

**Final Report**

SOEN 390

SoftCondo

By Team 10

Hisham Kitaz - 40188246

Paul Humennyj - 40209588

Jad Hanna - 40132590

Nour Hassoun - 40233077

Christa Abou Arraj - 40226631

Jack Spiratos - 40210312

Etienne Paquet - 40208596

Despina Koulisakis - 40190212

Jose Semaan - 40244141

Chris El-Kehdy - 40208593

Submitted To

Dr.

GINA CODY School of Engineering and Computer Science Department of Computer Science and Software Engineering

May 01, 2024

# Table of Content

[**Table of Content 2**](#_wn0f50o4yxns)

[**1. Product Vision 6**](#_kq19g4942ngq)

[● Introduction 6](#_pwkvxz5uspa)

[● Positioning 6](#_ypob38el7f0q)

[o Problem Statement 6](#_q2sbc6rvke37)

[o Product Position Statement 6](#_nrm86s8jfcf1)

[● Stakeholder and User Descriptions: 7](#_l66hbllx2v9n)

[o Stakeholder Summary 7](#_3pjeo3ox0jup)

[o User Summary 7](#_gjdgxs)

[o User Environment 8](#_k6d7uhdstzom)

[o Key Stakeholder or User needs 8](#_mlxcyfi450xq)

[o Alternatives and Competition 9](#_ed1mfknysywz)

[● Product Overview 9](#_6q2l6r7cxey7)

[● Product Features 9](#_mm8jqiy0djle)

[Public Profile Creation 9](#_oopbuuur3f28)

[Dashboard View 9](#_6cyat7vtaahq)

[Property Profile Creation 10](#_ssplo45syusa)

[Financial System 10](#_m7zuetuwn7yn)

[Reservation System 10](#_ucefknh1ru3s)

[Roles Set Up for Employees 11](#_7o13s44bknpo)

[Submit Request 11](#_97qw0papf03u)

[Notification Page 11](#_iuxcvvsoixy6)

[Multiple Platforms 11](#_fl3nvwvw0akr)

[Multi-Language 11](#_zf7zbtgsxdpd)

[Quick Sign In 11](#_7idirno4vwo2)

[● Other Product Requirements 11](#_8nltwmirl893)

[**1. Requirements and User Stories Backlog 12**](#_7i8ua1xuedkb)

[Feature 1: Public user profile creation 12](#_jmzc99lhc0mh)

[Feature 2: Property profile creation 13](#_u87tn632kmy)

[Feature 3: Condo owner dashboard 15](#_bbveedkyis49)

[Feature 4: Financial system 16](#_5rmi56htzzjz)

[Feature 5: Reservation System 17](#_6webbh3xickw)

[Feature 6: Roles 18](#_vwe7q5rhuzda)

[Feature 7: Request System 19](#_jvl81lo02ijg)

[Feature 8: Notification page 19](#_ikynst93dt4i)

[Additional features: 20](#_h9cpk1fa8ye8)

[Summary Story Points (without additional features) 23](#_y2ynqmtb8ar4)

[**3. Software Architecture 24**](#_tgdb9ej25d4w)

[1 - Introduction 24](#_g9hwoxvs6xgz)

[1.1 - Identifying information 24](#_17dp8vu)

[1.2 - Rationale for key decisions 24](#_3rdcrjn)

[2 - Stakeholders and Concerns 25](#_ru18m8ykc3z)

[3 - Views 27](#_3ba1htmgdh3x)

[3.1 - Scenarios 27](#_35nkun2)

[3.1.1 - Use Case Diagram for “Public User” and “Condo Owner” 28](#_joqa3q3tsqs4)

[3.1.2 - Use Case Diagram for “Condo Management Company” 29](#_325dgrr597u7)

[3.2 - Logical View 29](#_z337ya)

[3.2.1 - Domain Model 29](#_78smv4jb60u2)

[3.3 - Process View 31](#_vri6o2j3p36w)

[3.3.1 - Activity Diagrams 31](#_4i7ojhp)

[3.3.1.1 CMS-47 - CMS-49 31](#_bewuz1df82we)

[3.3.1.2 CMS-110 - CMS-48 - CMS-52 - CMS-52 32](#_sz8hml8dfmm3)

[3.3.1.3 CMS-78 - CMS-79 33](#_8l0zoqctyd82)

[3.3.1.4 CMS-51 - CMS-80 34](#_lz0xijj0lfjk)

[3.3.1.5 CMS-81 35](#_utwp0980q0jy)

[3.3.1.6 CMS-82 - CMS-83 - CMS-84 - CMS-85 36](#_o1mg0oivoi6g)

[3.3.1.7 CMS-86 - CMS-87 - CMS-103 37](#_mnq50l80sumo)

[3.4 - Development View 38](#_crtu79du4k0b)

[3.4.1 - Class Diagram for User Management 38](#_3whwml4)

[3.4.2 - Component Diagram 39](#_2bn6wsx)

[3.5 - Deployment View 40](#_qsh70q)

[3.5.1 - Deployment Diagram 40](#_3as4poj)

[3.6 - Known Issues with Views 40](#_1pxezwc)

[4 - Architectural Decisions 41](#_5cn48plghtu9)

[4.1 - Front-end Technologies 41](#_smoyo4k0yrkx)

[4.1.1 - React 41](#_3cofeuqukshh)

[4.1.2 - NodeJS 41](#_39h34voisu4z)

[4.2 - Back-end Technologies 41](#_bbrejsi40wqz)

[4.2.1 - C# 42](#_147n2zr)

[4.2.2 - .NET 42](#_3o7alnk)

[4.2.3 - Entity Framework Core 42](#_23ckvvd)

[4.2.4 - Microsoft SQL Server 42](#_ihv636)

[**4. Risk Assessment and Risk Management 43**](#_z34gcikp6qmh)

[1. Introduction 43](#_xlb3ba8032ji)

[2. Advantages of the risk assessment and management plan 43](#_ske0hhg2hgn4)

[3. How Risks and Assessment identified among our team 43](#_k783fdcdvihd)

[- Measurement information for the Risks 43](#_5ezrscn20lbf)

[4. Tables 44](#_a1mzxves77vd)

[4.1. Risk management chart 44](#_2nownvkyzp9w)

[4.2. List of identified risks 44](#_y0xj0ikzl996)

[**7 Retrospectives 53**](#_em4n5xhqkfqt)

[7.1 Short Sprint #1 Retrospective 53](#_1ssqv01jbxoa)

[7.2 Short Sprint #2 Retrospective 58](#_ub4a309fbpdu)

[7.3 Short Sprint #3 Retrospective 63](#_9eaiwyt0nd1r)

[7.4 Short Sprint #4 Retrospective 69](#_qk9fuy1dq22c)

[**8 Release Plans 74**](#_p7ktuvo9z7j6)

[8.1 Sprint #2 Release Plan 74](#_c75p72hbm0pa)

[8.2 Sprint #3 Release Plan 76](#_96gpoj5usltx)

[8.3 Sprint #4 Release Plan 78](#_g4t42xhapu6z)

[8.4 Sprint #5 Release Plan 81](#_7fywjfyifalu)

[**9 UI Prototypes 84**](#_200vmx1ctgx2)

[9.1 Sprint #2 UI Prototype 84](#_sdb354k4d4nt)

[9.2 Sprint #3 UI Prototype 91](#_2e4583cjvukh)

[9.3 Sprint #4 UI Prototype 96](#_ugbzl4kvbxwg)

[9.4 Sprint #5 UI Prototype 102](#_dapx16a9k246)

[**10 Testing Plans 107**](#_4y8k3josqobx)

[10.1 Testing Plan for Sprint #2 107](#_45cy19j34owr)

[10.2 Testing Plan for Sprint #3 110](#_1tm2eidxe0lj)

[10.3 Testing Plan for Sprint #4 114](#_ga7o6mw8i5g7)

[10.4 Testing Plan for Sprint #5 119](#_49bnoik3xbz4)

[**11- Code Management 125**](#_4owowkfjz99t)

[11.1 Software Implementation of planned user stories 125](#_v280jup7olns)

[11.2 Bug reports/fixing 128](#_2et92p0)

[11.3 Bug reports fixing techniques 129](#_tyjcwt)

[11.4 Defect Tracking Tool for Software Testing 129](#_3dy6vkm)

[11.5 Code Review Tools 130](#_at7ua0d9vj7a)

[11.6 Correct use of design patterns 131](#_m3s2xgwdimnt)

[11.6.1 Modified Design Patterns 131](#_tg8eutflr9ey)

[11.6.2 Backend Architecture 132](#_766hqsmxd2ff)

[11.6.3 Design patterns 132](#_6qm6gddyigc8)

[11.7 Code coverage 135](#_gn8wv7bwwzbb)

[11.8 Respect to code conventions 136](#_4htlimfnqeaw)

[11.9 Design quality (number of classes/packages, size, coupling, cohesion) 137](#_d3i1kkoh0ah8)

[11.10 Quality of source code documentation 137](#_26in1rg)

[11.10.1 Importance of Quality Source Code Documentation 137](#_i36rvuwd9e3c)

[11.10.2 Practices we used for Source Code Documentation on GitHub 137](#_yfcicqvvx96s)

[11.11 Refactoring activity documented in commit messages 139](#_lnxbz9)

[11.12 Quality/detail of commit messages 140](#_qku7725u1emp)

[11.13 Use of feature branches 140](#_qsb7mkjk8z1t)

[11.14 Atomic commits 141](#_1ksv4uv)

[11.15 Linking of commits to bug reports/features 142](#_2r0kqe6bii1a)

[11.16 Coding guides 143](#_44sinio)

# 

# 1. Product Vision

### **Introduction**

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

### **Positioning**

#### Problem Statement

| The problem of | Inefficient and manual condominium management processes |
| --- | --- |
| affects | Condominium owners, rental users, condo management companies |
| the impact of which is | Increased complexity, time consumption, errors in managing tasks |
| a successful solution would be | Implementation of a condo management application that provides key benefits, including:  -Simplified financial system  -Easy registration process (registration key system)  -Detailed and transparent property information  -User-friendly room reservation process |

#### Product Position Statement

| For | Condo owners, rental users, condo management companies |
| --- | --- |
| Who | Need a hassle-free way to manage condo information |
| The condo management system | is a web and app |
| That | Offers a comprehensive view, simplified financials, and other interesting features |
| Unlike | Traditional property and rental management systems |
| Our product | Offers user-friendly, all-in-one condo management, with enhanced community engagement |

### **Stakeholder and User Descriptions:**

#### Stakeholder Summary

| **Name** | **Description** | **Responsibilities** |
| --- | --- | --- |
| Service provider (maintenance and support team) | Ensures continued functionality, security, and usability of the system after deployment | * Provide updates and improvements * Resolve bugs and system issues * Ensures data integrity and system security * Offer technical support to users and administrators |
| Condo User | Main users of the application, may report bugs and suggest new features | * Report bugs * Suggest new features * Give feedback |
| Financial (Investors) | They assess the viability and potential return on investment | * Fund the project or approve the funding * Evaluate market demand for the project’s features * Monitor the project’s progress and potential risks |

#### User Summary

| **Name** | **Description** | **Responsibilities** | **Stakeholder** |
| --- | --- | --- | --- |
| Condo Owners | Individuals who own condominium units in the managed properties | Stay updated with financial status, submitted requests, etc. | Self-represented |
| Rental users | Individuals who are renting condominium units in the managed properties | Stay updated with financial status, submitted requests, etc. | Self-represented |
| Condo Management Companies | Responsible for managing and overseeing condominium properties | Organizing and managing condominium properties | Self-represented |
| Public Users | New users who are about to move into their condominium | * Create and manage profile * Provide a registration key from the condo management company to become condo owners or rental users | Condo Management Companies |

#### User Environment

* Condo Owners and Rental Users are expected to regularly check the system for any update concerning any request they have submitted, any updates concerning their properties, make payments, and update their profile when needed,...
* For condo management companies, multiple employees with different roles (finance, advertising, marketing, …) manage properties and handle user requests.

#### Key Stakeholder or User needs

1- User Profile creation: The problem is that public users need a registration key to access the condo, and become renters or owners. This might be an inconvenience as it causes friction for users during the onboarding process. The current solution is to obtain the keys from the condo management company. The proposed solution is to make user registration easier and user-friendly, by sending condo unit registration keys to the users.

2- Property profile creation: It might be cumbersome and time-consuming for condo management companies due to the current manual data entry. The proposed solution is to implement a smooth process for creating and managing the properties.

3- Reservation Systems: The manual way is not efficient. It is difficult to track availability and to perform bookings with this current solution. The proposed solution is to implement a more intuitive real-time availability calendar accessible within the application.

4- Financial Systems: The problem with this current solution is with the fee calculation and cost recording. The proposed solution would be to provide configurable options for the condo fee calculations and to track the remaining costs for the condo owners, as well as the rent for the rental users.

5-Condo owner dashboard: The problem is that the owners need a clear view of their properties and what’s included. Without it, there would be a lack of transparency which may lead to dissatisfaction. The proposed solution would be to enhance the dashboard to include as many details as possible such as financial status, request tracking, …

#### Alternatives and Competition

1- Using an existing condo management software: Buildium, AppFolio

→Strength: quick, and reducing time-to-market, possible costs

→ Weaknesses: Limited customization, challenges in integrating with the existing system, …

2- Using social media messaging apps such as WhatsApp:

→strengths: fast communication with condo management companies

→weaknesses: hard to keep track of submitted requests and finances, and any updates concerning the condo

3- Exploring Open Source Solutions such as OpenMAINT:

→ strengths: lower cost since there are probably no licensing fees

→weaknesses: Concerns in quality and security, may not be customizable

### **Product Overview**

Our product stands as an independent, self-contained solution, requiring no external dependencies or interfaces with other systems.

### **Product Features**

#### Public Profile Creation

Public users shall create their profile with the following information needed: profile picture, user name, contact email, and phone number. In addition, public users shall provide a registration key obtained from their condo management company to become condo owners or rental users in the system.

#### Dashboard View

Condo owners have a good view of their properties, including general information, personal information, condo information, financial status, status of the submitted request, etc.

#### Property Profile Creation

Condo management companies can create a profile for a property under their management. The property profile shall have at least the property name, unit count, parking count, locker count, and address.

Condo management companies can upload files for each property, which is accessible to all condo owners of that property.

Condo management companies can enter detailed information for each condo unit, each parking spot, and each locker in a building. Basic information about a condo unit includes unit ID, size, unit owner, occupant information (i.e., may be occupied by a rental user instead of the owner), as well condo fee associated with the unit. Basic information about a parking spot (or a locker) includes parking spot ID, spot owner, occupant information, and condo fee associated with the parking spot.

Condo management companies can send registration keys to unit owners or rental users for their dedicated units. Such registration keys will be used by unit owners or rental users to link a condo unit with their profiles.

#### Financial System

Condo management companies can enter condo fees per square foot, per parking spot, which will be calculated and presented for each unit owner.

The financial system records the operational budget (i.e., the collected condo fee) and cost. Condo management companies can enter the cost for each operation.

The financial system can generate an annual report. For example, all the condo fees are collected for a given year.

#### Reservation System

Condo management companies set up the common facilities, which require reservations. Examples include a sky lounge and a spa fitness.

The reservation system allows condo owners and rental users to reserve common facilities in a calendar-like interface.

The reservation system should show the availability of common facilities.

The reservation is first-come-first-serve. Once a facility is booked, it will become unavailable for the reserved time.

#### Roles Set Up for Employees

Condo management companies can set up different roles for different employees, who are responsible for the same property. Such roles include manager, someone who is responsible for daily operations, or someone responsible for finance.

#### Submit Request

Condo owners can submit requests which will be assigned to a corresponding employee based on the type of request.

#### Notification Page

All users have a notification page to see the latest activities in submitted or assigned requests.

#### Multiple Platforms

The app is accessible on Android, iOS, Linux, MacOS, and Windows.

