locker Number | Unit Size (sq ft) | Owner | Email | Condo Fees ($)
Unit 100 | 01 | 01 | 770 | John Smith | john_smith@gmail.com | 340
Unit 101 | 02 | 02 | 750 | Phil Sanders | phil.s@hotmail.com | 320
Unit 102 | 03 | 03 | 790 | Alice Anderson | anderson123@gmail.com | 360


Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
PropertyName
Done
Unit Number
Parking Spot
Locker Number
Unit Size (sq ft)
Owner
Email
Condo Fees ($)
Unit 100
01
01
770
John Smith
john_smith@gmail.com
340
Unit 101
02
02
750
Phil Sanders
phil.s@hotmail.com
320
Unit 102
03
03
790
Billy Bob
bob-b@gmail.com
360

Steps:

1. The user clicks on the name of the condo.
2. The user is redirected to the Properties page for the specific condo they clicked on.
3. The user clicks on the edit button.
4. The user edits the condo unit they want.
5. The user clicks on the Done button.
6. The user then sees the edited Property page of the condo they viewed.

5.4. TESTING PLAN

Condo Management System

Testing Plan for Sprint #2

07/02/2024

Unit and Integration Testing Tools

In this project, the technologies we use consist of Next.js framework for both the backend and the frontend. As such, the testing tools must be compatible with these frameworks. Since we will make use of Next.js, and given that it utilises React.js for the front end of our project, the decision for a testing tool to use for unit and integration tests that would be compatible with the technologies is JEST. It is a popular testing framework, providing a “zero-configuration” testing experience. It also has a built-in tool for code coverage which is essential for the purposes of this project.

Testing Approaches

The approaches to take in testing the system for this project are varied to ensure that we have an efficient and effective testing plan. For the duration of the project, we will use the Agile methodology to conduct testing iteratively throughout the sprints. In addition, and in line with the Agile method, we will also incorporate Test Driven Development to write tests before the code is written. We will also conduct exploratory tests to increase the user experience by allowing ourselves to test multiple functions of the system simultaneously. Lastly, we will also try to include Continuous Integration to integrate testing into the CI/CD pipeline to catch issues early.

Metrics and Coverage

To ensure a proper testing plan for the project, the team will prioritise certain testing metrics. Test pass rate is a testing metric important to ensure the quality of the code written by the team. The test pass rate is essentially the percentage of the passed tests during the development of the system. Another testing metric of our focus is defect density. It is related to the number of found defects per unit of code. We believe that this metric helps the testers to write helpful and efficient tests for the project. Lastly, we will incorporate the concept of code coverage. This metric is important to make sure we are covering most of the code written. This mostly includes methods and the different branches of outputs within each method. As such, we are aiming at a code coverage of at least 80%. We believe that this number is both feasible and a promising ratio for the proper functioning of the system.

Acceptance Test

Acceptance Test for Profile Creation (user story 1):

Feature: Profile Creation

Given I am a public user accessing the profile creation page when I provide a profile picture, a user name, a contact email, and a phone number and I submit the information, then my profile should be successfully created and I should receive a confirmation message.

Acceptance Criteria:

-The profile creation page allows the user to upload a profile picture.

-The user must enter a unique username.

-The user must provide a valid contact email address.

-The user must enter a valid phone number.

-Upon submitting the information, the system should create a unique profile for the user.

-The user should receive a confirmation message indicating the successful creation of the profile.

-The created profile should display the provided profile picture, user name, contact email, and phone number.

Example Test Data:

Profile Picture: [Upload an image file]

User Name: "JohnDoe123"

Contact Email: "john.doe@example.com"

Phone Number: "+1234567890"

Acceptance Test for Dashboard Display (User Story 4):

Feature: Dashboard Display

Given I am a condo owner logged into the system and I navigate to the dashboard, I should see general information about my properties, including personal profile, condo information, and financial status.

Acceptance Criteria:

-The dashboard should be accessible to condo owners upon logging into the system.

-The dashboard should display personal profile information of the condo owner.

-The dashboard should provide information about each property owned by the condo owner, including condo details.

-The financial status section should include relevant financial information such as dues, payments, and balances.

-The displayed information should be accurate and up-to-date.

-The dashboard should have a user-friendly layout for easy navigation and readability.

Example Test Data:

Condo Owner: "JaneSmith"

Personal Profile: [Display personal information]

Condo Information: [Display details of owned properties]

Financial Status: [Display financial information]

Acceptance Test for Property Profile Creation (User Story 7):

Feature: Property Profile Creation

Given I am a representative of the condo management company logged into the system and I navigate to the property profile creation page. Given I provide essential information such as property name, unit count, parking count, locker count, and address and I submit the information, then a profile for the property should be successfully created And I should receive a confirmation message.

Acceptance Criteria:

-The property profile creation page should be accessible to authorised representatives of the condo management company.

-The property name provided should be unique for each property profile.

-The unit count, parking count, and locker count should be numeric values representing the corresponding quantities for the property.

-The address field should accept valid address information.

-Upon submitting the information, the system should create a unique profile for the property.

-The user should receive a confirmation message indicating the successful creation of the property profile.

Example Test Data:

Property Name: "Sample Condo"

Unit Count: 100

Parking Count: 50

Locker Count: 20

Address: "123 Main Street, Cityville, State, ZIP"

Acceptance Test for Condo Files Upload (User Story 8):

Feature: Condo Files Upload

Given I am a representative of the condo management company logged into the system and I navigate to the file upload section for a specific property, given I upload condo files such as declarations, annual budgets, and board meeting minutes, then the files should be successfully uploaded and associated with the property and all condo owners should have access to these files.

Acceptance Criteria:

-The file upload section should be accessible to authorised representatives of the condo management company.

-The system should allow the upload of various file types, including declarations, annual budgets, and board meeting minutes.

-Uploaded files should be associated with a specific property.

-Condo owners for the respective property should have access to the uploaded files.

-The system should support versioning or date stamps for files to track updates over time.

-The file upload process should handle valid file formats and sizes.

Example Test Data:

Property Name: "Sample Condo"

File Types:

Declaration: [Upload a PDF file]

Annual Budget: [Upload a spreadsheet file]

Board Meeting Minutes: [Upload a text document]

Acceptance Test for Detailed Information Entry (User Story 9):

Feature: Detailed Information Entry

Given I am a representative of the condo management company logged into the system and I navigate to the detailed information entry section for a specific property, I provide information for each condo unit, parking spot, and locker, including unit size, owner information, and associated condo fees. Then I submit the detailed information for each unit, parking spot, and locker should be successfully entered and associated with the property.

Acceptance Criteria:

-The detailed information entry section should be accessible to authorised representatives of the condo management company.

-The system should allow the entry of detailed information for each condo unit, parking spot, and locker.

-The information for each unit should include unit size, owner information (name, contact details), and associated condo fees.

-The information for each parking spot and locker should include relevant details and associations.

-Upon submitting the information, the system should store and associate the detailed information with the respective property.

-The entered information should be accurate and reflect the current state of each condo unit, parking spot, and locker.

Example Test Data:

Property Name: "Sample Condo"

Condo Unit Information:

Unit Size: 1000 sq. ft.

Owner Information: "John Doe," "johndoe@example.com"

Condo Fees: $500/month

Parking Spot Information:

Parking Spot Number: P001

Owner Information: "Jane Smith," "janesmith@example.com"

Condo Fees: $50/month

Locker Information:

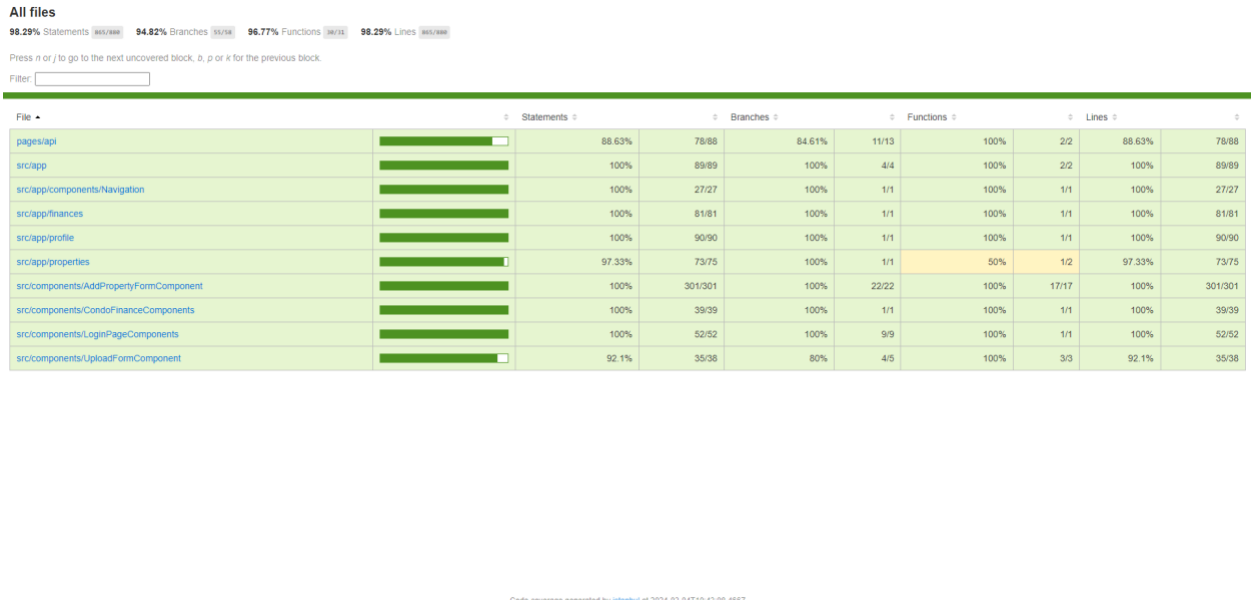
Locker Number: L001

Owner Information: "Alex Johnson," "alexjohnson@example.com"

Condo Fees: $10/month

6. SPRINT 2 DELIVERABLES

6.1. TESTING COVERAGE



All files

98.29% Statements 865/880 **94.82%** Branches 55/58 **96.77%** Functions 30/31 **98.29%** Lines 865/880

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
pages/api	88.63%	78/88	84.61%	11/13	100%	2/2	88.63%	78/88
src/app	100%	89/89	100%	4/4	100%	2/2	100%	89/89
src/app/components/Navigation	100%	27/27	100%	1/1	100%	1/1	100%	27/27
src/app/finances	100%	81/81	100%	1/1	100%	1/1	100%	81/81
src/app/profile	100%	90/90	100%	1/1	100%	1/1	100%	90/90
src/app/properties	97.33%	73/75	100%	1/1	50%	1/2	97.33%	73/75
src/components/AddPropertyFormComponent	100%	301/301	100%	22/22	100%	17/17	100%	301/301
src/components/CondoFinanceComponents	100%	39/39	100%	1/1	100%	1/1	100%	39/39
src/components/LoginPageComponents	100%	52/52	100%	9/9	100%	1/1	100%	52/52
src/components/UploadFormComponent	92.1%	35/38	80%	4/5	100%	3/3	92.1%	35/38

6.2. SPRINT RETROSPECTIVE

Behind the Scenes of Sprint 2

Introduction

As we complete our project's second sprint, it is essential to look back and analyze our progress in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Entreprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system, and a financial system.

During this sprint, our main goal was to achieve a basic version of our platform, which would serve as a building block where we could add the rest of the features piece by piece. By doing so, we would also ensure that we would have an MVP that would showcase a section of our platform to potential users. This was met by a lot of hardship, as many team members did not have previous experience with some of the used technologies.

When reflecting on this sprint's outcome, we were proud of our progress but still believe that there is room for improvement. Among various achievements, our GitHub is well-organized, and we have laid a solid foundation for the software architecture through both the SAD document and the running prototypes. As the next sprint is approaching, we have a better understanding of the Agile refinement process and the areas that the team needs to work on.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement towards the lack of communication amongst team members and the issues it led to.

What went wrong

1 - Lack of Communication

Throughout the second sprint, team members were often busy with midterms, assignments and other school projects which made it hard to communicate within the team as everyone had different schedules. The impact of this was that we ended up each doing our own tasks separately and in the end we encountered issues with merging our work together. There was a lot of confusion as well as of what was completed and what was left to be done.

We addressed it by giving updates every 3 days on discord on the progress that was made on each side. However, it would have been better to stick to a regular meeting schedule to better communicate by voice chat rather than by texts.

2 - Lack of Organization

Throughout the second sprint, it was not entirely clear of which task was assigned to who. This made the organization of our work confusing and messy because some parts rely on others. The file management was also an issue as mentioned previously because it caused conflicts when we attempted to merge the implementations together. The lack of organization also made it so that we had to scramble to finish our work at the last minute since we did not set clear boundaries as to who does what. Some tasks were left unassigned and we did not realize early on enough. Also, some tasks required another task to be done. However, the preceding tasks were sometimes done late, which, in turn, delayed the task that depended on it such as the testing. Additionally, some tasks on GitHub were not updated as they should have been which created ambiguity in terms of the progression of the sprint.

We addressed it by getting on a call to work on the remaining tasks together. However, it would have been better if we had done this earlier instead of at the last minute.

What went right

1 - Learning New Technologies

Throughout the second sprint, some members, who had never learned certain technologies such as writing automated tests and Next.js framework, got to learn these while doing their respective tasks. The impact of this was that we got the opportunity to use this project as a way to improve and surpass ourselves by challenging ourselves. Through this, we now have new tools under our belt that we can potentially use in a different setting in the future.

Some of the things that we learned, we each had to learn on our own for our own respective tasks. However, it would have been better to learn together so that we can share the knowledge across the team. This way, we can all help each other with our tasks and this could lessen the burden on certain people for their tasks.

2 - Flexibility of The Team Members

Throughout the second sprint, some members, as mentioned previously, team members were willing to challenge themselves and take on tasks even though they had little knowledge of the technology required for that part. Moreover, in desperate times, team members were willing to get together to put collective effort to help each other out with the remaining tasks. Other members were also willing to give tutorials to teach other members how to do their work.

The team’s flexibility and adaptability made it so that the work that needed to be done was ultimately finished in the end despite our lack of communication and organization at the beginning. However, had we been more flexible at the beginning, we would have encountered less difficulties during this sprint.

Conclusion

While reflecting on the second sprint, we realized that we encountered multiple challenges that originated from our lack of organization in terms of our files and tasks, and our lack of communication. However, we tackled these problems through the group's adaptability and flexibility. Although our teamwork is not perfect, we are in a learning process to improve on it to the best of our ability. Moreover, through learning new technologies, there was a lot of confusion throughout the sprint. However, things worked out fine in the end when we came together to collectively work on the project. We got to refine the rough edges and we were able to bond a bit more which improved the team dynamic.

As for the remaining upcoming sprints, we have learned many lessons that we will be applying for the rest of the project to ensure a smoother progression. For example, we now know that we can't delay the completion of our tasks as many are dependent on others. We also learned that we need to be more communicative with each other because we can't all be doing our tasks individually as it may cause conflicts. It is clearer to use now that we need to better assign tasks to members not only depending on what they are comfortable doing but also on their work style and availability. This sprint was an effective wake up call on the area for which we need more improvement as a team. By applying the lessons we learned, we will tackle future challenges being more informed and committed to developing the best product possible.

6.3. RELEASE PLAN

Table 1. User story points and priority of user stories that will be implemented in Sprint 3

User Story ID	User Story Points (USP)	Priority	Status
#41	3	High	TO DO
#43	3	High	TO DO
#44	5	High	TO DO
#48	12	High	TO DO
#49	5	High	TO DO
#50	11	High	TO DO
#51	2	Medium	TO DO
#52	8	Medium	TO DO
#53	4	Medium	TO DO
Total USP	53		

*The user stories (see below) only concern the web application. We decided to focus on completing the main website first before tackling the companion app.
**The User Story Points are approximately proportional to the expected number of hours a task should take (going from 1 to 5 for each task).

Sub-User Stories:

Public User

#41 - As a public user, I want to provide a registration key obtained from my condo management company to become a condo owner or renter in the system.

- Add a section for the condo key registration. (1 point)
- Implement validation for the registration key. (2 points)

Condo Owner

#43 - As a condo owner, I want to view the status of submitted requests, such as moving in/out requests, intercom changes, or reporting violations.

- Add a section for submitted requests in the user profile. (1 point)
- Implement the logic for requests' status. (2 points)

#44 - As a condo owner, I want to receive notifications about the latest activities in submitted or assigned requests.

- Implement an email notification system for submitted requests. (5 points)

Condo Management Company

#48 - As a condo management company, I want to send registration keys to unit owners or rental users for their dedicated units.

- Create the overview page for units (accessed from the properties page). (3 points)
- Create the details page for units (accesses from the units page). (3 points)

- Implement the logic to get properties for a company (fetching from the database). (2 points)
- Implement the logic to get units for a property (fetching from the database). (2 points)
- Implement the logic for unit details (fetching from the database). (2 points)

#49 - As a condo management company, I want to set up different roles for employees responsible for daily operations, finance, etc.

- Implement a building management page with information on the staff members and their roles. (2 points)
- Implement an edit page that lets the condo management company edit the staff information. (3 points)

#50 - As a condo management company, I want to set up the condo fee per square foot and per parking spot and calculate condo fees for each unit.

- Add a link to the parking page for a property. (1 point)
- Create a parking space overview page. (2 points)
- Implement the logic for the parking fee update. (2 points)
- Add fields to the unit table in the unit overview page. (1 point)
- Implement the logic for the unit field updates. (2 points)
- Implement the logic to return the total expense for a user. (2 points)
- Update the financial status of the user. (1 point)

Financial System

#51 - As a condo management company, I want to record operational budgets (collected condo fees) and costs.

- Add operational budget section in Finances page that include the monthly remaining budget (and collected condo fees) and expenses/costs (2 points)

#52 - As a condo management company, I want to enter costs for each operation.

- Add a page for the condo management company to enter new operation costs and the information on each expense. (3 points)
- Implement logic to calculate the month's remaining budget (2 points)
- Implement logic to calculate the month's total expenses (2 points)
- Update the budget section in Finances page (1 point)

#53 - As a condo management company, I want the financial system to generate an annual report showing all condo fees collected for a given year.

- Add button to generate annual report showing all condo fees collected for a given year. (1 point)
- Implement logic that calculates the annual condo fees and expenses. (2 points)
- Display the annual report as a popup window. (1 point)

Burndown Chart

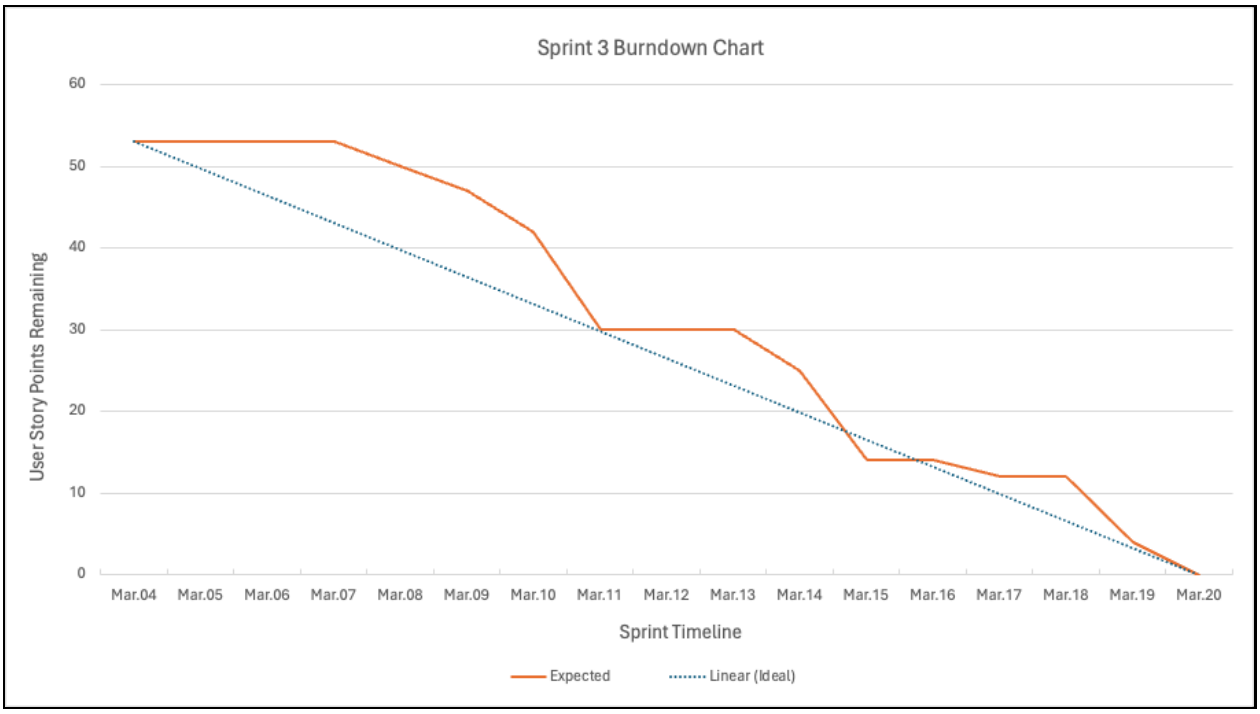
Let's consider that we start working on March 4 and that the end of the sprint is March 20. The sprint duration is 16 days. Let's also subtract 4 days of weekend for a more realistic result.