#### Multi-Language

The app is available in English and French

#### Quick Sign In

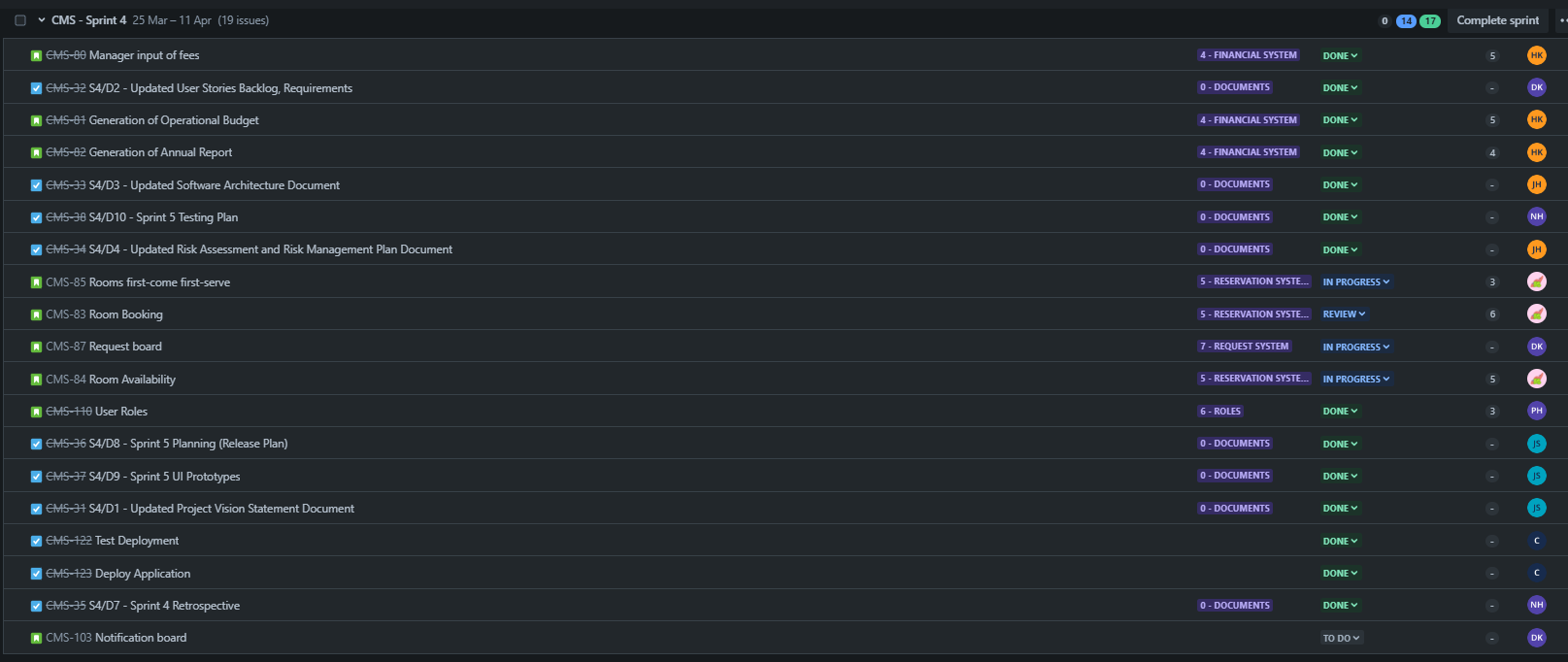
The app shall allow users to log in using their Gmail account, Facebook account, or Apple.

### **Other Product Requirements**

* + **Requirements in order of highest priority to lowest**
* The response time for operations should be relatively fast (fast loading times under typical workload conditions).
* The application should have the ability to continue functioning during errors.
* The user interface should be comprehensive and intuitive for users.
* The application should run on various platforms (e.g. Windows, Android …).
* Comprehensive user manuals/documentation should be available for users).

# 1. Requirements and User Stories Backlog

ScreenShot of Total Story Point and Completed Story Point for this Sprint



### Feature 1: Public user profile creation

| [CMS-47](https://soen390-cms.atlassian.net/browse/CMS-47?atlOrigin=eyJpIjoiODBhNTE0ODAxOWUzNDU5OWFiZGIxN2I5OGMwYjA3OTEiLCJwIjoiaiJ9): ~~User Profile Creation~~  (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to create my unique profile with a profile picture, user name, contact email, and phone number, so I can customise and enhance my overall experience within the system. | | | | |
| Feature 1: Public user profile creation | | | | |
| MoSCoW  Must have: create unique profile, profile picture, user name, contact email, phone number  Should have: customization options for profile (bio…)  Could have: option to add additional contact methods (social media handles…)  Won’t have: public display of phone numbers for regular users | | | | |
| Bus. Value: L | | | Risk: M | Effort: 5 |

| [CMS-49](https://soen390-cms.atlassian.net/browse/CMS-49?atlOrigin=eyJpIjoiY2UyZDk5YzU4ZjdmNDhiNGJjZTg1NjBhMThhOTJjOTYiLCJwIjoiaiJ9): ~~Assignment of Registration Keys~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to be recognized as a condo owner/renter by providing a registration key obtained from my condo management company, so I can get access to the system’s corresponding condo owner/renter features and link my condo unit on my profile. | | | | |
| Feature 1: Public user profile creation | | | | |
| MoSCoW  Must have: ability to input registration key, access to corresponding features  Should have: feedback on key verification status  Could have: option to update linked condo unit  Won’t have: free access to all condo management features | | | | |
| Bus. Value: L | | | Risk: M | Effort: 2 |

| [CMS-110:](https://soen390-cms.atlassian.net/browse/CMS-110?atlOrigin=eyJpIjoiOGI4M2Y0NDk2ODZkNDkxZmFjNzk4OGEwYzQ0ZTE2N2IiLCJwIjoiaiJ9) ~~User Roles~~ (completed) | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to only have access to the services relevant to my login user type (company, property owner, rentor, unit owner), so when I’m logged in, I can only use the features that are useful to me. | | | | |
| Feature 1: Public user profile creation | | | | |
| MoSCoW  Must have: ability to hide the feature that are not relevant to the corresponding user type  Should have: visual display of the user’s profile type (on the main page) when login is successful  Could have: user profile customization options that allow users to add or remove details based on their preferences and requirements  Won’t have: customization options that allow users to access features not intended for their user type. | | | | |
| Bus. Value: M | | | Risk: L | Effort: 3 |

### Feature 2: Property profile creation

| [CMS-48](https://soen390-cms.atlassian.net/browse/CMS-48?atlOrigin=eyJpIjoiMjUzYzI0OTAzMmU4NGRlMGJmNmRiODc5MmJlZDAzZjYiLCJwIjoiaiJ9): ~~Property Profile Creation~~  (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to create profiles for my properties, so I can efficiently manage all my profiles, provide accurate representations for each property, and ensure clear communication between rental users, condo owners, and myself, regarding my properties. | | | | |
| Feature 2: Property profile creation | | | | |
| MoSCoW  Must have: mandatory fields such as property name, company name, address, city  Should have: load pictures of property  Could have: analytics to track engagement per property profile  Won’t have: virtual tours of the property profile | | | | |
| Bus. Value: XL | | | Risk: M | Effort: 4 |

| [CMS-52](https://soen390-cms.atlassian.net/browse/CMS-52?atlOrigin=eyJpIjoiMjI2NjdhYTNkOGQ4NDUyZGI1NmUzYzViNjkxOTUwY2UiLCJwIjoiaiJ9): ~~Condo File Upload~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to upload condo files in the profiles of each one of my properties, so the files can be easily accessible to all condo owners and rental users, providing them with detailed representations of my properties. | | | | |
| Feature 2: Property profile creation | | | | |
| MoSCoW  Must have: functionality to upload condo files  Should have: user-friendly interface for uploading and viewing files  Could have: notification system to alert users when new files are uploaded  Won’t have: real time collaboration on documents | | | | |
| Bus. Value: L | | | Risk: M | Effort: 4 |

| [CMS-53](https://soen390-cms.atlassian.net/browse/CMS-53?atlOrigin=eyJpIjoiMWM3OWJmNjdmODE2NGVhOWI3NzgyYmIwZDE0NjgwZDMiLCJwIjoiaiJ9): ~~Condo Unit Management~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to upload detailed information for each condo unit, specifically unit id, size, unit owner, occupant information, and condo fee, so I can easily manage my property’s spaces, and users can view my property’s details. | | | | |
| Feature 2: Property profile creation | | | | |
| MoSCoW  Must have: ability to upload basic information for each condo unit (unit id, size, owner, occupant, condo fees)  Should have: ability to edit condo unit information after submission  Could have: advanced filtering settings to navigate through different condo units  Won’t have: a feature for occupants to directly edit their information (to ensure accuracy and security) | | | | |
| Bus. Value: L | | | Risk: M | Effort: 4 |

| [CMS-78](https://soen390-cms.atlassian.net/browse/CMS-78?atlOrigin=eyJpIjoiNmI1ZjU4ODBjMmE2NDM5Mzg0YmRmYjgzMjJjNzI0N2IiLCJwIjoiaiJ9): ~~Adding Parking Spots and Lockers to a Property Profile~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to upload detailed information for each parking spot, and each locker in my property’s profile, specifically parking spot id, spot owner, occupant information, and condo fee associated with the parking spot so I can easily manage my property’s spaces, and users can view my property’s facilities. | | | | |
| Feature 2: Property profile creation | | | | |
| MoSCoW  Must have: upload parking and locker information in the property profile  Should have: ability to view associated condo fee  Could have: advanced search filter to find specific parking spots or lockers  Won’t have: real time reservation system for parkings/lockers | | | | |
| Bus. Value: L | | | Risk: M | Effort: 3 |

| [CMS-79](https://soen390-cms.atlassian.net/browse/CMS-79?atlOrigin=eyJpIjoiZjc3N2U3MTlkYzFmNGEzMGIzNWEyODRkNDE0MTk3NGYiLCJwIjoiaiJ9): ~~Send Registration Keys from Condo Unit to Owner/Rentor~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to send registration keys to unit owners or rental users for their dedicated units, so they can link a condo to their profile and be recognized as either a condo owner or a rental user in the system. | | | | |
| Feature 2: Property profile creation | | | | |
| MoSCoW  Must have: ability for managers to send registration keys to unit owners or rental users  Should have: feedback on key status  Could have: batch processing for sending out registration keys (more efficient)  Won’t have: manual entry of unit owners or rental users (not as secure) | | | | |
| Bus. Value: M | | | Risk: M | Effort: 2 |

### Feature 3: Condo owner dashboard

| [CMS-51](https://soen390-cms.atlassian.net/browse/CMS-51?atlOrigin=eyJpIjoiMDg0MjdmOWFkNTkyNDZkZjg3YWE4YjBiNjQ2ODQyMmUiLCJwIjoiaiJ9): ~~Condo owner dashboard~~ (completed with acceptance test) | | | | |
| --- | --- | --- | --- | --- |
| As a condo owner, I want access to a dashboard of my properties which will include my profile, condo information, financial status, and status of submitted requests, so I can efficiently view, manage, and navigate all aspects of my condo ownership. | | | | |
| Feature 3: Condo owner dashboard | | | | |
| MoSCoW  Must have: access for condo owners to view profile, condo information, status of submitted requests  Should have: notification alerts for submitted requests  Could have: customizable dashboard views  Won’t have: real-time chat support | | | | |
| Bus. Value: L | | | Risk: M | Effort: 6 |

### 

### Feature 4: Financial system

| [CMS-80](https://soen390-cms.atlassian.net/browse/CMS-80?atlOrigin=eyJpIjoiNjlkZGVhODhkN2YxNGI4ZDk5MjEwY2IwNDczZmY4YmQiLCJwIjoiaiJ9): ~~Manager input of fees~~ (completed) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to enter the condo fee per square foot, and per parking spot for each unit in a property, so the financial system within the condo management system can generate the condo fee for each unit and present it to condo owners, making it easy for them to view their total condo fees. | | | | |
| Feature 4: Financial system | | | | |
| MoSCoW  Must have: form to enter condo fee per square food and per parking so the system can generate the condo fee  Should have: adjustment feature for rate changes  Could have: notifications to condo owners about fee updates  Won’t have: manual calculation or entry of total fees for each unit | | | | |
| Bus. Value: L | | | Risk: M | Effort: 5 |

| [CMS-81](https://soen390-cms.atlassian.net/browse/CMS-81?atlOrigin=eyJpIjoiMTVlYjdmNDNkMjJjNDI4M2IyZjczYmZkOGRkZDg0NGIiLCJwIjoiaiJ9): ~~Generation of Operational Budget~~ (completed) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want the financial system to generate an operational budget, so I can keep track of all fees and costs. | | | | |
| Feature 4: Financial system | | | | |
| MoSCoW  Must have: automated generation of operational budget with fees and costs  Should have: customizable budget  Could have: notifications being sent if costs go above the budget  Won’t have: reports for each condo unit | | | | |
| Bus. Value: XL | | | Risk: H | Effort: 5 |

| [CMS-82](https://soen390-cms.atlassian.net/browse/CMS-82?atlOrigin=eyJpIjoiZjE2YWVmNGVhYTA0NGZiODk0MjdiOTkwMWI4Yjc4YTkiLCJwIjoiaiJ9): ~~Generation of Annual Report~~ (completed) | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want the financial system to generate an annual report that describes all condo fees collected in a year, so I can easily view my yearly gross income. | | | | |
| Feature 4: Financial system | | | | |
| MoSCoW  Must have: annual report generation  Should have: customizable report parameters  Could have: additional security access  Won’t have: predictions for future income | | | | |
| Bus. Value: XL | | | Risk: H | Effort: 4 |

### 

### Feature 5: Reservation System

| [CMS-83](https://soen390-cms.atlassian.net/browse/CMS-83?atlOrigin=eyJpIjoiMTg5NDg4ODQ3ODYzNDA2Y2I1MzdiYmM4ODRmYzkxZWYiLCJwIjoiaiJ9): ~~Room Booking~~ (complete) | | | | |
| --- | --- | --- | --- | --- |
| As a user who wants to book a room, I would like to be able to reserve additional rooms provided by the condo management in a calendar-like interface so that I may have a better stay. | | | | |
| Feature 5: Reservation System | | | | |
| MoSCoW  Must have: calendar like interface for booking rooms with availabilities  Should have: ability to view information about each room  Could have: filtering settings to view certain rooms based on preferences  Won’t have: recommendation of rooms based on users past bookings | | | | |
| Bus. Value: L | | | Risk: M | Effort: 6 |

| [CMS-84](https://soen390-cms.atlassian.net/browse/CMS-84?atlOrigin=eyJpIjoiNTg2MzU5NWVmNmQ2NDA4MTkwMzdlNDlmMTVjMTNiN2EiLCJwIjoiaiJ9): Room Availability | | | | |
| --- | --- | --- | --- | --- |
| As a user who wants to book a room, I would like for the system to show the availability of common facilities, so that I may plan more services or receive better suggestions. | | | | |
| Feature 5: Reservation System | | | | |
| MoSCoW  Must have: ability to view the common facilities of each property and their availabilities  Should have: ability to reserve time slots for the facilities  Could have: notifications on facility availability changes  Won’t have: recommendation of facilities based on users past bookings | | | | |
| Bus. Value: M | | | Risk: M | Effort: 5 |

| [CMS-85](https://soen390-cms.atlassian.net/browse/CMS-85?atlOrigin=eyJpIjoiYjNmMTBkZGI0ZjRkNGFjNzg4NWE3YTU2NGY0MDhhYWQiLCJwIjoiaiJ9): Rooms first-come first-serve | | | | |
| --- | --- | --- | --- | --- |
| As a user who wants to book a room, I would like for the reservation service to be first come first served, so that I can be assured of my reservation. | | | | |
| Feature 5: Reservation System | | | | |
| MoSCoW  Must have: first-come first-serve mechanism  Should have: clear display of real time availability  Could have: waitlist feature  Won’t have: option to bid for rooms | | | | |
| Bus. Value: M | | | Risk: L | Effort: 3 |

# 

### Feature 6: Roles

| [CMS-86](https://soen390-cms.atlassian.net/browse/CMS-86?atlOrigin=eyJpIjoiYThhNWYzZTA0NDAwNGViN2IzMDI1ZGZhOTlhODY5ZDIiLCJwIjoiaiJ9) Assignment of Roles to Employee Users | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to be able to define different roles for employees, so that I can assign specific responsibilities related to property management. | | | | |
| Feature 6 - Roles | | | | |
| MoSCoW  Must have: ability to view and edit all employes and their roles  Should have: a save button to confirm changes  Could have: filter system by role  Won’t have: a revert feature to go back | | | | |
| Bus. Value: M | | | Risk: L | Effort: 4 |

### 

### Feature 7: Request System

| [CMS-87](https://soen390-cms.atlassian.net/browse/CMS-87?atlOrigin=eyJpIjoiNGY2ZjFiMmU3YTRiNGJhY2EyMjlmM2Y2MGIwNWRkZDEiLCJwIjoiaiJ9) Request board | | | | |
| --- | --- | --- | --- | --- |
| As a condo owner, I want to submit a request for moving in/out, intercom changes, reporting a violation, reporting a deficiency and/or asking any questions so that necessary action and assistance can be taken by the management. If applicable a corresponding employee needs to be assigned. | | | | |
| Feature 7 - Requests System | | | | |
| MoSCoW  Must have: View Model of requests for both owners and management  Should have: Buttons to add, remove, edit  Could have: employee assignation  Won’t have: ai support tools | | | | |
| Bus. Value: H | | | Risk: M | Effort: 6 |

### 

### Feature 8: Notification page

| [CMS-103](https://soen390-cms.atlassian.net/browse/CMS-103?atlOrigin=eyJpIjoiZGI0NDJjZDBhNTg5NDdhNzk0YzhlYzg5MmQ4Mzk5ZDYiLCJwIjoiaiJ9) Notification Board | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to have access to a notification page where I can see the latest activities submitted or assigned requests. | | | | |
| Feature 8 - Notification page | | | | |
| MoSCoW  Must have: tab that opens up to notifications view  Should have: it should be in form of a side bar  Could have: ability to remove notifications  Won’t have: ability to silence them | | | | |
| Bus. Value: M | | | Risk: L | Effort: 5 |

### 

### Additional features:

Additional features are optional and some might be discarded depending on further sprints

| [CMS-104](https://soen390-cms.atlassian.net/browse/CMS-104?atlOrigin=eyJpIjoiMDk5M2EwNTQ3ZmQ1NGZkYjkxZTExYzU3ZGNkMjg1YzMiLCJwIjoiaiJ9) Discussion Board | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to post and reply to new discussions, so that I can share my thoughts, questions, and experiences within the community. | | | | |
| Additional feature | | | | |
| MoSCoW  Must have: A place for discussion in which users can talk to management  Should have: The ability to reply to comments  Could have: The ability to remove comments  Won’t have: The ability to report comments | | | | |
| Bus. Value: L | | | Risk: L | Effort: 4 |

| [CMS-105](https://soen390-cms.atlassian.net/browse/CMS-105?atlOrigin=eyJpIjoiYTY1YmRkZDk3MTBjNGY3NWFlODZiYzkwODg4ZTNiOTAiLCJwIjoiaiJ9) Organizing and Participating in Community Events | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to organise events and invite other occupants to attend, as well as view events and accept/decline invitations so that I can facilitate community engagement and collaboration. | | | | |
| Additional feature | | | | |
| MoSCoW  Must have:A calendar or scheduling feature where users can create, view, and manage upcoming events.  Should have: Notifications for event invitations, reminders, and updates to ensure participants are informed.  Could have: Integration with external calendar services (e.g., Google Calendar, Outlook) for broader access and convenience  Won’t have: The ability to make events public to non-community members to maintain privacy and exclusivity. | | | | |
| Bus. Value: M | | | Risk: M | Effort: 3 |

| [CMS-106](https://soen390-cms.atlassian.net/browse/CMS-106?atlOrigin=eyJpIjoiODNlM2M2YWU5OTJmNDA3NmFlZDBkNmI3ODA0YTQ1YjUiLCJwIjoiaiJ9) Exclusive Offers Listing for Community Members | | | | |
| --- | --- | --- | --- | --- |
| As a manager in a condo management company, I want to list coupons/offers that are visible to unit owners or rental users for a specific property, so I can provide exclusive benefits to the community. | | | | |
| Additional feature | | | | |
| MoSCoW  Must have: A dedicated section on the community platform for posting and viewing exclusive offers and coupons.  Should have: The ability for users to save offers they are interested in and receive reminders before the offers expire.  Could have: Personalization of offers based on user preferences or past interactions to increase relevance and engagement.  Won’t have: The functionality for users to submit their own offers to prevent misuse and ensure offer quality. | | | | |
| Bus. Value: H | | | Risk: L | Effort: 2 |

| [CMS-107](https://soen390-cms.atlassian.net/browse/CMS-107?atlOrigin=eyJpIjoiMzk0OTQ0NmE0MDJkNDVmMThiM2I3NzU1N2FkNTViM2QiLCJwIjoiaiJ9) Cross-Platform System Accessibility | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want to be able to access the system on Android, iOS, Linux, MacOS, and Windows so that I can have the freedom to use whatever device is most convenient to me at the moment. | | | | |
| Additional features | | | | |
| MoSCoW  Must have:Responsive web design that ensures the system is accessible and usable across all mentioned platforms.  Should have: Native app versions for Android and iOS to provide an optimized experience on mobile devices.  Could have: Desktop applications for Linux, MacOS, and Windows for users who prefer or require offline access.  Won’t have: Custom-built web browsers or operating systems support outside of the major platforms listed to keep development and maintenance feasible. | | | | |
| Bus. Value: H | | | Risk: M | Effort: 5 |

| [CMS-108](https://soen390-cms.atlassian.net/browse/CMS-108?atlOrigin=eyJpIjoiZTZmNGM2NjU1NjJmNDBkOGJmZWZiYzJkMmM5ZDZhMDciLCJwIjoiaiJ9) Multilingual App Support | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want the option to use the app in English and at least one other language so that the app can be more accessible to everyone. | | | | |
| Additional features | | | | |
| MoSCoW  Must have: Support for both English and a second language that is widely used among the app's target audience, with the ability to switch languages within the app settings.  Should have: Language selection on first use or registration, allowing users to choose their preferred language from the start.  Could have: Automatic language detection based on the user's device settings to improve initial usability and convenience.  Won’t have: Support for every possible language, as this would be impractical and resource-intensive given the app's scope and target audience. | | | | |
| Bus. Value: H | | | Risk: L | Effort: 3 |

| [CMS-109](https://soen390-cms.atlassian.net/browse/CMS-109?atlOrigin=eyJpIjoiYmEyOWVmMzUyMDc1NGI4Y2I3NDI3Y2VlYjk1OTMyMGEiLCJwIjoiaiJ9) Seamless Email-Based Login Integration | | | | |
| --- | --- | --- | --- | --- |
| As a user, I want the option to log in with my Gmail account or other such email account, so that I can access the app seamlessly without creating a new set of credentials. | | | | |
| Additional feature | | | | |
| MoSCoW  Must have: Integration with Gmail for login, leveraging OAuth for secure authentication without sharing password details.  Should have: The ability to link multiple email accounts to a single app profile for users who want to use different email addresses.  Could have: Support for additional widely used email providers beyond Gmail, such as Outlook or Yahoo, to accommodate more users' preferences.  Won’t have: A requirement for users to provide additional unnecessary personal information during the login process to respect privacy and minimize data collection. | | | | |
| Bus. Value: H | | | Risk: M | Effort: 3 |

## 

### Summary Story Points (without additional features)

| Completed User Stories | Story points |
| --- | --- |
| CMS-47 | 5 |
| CMS-49 | 2 |
| CMS-48 | 4 |
| CMS-52 | 4 |
| CMS-53 | 4 |
| CMS-78 | 3 |
| CMS-79 | 2 |
| CMS-51 | 6 |
|  |  |
| *Total # completed user story points* | *30* |

| Remaining User Stories | Story points |
| --- | --- |
| CMS-84 | 5 |
| CMS-85 | 3 |
| CMS-86 | 4 |
| CMS-87 | 6 |
| CMS-103 | 5 |
|  |  |
| *Total # remaining user story points* | *23* |

# 3. Software Architecture

## 1 - Introduction

### 1.1 - Identifying information

The architecture being expressed in our case is a condo management system (CMS). Our tech stack includes programming languages, frameworks, a database, as well as front-end and back-end tools.

For the front-end, we are using React to build a progressive web app, along with NodeJS. Our back-end is an ASP.NET Web API written in C#, using Entity Framework Core and Microsoft SQL Server to compose the database. All of these technologies are part of the overall .NET coding environment.

The system of interest for which this document is written is our condo management system. Indeed, we are building a condo management app as well as its corresponding website. An important feature that our application possesses is its accessibility on major operating systems as a PWA, either as a website or downloadable web app.

### 1.2 - Rationale for key decisions

The specific technologies used for this architecture were chosen based on all the team members’ experiences from previous projects. Most importantly, they were chosen for their compatibility.

| **Front-End Technologies** | **Reasoning** |
| --- | --- |
| React | React was selected for its component-based architecture, which enables modular and reusable code. This framework simplifies the development of dynamic and interactive user interfaces, making it well-suited for creating the progressive web app aspect of our condo management system. Its wide adoption and robust community support ensure an abundance of resources and tools, facilitating development and problem-solving. |
| NodeJS | NodeJS was chosen for its efficiency and scalability, thanks to its non-blocking, event-driven architecture. It serves as the backbone for our React-based front-end, providing a lightweight and flexible server-side platform that can handle multiple connections simultaneously without compromising performance. NodeJS's extensive npm ecosystem allows easy integration of packages and tools, enhancing our development workflow and capabilities in building a progressive web app. |

| **Back-End Technologies** | **Reasoning** |
| --- | --- |
| C# | This programming language was chosen because it can be easily run on the .NET Framework, being its main language. |
| .NET | This framework is the main dependency for all other back-end technologies used. It provides the environment that allows us to develop using the Microsoft suite of tools. |
| Entity Framework Core | EF Core was chosen to handle .NET entities. This object-database mapper connects with a database such as Microsoft SQL Server and handles data transactions while minimizing the need to write in the SQL language. |
| Microsoft SQL Server | Microsoft SQL Server was chosen for its integration with most operating systems, which avoids the possibility of any compatibility issues. |

## 2 - Stakeholders and Concerns

* **Public User**: A new user who registers and makes use of the condo management app to obtain data, submit requests, and engage with the system.

| **Category** | **Concerns** |
| --- | --- |
| Purpose | Creating profiles, and personal data is secure and easily registered using the condo management company's registration key. |
| Suitability | User-friendly interface for profile management. |
| Feasibility | The app's accessibility and availability across multiple platforms. |
| Risks/Impacts | Security and privacy issues about profile data. |
| Maintenance/Evolution | Updating and providing user support. |

* **Condo Owner**: The owner of a condo unit, uses the app to manage their properties, monitor financial data, make reservations, and submit requests.

| **Category** | **Concerns** |
| --- | --- |
| Purpose | Easy request submission, accurate financial information, and effective management of condo units. |
| Suitability | A user-friendly dashboard of the property with general information and a property overview. |
| Feasibility | Availability across a range of platforms and devices. |
| Risks/Impacts | Accuracy of data and security of financial information. |
| Maintenance/Evolution | Regular upgrades for new features and refinements. |

* **Condo Management Company**: The organization in charge of managing several properties. It creates property profiles, uploads files, manages financial aspects, and sets up common facilities and roles for employees.

| **Category** | **Concerns** |
| --- | --- |
| Purpose | Appropriate personnel work assignment, financial tracking, and property management. |
| Suitability | Financial tools and a comprehensive property profile management system. |
| Feasibility | Compatibility across several platforms and integration with banking systems. |
| Risks/Impacts | Possible financial inequalities and data security. |
| Maintenance/Evolution | Ongoing assistance with system updates and personnel responsibility management. |

* **Rental User**: The tenant of a rented condo unit. They can review their financial information, make reservations, submit requests, and manage their rental properties using the app.

| **Category** | **Concerns** |
| --- | --- |
| Purpose | Easy request submission, accurate financial information, and effective management of rented units. |
| Suitability | Easy-to-use dashboard with financial information and a summary of the rental unit. |
| Feasibility | Availability across a range of platforms and devices. |
| Risks/Impacts | Reliability of data and financial information security. |
| Maintenance/Evolution | Regular updates for new features and improvements. |

## 3 - Views

The Condo Management System (CMS) is described here using the 4+1 architectural view model [4]. This model contains five viewpoints: scenarios (use-case view), the logical view, the process view, the development view, and the deployment view.

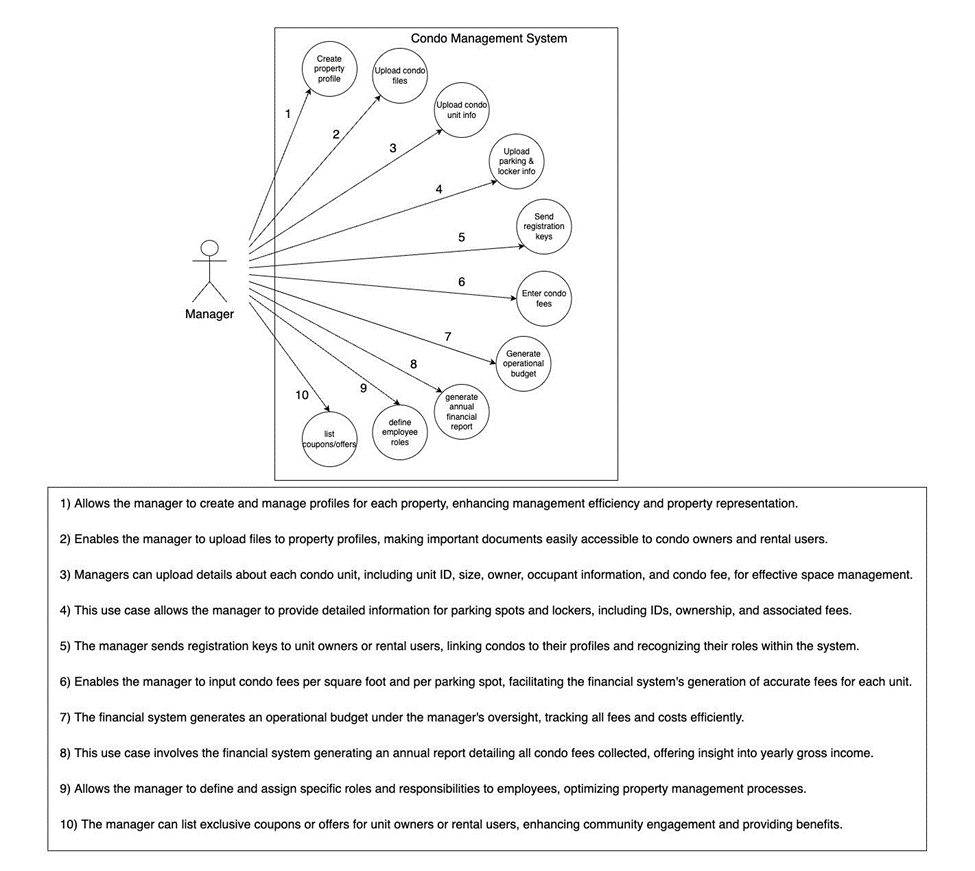
### 3.1 - Scenarios

The Use Case View is important to capture the functional requirements of the CMS and demonstrate the interactions between the system and its users.