Total Story Points = 3 + 3 + 5 + 12 + 5 + 11 + 2 + 8 + 4 = 53

Ideal Daily Burndown = Total Story Points / Sprint Duration = 53 / 14 ≃ 3.79

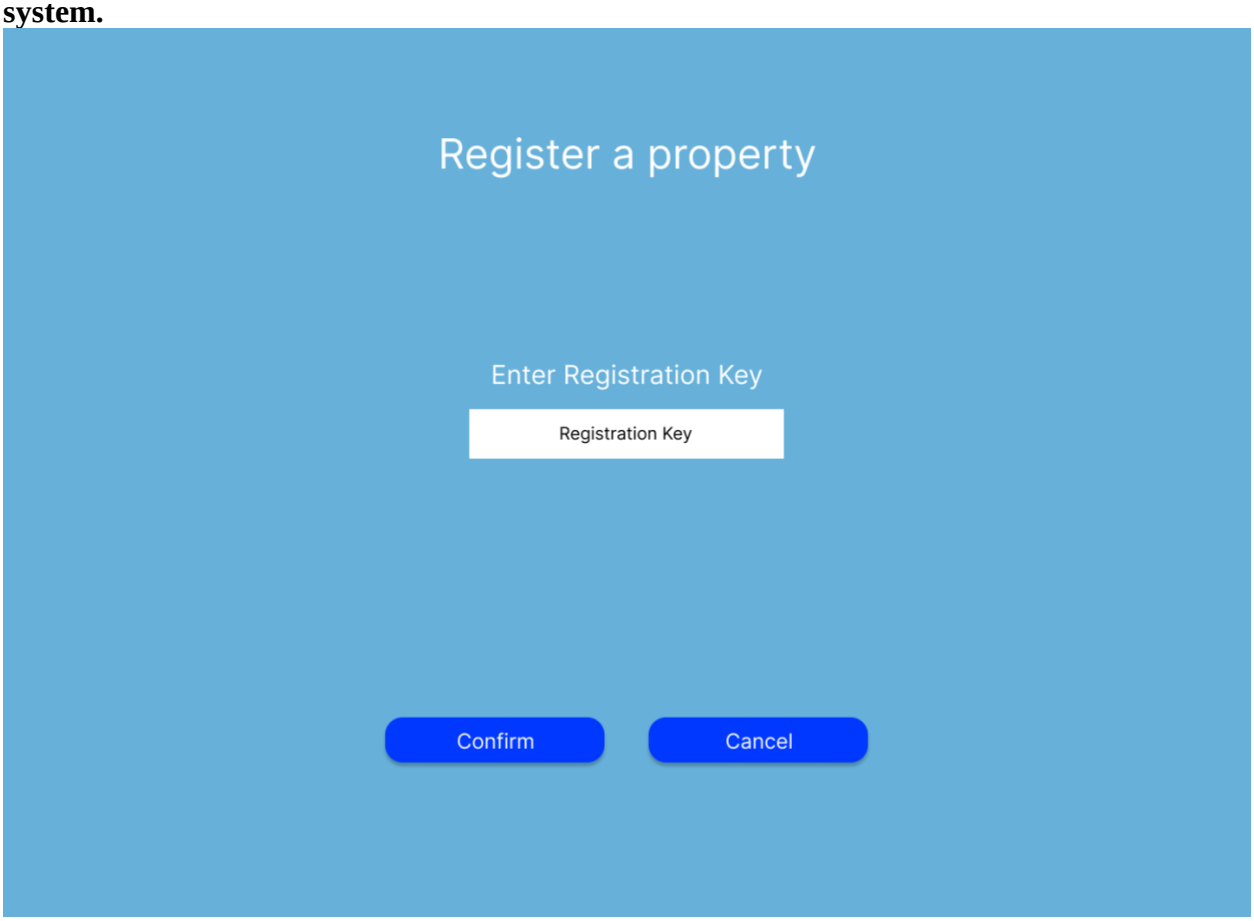
Therefore, we would ideally complete around 3.79 USP every day if we take only 2 days off.

The burndown chart below illustrates our expected progress rate.

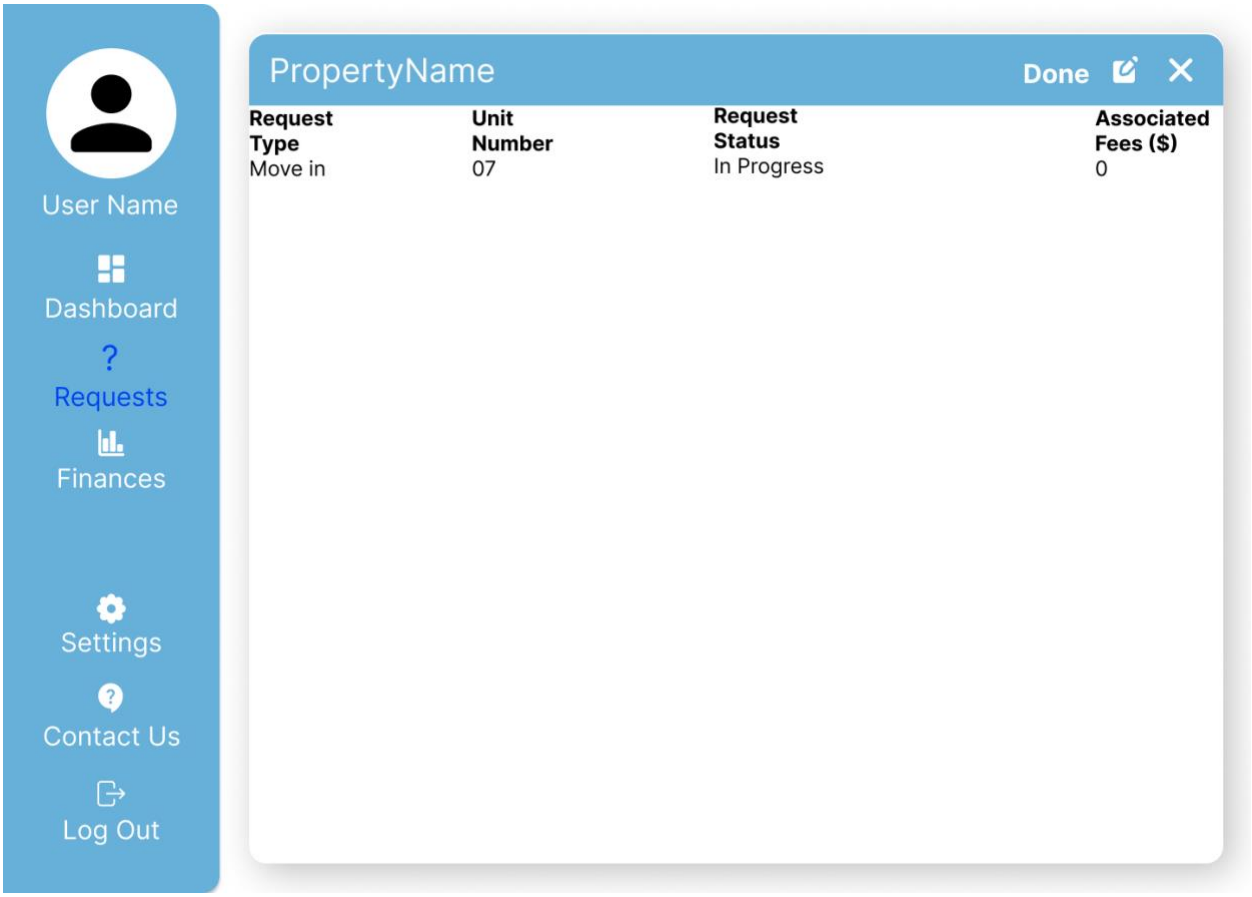


6.4. UI PROTOTYPE

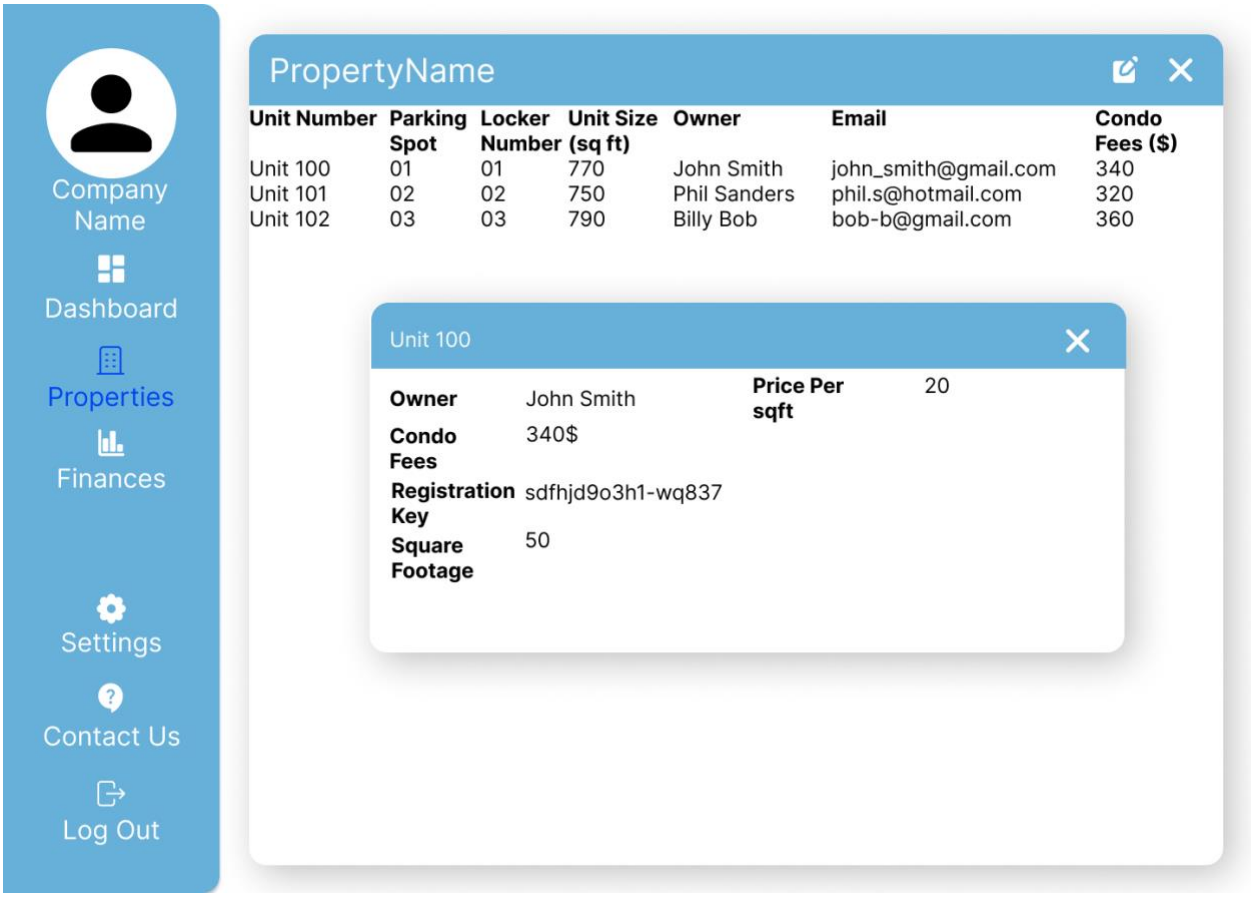
#41 - As a public user, I want to provide a registration key obtained from my condo management company to become a condo owner or renter in the system.



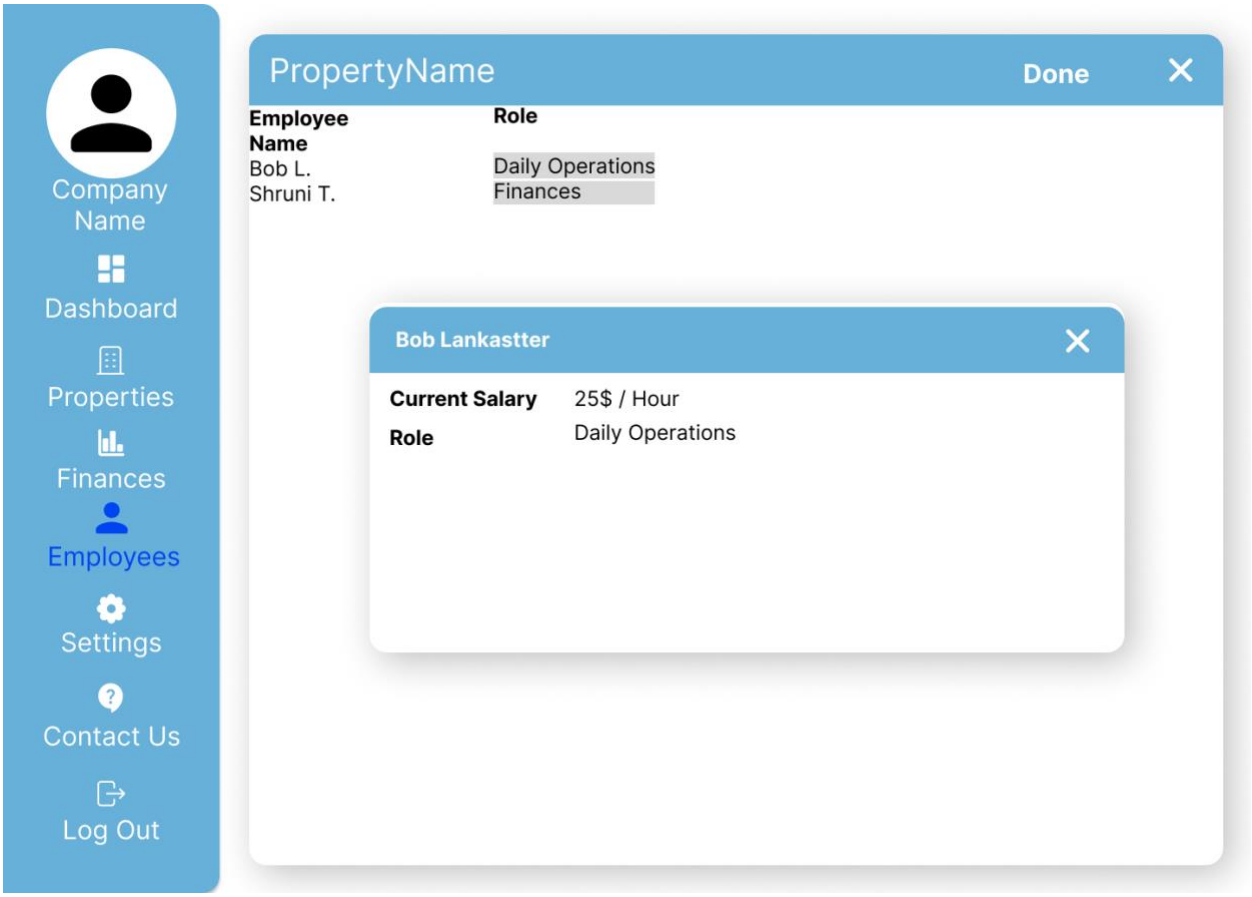
#43 - As a condo owner, I want to view the status of submitted requests, such as moving in/out requests, intercom changes, or reporting violations.



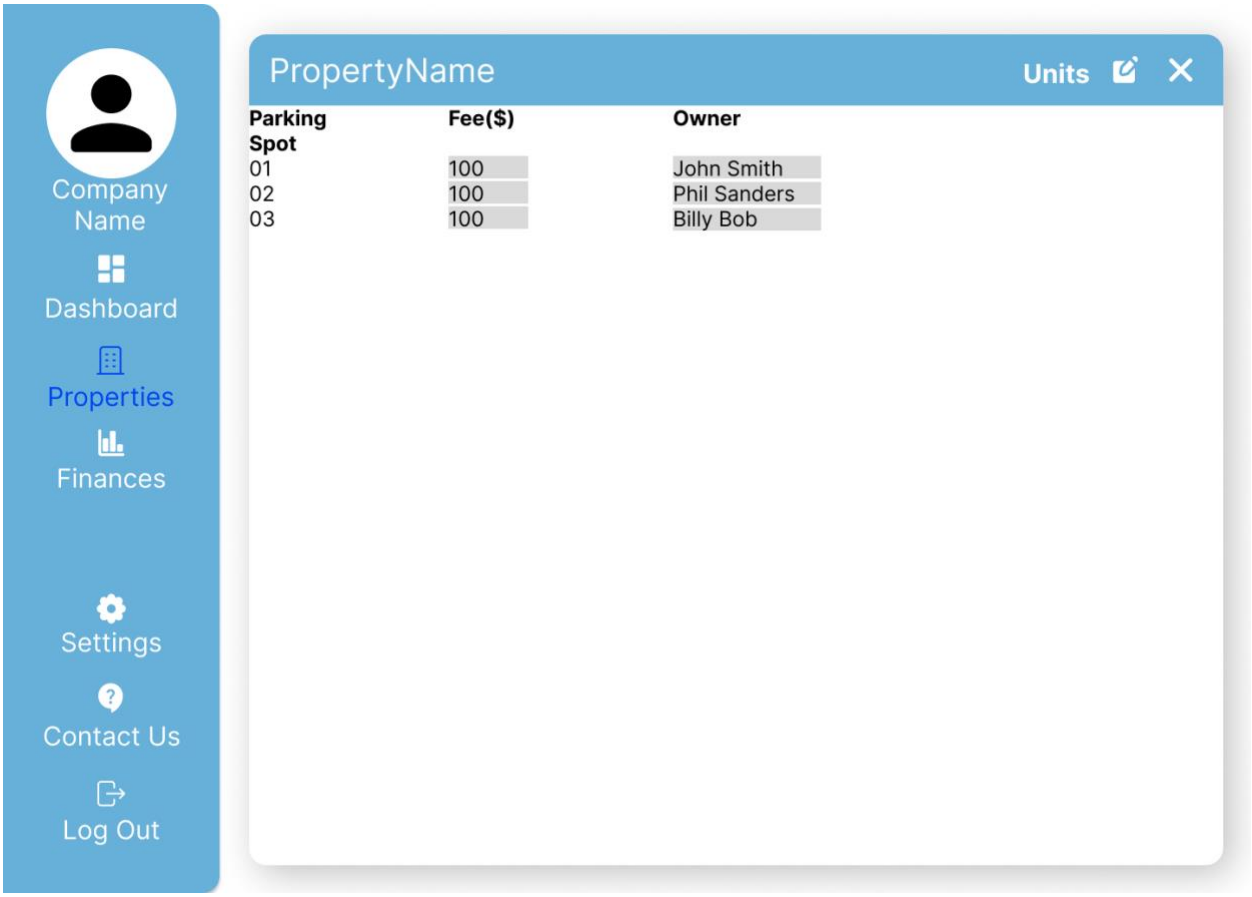
#48 - As a condo management company, I want to send registration keys to unit owners or rental users for their dedicated units.

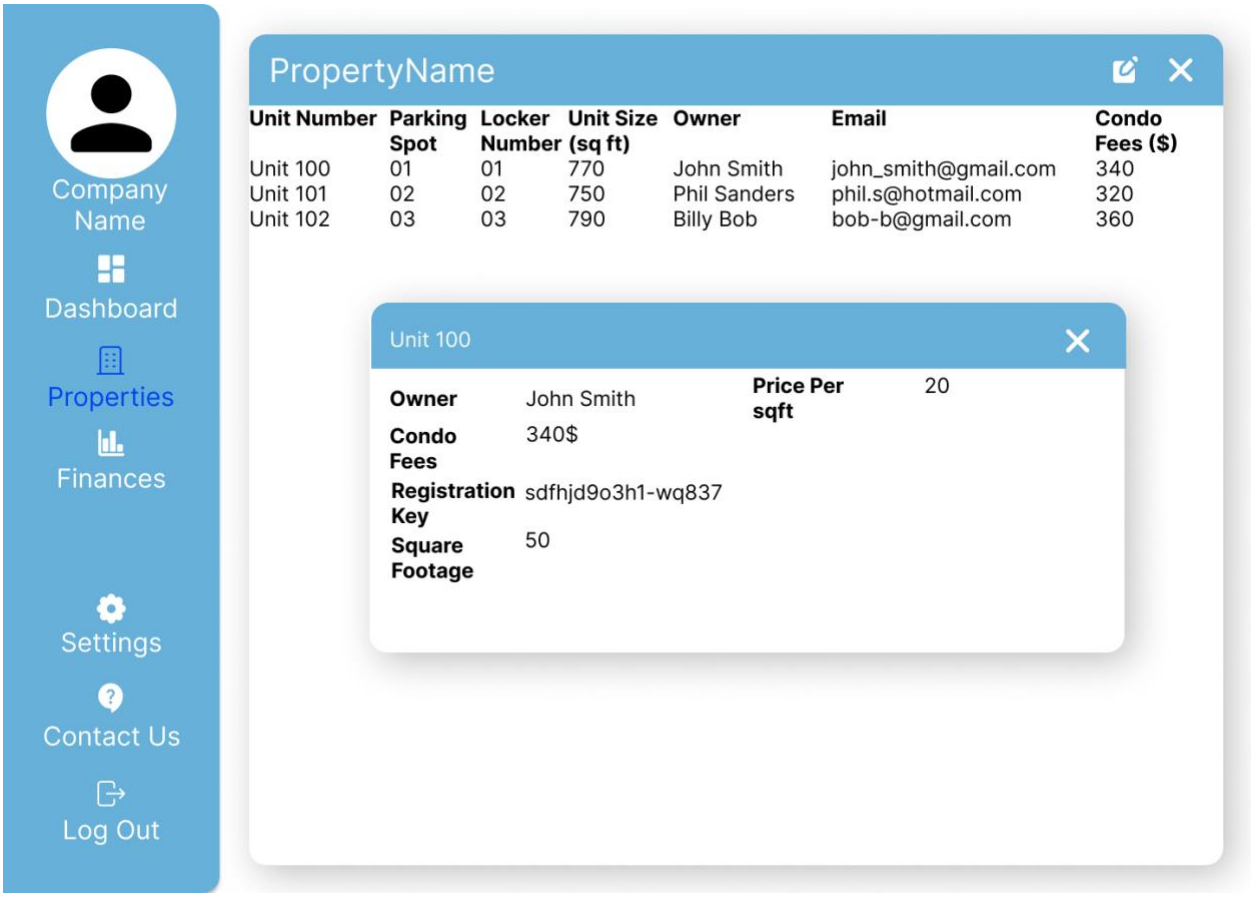


#49 - As a condo management company, I want to set up different roles for employees responsible for daily operations, finance, etc.

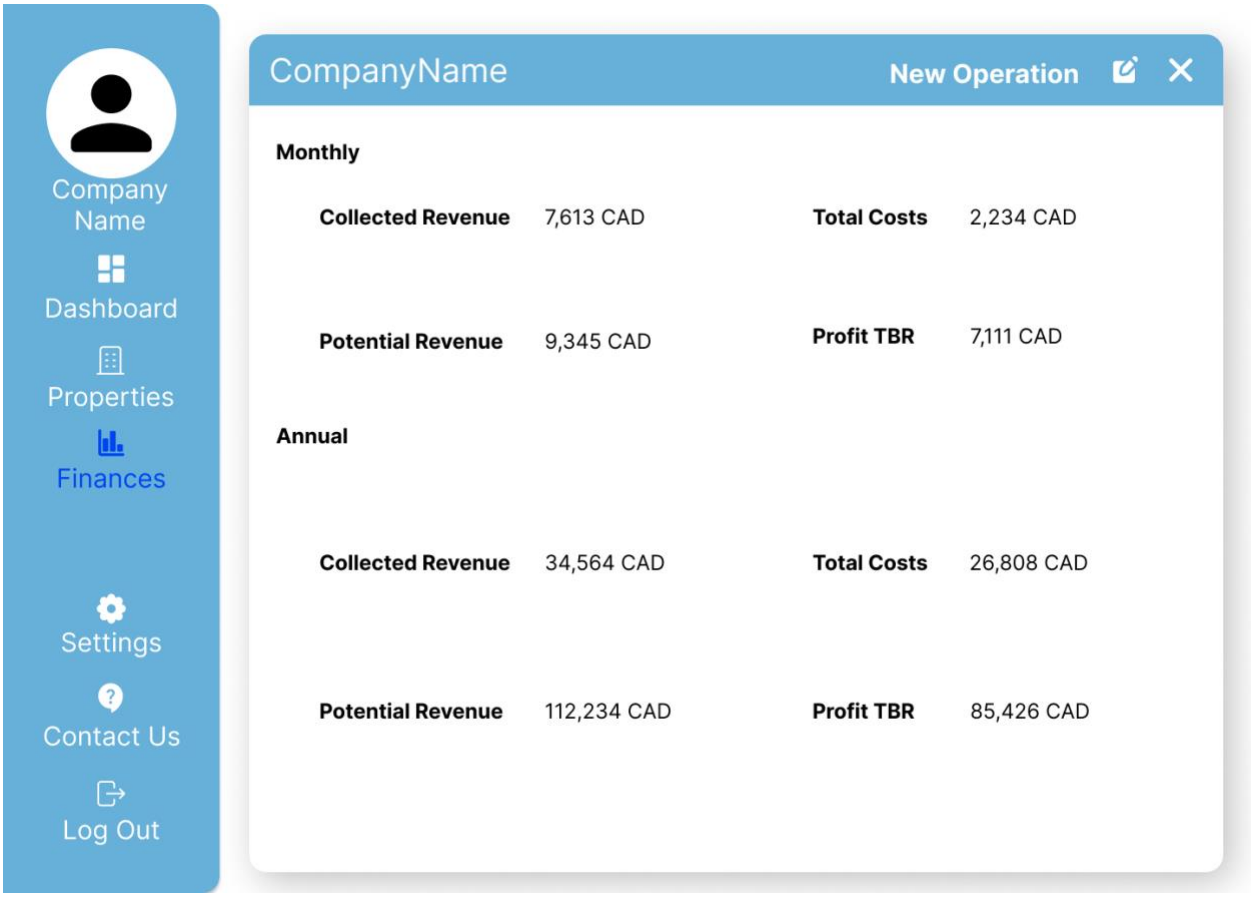


#50 - As a condo management company, I want to set up the condo fee per square foot and per parking spot and calculate condo fees for each unit.

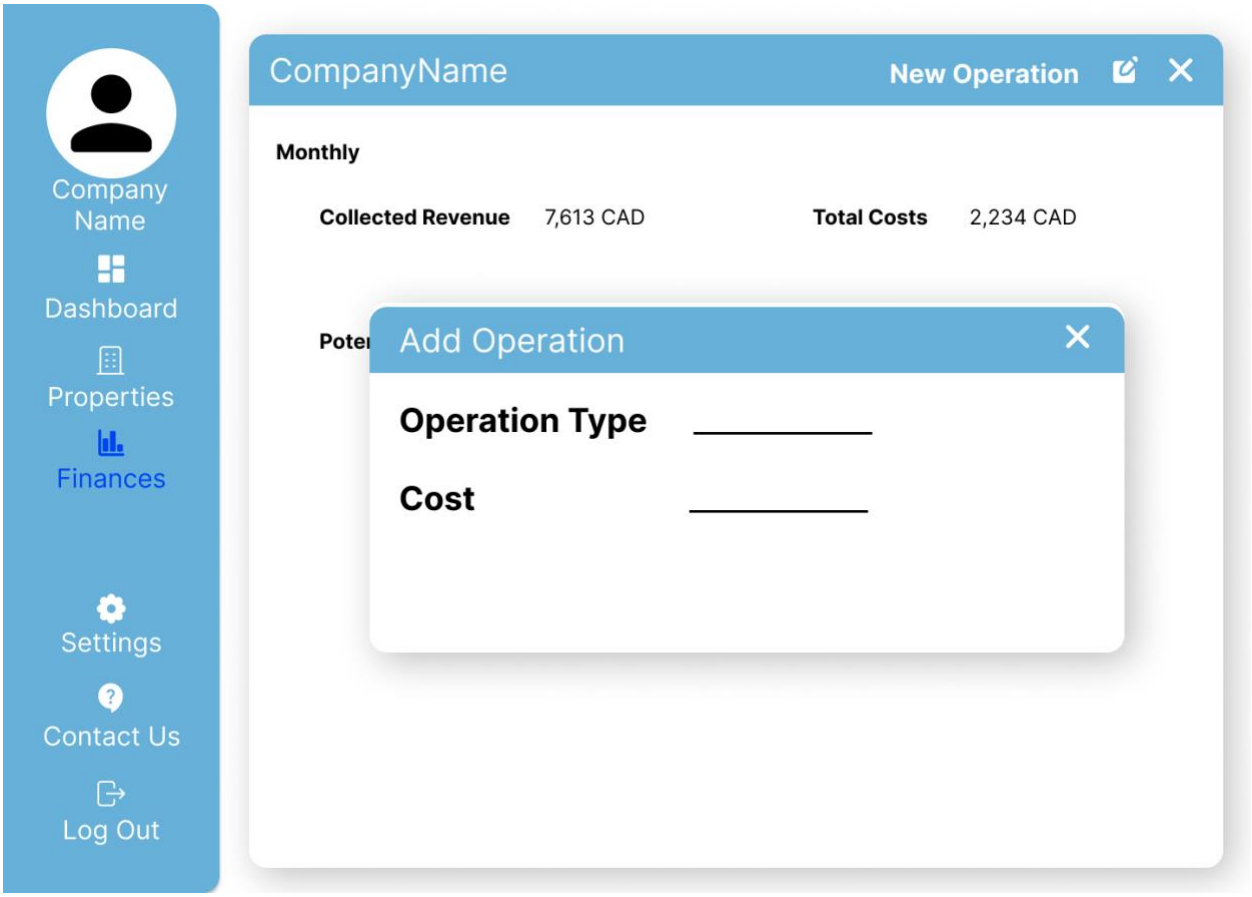

Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
PropertyName
Units
Parking Spot
Fee($)
Owner
01
100
John Smith
02
100
Phil Sanders
03
100
Billy Bob



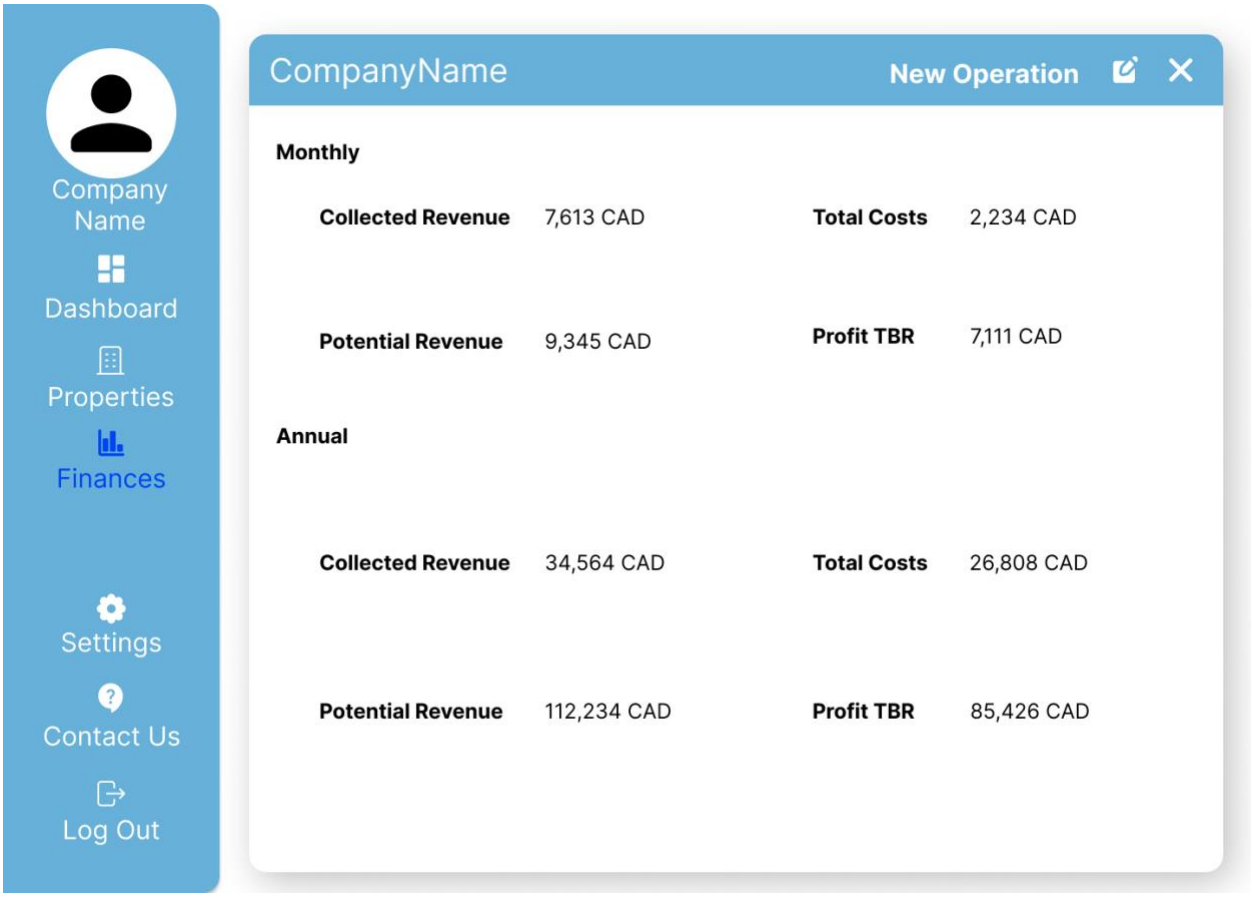
#51 - As a condo management company, I want to record operational budgets (collected condo fees) and costs.



#52 - As a condo management company, I want to enter costs for each operation.



#53 - As a condo management company, I want the financial system to generate an annual report showing all condo fees collected for a given year.


Company Name
Dashboard
Properties
Finances
Settings
Contact Us
Log Out
CompanyName
New Operation
Monthly
Collected Revenue
7,613 CAD
Total Costs
2,234 CAD
Potential Revenue
9,345 CAD
Profit TBR
7,111 CAD
Annual
Collected Revenue
34,564 CAD
Total Costs
26,808 CAD
Potential Revenue
112,234 CAD
Profit TBR
85,426 CAD

6.5. TESTING PLAN

Condo Management System

Testing Plan for Sprint #3

27/02/2024

Testing Approaches

The approaches to take in testing the system for this project are varied to ensure that we have an efficient and effective testing plan. For the duration of the project, we will use the Agile methodology to conduct testing iteratively throughout the sprints. In addition, and in line with the Agile method, we will also incorporate Test Driven Development to write tests before the code is written. We will also conduct exploratory tests to increase the user experience by allowing ourselves to test multiple functions of the system simultaneously. Lastly, we will configure our CI Pipeline to run the tests automatically to catch issues early in the development.

Testing Tools, Coverage and Metric

In this project, the technologies we use consist of Next.js framework for both the backend and the frontend. As such, the testing tools must be compatible with these frameworks. Since we will make use of Next.js, and given that it utilizes React.js for the front end of our project, the decision for a testing tool to use for unit and integration tests that would be compatible with the technologies is JEST. It is a popular testing framework, providing a "zero-configuration" testing experience. It also has a built-in tool for code coverage which is essential for the purposes of this project. We will be writing unit and integration tests for our project to produce testing analytics and verify the quality of the code.

For the testing metrics, JEST reports test coverage, test execution time and test pass/fail ratio. When "npx jest --coverage" is executed in the terminal of the project, it will create a coverage directory with coverage information such as statements, branches, functions and lines. Another command is executed in the terminal for execution time. The results of these tests will be saved and analyzed in a short testing report. We are aiming for a code coverage of at least 80%. We believe that this number is both feasible and a promising ratio for the proper functioning of the system.

For the code quality metrics, we will be using ESLint, which is a static code analysis tool for identifying and reporting potential errors and improving code quality. It is integrated in the IDEs to provide feedback as the team writes code. It is possible to install many plugins and extensions to cover many code quality metrics.

ESLint reports static code analysis violation, code duplication, code smells and performance metrics when ESLint is executed in the terminal for the entire project. It gives feedback, shows potential errors and improvements. If fixable errors are detected, it is possible to run the command " npx eslint --fix " to fix all the errors detected. For code duplication, we will be using an ESLint plugin and updating the ESLint configuration file. This will detect code duplication when we run ESLint in the terminal of the entire project.

Using these tools will improve code readability, catch bugs early in development and will help us follow a consistent coding standard. The static code analysis tools will be integrated in the continuous integration pipeline.

Another testing metric of our focus is defect density. It is related to the number of found defects per unit of code. We believe that this metric helps the testers to write helpful and efficient tests for the project. We will be using the results of the test coverage to calculate the defect density.

Acceptance Tests

Feature: Viewing Annual Condo Fee Reports (US #53)

Given I am a condo management company authorized representative accessing the financial system page, I want to see and open annual reports showing all condo fees collected when I select a specific year.

Acceptance Criteria	Example Test Data
• Financial system page prompts the user to select an year in a dropdown list. • The user must select a year. • All documents related to the selected year appear on the page with appropriate title of what the document contains. • The user clicks on the document to open. • The page should display the pdf file in a new tab. • The condo fee page should be accessible to authorized representatives of the condo management company.	Selected year: 2023 Selected Document: Report_2023.pdf

Feature: Record Operational Budget (US #51)

Given I am a condo management company authorized representative accessing the operation fees page, I should see a form to enter information about a fee I just collected and a form to enter the cost of a service.

Acceptance Criteria	Example Test Data
• The page should display a form with the following information: type, condo, service, cost, date, people involved and a submit button. • The form should have a user-friendly layout. • The form should display a confirmation message once submitted. • The form should check for invalid inputs. • The condo fee page should be accessible to authorized representatives of the condo management company. • The form should be associated with a condo unit	Type: Income Date: 23 March 2024 Condo: 54 Servive: Rent Cost: 2000 Condo Renter: Jane Doe

Feature: Set Condo Fees per Area and Parking (US #50)

Given I am a condo management company authorized representative accessing the condo fee page, I should be able to set up a condo fee per square foot and per parking spot when I enter the area and the number of parking spots. I would like the page to automatically calculate the price and I am allowed to adjust the price.