#### 3.1.1 - Use Case Diagram for “Public User” and “Condo Owner”

### 

#### 3.1.2 - Use Case Diagram for “Condo Management Company”

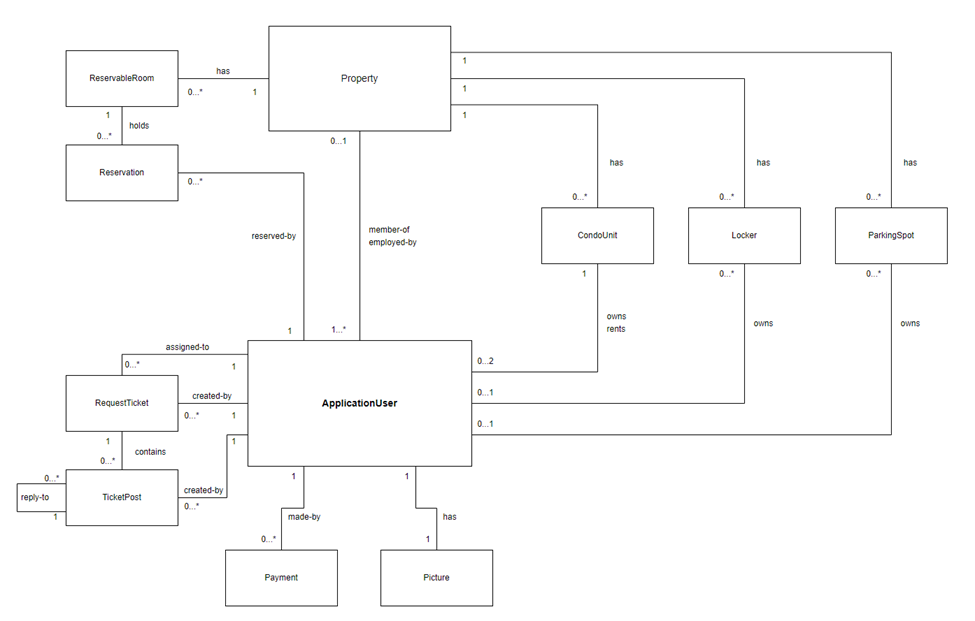


### 3.2 - Logical View

The Logical View focuses on the system’s functionality and the conceptual organization of the system.

#### 3.2.1 - Domain Model

The domain model provides a high-level overview of the main entities and their relationships with the CMS.



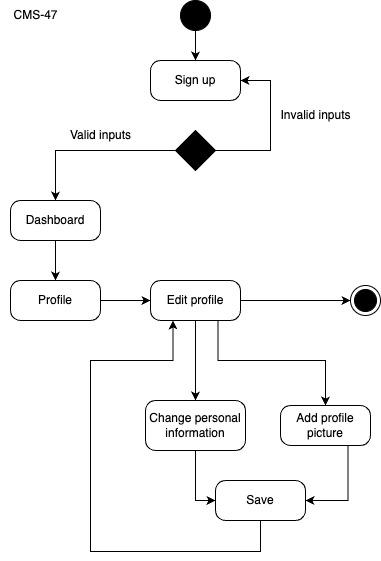
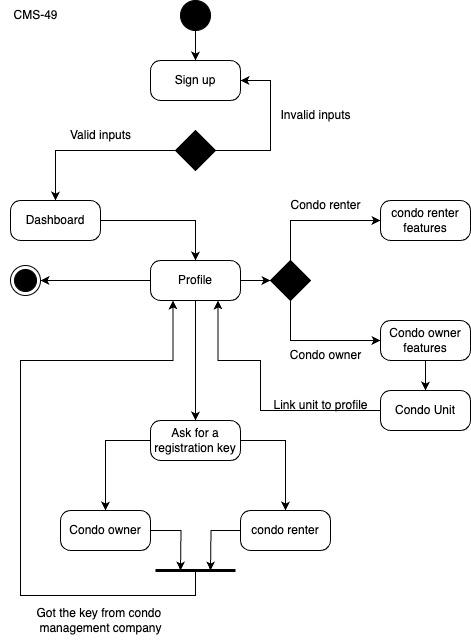
## 

### 3.3 - Process View

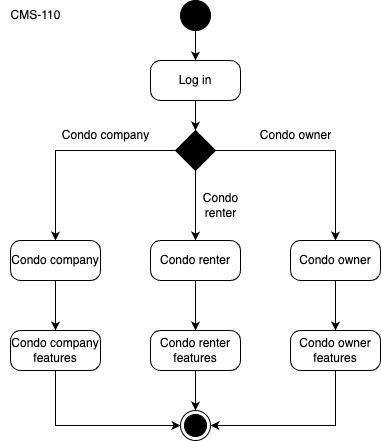
The Process View focuses on the dynamic aspects of the system, illustrating how the system responds to events.