Acceptance Criteria	Example Test Data
• The condo fee page should be accessible to authorized representatives of the condo management company. • The form should prompt the user for condo unit, condo area and the number of parking spots • Once the form is submitted, the system calculates its price. • The representative is allowed to adjust the price. • Once the price is confirmed, the system should save and associate the price with the condo unit • The form should be associated with a condo unit	Condo unit: 45 Condo area: 300 Parking Spot: 2

Feature: Detailed Information Entry (US #47)

Given I am a representative of the condo management company logged into the system and I navigate to the detailed information entry section for a specific property, I provide information for each condo unit, parking spot, and locker, including unit size, owner information, and associated condo fees. Then I submit the detailed information for each unit, parking spot, and locker should be successfully entered and associated with the property.

Acceptance Criteria	Example Test Data
• The detailed information entry section should be accessible to authorized representatives of the condo management company. • The system should allow the entry of detailed information for each condo unit, parking spot, and locker. • The information for each unit should include unit size, owner information (name, contact details), and associated condo fees. • The information for each parking spot and locker should include relevant details and associations.	Property Name: Sample Condo Condo Unit Information: Unit Size: 1000 sq. ft. Owner Information: John Doe, johndoe@example.com Condo Fees: $500/month Parking Spot Information:

• Upon submitting the information, the system should store and associate the detailed information with the respective property. • The entered information should be accurate and reflect the current state of each condo unit, parking spot, and locker.	Parking Spot Number: P001 Owner Information: Jane Smith, janesmith@example.com Condo Fees: $50/month Locker Information: Locker Number: L001 Owner Information: Alex Johnson, alexjohnson@example.com Condo Fees: $10/month

Feature: Registration key processing to make a public user a condo owner or renter (US #41)

Given I am a public user with a registration key obtained from my condo management company assessing my profile page, when I enter my valid key, the system should update my status and make me a condo owner/renter and show appropriate information.

Acceptance Criteria	Example Test Data
• A registration key form should be available on the public user's profile page • The user must provide a valid key and submit the form • The user should receive a confirmation message that the key is valid • Upon submitting the valid key, the public user's status changes to condo owner/renter and the system should let the user have access to all functionalities related to condo owners/renters.	Registration key: RC-93752

Feature: Set Operation Costs (US #52)

Given I am a representative of the condo management company accessing the operation fee page, I provide information about the price and operation name. When I submit the information, each operation is associated with a cost. I can also see all operations in the system with their costs.

Acceptance Criteria	Example Test Data
• The operation fee page should only be accessible to authorized representatives of the condo management company. • The system should display a form to add the name of the operation and its cost. • The user enters valid data into the form and submits the form. • The system should display a message if the submission was successful or if an error occurred. • The added operation gets added on the list of operations with their associated fee and is visible to the user on the same page.	Operation: Window cleaning Cost: 1400

7. SPRINT 3 DELIVERABLES

7.1. SPRINT RETROSPECTIVE

Behind the Scenes of Sprint 3

Introduction

As we complete our project's third sprint, it is essential to look back and analyze our progress in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Enterprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system,a management system, and a financial system.

During this sprint, our main goal was to complete the main parts of our web platform, which would serve as a foundation for our mobile app that will be implemented in the fourth sprint. As team members are familiarizing with the technologies used throughout the past sprints, sprint 3 went relatively smoother.

When reflecting on this sprint's outcome, we were proud of our progress but still believe that there is room for improvement. Among various achievements, we had less problems with our branch merges, we used pull requests instead of force merging branches like in the last sprint, and we finished the SAD document and most of the frontend of the website. As the next sprint is approaching, we have a better understanding of the Agile refinement process and the areas that the team needs to work on. We will also be tackling the mobile platform in the upcoming sprint and we will additionally be fine-tuning our website to add the finishing touches.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement concerning our organization. Although it has significantly improved from last sprint, there is still some confusion about the overlap of our individual tasks. In other words, it is sometimes unclear as to who does which task which may affect us doing our tasks as we are sometimes waiting on others.

What went wrong

1 - Lack of Organization

Throughout the third sprint, team members were often busy with assignments and other projects as the semester is coming to an end in less than a month. Once again, everyone had very different schedules which made it challenging to assign tasks as we were often unavailable to hold a meeting altogether with all the members. The impact of this was that we still ended up with some

unfinished tasks for which we will have to catch up on in sprint 4. There was a lot of confusion as well as of what was completed and what was left to be done.

We addressed it by holding small meetings with whichever members were available at the chosen time and so we were able to assign most of the tasks. However, it would have been better to hold more frequent meetings to update each other on what is left to be done.

2 - Lack of Communication

Throughout the previous sprints, team members, as mentioned before, had widely differing schedules. The impact of this was that progress was slow because members were not always available or present to answer questions or to offer help when needed.

We addressed it by holding ourselves more accountable this sprint. Members made an effort to be more active in order to help each other out when needed. However, it would have been better to be more active since the start as this has delayed our progress already.

What went right

1 - Having a Clearer Idea Of The Design

Throughout the third sprint, we were able to collectively come up with some sketches and templates of what we want our final UI design to look like for the website and the upcoming mobile app. The impact of this was that we better understood what needed to be added, removed or modified in the web app we were creating. Through this, our teamwork has gotten better and, even though we lacked communication, establishing our vision of the web app helped with the progress nonetheless.

However, the UI for a public user should be different from a building manager for example. Therefore, it was tricky to figure out which design we should use for which type of user. We should have better established that and in more detail in the first place.

2 - Flexibility of The Team Members

Throughout the third sprint, some members, as mentioned previously, team members were willing to challenge themselves and take on tasks even though they had little knowledge of the technology required for that part. Moreover, in desperate times, team members were willing to get together to put collective effort to help each other out with the remaining tasks. Other members were also willing to give tutorials to teach other members how to do their work.

The team’s flexibility and adaptability made it so that the work that needed to be done was ultimately finished in the end despite our lack of communication and organization at the beginning. However, had we been more flexible at the beginning, we would have encountered less difficulties during this sprint.

Conclusion

While reflecting on the third sprint, we realized that we encountered multiple challenges that originated from our lack of organization in terms of the tasks, still, and our lack of communication. However, we tackled these problems through the group's adaptability and flexibility. Although our teamwork is not perfect, we are in a learning process to improve on it to the best of our ability. Things worked out better in sprint 3 than the previous one because we established earlier on what our vision was for what was to be done for this sprint. We got to refine the rough edges and we were able to bond a bit more which improved the team dynamic.

As for the remaining upcoming sprints, we have learned many lessons that we will be applying for the rest of the project to ensure a smoother progression. For example, we now know that we need to have a better idea of the tasks that are still to be completed so as to not further delay our project's delivery. We also learned that we need to be more communicative with each other because we need to clearly establish the tasks to be assigned and done for this sprint and for the next. This sprint was an effective wake up call on the area for which we need more improvement as a team. By applying the lessons we learned, we will tackle future challenges being more informed and committed to developing the best product possible.

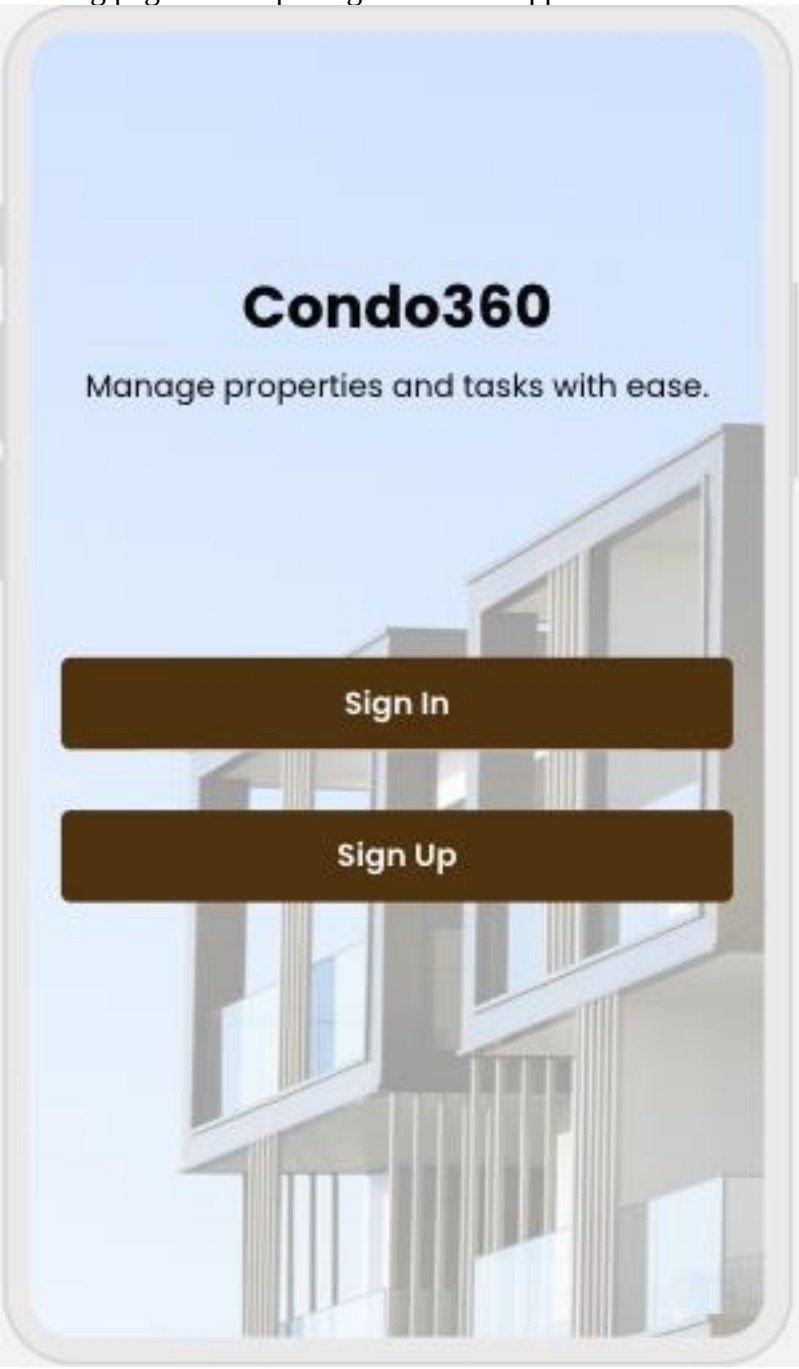
7.2. RELEASE PLAN

Title	URL	Estimate	Labels
As a condo management company, I want to access the details of my property from the mobile app	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/163		Mobile, User Story
Implement overview of properties UI page	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/164	2	development, frontend, Mobile, task
Implement property details UI page	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/165	3	development, frontend, Mobile, task
As a condo owner or rental user, I want to reserve common facilities in a calendar-like interface.	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/63		Mobile, tasks TBA, User Story, Web
Add backend code to fetch reservations by facility	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/156	2	backend, development, task, Web
Create parking space overview page.	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/106	3	development, frontend, task, Web
Create frontend page for reservations	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/155	3	development, frontend, task, Web
Add backend implementation to save reservations	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/154	2	backend, development, task, Web
Add db tables for reservations	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/153	1	database, development, task, Web
As a user, I want to access the details of my profile	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/161		Mobile, User Story
Implement profile page UI	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/162	2	development, frontend, Mobile, task
As a user, I want to access my account on the mobile version of the platform	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/157		Mobile, User Story
Add a home page for the app	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/160	2	development, frontend, Mobile, task
Implement Signup UI	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/159	2	development, frontend, Mobile, task
Implement login UI	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/158	2	development, frontend, Mobile, task
Carryover tasks from past sprints, reevaluated			
As a condo management company, I want to set up the condo fee per square foot and per parking spot and calculate condo fees for each unit.	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/50		tasks TBA, User Story, Web
Add backend code for parking fee update	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/111	2	backend, development, task, Web
Add fields to unit table	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/112	1	database, development, task, Web
Modify the unit overview page	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/113	2	development, frontend, task, Web
Add backend code for unit field updates	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/114	2	backend, development, task, Web
Create backend code to return total expense for a user.	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/115	3	backend, development, task, Web
Update financial status page	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/116	2	development, frontend, task, Web
Add link of parking page for a property	https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/issues/105	1	development, frontend, task, Web
Total		37	
Sprint Burndown Chart			
Assuming we start on the 25th, we will have 17 days in this sprint.			
Total Story Points = 37			
Ideal Daily Burndown Rate = 37/17 = 2.17 USP			

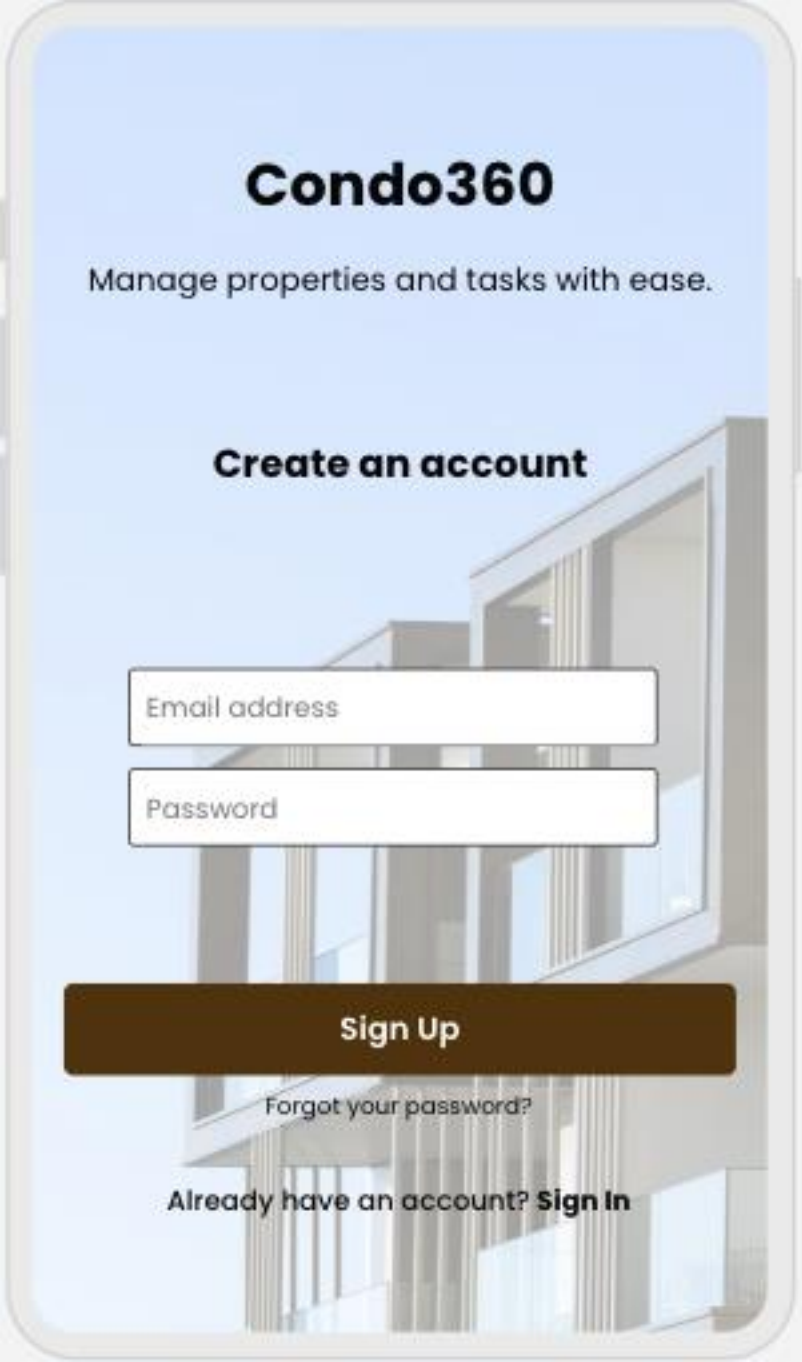
7.3. UI PROTOTYPE

User Story #157 – As a user, I want to access my account on the mobile version of the platform.

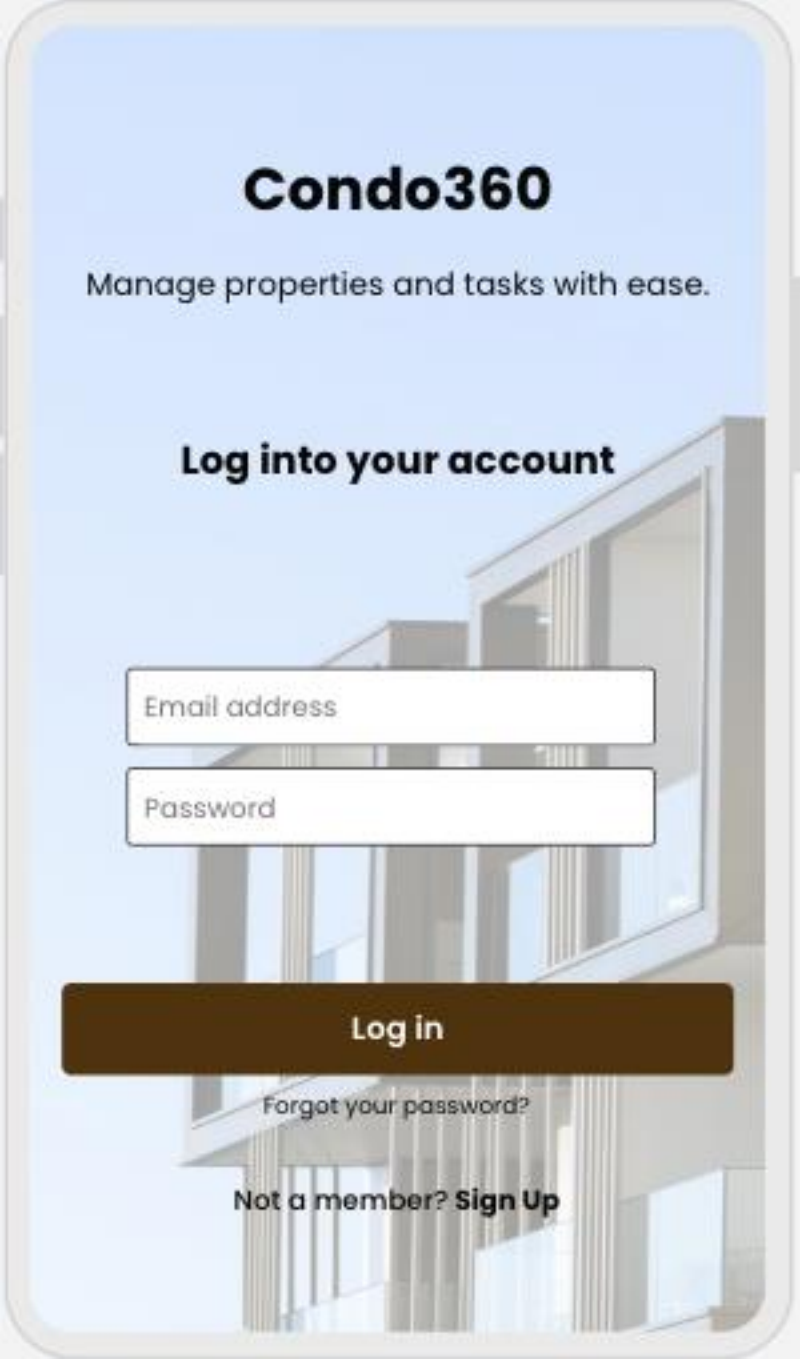
Landing page when opening the mobile app



Sign Up page


Condo360
Manage properties and tasks with ease.
Create an account
Email address
Password
Sign Up
Forgot your password?
Already have an account? Sign In

Sign In page

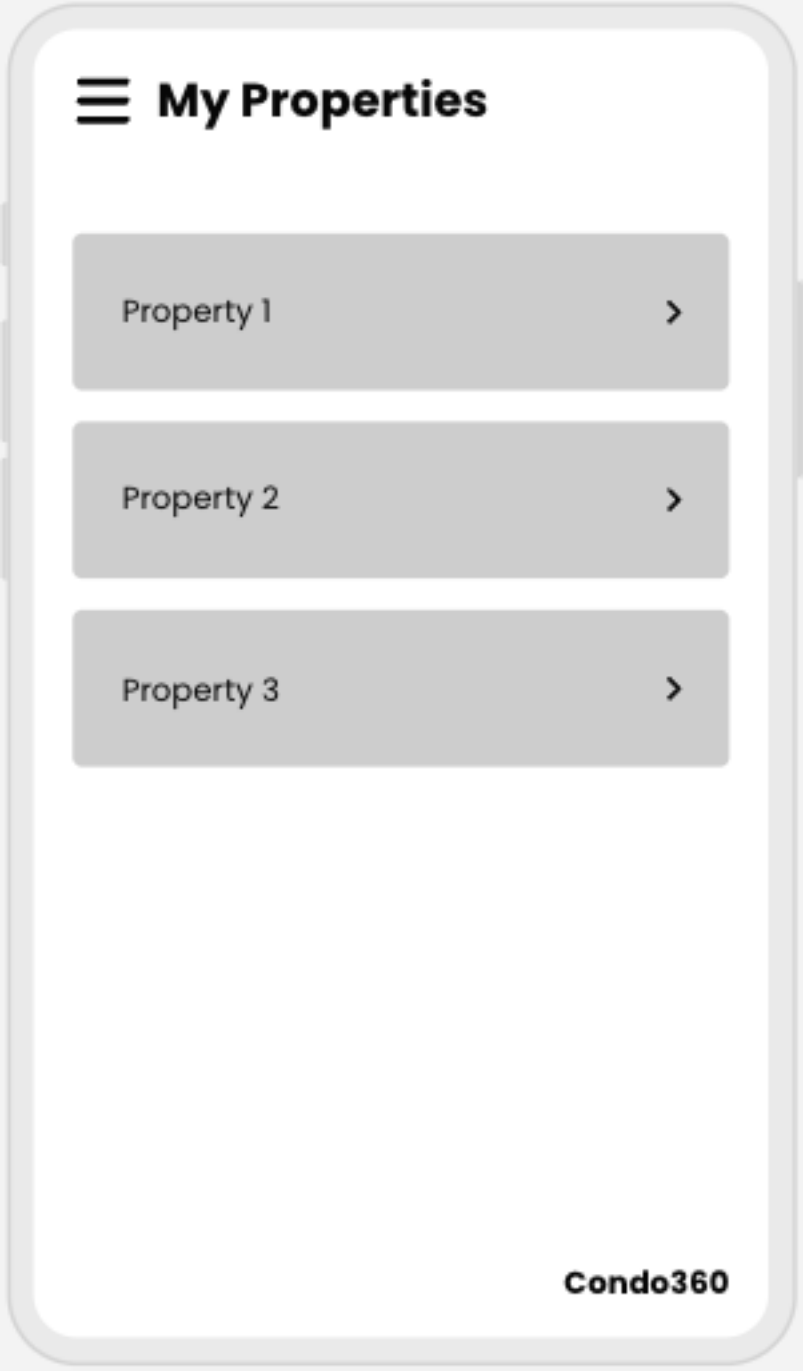


User Story #161 – As a user, I want to access the details of my profile.

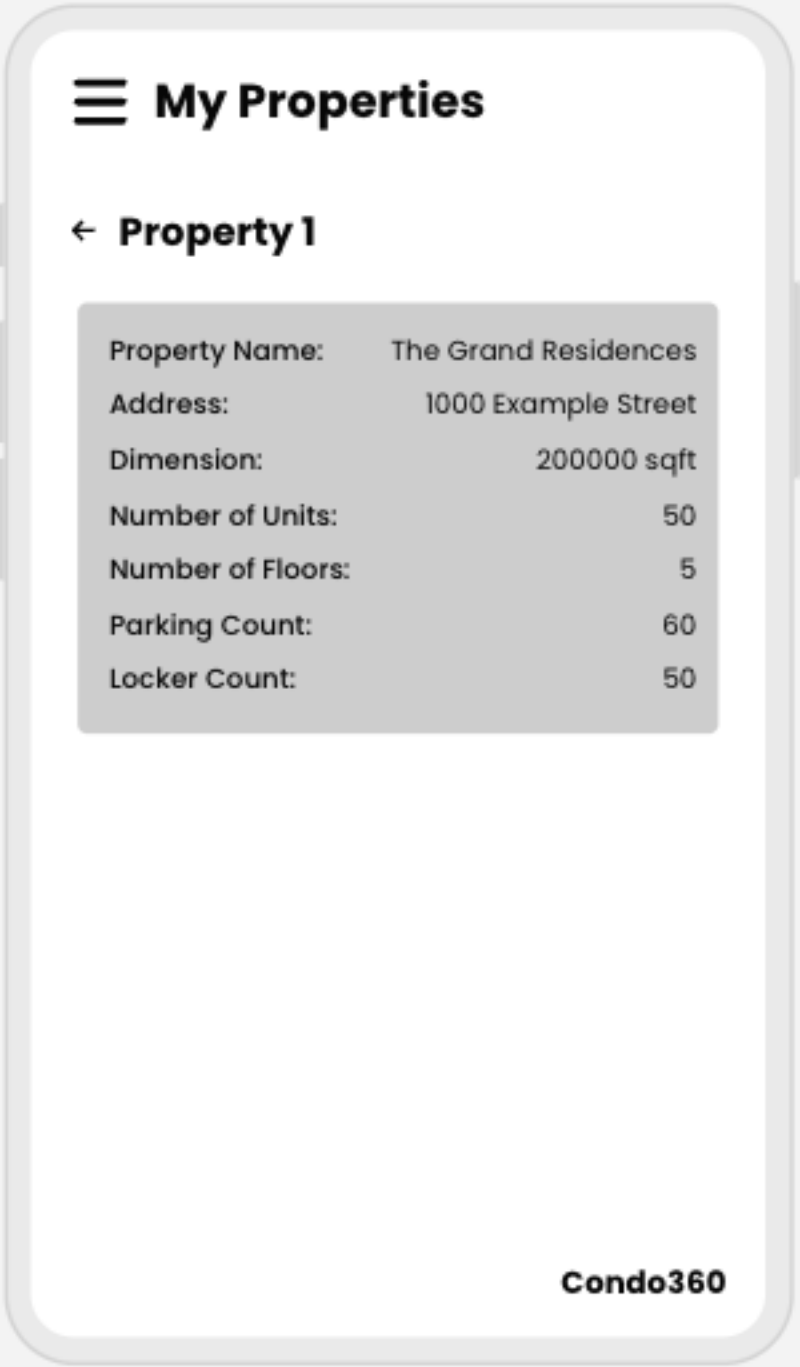
Profile page

User Story #163 – As a condo management company, I want to access the details of my property from the mobile app.

Overview page for properties



Details page for a property


My Properties
Property 1
Property Name: The Grand Residences
Address: 1000 Example Street
Dimension: 200000 sqft
Number of Units: 50
Number of Floors: 5
Parking Count: 60
Locker Count: 50
Condo360

7.4. TESTING PLAN

Condo Management System

Testing Plan for Sprint #4

22/03/2024

Testing Approaches

Testing strategies for this project will encompass a diverse range to optimize our testing plan's efficiency and effectiveness. Throughout the project timeline, we will employ Agile methodology, conducting iterative testing during sprints. Complementing Agile, Test Driven Development will be integrated, prioritizing test writing before code implementation. Exploratory testing will also be conducted to enhance user experience, allowing simultaneous testing of various system functions. Moreover, our CI Pipeline will be configured to automatically execute tests, facilitating early issue detection during development.

Testing Tools, Coverage and Metric

For our project, we're utilizing the Next.js framework for both backend and frontend development. Therefore, our choice of testing tools must seamlessly integrate with these frameworks. Since Next.js leverages React.js for frontend development, JEST emerges as the optimal testing tool for unit and integration tests due to its compatibility. JEST is renowned for its "zero-configuration" testing environment and includes a built-in code coverage tool, crucial for our project's objectives. We'll be extensively utilizing unit and integration tests to generate testing analytics and validate code quality.

In terms of testing metrics, JEST provides insights into test coverage, execution time, and pass/fail ratios. Running "npx jest --coverage" in the project terminal generates a coverage directory containing details on statements, branches, functions, and lines. Similarly, a command is used to assess execution time. Results from these tests will be stored and evaluated in a concise testing report. Our target is to achieve a minimum code coverage of 80%, a figure we deem both achievable and indicative of the system's robust functionality.

To gauge code quality, we'll leverage ESLint, a static code analysis tool renowned for pinpointing potential errors and enhancing overall code quality. Integrated into IDEs, ESLint offers real-time feedback as the team codes, ensuring adherence to best practices. With the flexibility to install numerous plugins and extensions, we can encompass a wide array of code quality metrics.

ESLint performs static code analysis during execution in the terminal for the entire project, highlighting violations, code duplication, and performance metrics. It provides valuable feedback, indicating potential errors and areas for improvement. Fixable errors can be addressed by running the command "npx eslint --fix" to automatically resolve detected issues. To address code duplication, an ESLint plugin will be utilized, and the ESLint configuration file will be updated to detect duplication during project-wide terminal execution.

Employing these tools will enhance code readability, detect bugs at an early stage of development, and ensure adherence to a uniform coding standard. Integration of static code analysis tools will be seamlessly incorporated into the continuous integration pipeline.

Another key testing metric we prioritize is defect density, which quantifies the number of defects discovered per unit of code. We view this metric as instrumental in guiding testers to craft effective and beneficial tests for the project. Test coverage results will be utilized to compute defect density, aiding in our assessment of testing efficacy.

Acceptance Tests

Feature: Notifications (US #44)

As a condo owner, I want to receive notifications about the latest activities in submitted or assigned requests

Acceptance Criteria	Example Test Data

• The system should send an email detailing the newly submitted request
• The system should send an email detailing the newly assigned request
• The system should send an email detailing any updates on the requests
• The updated status should be reflected on the condo owner's request page when they log in to check the status of their assigned requests on the web platform
	• Request Submitted Notification:
	  • Request ID: 011
	  • Submitted By: John Doe
	  • Date Submitted: April 10, 2024
	  • Status: Pending
	  • Description: Moving in request.

Feature: File upload (US #46)

As a condo management company, I want to upload condo files for each property, accessible to all condo owners, including declarations, annual budgets, and board meeting minute

Acceptance Criteria	Example Test Data
• The system should reflect the updated list of documents accessible by all condo owners in the Properties page	• Upload File named "Declaration 2023" • For Property: 100 Random Street

Feature: Mobile platform (US #214)

As a condo owner, I want to access my account on the mobile version of the platform.

Acceptance Criteria	Example Test Data
• User should be able to open the app and navigate to their account and have it be displayed on the screen	PROFILE PICTURE Name: Gladius Maximus Email: ramranch@gmail.com Phone number: 514 7346784673290234

Feature: Mobile platform (US #157)

As a user who has a working account, I want to access my account on the mobile version of the platform.

Acceptance Criteria	Example Test Data
• The platform should have a responsive design that adapts to various screen sizes and orientations, ensuring optimal usability and readability on mobile devices. • Users should be able to access their accounts by logging in using their credentials (e.g., username/email and password) on the mobile version of the platform. • The login process should be intuitive and user-friendly, with appropriate error messages displayed for invalid credentials or login failures. • Once logged in, users should be able to access their account information, including profile details, and settings through the mobile interface.	Username: Scooby Password: Secret123?

Feature: Property Details on Mobile (US #163)

As a condo management company, I want to access the details of my property from the mobile app.

Acceptance Criteria	Example Test Data

• Condo management company should be able to open the mobile app, go to the Properties page, and see their properties' details displayed on the screen	• Property Details
	  • Name: 100 Random Street
	  • Number of Units
	  • Number of Parking Spots
	  • Number of Lockers
	  • Square feet of each unit

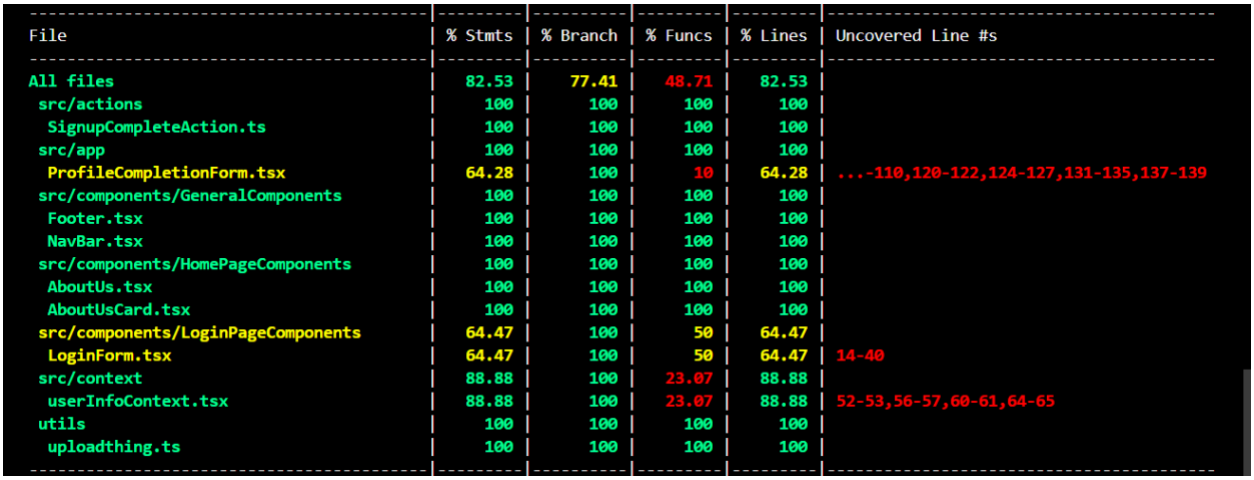
Feature: Access Account on Mobile (US #215)

As an employee, I want to access my account on the mobile version of the platform.

Acceptance Criteria	Example Test Data
• User should be able to display and upload a profile picture • Users should be able to set and change their name • Users should be able to display their gmail. • Users should be able to display and edit their phone number	PROFILE PICTURE Name: Gladius Maximus Email: ramranch@gmail.com Phone number: 514 7346784673290234

8. SPRINT 4 DELIVERABLES

8.1. CODE COVERAGE



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	82.53	77.41	48.71	82.53	
src/actions	100	100	100	100	
SignupCompleteAction.ts	100	100	100	100	
src/app	100	100	100	100	
ProfileCompletionForm.tsx	64.28	100	10	64.28	...-110,120-122,124-127,131-135,137-139
src/components/GeneralComponents	100	100	100	100	
Footer.tsx	100	100	100	100	
NavBar.tsx	100	100	100	100	
src/components/HomePageComponents	100	100	100	100	
AboutUs.tsx	100	100	100	100	
AboutUsCard.tsx	100	100	100	100	
src/components/LoginPageComponents	64.47	100	50	64.47	
LoginForm.tsx	64.47	100	50	64.47	14-40
src/context	88.88	100	23.07	88.88	
userInfoContext.tsx	88.88	100	23.07	88.88	52-53,56-57,60-61,64-65
utils	100	100	100	100	
uploadthing.ts	100	100	100	100	

8.2. SPRINT RETROSPECTIVE

Behind the Scenes of Sprint 4

Introduction

As we complete our project's fourth sprint, it is essential to look back and analyze our progress in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Enterprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system,a management system, and a financial system.

During this sprint, our main goal was to complete the entire web and mobile platforms. As team members are familiarizing with the technologies used throughout the past sprints, sprint 4 went relatively smoother.

When reflecting on this sprint's outcome, we were proud of our progress but still believe that there is room for improvement. Among various achievements, we had less problems with our branch merges, we used pull requests instead of force merging branches like in sprint 2, we refined the documentation and finished the web and mobile apps. As the end of the sprint is approaching, we have a better understanding of the Agile refinement process and the areas that the team needs to work on. We will also be working on tying up any loose ends before the deployment and the product presentation for sprint 5.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement concerning our organization. Although it has significantly improved from last sprint, there is still some confusion about the overlap of our individual tasks. In other words, it is sometimes unclear as to who does which task which may affect us doing our tasks as we are sometimes waiting on others.

What went wrong

1 - Lack of Organization

Throughout the fourth sprint, team members were often busy with assignments and other projects as the semester is coming to an end in less than a month. Once again, everyone had very different schedules which made it challenging to assign tasks as we were often unavailable to hold a meeting altogether with all the members. The impact of this was that we still ended up with some unfinished tasks for which we will have to catch up on in sprint 4. There was a lot of confusion as well as of what was completed and what was left to be done.

We addressed it by holding small meetings with whichever members were available at the chosen time and so we were able to assign most of the tasks. However, it would have been better to hold more frequent meetings to update each other on what is left to be done.

2 - Lack of Communication

Throughout the previous sprints, team members, as mentioned before, had widely differing schedules. In addition, as we are approaching the end of the semester, members are juggling multiple team projects, assignments and the preparation for the final exams at the same time. The impact of this was that progress was slow because members were not always available or present to answer questions or to offer help when needed.

We addressed it by holding ourselves more accountable this sprint. Members made an effort to be more active in order to help each other out when needed. However, it would have been better to be more active since the start as this has delayed our progress already.

What went right

1 - Having a Clearer Idea of The Mobile Design

Throughout the fourth sprint, we were able to collectively come up with some sketches and templates of what we want our final UI design to look like for the website and the upcoming mobile app. The impact of this was that we better understood what needed to be added, removed or modified in the web app we were creating. Through this, our teamwork has gotten better and, even though we lacked communication, establishing our vision of the web app helped with the progress nonetheless.

However, it was tricky to figure out at first how much we needed to implement for the mobile app compared to the web app so we needed to ask the

2 - Having a Clearer Idea of What Tasks To Focus On

At the beginning of the fourth sprint, we had a meeting to discuss how we were going to proceed with sprint 4. We decided to split into sub-units who will work on different types of tasks. For example, two members were responsible for correcting and refining the documentations. Another few members worked on the mobile app while the rest worked on finishing up the web platform. That way, there was more collaboration between the teammates while ensuring that we are covering everything that needs to be done.

However, we did not hold each other accountable enough so we ended up doing some of the tasks at the last minute.

Conclusion

While reflecting on the fourth sprint, we realized that we encountered multiple challenges that originated from our lack of organization in terms of the tasks, still, and our lack of

communication. However, we tackled these problems by having a clearer picture of what is left to be done and who is doing what. Although our teamwork is not perfect, we are in a learning process to improve on it to the best of our ability. Things worked out better in sprint 5 than the previous one because we established earlier on what our vision was for what was to be done for this sprint. We got to refine the rough edges and we were able to bond a bit more which improved the team dynamic.

As for the last remaining sprint, we have learned many lessons that we will be applying for the rest of the project to ensure a smooth delivery and deployment. For example, we now know that we need to have a better idea of the tasks that are still to be completed so as to not further delay our project's delivery. We also learned that we need to be more communicative with each other because we need to clearly establish the tasks to be assigned and done for this sprint and for the next. By applying the lessons we learned, we will tackle future challenges being more informed and committed to developing the best product possible. We will also be able to carry on these newfound skills onto future projects.

8.3. RELEASE PLAN

Table 1. User story points and priority of user stories that will be implemented in Sprint 3

User Story ID	User Story Points (USP)	Priority	Status
#44	2	High	TO DO
#43	6	High	TO DO
#157	2	Medium	TO DO
#214	2	Medium	TO DO
#163	2	Medium	TO DO
#215	2	Medium	TO DO
Total USP	16		

*The User Story Points are approximately proportional to the expected number of hours a task should take (going from 1 to 5 for each task).

Sub-User Stories:

Condo Owner

#44 – As a condo owner, I want to receive notifications about the latest activities in submitted or assigned requests.

- Add a notification popup on the profile page for each new update in requests (2 points).

Condo Management Companies

#46 – As a condo management company, I want to upload condo files for each property, accessible to all condo owners, including declarations, annual budgets, and board meeting minutes.

- Create the upload form for documents (1 point).
- Create the Documents page (1 point).
- Get documents by user ID of renter/owner of unit (1 point).
- Add file table to the database (1 point).
- Post pdf files to the database (2 points).