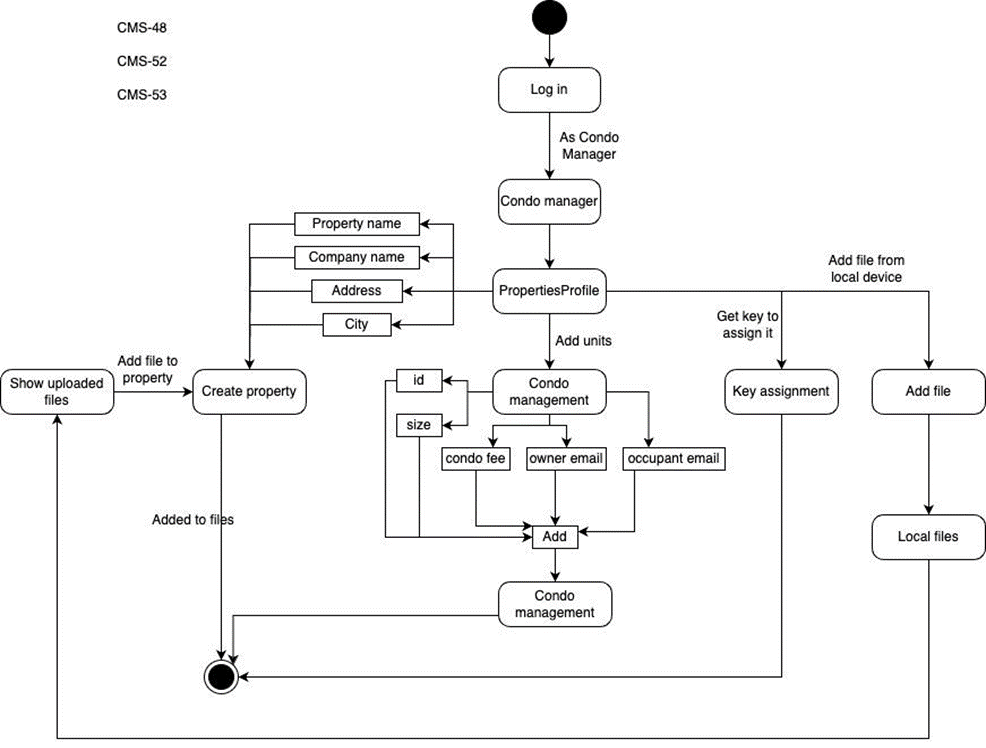
#### 3.3.1 - Activity Diagrams

##### 3.3.1.1 CMS-47 - CMS-49

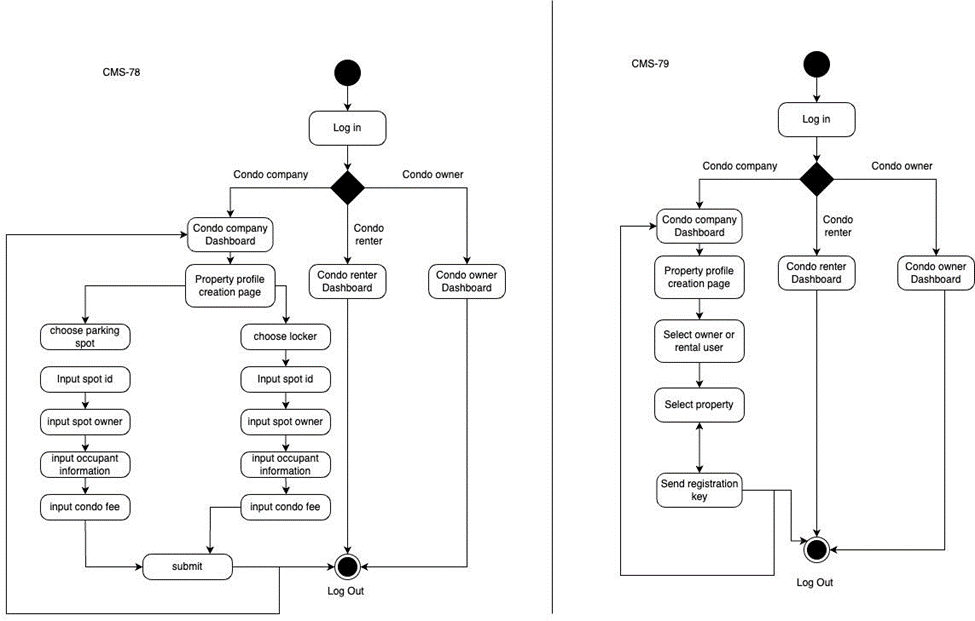
 

##### 3.3.1.2 CMS-110 - CMS-48 - CMS-52 - CMS-52

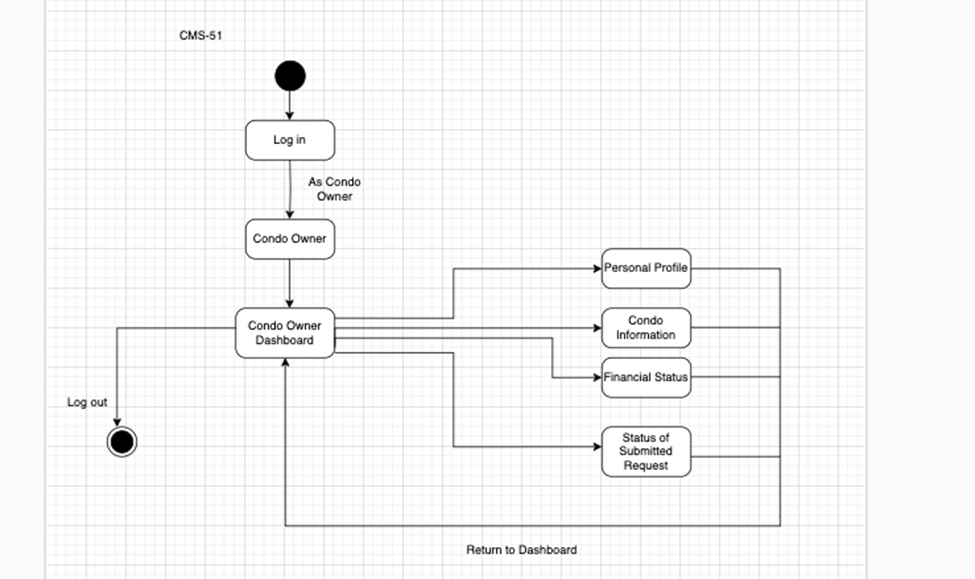


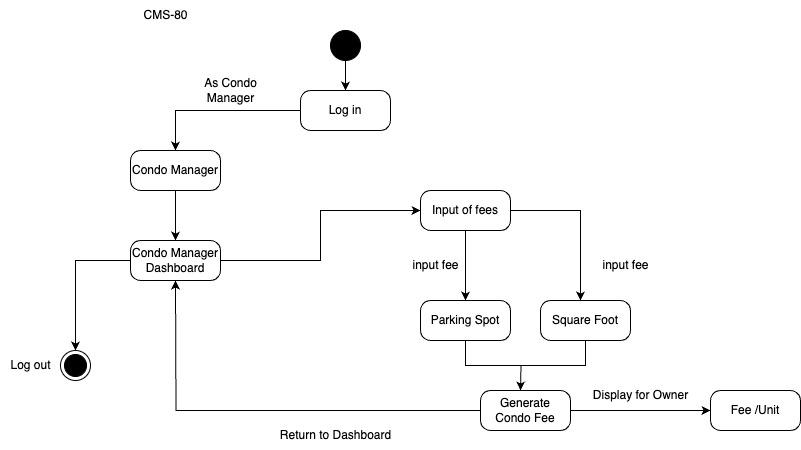


##### 3.3.1.3 CMS-78 - CMS-79

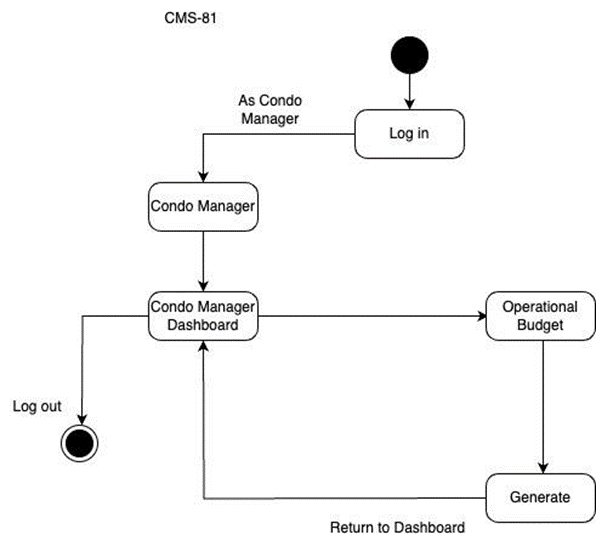


##### 3.3.1.4 CMS-51 - CMS-80





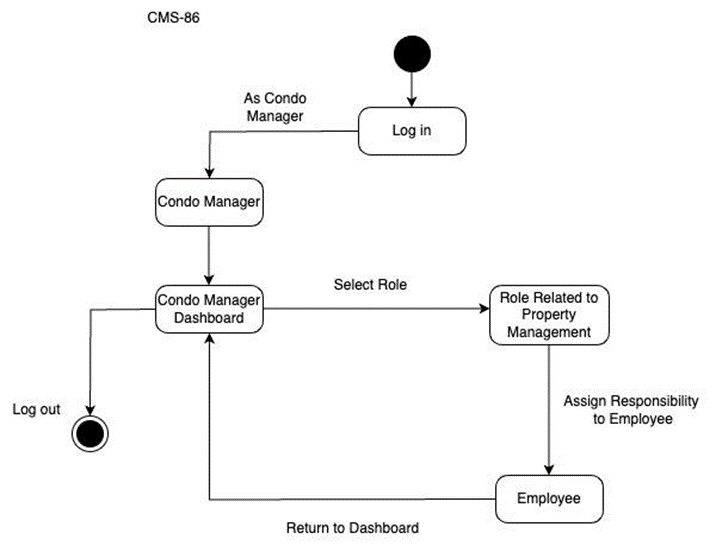
##### 3.3.1.5 CMS-81

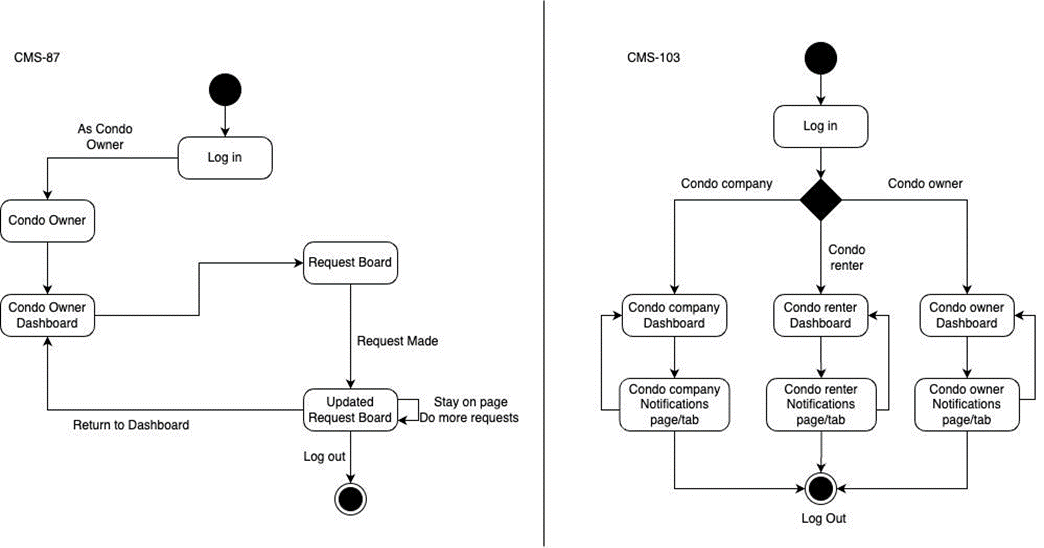


##### 3.3.1.6 CMS-82 - CMS-83 - CMS-84 - CMS-85

## 

##### 3.3.1.7 CMS-86 - CMS-87 - CMS-103



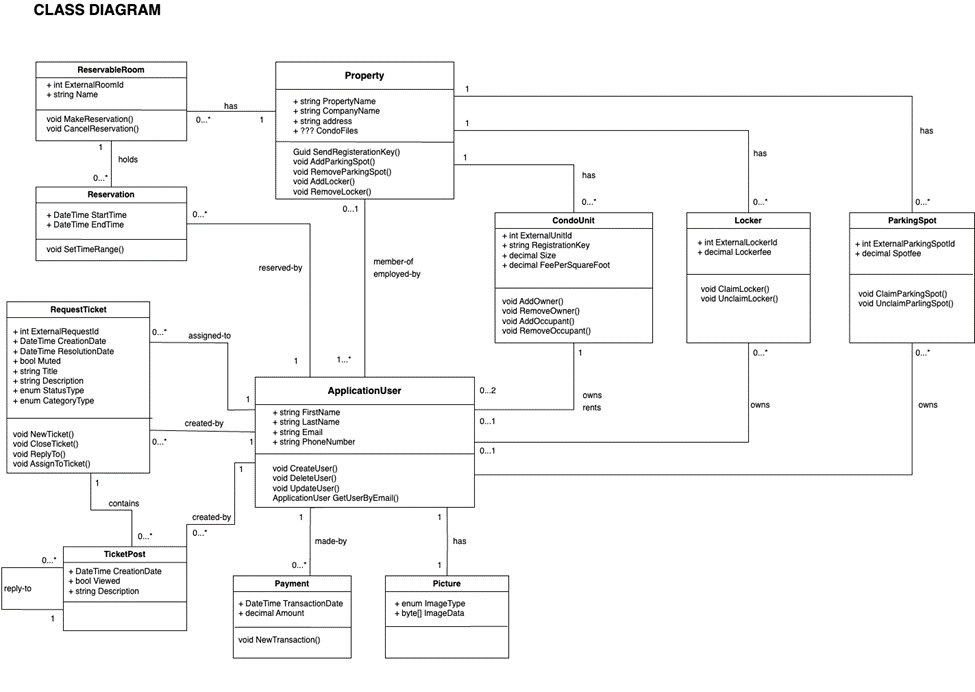


### 3.4 - Development View

The Development View is concerned with the actual realization of the system in source code.

#### 3.4.1 - Class Diagram for User Management

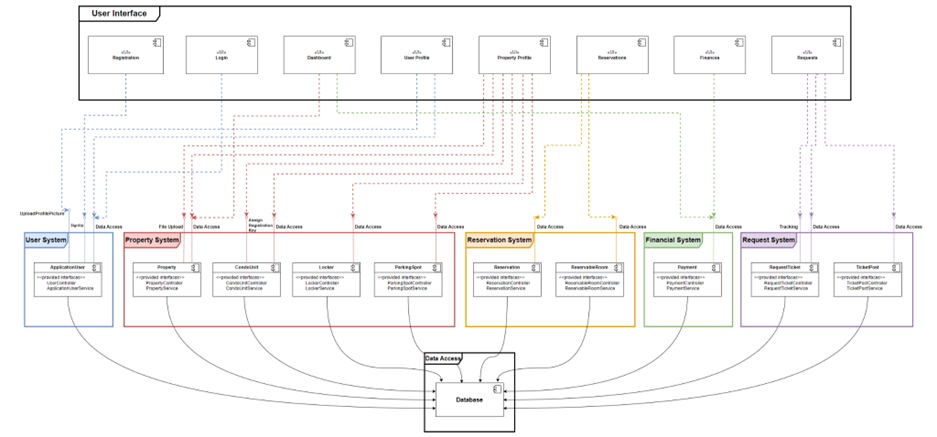
This diagram represents the classes involved in managing user information and their relationships.



### 

#### 3.4.2 - Component Diagram

This diagram illustrates the main software components of the CMS and their interactions.



## 

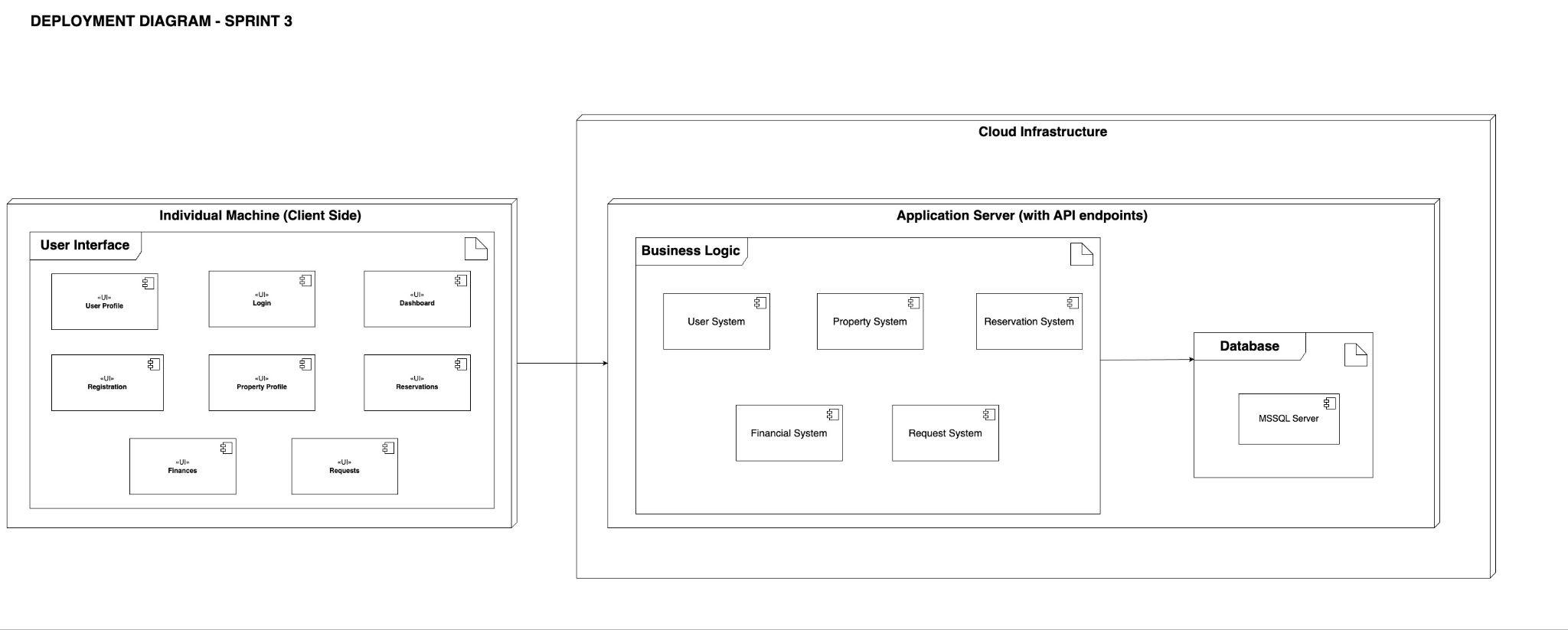
## 

### 3.5 - Deployment View

The Deployment View shows the physical distribution of software components across the hardware infrastructure.

#### 3.5.1 - Deployment Diagram

This diagram illustrates the deployment of the CMS components on the cloud infrastructure. The client side is basically the User Interface containing all the UI components like the Dashboard, Property Profile, User Profile, etc; which is linked to the Application Server that includes the Business Logic, connected to the Database. The former has the logic for the system functionalities and the latter is built on MSSQL server.



### 3.6 - Known Issues with Views

At this stage, there are no known discrepancies between the views and their viewpoint conventions. However, as the project evolves, we will document any inconsistencies, incomplete items, open or unresolved issues, and deviations from the conventions established by the viewpoints. These will be addressed through decision-making processes and documented as outcomes and rationales.

# 

## 

## 4 - Architectural Decisions

### 4.1 - Front-end Technologies

The front-end of our Condo Management System is engineered to provide a seamless, user-friendly experience across various platforms and devices. To achieve this, we have opted for technologies that are both versatile and powerful, ensuring our application is accessible, responsive, and easy to use. The choice of React and NodeJS specifically caters to these requirements, facilitating the development of a progressive web app that meets the modern standards of web applications.

#### 4.1.1 - React

React was chosen for its exceptional capability to build dynamic and complex user interfaces with efficiency and ease [3]. This JavaScript library offers a component-based architecture, enabling our team to develop reusable UI components that can manage their state, leading to more manageable code and faster development cycles. React's virtual DOM feature optimizes rendering, making the user experience smooth and responsive. Furthermore, its vast ecosystem, including tools, libraries, and community support, provides valuable resources for solving development challenges. React's popularity and wide adoption also mean that it stays at the forefront of web development technologies, ensuring our application remains current and robust.

#### 4.1.2 - NodeJS

The decision to utilize NodeJS as part of our front-end technology stack is grounded in its efficiency and scalability, which are crucial for handling the server-side aspects of our progressive web app [5]. NodeJS's event-driven, non-blocking I/O model makes it particularly suited for data-intensive, real-time applications that run across distributed devices. It enables our application to handle numerous simultaneous connections with high throughput, which is essential for a condo management system expected to serve a large number of users. Additionally, NodeJS's compatibility with JavaScript allows for a unified development experience across both client and server sides, streamlining the development process and reducing the learning curve for developers familiar with JavaScript. The extensive npm package ecosystem further enhances NodeJS's functionality, offering a wealth of modules and tools that can be easily integrated to extend the application's capabilities and accelerate development.

### 4.2 - Back-end Technologies

The majority of the technologies that the Condo Management System’s back-end will be built upon are part of Microsoft’s developer environment. This will help the team avoid any compatibility issues between components as we build the back-end web API, as all of these are designed to work together. The following sections describe the reasoning behind our choices in more detail.

#### 4.2.1 - C#

The decision to use C# as the main language for this project was made largely to be able to use the other parts of the tech stack listed in the next sections as they are designed for this language in particular. Indeed, the choice of these frameworks is what will determine how well the concerns of the stakeholders are ultimately addressed, not the coding language itself. Nonetheless, choosing C# still provides certain benefits over other languages for our team for two main reasons. First, it is quite similar to Java, a language that the majority of the team has extensively worked with throughout our engineering programs. Second, certain members of the team have experience with C# itself. Both of these factors indicate that C# will be easy for the team to work with during this project.

#### 4.2.2 - .NET

The .NET platform [1], built by Microsoft, is the ideal infrastructure for the Condo Management System. The large community support that .NET has will help the team more easily find solutions to any problems that may arise. Indeed, the ASP.NET Web API we are building has great documentation available, providing us with guidance as to how we should structure our files and follow code conventions.

#### 4.2.3 - Entity Framework Core

A framework used to allow .NET to easily access a database and store entities, Entity Framework Core is the obvious choice for a system using Microsoft technologies. One of the best benefits it provides is for developers, allowing us to almost completely avoid writing SQL queries [2].

#### 4.2.4 - Microsoft SQL Server

In keeping with the theme of the back-end tech stack, Microsoft SQL Server allows us to easily implement our domain model while remaining as conflict-free as possible due to being part of the Microsoft environment.

# 4. Risk Assessment and Risk Management

## **Introduction**

The risk assessment and management plan: documents that lists hazards and risks that could affect a software project, along with mitigation or avoidance techniques. The plan's objective is to guarantee that the project is finished on schedule, on budget, and with the appropriate standard.

## **Advantages of the risk assessment and management plan**

* Early identification of any possible problems allows for the necessary mitigation through suitable action.
* Risks are prioritized according to their likelihood and probable consequences, which helps the team concentrate on the most crucial problems.
* Improve project performance by lowering the risk of overruns in budgets and delays.

## **How Risks and Assessment identified among our team**

Through discussion sessions where team members shared their knowledge and experiences and explored potential risks, the project team determined the risks connected with the project. The group also carried out analysis to determine the most common hazards on internet platforms and the kinds of vulnerabilities to which the project was most at risk.

The team evaluated and ranked the hazards according to their likelihood and possible consequences. The group produced a risk management strategy that comprised tactics to reduce or remove the risks that were identified as well as a method for tracking the effectiveness of the risk management measures through reporting and monitoring. The team was able to concentrate on the most significant risks and raise the likelihood that the project would succeed thanks to this thorough approach.

## **Measurement information for the Risks**

Impact measures: Low = 1 day delay, Medium = 3 days delay, High = 5 days delay

Probability measures: Low = (0 - 30%), Medium = (30 - 70%), High = (70% - 100%)

## **Tables**

### **4.1. Risk management chart**

| Impact  Probability | Low | Medium | High |
| --- | --- | --- | --- |
| Low | R07 |  | R02, R04, R05,  R08, R15 |
| Medium | R16 | R06, R10, R12 | R01, R03, R11, R13, R14 |
| High |  | R09 |  |

**Figure [4.1]**: Risk management chart

### **4.2. List of identified risks**

| Risk ID | Risk Type and Description | Risk Score | Resolved in Sprint | Strategy and Effectiveness |
| --- | --- | --- | --- | --- |
| US-1.1 | * Technical * Management * External * Budget * Schedule * Etc. | * Medium * High |  | * Mitigate * Accept * Avoid * Transfer |
| R01 | **Time Risk:** The project islimited by the capstone timeframe, meaning once the timeframe ends, the contract with the client also ends and the project should be completed. | Impact: High (5 days delay)  Probability: Medium (50%)  Risk Score: High  Consequences:  1) Team members are under more pressure to meet their deadlines.  2) Potential quality reduction caused by time restraints. | Sprint 5 | **Strategy:** By efficient planning, we can make sure all the features we need are complete.  **Strategy type:** Mitigate  **Contingency plan:** Removing the more unnecessary features and saving them for last.  **When to invoke the contingency plan:** Invoke the plan when the team is deviating too much off schedule. |
| R02 | **Application Risk:** there exists the risk that our project doesn't solve the problem we intended. | Impact: High (5 days delay)  Probability: Low (5%)  Risk Score: Medium  Consequences: If the application fails to achieve its goal, the time and resources spent developing it were wasted. | Sprint 5 | **Strategy:** Survey our solution to see if it theoretically makes sense. Proper testing before more costly implementations.  **Strategy type:** Mitigate  **Contingency plan:** do it again next semester or replan the solution.  **When to invoke the contingency plan:** when the app fails or when user testing is very negative. |
| R03 | **Skill Risk:** there is a chance the team won’t have the skill set in order to complete the project in the time frame or at all. | Impact: High (5 days delay)  Probability: Medium (30%)  Risk Score: Medium  Consequences:1) Potential delays in the project brought on by a lack of team knowledge.  2) a higher chance of mistakes and poor quality output if team members are not equipped with the required skills. | Sprint 5 | **Strategy:** Proper planning and use of our programmers' skills.  **Strategy type:** Mitigate  **Contingency plan:** hire additional more skilled individuals to make up for the lack and teach.  **When to invoke the contingency plan:** when straying too far off the schedule. |
| R04 | **Legal risk:** The risk that the execution of the project might violate a legal convention or rule. | Impact: High (5 days delay)  Probability: Low (0 - 5%)  Risk Score: Medium  Consequences: If the project is discovered to be in violation of the law, there could be harm to its trust and reputation. | Sprint 5 | **Strategy**: Reviewing legal conventions or precedent is beyond the scope of our project, but making a minor change  **Strategy type:** Transfer  **Contingency plan:** We will revert to the old state of the system if something serious arises  **When to invoke the contingency plan:** If our project leads to a serious legal problem, but this is extremely unlikely in our opinion. |
| R05 | **Fraud risk:** The risk that our project will be exploited by one of the team members with malicious intent, for example, to steal user data. | Impact: High (5 days delay)  Probability: Low (0 - 5%)  Risk Score: Medium  Consequences: Bad reputation and a loss of user trust in the event that private information is lost or stolen. | Sprint 5 | **Strategy:** We believe that the likelihood that any of our team would do something like that is extremely low since there is almost no incentive to steal our clients' information, since it has little utility outside of the system and the legal repercussions would far outweigh the upside.  **Strategy type:** Mitigate  **Contingency plan:** We will forward the pertinent information to law enforcement if we believe a crime was committed.  **When to invoke the contingency plan:** If we believe a malicious action has been committed. |
| R06 | **Scalability Risk:** The website may not perform well if the number of users or operations is high due to imperfect infrastructure. | Impact: Medium (3 days delay)  Probability: Medium (30 - 40%)  Risk score: Medium  Consequences: lower user satisfaction if the website performs poorly during periods of high number of users. | Sprint 5 | **Strategy:** Mitigate byusing a scalable and reliable infrastructure and use load testing for the platform to ensure that its scalability is acceptable  **Strategy type:** Mitigate  **Contingency plan:**  Try alternate deployment methods and services and change the faulty algorithms if applicable  **When to invoke contingency plan:** When we are severely limited by our application's performance |
| R07 | **Team dynamics risk:** Some team members might have difficulty working together due to personal conflicts and different working styles | Impact: Low (1 days delay)  Probability: Low (10%)  Risk score: Low  Consequences: 1) Conflicts between teammates break up team spirit and communication, there will be a decrease in production and collaboration.  2) Possible quality problems or project delays caused by conflict and misunderstanding amongst team members. | Sprint 5 | **Strategy:** Proper team meetings and encourage team communication.  **Strategy type:** Mitigate  **Contingency plan:** Assess the teams composition and explore the possibility to invoke sub teams to avoid any more conflicts within team members.  **When to invoke contingency plan:** When the conflict of team members persists despite initial attempts to resolve it. |
| R08 | **Security risk:**  There is a risk that our application might be vulnerable to cyberattacks, especially since none of our team members are cybersecurity experts. | Impact: High (5 days delay)  Probability: Low (0 - 10%)  Risk score: Medium  Consequences: 1) Possible security breaches or unapproved access to private data might damage user confidentiality and confidence. | Sprint 5 | **Strategy:** Avoid by implementing security measures like 2FA, encryption and DDOS protection  **Contingency plan:** Make a subteam that will be in charge of cybersecurity if it becomes a relevant problem  **When to invoke contingency plan:** If cybersecurity threats become a credible threat to our product |
| R09 | **Availability of Resources Risk:** An unexpected trip by a team member could affect the status of the project and connections within the team. | Impact: Medium (3 days delay)  Probability: High (90 %)  Risk score: High  Consequences: 1) Postponed completion of the traveling team member's allocated job.  2) Possible slowing in decision-making procedures.  3) Additional tasks for the remaining team members. | Sprint 3 | **Strategy:** Assign tasks to the remaining team members and modify deadlines as necessary.  **Strategy type:** Mitigate  **Contingency plan:** assign responsibilities to other team members and use remote collaboration technologies to keep the traveling team member engaged.  **When to invoke contingency plan:** If the traveling team member's absence significantly impacts team communication or project progress, then implement the plan. |
| R10 | **Risk Associated with Technology Compatibility**: Team members who use MacBooks might have trouble using some of the technologies selected for the project. | Impact: Medium (3 days delay)  Probability: Medium (30%)  Risk score: Medium  Consequences: 1) Delays in task completion.  2) A greater dependence on pair programming.  3) Possibility of lower team spirit or annoyance if certain technologies aren't completely utilized. | Sprint 3 | **Strategy:** depending on pair programming as a means of cooperation.Assist team members in learning how to use pair programming or adjust to new technologies by offering guidance and assistance.  **Strategy type:** Mitigate  **Contingency plan:** If required, look into other tools or development environments that work with MacBooks. Set aside more time for pair programming sessions to make sure that everyone is working together effectively.  **When to invoke contingency plan:** Use the strategy if team members who use MacBooks continue to face major challenges because of the selected technologies, which is impeding the project's progress or the quality of their work. |
| R11 | **Platform Change risk:**  Risk that changing platforms from Uno to React might create problems | Impact: Medium (delayed the development of the frontend by at least a week)  Probability: Certain  Risk Score: Medium-High  Consequences:  1) Delay in task completion for the frontend.  2) Domino effect: Delays in frontend development caused setbacks for backend development as well. | Sprint 2 | **Strategy:**  Use Chris as a resource for frontend developers since he has by far the most experience with the new framework. Chris is able to help the other developers with most of their bugs.  **Strategy type:** Mitigate  **Contingency plan:** If required, use more time-consuming learning resources for React to improve our developers' knowledge  **When to invoke contingency plan:**  Use the contingency plan if the number of setbacks for the frontend becomes too much. |
| R12 | **Code Coverage Testing Delay Risk:** The code coverage testing would take longer than expected due to external matters, which would affect the project overall schedule. | Impact: Medium (4 days delay)  Probability: Medium (30%)  Risk Score: Medium  Consequences: 1) it will increase the chance of errors and problems in the code.  2) Possible hold-ups to find and fix the problems.  3) It will decrease the trust in the reliability and dependability of the system. | Sprint 3 | **Strategy:**  Code coverage testing is made as a top priority in the next sprint and assigns multiple developers from the team to complete testing.    **Strategy type:** Mitigate  **Contingency plan:** Provide more resources and continue the testing process into later sprints, if the code coverage testing can’t be done in Sprint 3. We will make sure that other important tasks are not missed.  **When to invoke contingency plan:** if some problems happen during the code coverage testing and stop it from being done in the allocated sprint time. |
| R13 | **Security Risk:** The condo management system could be exposed to security risks, such as virus attacks, hacking, and data breaches, which is dangerous to the data and system reliability. | Impact: High (5 days delay)  Probability: Medium (30%)  Risk Score: High  Consequences: 1) It will harm the system's reputation.  2) Users might lose trust in the system.  3) Potential legal consequences. | Sprint 5 | **Strategy:**  Reduce the risk by increasing the security measures, such as firewalls, robust security measures, regular evaluation of the security..    **Strategy type:** Mitigate  **Contingency plan:** Create backup plans, communicate more with the stakeholders, and create strategies for the security so we can react quickly to any harm.  **When to invoke contingency plan:** If a security breach happens. |
| R14 | **Compliance with Regulations Risk:** laws or industry standards might change at any point in time, that will impact how the condo management system will operate or what it will be able to do. | Impact: High  Probability: Medium (30%)  Risk Score: High  Consequences: 1) It might lead to legal consequences and fines.  2) There will be some disturbances in the system's operation.  3) It will impact the system's reputations, if found non-compliant. | Sprint 5 | **Strategy:**  The risk of non-compliance can be reduced by outsourcing changes or updates to guarantee that the system complies with industry standards and laws.    **Strategy type:** Transfer  **Contingency plan:** Make a contract with third-party providers to keep up with the updates and changes, and they can inform the team on what to do and how to react to these changes.  **When to invoke contingency plan:** There are significant changes to laws or industry standards. |
| R15 | **Management Risk:** The likelihood of internal mismanagement that might result in project failures or errors. This may involve poor decisions, insufficient allocation of the resources, poor communication, and failure to adapt to new changing conditions. | Impact: High  Probability: Low (40%)  Risk Score: High  Consequences: 1) Project Delays: Missing deadlines.  2) Budget Overcharges: poor decisions and insufficient allocation of the resources might lead to time wasting more than what was planned. | Sprint 5 | **Strategy:**  Develop strong management procedures and plan more ways to communicate in order to reduce management risk. This needs solid planning, continuous reviewing, and having backup plans.    **Strategy type:** Mitigate  **Contingency plan:** Develop a good risk strategy that will show detailed steps that needs to be done when mismanagement occurs. To better solve identified issues, the plan should outline how resources should be reallocated, adjust the scopes of the project or change the management methods.  **When to invoke contingency plan:** When there is an early sign of project changing, like missing deadlines, going over time, drop on team performance, then the contingency plan should be invoked. |
| R16 | **Technical Development Risk:** The chance that the condo management system's technology will age and become less effective, and may become unsuitable with the new hardware and software, that leads to a poorer user experience than the new systems. | Impact: Low  Probability: Medium (40%)  Risk Score: Medium  Consequences: 1) Decreased in user's satisfaction.  2) Problems with the interface that come for difficulties integrating the system with new technologies.  3) Maintenance costs will increase because older systems require more time and money to maintain. |  | **Strategy:**  Stay up to date with the customer and users needs and keep up with the technological developments. That will make it easier to upgrade system components and schedule regular updates.    **Strategy type:** Mitigate  **Contingency plan:** Develop a strategy to make continuous reviews on technology and identify the components that may become outdated.  **When to invoke contingency plan:** Every specific period of time (a year), when the technical development creates that the system or the system components might become outdated, after that the contingency plan should be invoked. |

**Figure [4.2]:** List of identified risks

# 7 Retrospectives

## 7.1 Short Sprint #1 Retrospective

Introduction

Firstly, coming into this sprint, our initial expectation was to establish a clear vision of our product vision and the overall mission behind this project. To do so, it was important to identify the stakeholders and their interests. Secondly, it was important to ensure that all team members understood the requirements and constraints this project was to be built on. User stories helped immensely with that since they highlighted the key aspects of our project. Finally, multiple team meetings and significant research have been conducted to establish which tools and architecture our project was going to be based on.

What went wrong

1- New tools and technologies

One of the most challenging parts of this sprint was the choice of the appropriate technologies, databases, and frameworks to use. Due to the vast and constantly evolving landscape of available tools, it was challenging to navigate through the multitude of options and determine all together which ones best suit the project’s requirements.

The difficulty in adapting to new tools has led to long meeting hours as well as delays in development, impacting our project timelines. To address this issue, the team made sure all members spent additional time becoming comfortable with the new technologies needed for this project.

A method that should have been implemented from the beginning is pair programming. Encouraging pair programming sessions within the team would have facilitated faster adoption of the new tools and promoted mutual learning.

2- User stories

One aspect that did not work well during the development is the development of user stories. This issue occurred, because of a possible lack of prioritization during the sprint planning sessions regarding the most valuable features for the end user. The user stories seem to be divided into small distinct parts instead of providing one clear objective.

This led to complicating the sprint planning sessions, making it difficult for the team to accurately estimate the effort required for each task and allocate resources accordingly. To fix this issue, the team conducted refinement sessions to review and determine how to refine the existing user stories in the upcoming sprints. The granular user stories will be broken down into more manageable and clearer chunks.

A method that could have been used to avoid this issue is the use of story mapping techniques. These techniques could help visualize the flow of features and user interactions, allowing the team to identify and prioritize high-level user activities rather than getting bogged down in overly detailed tasks.

3- Codebase and Testing

Another aspect that presented challenges during this sprint was the effectiveness of the codebase. Indeed, the codebase was not as appropriate and efficient as anticipated, leading to challenges in testing and quality assurance efforts. This aspect occurred due to a lack of code reviews and limited resources for refactoring.

This issue caused increased time and effort required for testing and reduced overall test coverage. To address this, the team conducted a code quality assessment to identify areas for improvement and prioritize refactoring efforts.

Improvement could have been made by allocating more time for code refactoring and optimization from the start of code-writing.

4- Uno Platform

The Uno Platform is an aspect that did not work for the development of this first sprint. This happened due to this platform’s insufficient documentation, making it difficult for the team to understand and utilize its features effectively.

The impact included increased time and effort trying to adapt to the new platform and the likelihood of errors or misuse of Uno’s platform’s features. To address this issue, the team conducted a thorough review of existing documentation to update the outdated information about the platform.

More in-depth research on the platform’s documentation could have been done before choosing the platform.

5- C# experience

A challenge that the team encountered was the lack of experience of some team members with the new technologies introduced for this project. This occurred due to factors such as the limited exposure to C# in previous projects and the limited time assigned to achieving the deliverables.

This aspect has led to slower adoption and learning curves, delays in task completion, as well as increased workload on specific team members. To fix this issue, the team has leveraged online resources and documentation to provide additional support. Furthermore, the team has assigned tasks based on team members’ strengths and areas for growth, allowing for gradual hands-on experience.

Improvements could have been made by conducting a skills assessment at the outset of the project to identify the gaps within the team. Based on that assessment, the choice of language used for this project could have differed.

What went right

1- Good pace

One aspect that did work well during the development was the pace at which our team worked during the sprint. This good pace in executing tasks can be attributed to effective sprint planning and task breakdown. Initially, our team invested sufficient time to define clear sprint goals, prioritize tasks, and break them down into manageable units by assigning them to all team members.

The good pace in executing tasks resulted in increased productivity, allowing our team to deliver more value within the sprint timeframe. This positively impacts overall project progress and contributes to meeting project milestones and deadlines. To address this aspect, our team prioritized tasks based on their importance and urgency, focusing on high-priority items first to ensure maximum value delivery within the sprint timeframe.

While the good pace in executing tasks is commendable, there is always room for improvement. Identifying and mitigating risks that could potentially impact task execution could’ve helped maintain a more consistent pace throughout the sprint.

2- Continuous Monitoring

An aspect that worked well during the development is the continuous monitoring and adjustment. This aspect was well executed due to clear and open communication among team members. Regular meetings and updates on task progress helped keep everyone aligned and focused on their respective responsibilities. This could include reallocating resources, re-prioritizing tasks, or providing additional support to team members as needed.

By doing so, our team was able to adapt quickly to evolving requirements, priorities, and challenges. It enhanced our ability to stay aligned with project goals and respond to changes in the project environment. To address this aspect, daily stand-up meetings were conducted. These meetings have provided an opportunity for team members to share updates, discuss progress, and raise any issues or concerns.

However, a better and more constant use of the Jira platform by all team members could’ve helped document our progress.

3- Agile Methodology Implementation

The implementation of Agile methodologies worked well during the development. This aspect occurred due to clear understanding and commitment from the team. Agile principles such as iterative development, frequent feedback, and adaptive planning were embraced and integrated into the team’s workflow.

The impact of effective Agile methodology implementation included increased productivity within the team, as well as improved collaboration among all team members. To ensure this implementation, the team established daily stand-ups and sprint planning meetings. Furthermore, regular retrospectives were held to keep up with the team’s processes and identify areas that needed more work.

This implementation could have been better by strengthening communication channels between the different parts of the sprints. For example, if some members finished earlier than anticipated, and other members needed help with their parts, teammates should be able to move from part to part to help each other.

4- Team Communication

Good communication was a key aspect during this sprint. Clear and transparent communication channels were established through the use of Slack and Discord. These channels encouraged team members to communicate openly and proactively share updates, questions, and concerns.

This aspect improved coordination between team members, reduced misunderstandings, and enhanced overall team cohesion. Also, it ensured everyone was aligned with the sprint goals and aware of their responsibilities, which led to higher productivity. Team communication was addressed by setting communication expectations from the start of the project and encouraging active participation in team meetings and discussions.

To further improve communication, the team could have implemented more structured guidelines for providing constructive feedback.

5- Collaborative Problem Solving

An aspect that did work well during the development is Collaborative Problem Solving. Each member of the team brought unique perspectives and problem-solving approaches to the table.

This aspect increased creativity, faster problem resolution, and a stronger sense of ownership and accountability among team members. To address Collaborative Problem Solving, the team organized brainstorming sessions and workshops to generate ideas collaboratively.

To further enhance Collaborative Problem Solving, the team could organize more opportunities to share knowledge and expertise such as in-person meetings.