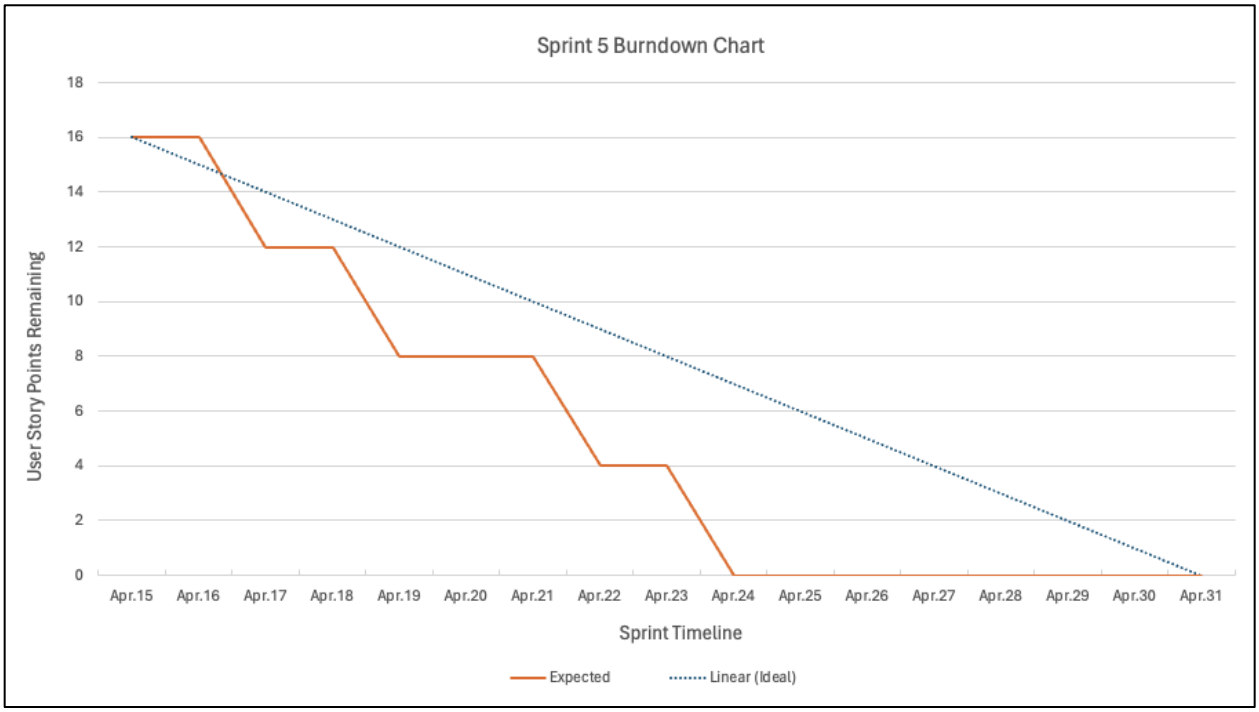
Mobile App

#157 – As a public user, I want to access my account on the mobile version of the platform.
#214 – As a condo owner, I want to access my account on the mobile version of the platform.
#163 – As a condo management company, I want to access the details of my property from the mobile app.
#215 – As an employee, I want to access my account on the mobile version of the platform.

Since the UI has been completed, we simply need to link to front-end of the mobile app to the back-end of the web app for each user to see their respective account (2 points for each user type).

Burndown Chart

Let’s consider that we start working on April 15 and that the end of the sprint is May 1st. The sprint duration is 16 days. Since this is the last sprint, we need to leave the last week for the final presentation. Therefore, we can estimate that we have 9 days for the user stories completion. Let’s also consider the weekend (2 days) as off.
Total Story Points = 16
`Ideal Daily Burndown = Total Story Points / Sprint Duration = 16 / 7 ≃ 2.29`
Therefore, we would ideally complete around 2.29 USP every day if we take only 2 days off. The burndown chart below illustrates our expected progress rate.



8.4. TESTING PLAN

Condo Management System

Testing Plan for Sprint #5

14/04/2024

Testing Approaches

Testing strategies for this project will encompass a diverse range to optimize our testing plan's efficiency and effectiveness. Throughout the project timeline, we will employ Agile methodology, conducting iterative testing during sprints. Complementing Agile Development, Test Driven Development will be integrated, prioritizing test writing before code implementation. Exploratory testing will also be conducted to enhance user experience, allowing simultaneous testing of various system functions. Moreover, our CI Pipeline will be configured to automatically execute tests, facilitating early issue detection during development.

Testing Tools, Coverage and Metric

For our project, we're utilizing the Next.js framework for both backend and frontend development. Therefore, our choice of testing tools must seamlessly integrate with these frameworks. Since Next.js leverages React.js for frontend development, JEST emerges as the optimal testing tool for unit and integration tests due to its compatibility. JEST is renowned for its "zero-configuration" testing environment and includes a built-in code coverage tool, crucial for our project's objectives. We'll be extensively utilizing unit and integration tests to generate testing analytics and validate code quality.

In terms of testing metrics, JEST provides insights into test coverage, execution time, and pass/fail ratios. Running "npx jest --coverage" in the project terminal generates a coverage directory containing details on statements, branches, functions, and lines. Similarly, a command is used to assess execution time. Results from these tests will be stored and evaluated in a concise testing report. Our target is to achieve a minimum code coverage of 80%, a figure we deem both achievable and indicative of the system's robust functionality.

To gauge code quality, we'll leverage ESLint, a static code analysis tool renowned for pinpointing potential errors and enhancing overall code quality. Integrated into IDEs, ESLint offers real-time feedback as the team codes, ensuring adherence to best practices. With the flexibility to install numerous plugins and extensions, we can encompass a wide array of code quality metrics.

ESLint performs static code analysis during execution in the terminal for the entire project, highlighting violations, code duplication, and performance metrics. It provides valuable feedback, indicating potential errors and areas for improvement. Fixable errors can be addressed by running the command "npx eslint --fix" to automatically resolve detected issues. To address code duplication, an ESLint plugin will be utilized, and the ESLint configuration file will be updated to detect duplication during project-wide terminal execution.

Employing these tools will enhance code readability, detect bugs at an early stage of development, and ensure adherence to a uniform coding standard. Integration of static code analysis tools will be seamlessly incorporated into the continuous integration pipeline.

Another key testing metric we prioritize is defect density, which quantifies the number of defects discovered per unit of code. We view this metric as instrumental in guiding testers to craft effective and beneficial tests for the project. Test coverage results will be utilized to compute defect density, aiding in our assessment of testing efficacy.

Acceptance Tests

Feature: Access the details of the property from the mobile app (US #163)

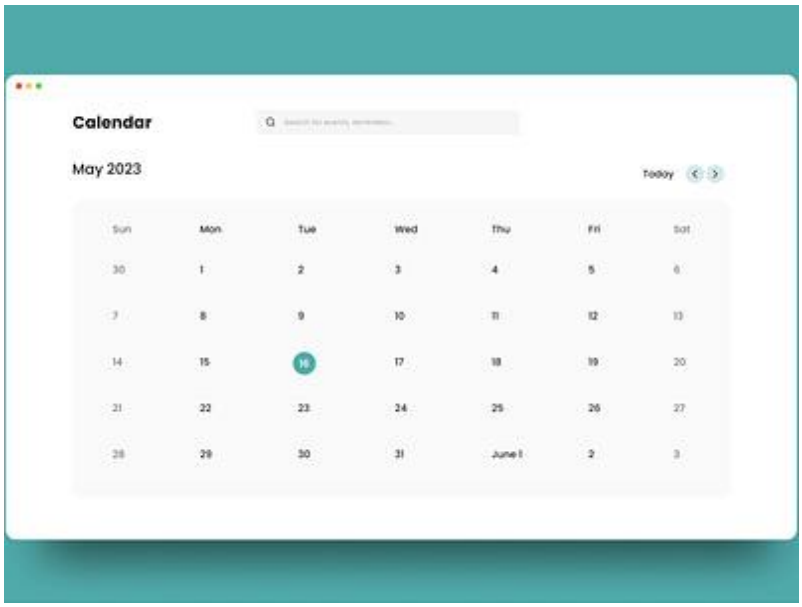
Given I am a condo management company authorized representative accessing the financial system page, I want to have access to all the details, such as annual reports showing all condo fees collected on the mobile app.

Acceptance Criteria	Example Test Data
• The detailed information entry section should be accessible to authorized representatives of the condo management company. • The system should allow the entry of detailed information for each condo unit, parking spot, and locker. • The information for each parking spot and locker should include relevant details and associations. • The entered information should be accurate and reflect the current state of each condo unit, parking spot, and locker. • All of this should be on mobile	-Name: Elderwood condos -Condo unit: 45k -Condo area: 300 -Parking Spot: 2 -Unit Size: 1000 sq. ft

Feature: Reserve common facilities in a calendar-like interface. (US #63)

Given I am a condo management company authorized representative, I want to have an easy to use and clear interface where I can reserve facilities. It has to look like a physical calendar and have quick and easy navigation.

Acceptance Criteria	Example Test Data

• The system should provide a calendar-like interface where users can view availability and make reservations for common facilities such as meeting rooms, gym, pool, etc. • The calendar should display dates clearly and allow users to navigate between months and years easily. • Users should be able to select the date and time they want to reserve a common facility. • The system should display the availability of the facility for the selected date and time, showing booked slots as unavailable. • The system should validate the selected date and time to ensure it falls within the operating hours of the common facility and does not conflict with existing reservations. • If the selected date or time is invalid or conflicts with existing reservations, appropriate error messages should be displayed, guiding users to select alternative options.	

Feature: Access details of profile (US #161)

Given I am a user of the app, I should be able to have access to the details of my profile. I should be able to not only see my details, but also edit them in case they change. It should display things such as my profile picture, name and list of properties.

Acceptance Criteria	Example Test Data
• User should be able to display and upload a profile picture • Users should be able to set and change their name • Users should be able to display their gmail. • Users should be able to display and edit their phone number	PROFILE PICTURE Name: Gladius Maximus Email: ramranch@gmail.com Phone number: 514 7346784673290234

Feature: Access account on the mobile version of the platform (US #157)

As a user who has a working account, I want to access my account on the mobile version of the platform.

Acceptance Criteria	Example Test Data
• The platform should have a responsive design that adapts to various screen sizes and orientations, ensuring optimal usability and readability on mobile devices. • Users should be able to access their accounts by logging in using their credentials (e.g., username/email and password) on the mobile version of the platform. • The login process should be intuitive and user-friendly, with appropriate error messages displayed for invalid credentials or login failures. • Once logged in, users should be able to access their account information, including profile details, and settings through the mobile interface.	Username: Scooby Password: Secret123?

9. SPRINT 5 DELIVERABLES

9.1. CODE COVERAGE

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	83.02	69.23	44.11	83.02	
actions	100	100	100	100	
SignupCompleteAction.ts	100	100	100	100	
app/(auth)/signup/complete	100	100	100	100	
page.tsx	100	100	100	100	
app/api/getSalesByCompanyId	89.18	50	100	89.18	
route.js	89.18	50	100	89.18	20-23
app/api/getUnitDetails	88.23	50	100	88.23	
route.js	88.23	50	100	88.23	21-24
app/api/getUnitFeesByUserId	90	50	100	90	
route.js	90	50	100	90	23-26
app/api/getUnitsFromProperty	88.23	75	100	88.23	
route.js	88.23	75	100	88.23	18-21
app/api/handleLockerRegistration	77.77	25	100	77.77	
route.js	77.77	25	100	77.77	23-26,29-32,41-44
components/CompleteProfileComponents	63.63	50	10	63.63	
ProfileCompletionForm.tsx	63.63	50	10	63.63	...2,124-127,131-135,137-13
9					
components/GeneralComponents	100	100	100	100	
Footer.tsx	100	100	100	100	
NavBar.tsx	100	100	100	100	
context	88.88	100	23.07	88.88	
userInfoContext.tsx	88.88	100	23.07	88.88	52-53,56-57,60-61,64-65
utils	100	100	100	100	
uploadthing.ts	100	100	100	100	

9.2. SPRINT RETROSPECTIVE

Behind the Scenes of Sprint 5

Introduction

As we complete our project's fifth sprint, it is essential to look back and analyze our progress in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Enterprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system,a management system, and a financial system.

During this sprint, our main goal was to complete, finalise, and deploy the entire web and mobile platforms. As team members are familiarised with the technologies used throughout the past sprints, sprint 5 went relatively smoothly.

When reflecting on this sprint's outcome, we were proud of our progress but still believe that there is room for improvement. Among various issues encountered, were build issues when deploying the various applications. We were able to adapt to these inconsistencies by working on different branches and merging them to resolve the various conflicting issues. Like in sprint 2, we used pull requests instead of directly merging to main branches, we refined the documentation and finished the web and mobile apps. As the end of the sprint is approaching, we have a better understanding of the Agile refinement process and the areas that the team needs to work on. We were able to tie up any loose ends before deployment and prepared the product presentation as hoped in the sprint 4 retrospective.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement concerning our organisation. Although it has significantly improved from last sprint, there is still some confusion about the overlap of our individual tasks. In other words, it is sometimes unclear as to who does which task which may affect us doing our tasks as we are sometimes waiting on others.

What went wrong

1 - Lack of communication

Throughout the fifth sprint, team members were often busy with assignments, other projects, and preparing for finals as it is the last few weeks of the semester. Once again, everyone had very different schedules which made it challenging to assign tasks as we were often unavailable to hold a meeting altogether with all the members. The impact of this was that we encountered unnecessary merging issues since team members would start working on tasks

without notifying others which caused a lot of time to be wasted trying to fix resulting merge and build conflicts.

We addressed it by holding small meetings with whichever members were available at the chosen time and so we were able to assign most of the tasks. However, it would have been better to hold more frequent meetings to update each other on what is left to be done.

2 - Deploying issues encountered

Throughout this last sprint as we started to deploy the WebApp using Vercel, we encountered many deployment failures. This was the result of lack of communication once again since we automated the deployment platform without considering many features that still needed to be fixed as well as many branching issues. This lack of communication can be attributed to the significant amount of projects due and preparation for finals, leaving every team member with different schedules which impacted the availability to hold frequent meetings to discuss task advancements and deployment status.

We addressed this issue by communicating towards the beginning and end of the sprint and dividing all the required tasks. Notably, the required fixes were discussed and implemented for the application to deploy successfully. However, it would have been better to have more frequent meetings from the beginning of the sprint, in order to prevent unnecessary overhead to fix merging, branching, and deployment issues.

What went right

1 - Teamworking skills were improved

Throughout the fifth sprint, we were able to collectively work together to resolve issues caused by lack of communication. Although short in time, we were able to divide tasks and work on fixing issues in our codebase that were causing builds to fail and preventing us from successfully deploying our applications. Through this, our teamwork has gotten better and, even though we lacked communication, redirecting our focus on tasks that required immediate attention enabled us to successfully deploy our Web App.

However, it was tricky to figure out at first how to approach the existing issues in our codebase, we were nonetheless able to communicate and resolve the various inconsistencies.

2 - Prioritising tasks to ensure a successful deployment

At the beginning of the fifth sprint, we had a meeting to discuss how we were going to proceed with sprint 5. We decided to split into subgroups of members that will work on different types of tasks. For example, two members were responsible for correcting and refining the documentation. Others worked on fixing build issues, while the rest worked on finishing up the web and mobile platforms. That way, there was more collaboration between teammates while

making sure that we were covering everything that needed to be done to ensure a successful deployment of the platforms.

However, we did not hold each other accountable enough so we ended up doing some of the tasks at the last minute.

Conclusion

While reflecting on the fifth sprint, we realised that we encountered multiple challenges that originated from our lack of communication, in terms of the tasks. However, we tackled these problems by assigning subgroups of members to certain tasks to ensure successful deployment. Although our teamwork is not perfect, it has significantly improved by trying to communicate more often to the best of our abilities. Things worked out better in sprint 5 compared to previous ones since we established earlier on what our vision was for what was to be completed for this sprint. We got to refine the rough edges and we were able to bond a bit more which improved the team dynamic.

In this last sprint, looking back at the various sprints and resulting product, we have learned many lessons. Notably, the importance of communication, organisation, and teamwork are crucial when it comes to software development, in a team-based environment. In regards to the organisation of the project, the importance stems from the need to prioritise tasks and be realistic about what can be accomplished by implementing effective communication to ensure smooth delivery and deployment. By applying the lessons we learned, we will tackle future challenges by being more informed and committed to developing the best product possible. We will also be able to carry on these newfound skills in future projects.

9.3. PROJECT RETROSPECTIVE

Behind the Scenes of the Project

Introduction

As our mini-capstone project comes to an end, it is essential to look back and analyse the process in this postmortem analysis. The objective of this project is to develop a condo management website and its companion mobile app for potential buyers, condo owners and condo management companies. The core structure of this application is an Enterprise Resource Planning (ERP) system, which encompasses various functionalities such as user profile creation, a reservation system,a management system, and a financial system.

Our main goal was to offer the client and end users a web and mobile platform that will allow condo management companies to use our software to manage their condo buildings, for condo owners to manage their respective condo building and for public users to be able to access information on their condo units through this app.

When reflecting on this project's outcome, we were proud of our accomplishments but still believe that there is room for improvement. Among various issues encountered, were our bad time management. Throughout the semester, team members were busy with their other classes and projects, which made it difficult to coordinate when to finish our tasks. We also had issues involving understanding the requirements. However, we were able to tie up any loose ends before deployment and prepared the product presentation as hoped in the sprint 4 retrospective.

After filling out a retrospective board anonymously, we saw that there was a mutual agreement concerning our organisation. Although it has significantly improved since the first sprint, there is still some confusion as to what needs to be refined and reworked on. We also encountered a few merge issues on our GitHub as well.

What went wrong

1 - Lack of communication

Throughout the project, team members were often busy with assignments, other projects, and exams. Once again, everyone had very different schedules which made it challenging to assign tasks as we were often unavailable to hold a meeting altogether with all the members. The impact of this was that we encountered unnecessary merging issues since team members would start working on tasks without notifying others which caused a lot of time to be wasted trying to fix resulting merge and build conflicts.

We addressed it by holding small meetings with whichever members were available at the chosen time and so we were able to assign most of the tasks. However, it would have been better to hold more frequent meetings to update each other on what is left to be done.

2 - Time management

Throughout the project, the team struggled with time management, because, as mentioned in the last point, every team member has very different schedules and responsibilities. We all had assignments and group projects in other classes as well. This made it so that we often left our task to be done at the last minute. As we scramble to get our work done in the last few days of each sprint, we fail to pay attention to small details that end up costing us points that could have easily been avoided. We also ended up having to stay up late trying to fix our errors and make as much significant progress as possible. This could have been avoided had we given ourselves more strict deadlines and held each other accountable.

We addressed this issue by communicating to each other in the last week each sprint what is left to be done. Notably, we make sure to communicate to each team member the tasks they need to accomplish for the sprint. However, it would have been better if we had communicated that from the very beginning and checked up on each other to ensure that progress is made in an appropriate time frame instead of all near the end of the sprint.

3 - Better understanding of the requirements from the get-go

Throughout the project, the team struggled with understanding the requirements. As we had bad time management, we often overlooked small details that mattered. As a result, we lost points due to our lack of clear understanding of the deliverable requirements. For example, there were mistakes or missing information in documents in our Sprint 2 deliverables that were not fixed until Sprint 4. We were not careful enough in trying to understand our errors which led to us losing points that we could have easily avoided losing. This could have been avoided had we taken better care and more time to fully understand the requirements as well as asked questions to the client (the TA) earlier on to take those new information into account for our sprint.

We addressed the issue by making a list of questions to be asked to the client (the TA). We also made use of our demo time with the TA to verbally ask questions and clarifications on the expectations regarding our software and our documentations. However, it would have been better to have had a clearer idea of what was needed from the beginning or early on throughout the sprints so that we don’t waste time fixing errors caused by our misunderstandings.

What went right

1 - Teamworking skills were improved

Throughout the sprints, we were able to collectively work together to resolve issues caused by lack of communication. Although we had a rocky start, we were able to more effectively divide

our tasks through our newfound understanding of each other's strengths and weaknesses. Through the project, our teamwork has gotten better and, even though we lacked communication, redirecting our focus on tasks that required immediate attention enabled us to successfully deploy our Web App.

However, it was tricky to figure out at first how to approach the existing issues in our codebase, we were nonetheless able to communicate and resolve the various inconsistencies.

2 - Flexibility and adaptability of the team members

Throughout the project, team members were willing to challenge themselves and take on tasks even though they had little knowledge of the technology required for that part. Moreover, team members were willing to get together to put collective effort to help each other out with the remaining tasks. Other members were also willing to give tutorials to teach other members how to do their work.

The team's flexibility and adaptability made it so that the work that needed to be done was ultimately finished in the end despite our lack of communication and organisation at the beginning. However, had we been more flexible at the beginning, we would have encountered less difficulties during this sprint.

Lessons Learnt

Positive

Through this mini-capstone, we learned the importance of teamwork in a group setting project. We started off working on our tasks individually with little communication and teamwork. This resulted in a lot of conflicts when it came to merging our code branches. There were conflicts in our frontend designs as well for which we had to fix later on in the project. We were able to recuperate through better teamwork as we learned to work together more and share responsibilities instead of individually working on our tasks.

Additionally, we were able to learn new technologies for the sake of this project. For example, we had to look into cloud deployment methods and services. Some of us also had to learn Next.js for this project. Therefore, there were definitely some useful takeaways that came as an outcome of this mini-capstone.

Negative

On the other hand, some negative learning outcomes would be, as mentioned before, poor and unclear understanding of the requirements from the beginning, time management and lack of communication. These were all things that we needed to work on and that we still need to improve on.

Our misunderstanding or lack of in-depth understanding of the requirements led to wasted time fixing our errors, had we just been more careful and paid more attention from the start. It also led to our grades suffering. Our bad time management caused us to always rush at the end of the sprint to get things done which then made the quality of our work suffer. Finally, our lack of communication caused frustration within the team. It was also a factor that influenced our bad time management.

All this to say that, through this project, we were able to learn the importance of these factors in a group project as they can lead to poor productivity and poor quality of the product.

Conclusion

While reflecting on the project, we realised that we encountered multiple challenges that originated from our lack of communication, time management and poor understanding of the requirements. However, we tackled these problems by assigning subgroups of members to certain tasks to ensure successful deployment. Although our teamwork is not perfect, it has significantly improved by trying to communicate more often to the best of our abilities. Things worked out better in the end compared to previous sprints, since we established earlier on what our vision was for what was to be completed for the end of the project. We got to refine the rough edges and we were able to bond a bit more which improved the team dynamic.

Looking back at the various sprints and resulting product, we have learned many lessons. Notably, the importance of communication, organisation, and teamwork are crucial when it comes to software development, in a team-based environment. In regards to the organisation of the project, the importance stems from the need to prioritise tasks and be realistic about what can be accomplished by implementing effective communication to ensure smooth delivery and deployment. By applying the lessons we learned, we will tackle future challenges by being more informed and committed to developing the best product possible. We will also be able to carry on these newfound skills in future projects.

10. CODE MANAGEMENT

SOEN 390 – Sprint 5

Code Management Documentation

1. Quality of source code reviews

The quality of source code review is a very important process in web development. The source code reviews were done for every pull request. At least one reviewer is required to review the changes and inspect code quality such as respecting the coding conventions.

Here is an example of a pull request with source code reviews: https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/pull/206

The reviewer leaves comments on sections of the code that require improvement. The creator of the pull request takes into consideration the comments and refactors the code. The pull request is only merged when the reviewer approves the pull request. For a major pull request, such as code refactoring, the team has a meeting to review all changes.

Figure 1. Source code reviews and comments and the pull request author's refactorings.

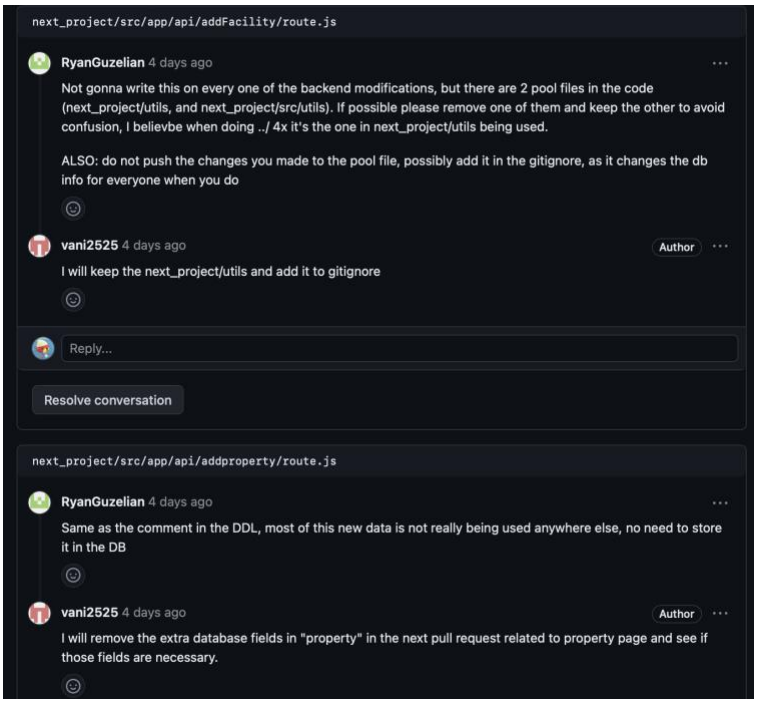
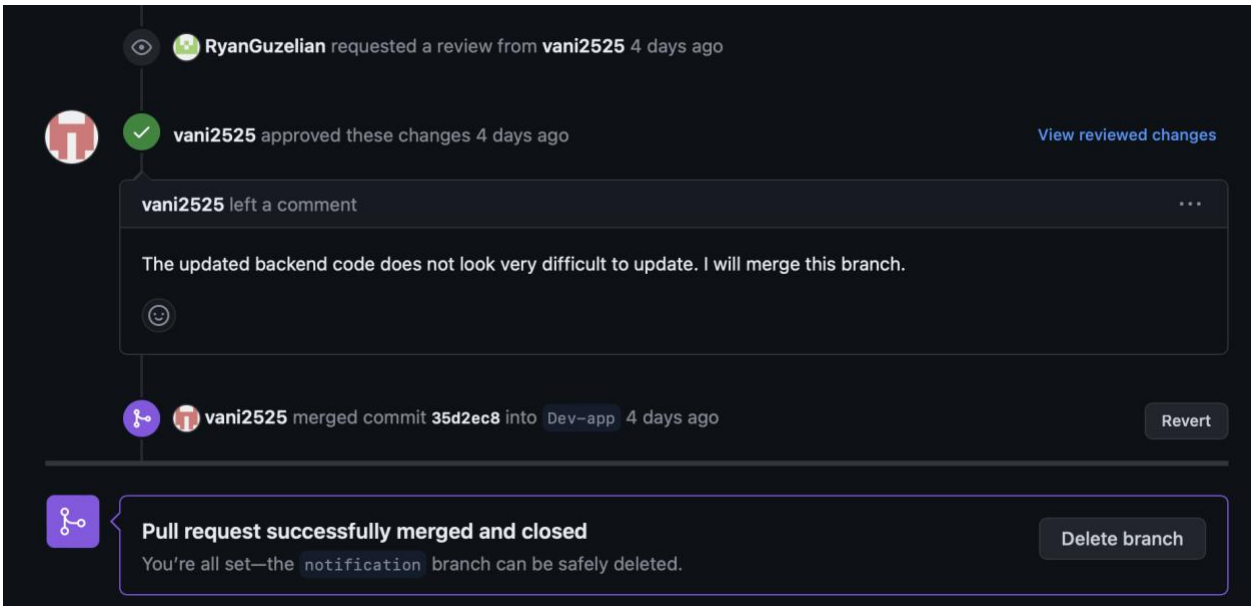


Figure 2. Reviewer's approval of the pull request



2. Correct use of design patterns

In order to follow proper software engineering techniques, and considering our tech stack, we decided to follow the MVC (Model-View-Controller) pattern. This would solidify our use of TypeScript, since under the hood, it relies on JavaScript's prototype-based inheritance, which slightly strays from known OOP concepts.

Model:

The model is represented by our database structure, which uses PostgreSQL to define data models and perform CRUD operations.

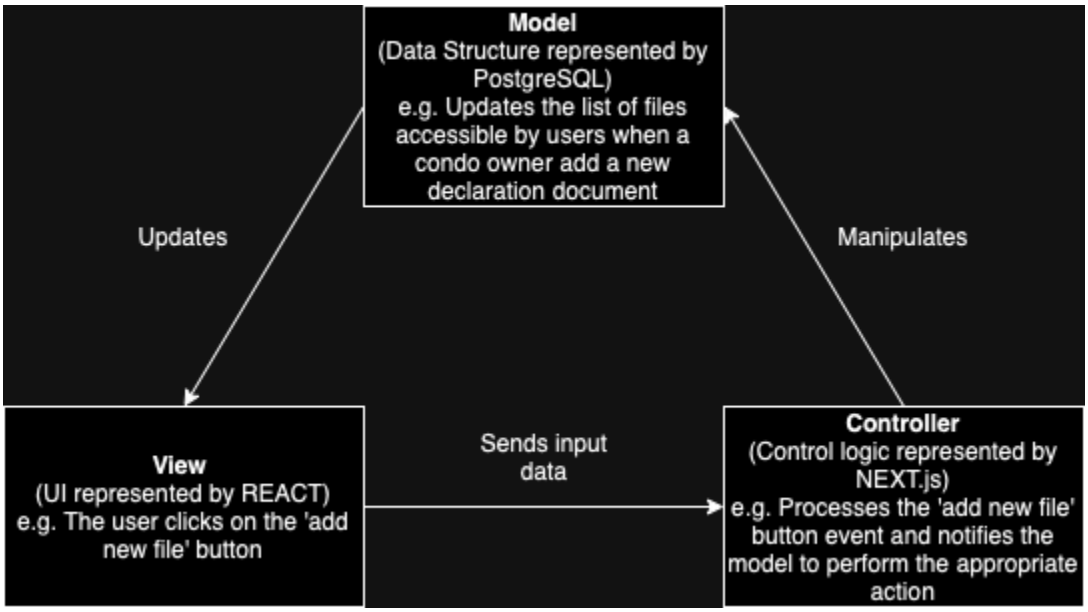
View:

The view is represented by our React components, which present data to the user and handle the interactions with them.

Controller:

The controller is represented by the API routes created using Next.JS, which allow interaction between the application and server-side functions.

Figure 3. Model-View-Controller



3. Respect to code conventions

Coding conventions are programming language-specific guidelines that offer recommendations for keeping your code clean. Code conventions promote code consistency. This is especially important when multiple developers are working on the same project. Thanks to code conventions, different teams can avoid unnecessary confusion and reduce the likelihood of errors caused by inconsistencies in coding styles. Moreover, appropriate conventions and practices ensure that any developers who will work on the application in the future can easily understand the code.

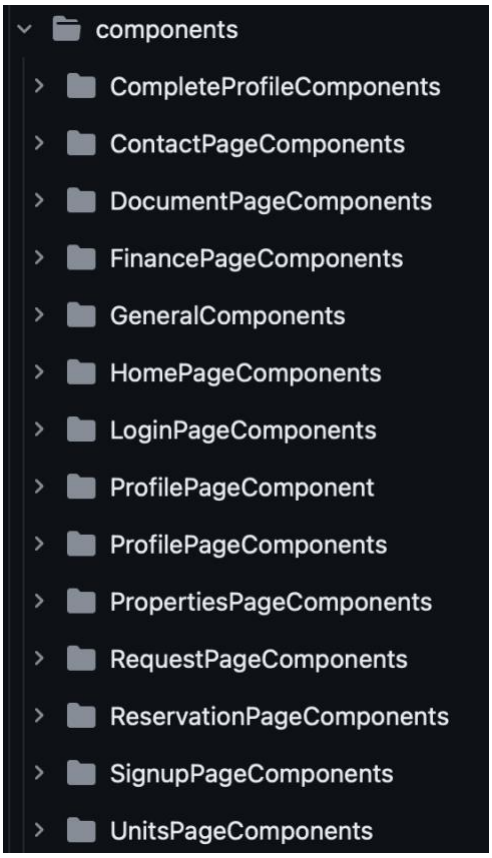
Guideline followed examples:

- Avoid one-letter names
- Nested code should be indented
- Avoid code duplication
- Avoid multiple inheritance
- Lowercase letters represent variables for which you must substitute information or specific values.

Figure 4. Example of code convention: Descriptive lowercase variable names with comments

```
//-------------------------HARDCODED VALUES--------------------------------------
const userId = 1; // company user
const accountType = "company";
//const userId = 3;
//const accountType = "reg_user";
//-------------------------------------------------------------------------------
```

Figure 5. Separate components files for code reusability



4. Quality of source code documentation

Good source code documentation is essential for ensuring that the code written has logic, intent and function. Proper documentation reduces the risk of misinterpretation and costly errors. In addition, it promotes productive and efficient collaboration between different teams when bug fixing, code reviewing, and other maintenance processes.

Guideline followed:

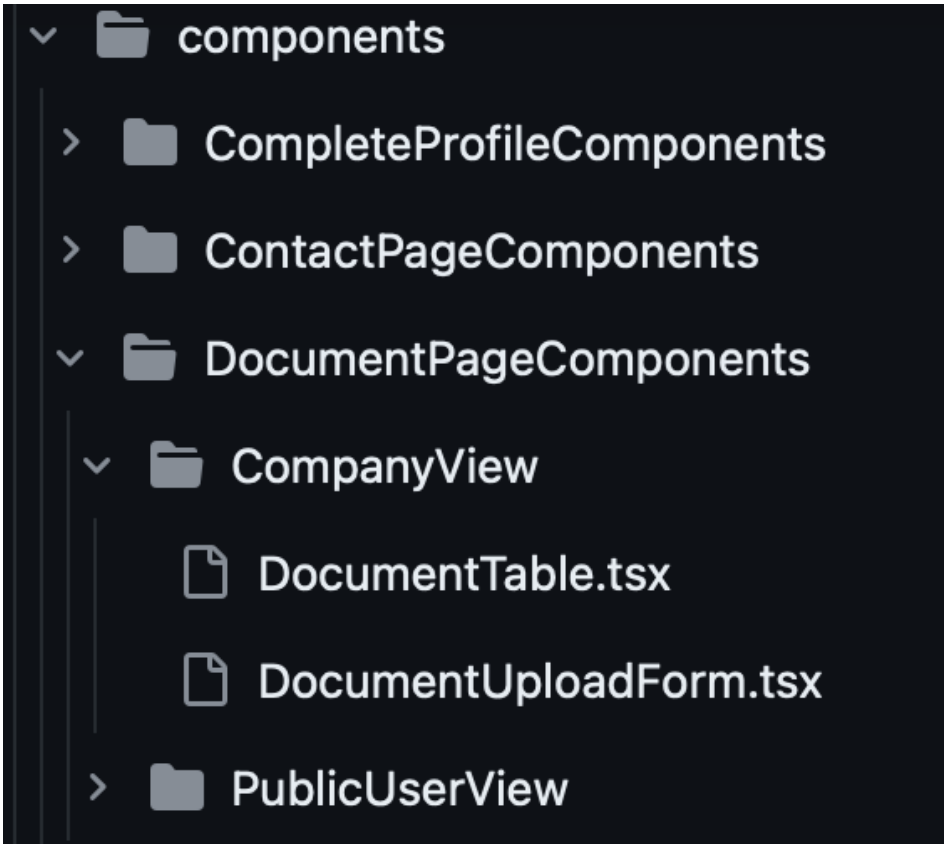
- Clearly state the intent and goal of the code, along with any unique functions, classes, or modules.
  - Use easy-to-understand language so that both veterans and beginners can understand it
- Use the proper headers, subheadings, and sections to arrange your material.
- Provide references
- Comments

Figure 6. Example of comments

```
  // page in progress
  //-----------------------------------FIX PAGE RENDERING HERE ----------------------------------------------------------
const page = 'company';
//const userId = 3 // reg_user
const userId = 1; // company user

  //-------------------------------------------------------------------------------------------------------------------
```

Figure 7. Example of clear files names and category names that convey the goal of the code

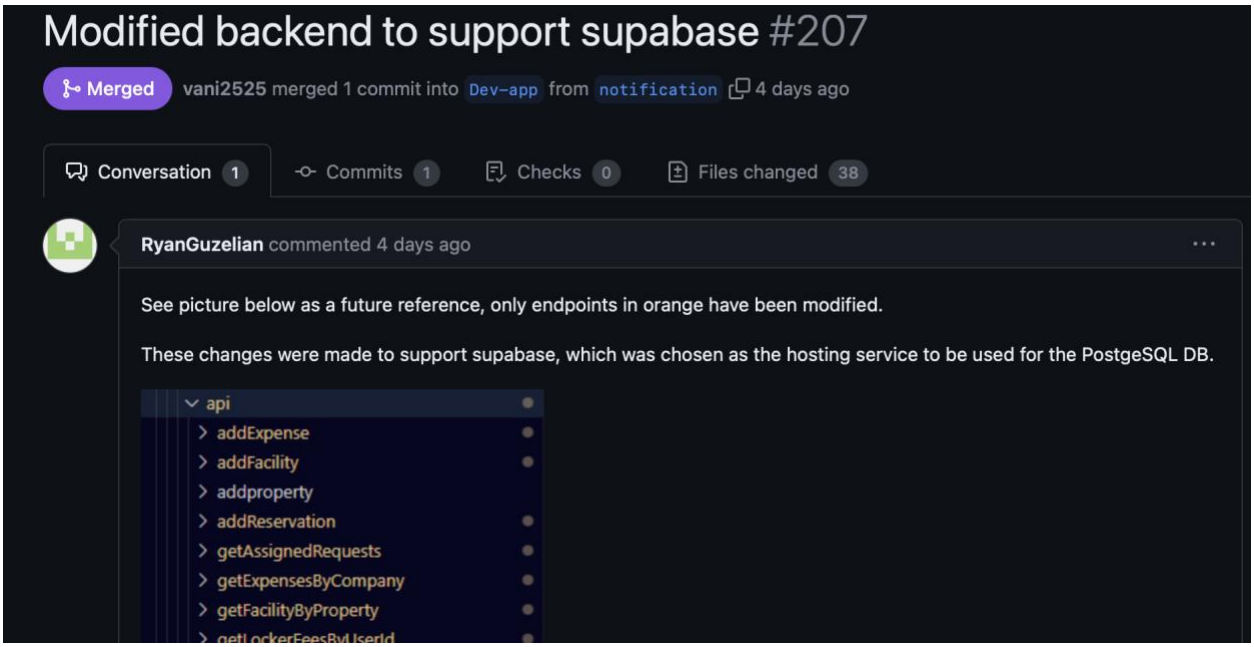


5. Refactoring activity documented in commit messages

The team undertook significant refactoring activities to enhance the project's codebase. Key efforts included file restructuring, standardization of front-end components code, replacing redundant codes with reusable components. These changes were driven by focus on improving the system's maintainability and scalability, providing a clear and structured basis for future development.