Conclusion

In conclusion, the team has encountered both successes and challenges in equal measure. One of the main takeaways from this sprint is the importance of effective communication and collaboration within the team. Despite facing obstacles such as technical complexities and time constraints, the team’s ability to work together, share knowledge, and support one another has been instrumental in overcoming obstacles. Additionally, the team has gained a deeper understanding of the technologies and processes involved, highlighting areas for improvement and growth as this project continues. Finally, this sprint has highlighted the significance of adaptability when encountering hurdles. As the team navigated through unfamiliar territories, all team members demonstrated their ability to learn quickly, pivot when necessary, and remain focused on the project’s requirements.

# 

## 7.2 Short Sprint #2 Retrospective

Introduction

Firstly, coming into this sprint, our team knew that multiple elements needed an update. Our product vision statement, requirements and user stories needed minor changes to make them more specific and clearer to what we wanted for our application. Secondly, our team took a collective decision to change the environment we had set up for our application. Indeed, for our front-end, we decided to switch from the Uno Platform to using React.

What went wrong

1. Transition of Front-End Tools

One of the more challenging parts of this sprint was the change of tools done for our front-end part. Our team encountered multiple issues and limitations with the Uno Platform, initially chosen, such as compatibility issues, performance concerns, or lack of community support. Therefore, our team decided to choose a different route and utilize React instead for the continuation of the project.

However, the shift in tools necessitated the rewriting of code, and the adjustment of development processes, resulting in a temporary slowdown of project progress. To address this challenge, our team recalibrated project milestones, revised timelines and allocated additional resources to facilitate the transition to React.

This transition could have been avoided by choosing the right tool for all team members at the beginning of the project. To do so, our team could have conducted a more thorough evaluation of the potential tools’ considering factors such as platform compatibility, learning curves, long-term maintenance requirements, integration capabilities, etc.

1. React

An aspect that went wrong for some, but not all team members, was the utilization of React for our front-end. Since React introduces new concepts such as JSX syntax, virtual DOM, and component-based architecture, it was challenging for some team members that haven’t had hands-on experience with it.

The complexity of React concepts led to confusion, frustration, and slower adoption among team members, affecting productivity and overall motivation. To address the issue, the team provided comprehensive documentation and tutorials to familiarize team members with React concepts gradually.

Implementing a structured pair programming where experienced React team members are paired with those new to React could have facilitated this issue for the whole team.

1. C# experience

A challenge that the team continued to deal with during this sprint is the lack of experience of some team members with C#, used for this project. This occurred due to factors such as the limited exposure to C# in previous projects and the limited time assigned to achieving the deliverables.

This aspect has led to slower adoption and learning curves throughout sprint #2. To fix this issue, the team continues to leverage online resources and documentation to provide additional support. Furthermore, the team has assigned tasks based on team members’ strengths and areas for growth, allowing for a gradual hands-on experience.

Improvements could have been made by conducting a skills assessment at the outset of the project to identify the gaps within the team. Based on that assessment, the choice of language used for this project could have differed.

1. 80% code coverage

The most challenging aspect of this sprint was the implementation of necessary tests to achieve of 80% code coverage. The failure of doing so may have been caused due to our team facing time constraints that led to deprioritizing testing efforts.

Without sufficient test coverage, the project is susceptible to undetected bugs and regressions, potentially leading to degraded product quality and increased maintenance efforts. To address this issue, our team has conducted a retrospective analysis to identify the root causes of the testing gaps. Furthermore, we’ve reevaluated project priorities and allocated dedicated time and resources for test implementation.

An approach that may have been implemented from the start is a test-driven development (TDD). This approach could have ingrained testing as a fundamental aspect of the development process, ensuring comprehensive coverage from the outset.

1. Pressing Deadline

An aspect that our team struggled with during this second print, but not the first, was completing the requirements for this sprint before the deadline initially imposed (before the extension was given). This was caused by high-pressure of trying to finish everything on time while dealing with other academic work and personal life.

Team members felt compelled to rush certain tasks or skip thorough testing and validation processes to meet the deadlines. To address this issue, the team prioritized tasks based on criticality and potential impact.

Setting realistic and achievable deadlines based on thorough project planning and estimation could have prevented excessive pressure on team members and allowed for a more balanced allocation of resources and efforts.

What went right

1. Good pace

One aspect that continued to work well during the development was the pace at which our team worked during the sprint. This good pace was reached by the effective sprint planning and task breakdown our team maintained from the first print. This time around, the prioritizing and the assignment of tasks was done more quickly.

The good pace in executing tasks resulted in increased productivity, allowing our team to deliver more value within the sprint timeframe. This positively impacts overall project progress and contributes to meeting project milestones and deadlines. To address this aspect, our team prioritized tasks based on their importance and urgency, focusing on high-priority items first to ensure maximum value delivery within the sprint timeframe.

While the good pace in executing tasks is commendable, there is always room for improvement. Indeed, when it comes to the implementation of code, identifying and mitigating risks concerning the implementation could’ve avoided the large amount of time that was spent on resolving coding issues on the back-end part.

1. Continuous Monitoring

An aspect that continued to work well during the development is the continuous monitoring and adjustment. Regular meetings and updates on task progress helped keep everyone aligned and focused on their respective responsibilities once again. This included reallocating resources, re-prioritizing tasks, or providing additional support to team members as needed.

By doing so, our team was able to adapt quickly to evolving requirements, priorities, and challenges. It enhanced our ability to stay aligned with project goals and respond to the bigger changes that were made in the project environment. To address this aspect, daily stand-up meetings were conducted. These meetings have provided an opportunity for team members to share updates, discuss progress, and raise any issues or concerns.

However, the integration of monitoring tools with automated incident response systems could have helped us more. By automating the response to common alerts and predefined scenarios, our team could have reduced manual intervention, minimized downtime, and improved overall system reliability.

1. Agile Methodology Implementation

The implementation of Agile methodologies remained something that worked well during the development. This aspect occurred due to clear understanding and commitment from the team. Agile principles such as iterative development, frequent feedback, and adaptive planning were embraced and integrated into the team’s workflow.

The impact of effective Agile methodology implementation included increased productivity within the team, as well as improved collaboration among all team members. To ensure this implementation, the team established daily stand-ups and sprint planning meetings. Furthermore, regular retrospectives were held to keep up with the team’s processes and identify areas that needed more work.

However, this implementation could have been better by putting more effort in strengthening communication channels between the different parts of the sprints. For example, if some members finished earlier than anticipated, and other members needed help with their parts, teammates should be able to move from part to part to help each other. This aspect is something our team still must work on.

1. Jira

The use of the Jira Platform was improved during this sprint and was a result of something working well for our team. The effort in using a user-friendly platform like Jira originated from our team’s need to efficiently organize tasks, assign them to each other, and track our progress throughout the development cycle.

This team effort significantly improved team coordination and helped in meeting project deadlines effectively. To ensure every member of the team put in this effort, regular stand-up meetings were held where team members discussed their task’s status, identified any blockers, and collaborated on solutions.

While the task management aspect was efficient, providing more detailed task descriptions and acceptance criteria upfront could have improved clarity and reduced the need for frequent clarifications.

1. Collaborative Problem Solving

An aspect that did work well and was maintained during the development of sprint #2 is

our team’s collaborative problem solving. This was reached, because each member of the team brought unique perspectives and problem-solving approaches to the table.

This aspect increased creativity, faster problem resolution, and a stronger sense of ownership and accountability among team members. To address collaborative problem solving, the team organized brainstorming sessions and workshops to generate ideas collaboratively.

However, for future sprints, our team could organize more opportunities to share knowledge and expertise such as in-person meetings to further enhance this aspect.

Conclusion

In conclusion, the team has encountered both achievements and challenges in equal measure during this second sprint. One of the main takeaways from this sprint is the importance to remain consistent in communicating and collaborating with one another. Despite facing obstacles such as technical complexities in the back end and time constraints, the team’s ability to work together, share knowledge, and support one another has been instrumental in overcoming obstacles. Additionally, the team has gained a deeper understanding of the technologies that were retained from sprint #1. However, when it comes to the new technologies that were installed, there is room for improvement and growth for all team members as this project continues. Finally, this sprint has highlighted the significance of adaptability when encountering hurdles. As the team navigated through unfamiliar territories implemented in this sprint, all team members demonstrated their ability to adapt quickly and remain focused on the project’s requirements and deadlines to respect.

# 

## 7.3 Short Sprint #3 Retrospective

Introduction

Firstly, coming into this sprint, our team’s focus was to focus on testing to achieve at least 80% of code coverage. To do so, we collectively agreed to put aside the implementation of new features and dedicate more time and effort on back-end and front-end unit testing the increase. Furthermore, based on the sprint #2 feedback, a deployment diagram was added to our system architecture document.

What went wrong

1. Transition of Front-End Tools

One of the more challenging parts that was found again in this sprint was the change of tools done in sprint#2 for our front-end part. Indeed, our team encountered multiple issues and limitations with the Uno Platform, initially chosen, such as compatibility issues, performance concerns, or lack of community support. Therefore, our team decided to choose a different route and utilize React instead for the continuation of the project.

However, the shift in tools necessitated the rewriting of code, and the adjustment of development processes, resulting in a temporary slowdown of project progress. While many of these concerns were resolved during sprint#2, many members of the team are still accommodating to this transition. To address this challenge, our team recalibrated project milestones, revised timelines and allocated additional resources to facilitate the transition to React.

This transition could have been avoided by choosing the right tool for all team members at the beginning of the project. To do so, our team could have conducted a more thorough evaluation of the potential tools’ considering factors such as platform compatibility, learning curves, long-term maintenance requirements, integration capabilities, etc.

1. React

An aspect that went wrong once again for some, but not all team members, was the utilization of React for our front-end. Since React introduces new concepts such as JSX syntax, virtual DOM, and component-based architecture, it was challenging for some team members that haven’t had hands-on experience with it.

The complexity of React concepts led to confusion, frustration, and slower adoption among team members, affecting productivity and overall motivation. To address the issue, the team provided comprehensive documentation and tutorials to familiarize team members with React concepts gradually.

Implementing a structured pair programming where experienced React team members are paired with those new to React could have facilitated this issue for the whole team.

1. C# experience

A challenge that the team continued to deal with during this sprint is the lack of experience of some team members with C#, used for this project. This occurred due to factors such as the limited exposure to C# in previous projects and the limited time assigned to achieving the deliverables.

This aspect has led to slower adoption and learning curves throughout sprint #3, especially with addition of test cases. To fix this issue, the team continues to leverage online resources and documentation to provide additional support. Furthermore, the team has assigned tasks based on team members’ strengths and areas for growth, allowing for a gradual hands-on experience.

Improvements could have been made by conducting a skills assessment at the outset of the project to identify the gaps within the team. Based on that assessment, the choice of language used for this project could have differed.

1. Documentation

An aspect that our team struggled with during this third print, but not the first two, was not completing the documentation at the last minute and before the deadline imposed. This was caused by the high-pressure and extreme focus of successfully achieving 80% code coverage on time.

By focusing on testing, team members neglected the importance of the documentation aspect of our sprint and felt compelled to leave it to the last day. To address this issue, the team agreed to find a balance between achieving the testing or the implementation of new features and the writing of documents. To do so, we’ve decided to include specific days where every member of the team would work on the documentation rather than implementation/testing.

Finally, setting realistic and achievable deadlines based on thorough project planning and estimation could have prevented excessive pressure on team members and allowed for a more balanced between implementation/testing and completion of documentation.

1. Jira

The use of the Jira Platform was not as efficient during this sprint as it was during sprint#2. Despite its capabilities, we fell short in maximizing its benefits for organizing tasks, assigning responsibilities, and monitoring progress effectively throughout the development cycle.

Acknowledging Jira's value to our team, we understand that our usage during this sprint didn't reflect its full potential. To address this issue, we’ve agreed to provide more detailed task descriptions and acceptance criteria in future sprints to improve clarity and streamline our processes effectively.

This issue could have been avoided by setting frequent reminders for every member of the team to participate in Jira and to hold them accountable if they haven’t shown consistent participation.

What went right

1. 80% Code Coverage

In this new sprint, our team had effortlessly achieved an impressive 80% code coverage, marking a significant milestone for our team. The implementation of necessary tests that were not achieved in prior sprints was seamless, reflecting our team's proactive approach to quality assurance.

By ensuring 80% code coverage, we can now state that are our project is well-protected against undetected bugs and regressions. This ensures that our product maintains its high quality and reduces the need for extensive maintenance efforts in the future. To reach this level of code coverage, our team focused on prioritizing testing and allocating dedicated time and resources effectively to address any potential gaps in our testing strategy.

However, this achievement could have been reached in earlier sprints if our team hadn’t underestimated the importance of testing in our project. Testing should have been a priority over the implementation of new features from the start of the project. Moving forward, we will continue to uphold this standard of excellence in testing, leveraging our newfound success as motivation to maintain and even surpass our current levels of code coverage.

1. Continuous Monitoring

An aspect that continued to work well during the development is the continuous monitoring and adjustment. Regular meetings and updates on task progress helped keep everyone aligned and focused on their respective responsibilities once again. This included reallocating resources, re-prioritizing tasks, or providing additional support to team members as needed.

By doing so, our team was able to adapt quickly to evolving requirements, priorities, and challenges. It enhanced our ability to stay aligned with project goals and respond to the bigger changes that were made in the project environment. To address this aspect, daily stand-up meetings were conducted. These meetings have provided an opportunity for team members to share updates, discuss progress, and raise any issues or concerns.

However, the integration of monitoring tools with automated incident response systems could have helped us more. By automating the response to common alerts and predefined scenarios, our team could have reduced manual intervention, minimized downtime, and improved overall system reliability.

1. Agile Methodology Implementation

The implementation of Agile methodologies remained something that worked well during the development. This aspect occurred due to clear understanding and commitment from the team. Agile principles such as iterative development, frequent feedback, and adaptive planning were embraced and integrated into the team’s workflow.

The impact of effective Agile methodology implementation included increased productivity within the team, as well as improved collaboration among all team members. To ensure this implementation, the team established daily stand-ups and sprint planning meetings. Furthermore, regular retrospectives were held to keep up with the team’s processes and identify areas that needed more work.

However, this implementation could have been better by putting more effort in strengthening communication channels between the different parts of the sprints. For example, if some members finished earlier than anticipated, and other members needed help with their parts, teammates should be able to move from part to part to help each other. This aspect is something our team still must work on.

1. Good Pace

One aspect that continued to work well during the development was the pace at which our team worked during the sprint. This good pace was reached by the effective sprint planning and task breakdown our team maintained from the first print. This time around, the prioritizing and the assignment of tasks was done more quickly.

The good pace in executing tasks resulted in increased productivity, allowing our team to deliver more value within the sprint timeframe. This positively impacts overall project progress and contributes to meeting project milestones and deadlines. To address this aspect, our team prioritized tasks based on their importance and urgency, focusing on high-priority items first to ensure maximum value delivery within the sprint timeframe.

While the good pace in executing tasks is commendable, there is always room for improvement. Indeed, when it comes to the implementation of code, identifying and mitigating risks concerning the implementation could’ve avoided the large amount of time that was spent on resolving coding issues on the back-end part.

1. Collaborative Problem Solving

An aspect that did work well and was maintained during the development of sprint #2 is

our team’s collaborative problem-solving. This was reached, because each member of the team brought unique perspectives and problem-solving approaches to the table.

This aspect increased creativity, faster problem resolution, and a stronger sense of ownership and accountability among team members. To address collaborative problem solving, the team organized brainstorming sessions and workshops to generate ideas collaboratively.

However, for future sprints, our team could organize more opportunities to share knowledge and expertise such as in-person meetings to further enhance this aspect.

Conclusion

In conclusion, the team has encountered both achievements and challenges in equal measure during this third sprint. One of the main takeaways from this sprint is the importance of testing and consistently reaching at least 80% code coverage. Despite facing obstacles such as learning curves of front-end tools and lack of use of the Jira platform, the team’s ability to work together, share knowledge, and support one another has been instrumental in overcoming obstacles. Additionally, the team has continued to gain a deeper understanding of the technologies that were retained from sprint #1. However, when it comes to the new technologies that were installed in sprint#2, there is always room for improvement and growth for some team members as this project continues. Finally, this sprint has highlighted the significance of prioritizing testing and not underestimating it when having to implement new features. As the team navigated through unfamiliar territories implemented in this sprint, all team members demonstrated their ability to adapt quickly and remain focused on the project’s requirements and deadlines to respect.