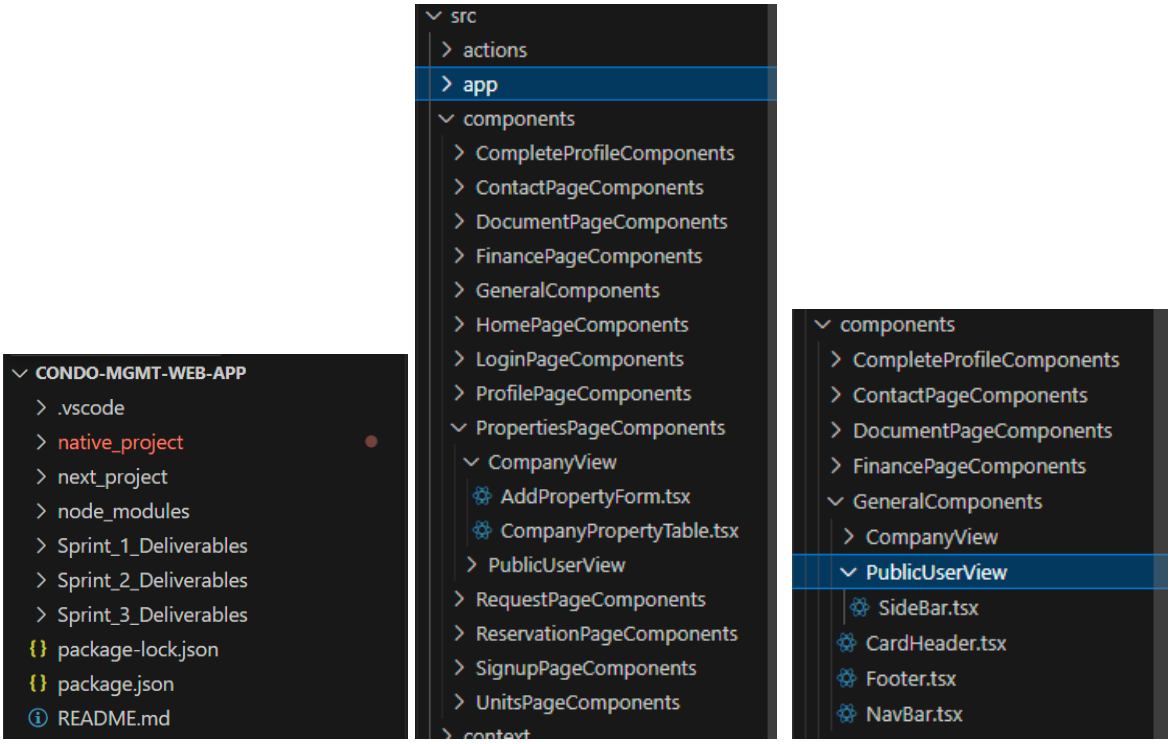
Furthermore, for sprint 4, code refactoring was performed in order to support Supabase as our hosting service for our PostgreSQL database.

Figure 8. Pull request opened to document and review the changes made



https://github.com/CONCORDIA-SOEN-390/Condo-Mgmt-Web-App/pull/207

Figure 9. Structure of the project



6. Quality/detail of commit messages

A good commit message provides an essential description of the changes introduced by the commit. Its goal is to explain the change, while providing enough context for others (including future you) to understand the code without having the need to inspect code.

Guideline followed:

- Add a significant title.
- Describe the commit.
- Do not includes personal information, such as passwords
- If a commit solves a bug or closes an issue, specify it.
- Avoid massive sizes. There is no limit to how much text may be stuffed into the commit body. However, a giant message would destroy performance.

Figure 10. Examples of short but descriptive commit titles

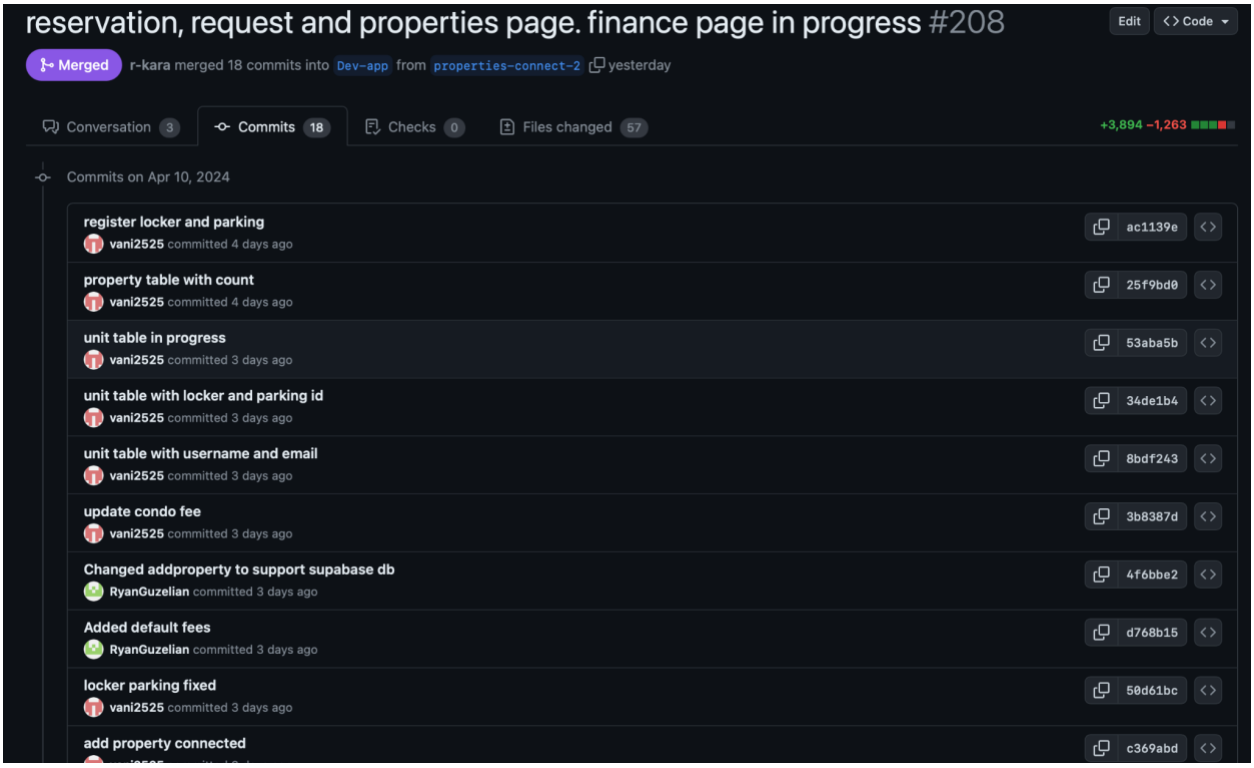
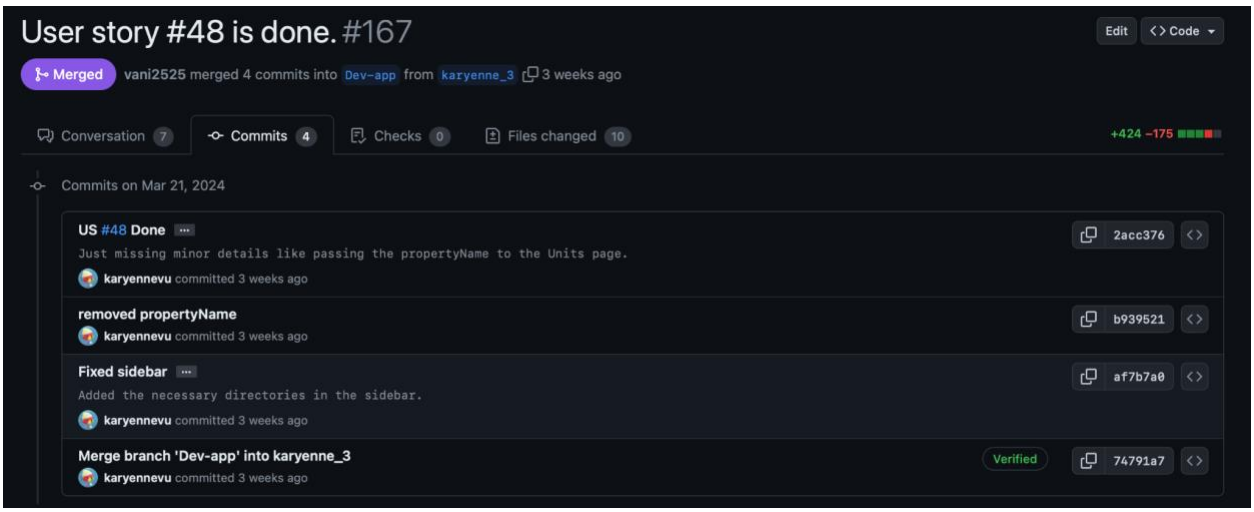


Figure 11. Example of commits with comments to provide further information



7. Use of feature branches

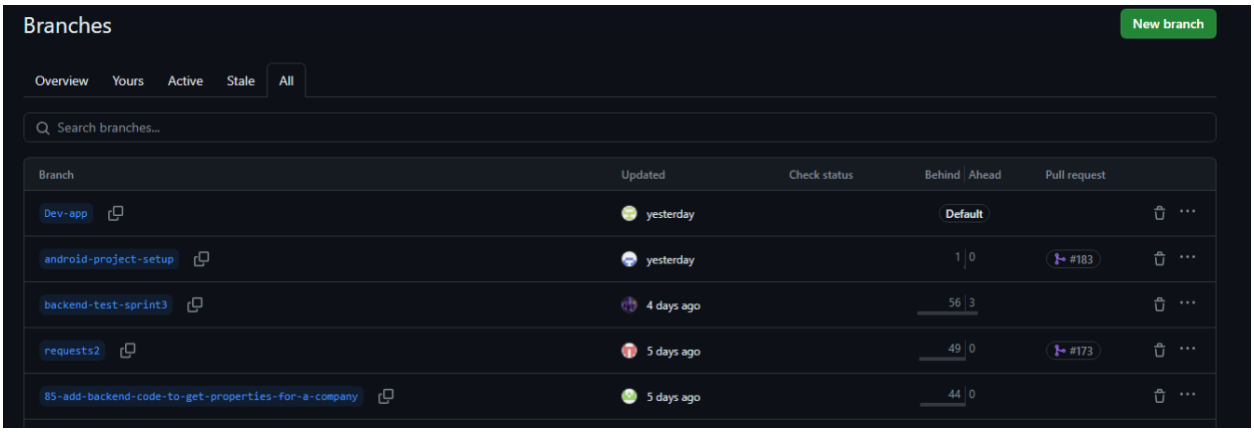
The use of feature branches is an essential example of good software development practice, especially when working in a group setting, to avoid negative results. It allows developers to work on new features of an application or a software without disturbing the progress already made. Without making feature branches, someone could easily delete the changes made and make the team lose all its progress which would consequently delay the delivery of the project.

Additionally, feature branches ensure that the new code has to be reviewed in a pull request first before merging it into the main branch to avoid conflict. The review is also usually done by someone else on the team which makes it so that the person merging their code needs the approval from at least one teammate before merging. This ensures that the person who worked on the feature branch can't just decide things on their own, and it also allows them to receive feedback on their work.

Guideline followed:

- Descriptive feature branch titles
- Existing pull requests for code review before merging
- Different branches for different features

Figure 12. Example of feature branches used for this project on GitHub



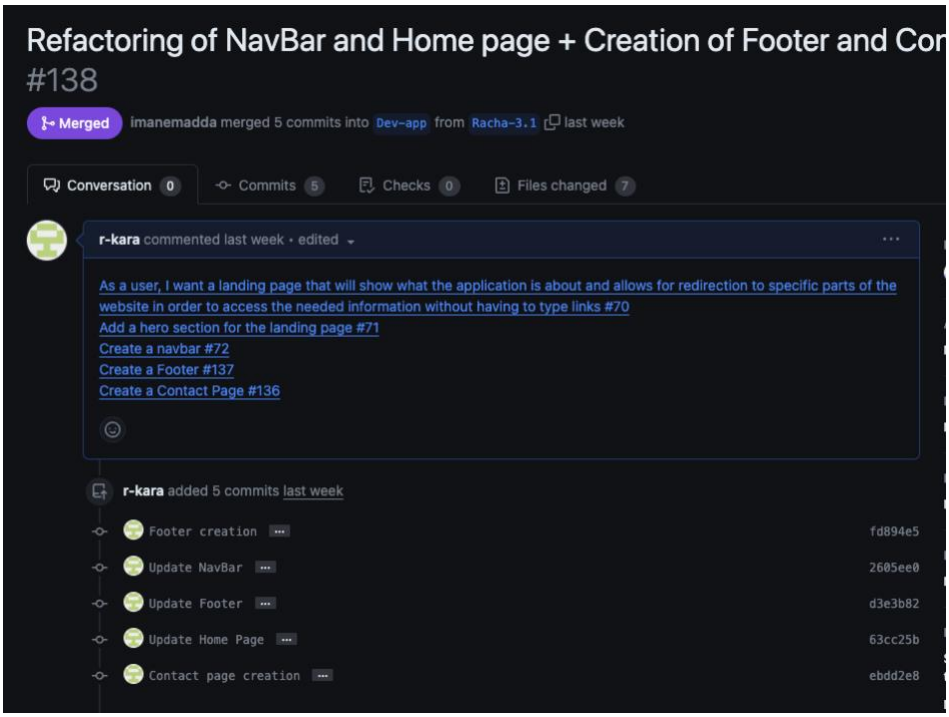
8. Atomic commits

Atomic commits are a widely used good practice among developers. Their main purpose is to ensure that changes are isolated from each other and ultimately reversible if needed. They also make the reviewing process easier since the changes are divided in logical blocks. Overall, a clear history of commits allows for better version control and task management. Although this practice may not seem useful at times, its necessity is highlighted when a bug is detected after merging commits, meaning that it can only be uncovered by reviewing recent changes.

Guideline followed:

- Address one specific task at the time
- Commit all changes made that are related to this task
- If needed, break down a large feature into smaller tasks (should already be done through the issues backlog)
- Avoid unrelated changes as much as possible
- Specify your changes in the commit message

Figure 13. Example of atomic commits that targeted separate tasks



9. Bug reporting

Although formal bug reporting is crucial to the documentation of a project, we initially followed an informal approach by reporting any bugs directly to the developer that worked on the feature, either through text message or in-person. However, we understood the importance of keeping a trace since a bug may emerge again and having a link to the commit that solved it can make us gain a significant amount of time. Therefore, we started reporting bugs, when discovered through testing, through issues directly on GitHub, allowing everyone on the team to be updated at once.

Guideline followed:

- Use a short, descriptive title.
- Add the “bug” tag to the issue.
- Specify the severity and priority.
- Describe the steps needed to reach the bug.
- If possible, add screenshots to the commit message.
- Assign the developer that worked on this feature.

Figure 14. Example of a bug report through a GitHub issue

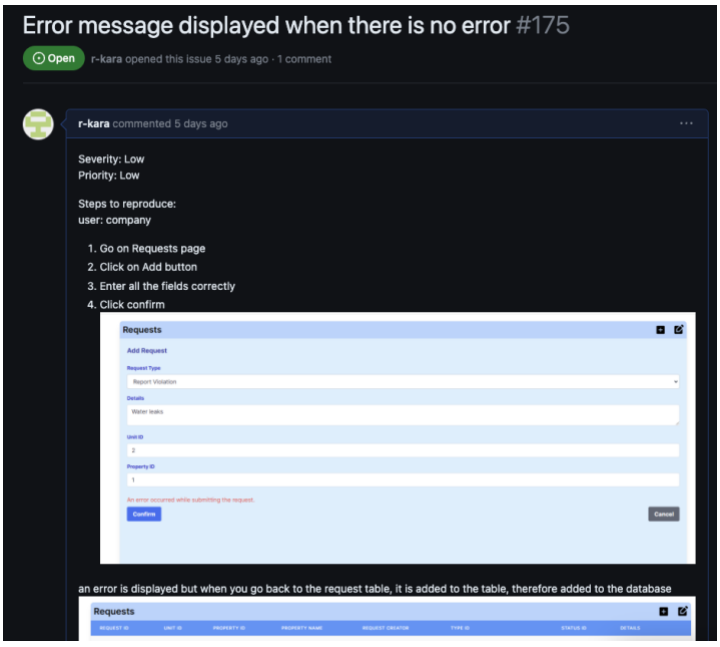


Figure 15. The use of the ‘bug’ label for issues pertaining application bugs on GitHub

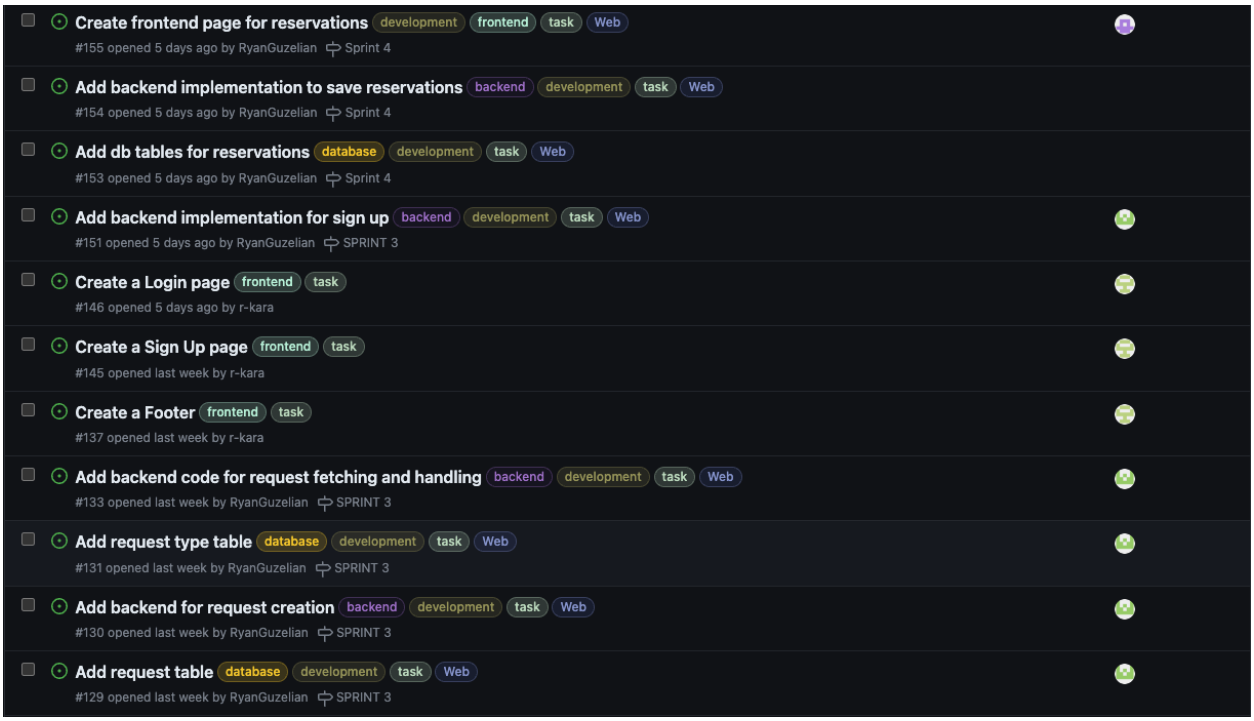
10. Use of issue labels for tracking and filtering

Issue labels are used to identify and organize the HitGub issue tracking system in order to increase understandability and efficient tracking. We defined the different labels at the beginning of the first sprint and consistently used them when defining user stories, tasks and bugs.

Types of Labels:

- Mobile, Web
- Backend, Frontend, Database, Testing
- User Story, Task, Tasks TBA, Bug
- Development, Documentation
- Extra (rarely used): High Priority, Enhancement

Figure 16. Small portion of the issues backlog



11. Links between commits and bug reports/features

Linking each commit to their corresponding bug reports or features may be one of the most essential practices to follow when contributing to a large project. First, it allows the developer to focus on a single issue at the time and make him accountable for this part of the project. It also enables traceability, allowing other developers to know what this commit is solving or contributing to when reviewing it. This way, collaboration within a team is more efficient and progress can be tracked accurately.

Guideline followed:

- Add a significant title.
- Include a link to all related issues in the commit message (user stories, tasks or bugs).
- Include the issue number.
- If a commit solves a bug or closes an issue, specify it.
- If possible, assign a reviewer that worked on these issues.

Figure 17. Example of links between commits and issues