# 

## 7.4 Short Sprint #4 Retrospective

Introduction

Firstly, coming into this sprint, our team’s focus was to implement all features to have a lighter succeeding and final sprint as it coincides with finals. To do so, we collectively agreed to spend less time on testing and dedicate more time and effort on back-end and front-end implementation. Furthermore, based on the sprint #3 feedback, the missing scenarios in the Use-Case diagram and Activity diagram were added, additional acceptance tests were provided, and the overall content of the code management document was improved.

What went wrong

1. Pressing deadline

An aspect that our team struggled with during this fourth sprint was completing the requirements for this sprint before the imposed deadline (before and after the extension was given). This was caused by the high pressure of trying to finish everything on time while dealing with final exams season.

Team members felt compelled to rush certain tasks to meet deadlines or leave things to the last-minute due to other more urgent coursework. To address this issue, the team prioritized tasks based on criticality and potential impact.

Setting realistic and achievable deadlines based on thorough project planning and estimation could have prevented excessive pressure on team members and allowed for a more balanced allocation of resources and efforts.

1. React

An aspect that continued to be challenging for some, but not all, team members, was the utilisation of React for our front-end. Since React introduces new concepts such as JSX syntax, virtual DOM, and component-based architecture, it was challenging for some team members who haven’t had hands-on experience with it.

The complexity of React concepts led to confusion, frustration, and slower adoption among team members, affecting productivity and overall motivation. To address the issue, the team provided comprehensive documentation and tutorials to familiarize team members with React concepts gradually.

Implementing structured pair programming where experienced React team members are paired with those new to React could have facilitated this issue for the whole team.

1. C# experience

A challenge that the team continued to deal with during this sprint was the lack of experience of some team members with C#, used for this project. This occurred due to factors such as the limited exposure to C# in previous projects and the limited time assigned to achieving the deliverables.

This aspect has led to slower adoption and learning curves throughout sprint #3, especially with the addition of test cases. To fix this issue, the team continues to leverage online resources and documentation to provide additional support. Furthermore, the team has assigned tasks based on team members’ strengths and areas for growth, allowing for a gradual hands-on experience.

Improvements could have been made by conducting a skills assessment at the outset of the project to identify the gaps within the team. Based on that assessment, the choice of language used for this project could have differed.

1. Documentation

An aspect that our team struggled with during this sprint was not completing the documentation at the last minute and before the imposed deadline. This was caused by the high-pressure and extreme focus of successfully implementing all features, so that our final sprint can only be dedicated to polishing and fixing existing elements. By focusing on the completion of our features, the team neglected the importance of the documentation aspect of our sprint and felt compelled to leave it to the last day.

To address this issue, the team agreed to find a balance between achieving the implementation of new features and the writing of documents. To do so, we’ve decided to include specific days where every member of the team would work on the documentation rather than implementation.

Finally, setting realistic and achievable deadlines based on thorough project planning and estimation could have prevented excessive pressure on team members and allowed for a balance between implementation and completion of documentation.

1. Jira

The use of the Jira Platform was not as efficient during this sprint as it was during our two first sprints. Despite its capabilities, we fell short in maximizing its benefits for organizing tasks, assigning responsibilities, and monitoring progress effectively throughout the development cycle.

Acknowledging Jira's value to our team, we understand that our usage during this sprint didn't reflect its full potential. To address this issue, we’ve agreed to provide more detailed task descriptions and acceptance criteria in future sprints to improve clarity and streamline our processes effectively.

This issue could have been avoided by setting frequent reminders for every member of the team to participate in Jira and holding them accountable if they haven’t shown consistent participation.

What went right

1. Deployment

In this new sprint, our team successfully achieved the deployment of our website, marking a significant milestone for our team. Our streamlined deployment process ensured that all necessary updates and features were seamlessly integrated into the live site.

By doing so, it minimized downtime and disruptions for our users, enhancing their overall experience. To address potential issues, we conducted thorough testing across various environments, allowing us to identify and resolve any issues prior to deployment. Also, clear communication channels were established to promptly address any unforeseen challenges that arose during the process.

While the deployment was executed smoothly, there is always room for improvement. Enhancing automation tools and further refining our testing protocols could expedite future deployments and enhance overall efficiency.

1. Continuous Monitoring

An aspect that continued to work well during the development was the continuous monitoring and adjustment. Regular meetings and updates on task progress helped keep everyone aligned and focused on their respective responsibilities once again. This included reallocating resources, re-prioritizing tasks, or providing additional support to team members as needed.

By doing so, our team was able to adapt quickly to evolving requirements, priorities, and challenges. It enhanced our ability to stay aligned with project goals and respond to the bigger changes that were made in the project environment. To address this aspect, daily stand-up meetings were conducted. These meetings have provided an opportunity for team members to share updates, discuss progress, and raise any issues or concerns.

However, the integration of monitoring tools with automated incident response systems could have helped us more. By automating the response to common alerts and predefined scenarios, our team could have reduced manual intervention, minimized downtime, and improved overall system reliability.

1. Agile Methodology Implementation

The implementation of Agile methodologies remained something that worked well during the development. This aspect occurred due to clear understanding and commitment from the team. Agile principles such as iterative development, frequent feedback, and adaptive planning were embraced and integrated into the team’s workflow.

The impact of effective Agile methodology implementation included increased productivity within the team, as well as improved collaboration among all team members. To ensure this implementation, the team established daily stand-ups and sprint planning meetings. Furthermore, regular retrospectives were held to keep up with the team’s processes and identify areas that needed more work.

However, this implementation could have been better by putting more effort into strengthening communication channels between the different parts of the sprints. For example, if some members finished earlier than anticipated, and other members needed help with their parts, teammates should be able to move from part to part to help each other. This aspect is something our team still needs to work on.

1. Camaraderie

One aspect that was highlighted throughout this sprint was the remarkable teamwork and support within our team when faced with challenges. This camaraderie stemmed from a shared understanding that our collective success depended on each member’s contributions. If one team member encountered difficulties with their tasks or features, others readily stepped in to help and ensure progress was not impeded. This proactive approach ensured that no individual was left struggling alone and that critical project milestones were consistently met.

To address potential bottlenecks, regular check-ins were made by the team leader and status updates were conducted, allowing us to redistribute workload as needed and provide additional support where necessary.

However, while our team’s willingness to assist each other was commendable, there’s always room for improvement. Establishing clearer escalation procedures and refining task prioritization methods could further enhance our ability to respond swiftly to challenges and optimize project efficiency.

1. Collaborative Problem Solving

An aspect that did work well and was maintained during the development of sprint #2 is

our team’s collaborative problem solving. This was reached, because each member of the team brought unique perspectives and problem-solving approaches to the table.

This aspect increased creativity, faster problem resolution, and a stronger sense of ownership and accountability among team members. To address collaborative problem solving, the team organized brainstorming sessions and workshops to generate ideas collaboratively.

However, for future sprints, our team could organize more opportunities to share knowledge and expertise such as in-person meetings to further enhance this aspect.

Conclusion

In conclusion, the team has encountered both achievements and challenges in equal measure during this fourth sprint. Despite facing obstacles such as learning curves for front-end and back-end tools and a lack of use of the Jira platform, the team’s ability to work together, share knowledge, and support one another has been instrumental in overcoming obstacles. Additionally, the team has continued to gain a deeper understanding of the technologies that were retained from sprint #1. However, when it comes to the new technologies that were installed in sprint #2, there is always room for improvement and growth for some team members as this project continues. One of the main takeaways taken from this sprint is the importance of teamwork and communication when implementing new complex features. Finally, as the team navigated through the unfamiliar territories implemented in this sprint, all team members demonstrated their ability to adapt quickly and remain focused on the project’s requirements and deadlines.

# 

# 8 Release Plans

## 8.1 Sprint #2 Release Plan

Generally speaking, the tasks that will be accomplished during Sprint 2 concern property profile creation by the condo management companies (condo details, parking, fees, …), the assignment of registration keys to the condo owners and rental users from the condo management companies, and finally the user’s condo dashboard.

More specifically, these are the points with a short explanation of each:

1- “User profile creation”: The ability of the user (condo owner or rental user) to create an account, and log in to save their information.

2- “Property profile creation”, “condo file upload”, “condo unit management”, and “adding parking spots and lockers to a property profile”. These 4 user stories fall under the creation of the property profile. To explain more, the first user story (“property profile creation”) consists of only creating the condo (condo number). The second user story (“condo file upload”) consists of uploading a file for the condo containing all information related to it. The third user story (“condo unit management ”) consists of writing all the details about the condo (from the condo size to the occupant information, etc.). Finally, the last user story “Adding parking spots and lockers to a property profile” consists of reserving parking spots to the condo.

3- “Assignment of registration keys” and “send registration keys” fall in the same category. The first user story is about the condo owner or rental user receiving a registration key to access their condo dashboard on the app. The second user story is about the condo management company sending the registration key to the condo owner or the rental user.

4- “User dashboard”: This feature allows the condo owner or the rental user to access any information they might need concerning the condo (managing finances, being up to date with submitted requests, checking available amenities, etc.).

As an initial thought, we’re planning on using Microsoft Azure for future deployment.

| **User Story ID** | **User Story Points**  **(USP)** | **Priority** | **Status** |
| --- | --- | --- | --- |
| User Profile Creation | 5 | High | DONE |
| Property Profile Creation | 4 | High | TODO |
| Assignment of Registration Keys | 2 | Medium | TODO |
| Condo File Upload | 4 | Medium | TODO |
| Condo Unit Management | 4 | High | TODO |
| Adding Parking Spots and Lockers to a Property Profile | 3 | Medium | TODO |
| Send Registration Keys | 2 | Medium | TODO |
| User Dashboard | 6 | High | TODO |
| **Total USP** | 30 |  |  |

## 

## 8.2 Sprint #3 Release Plan

Generally speaking, the tasks that will be accomplished during Sprint 3 are related to:

-room booking for condo owners and rental users.

-Financial management for the condo management companies (operational budget, annual report, decision about condo fees).

Diving deep into each user story:

1- “Manager Input of Fees”: The manager of the respective condo management company enters the condo fee per square foot, per parking spot for each unit.

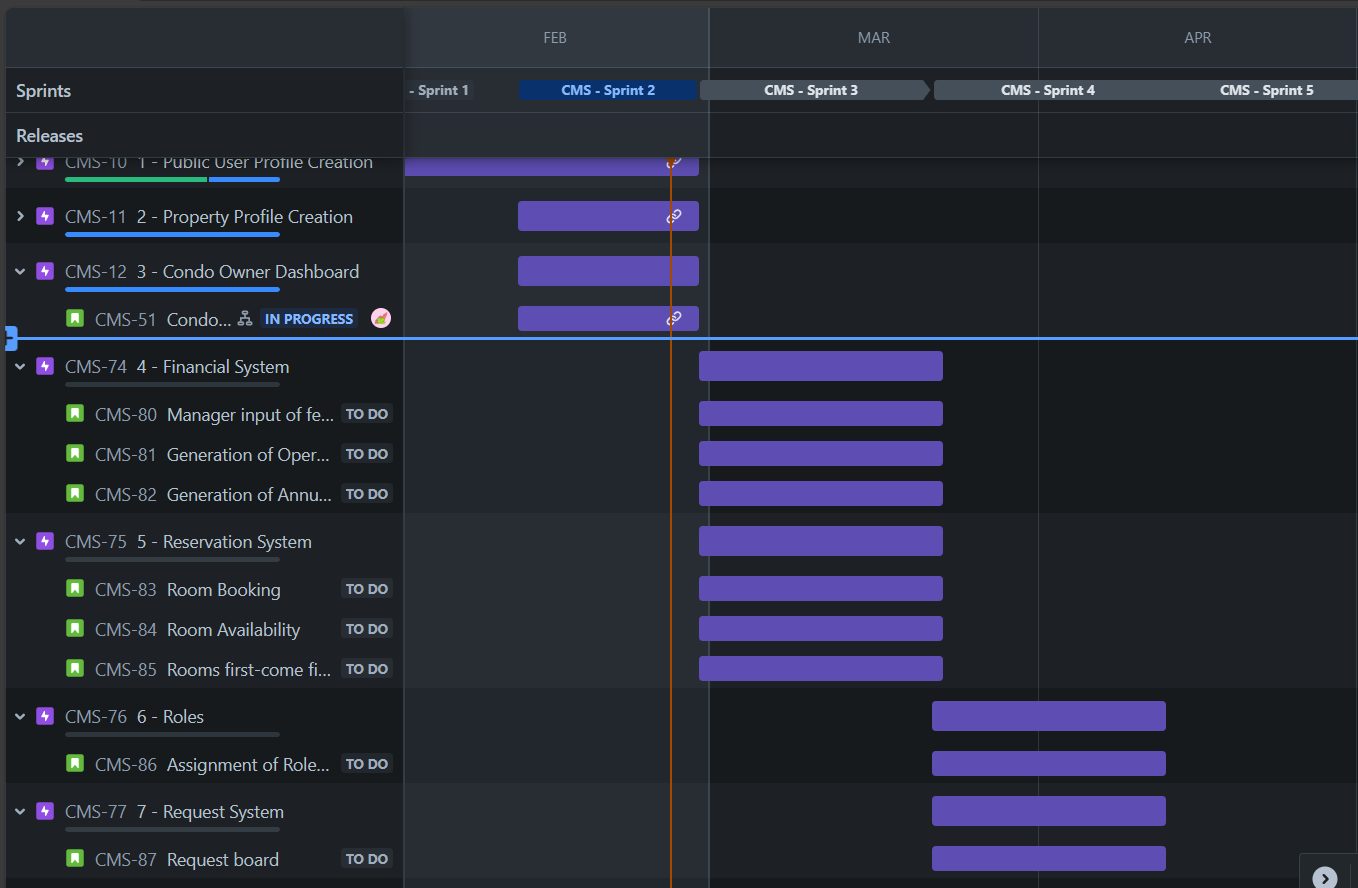
2- “Generation of operational budget”, and “Generation of annual report”: these two user stories fall under the same roof. The first user story consists of creating a financial system for the condo management company so the manager can keep track of all fees and costs. The second user story consists of generating an annual report describing the fees collected in a year, so the manager can view the yearly gross income.

3- “Room booking”, “Room availability”, “Room first-come first-serve”: These 3 user stories fall under the same category. The first one allows the condo owner and the rental user to book any available common room. The second one checks for the availability of the desired common room that the condo owner or rental user wants to book. That way, the user can plan in advance. Finally, the third user story adds order to the booking process. This process works as first-come, first-served, which means that the first condo owner or rental user that books a common room will have access to this common room first.

| **User Story ID** | **User Story Points**  **(USP)** | **Priority** | **Status** |
| --- | --- | --- | --- |
| Manager Input of Fees (CMS-80) | 4 | High | TODO |
| Generation of operational budget (CMS-81) | 4 | High | TODO |
| Generation of annual report (CMS-82) | 5 | Medium | TODO |
| Room booking (CMS-83) | 4 | High | TODO |
| Room availability (CMS-84) | 4 | Medium | TODO |
| Room first-come first-serve (CMS-85) | 5 | High | TODO |
| **Total USP** | 30 |  |  |

*The user story points were chosen based on 2 factors: difficulty of implementation and importance of implementation (how important it is to have this feature).*

Future deployment plans: Microsoft Azure



*bar graph that shows our team’s progress*

## 8.3 Sprint #4 Release Plan

Generally speaking, the tasks that will be accomplished during Sprint 4 are related to:

* Managing input fees for units and parking spots.
* Generation of Operational Budget and Annual Report.
* Assignment of Roles to Employees.
* Request board, notification board, discussion board, event board, and offers board.

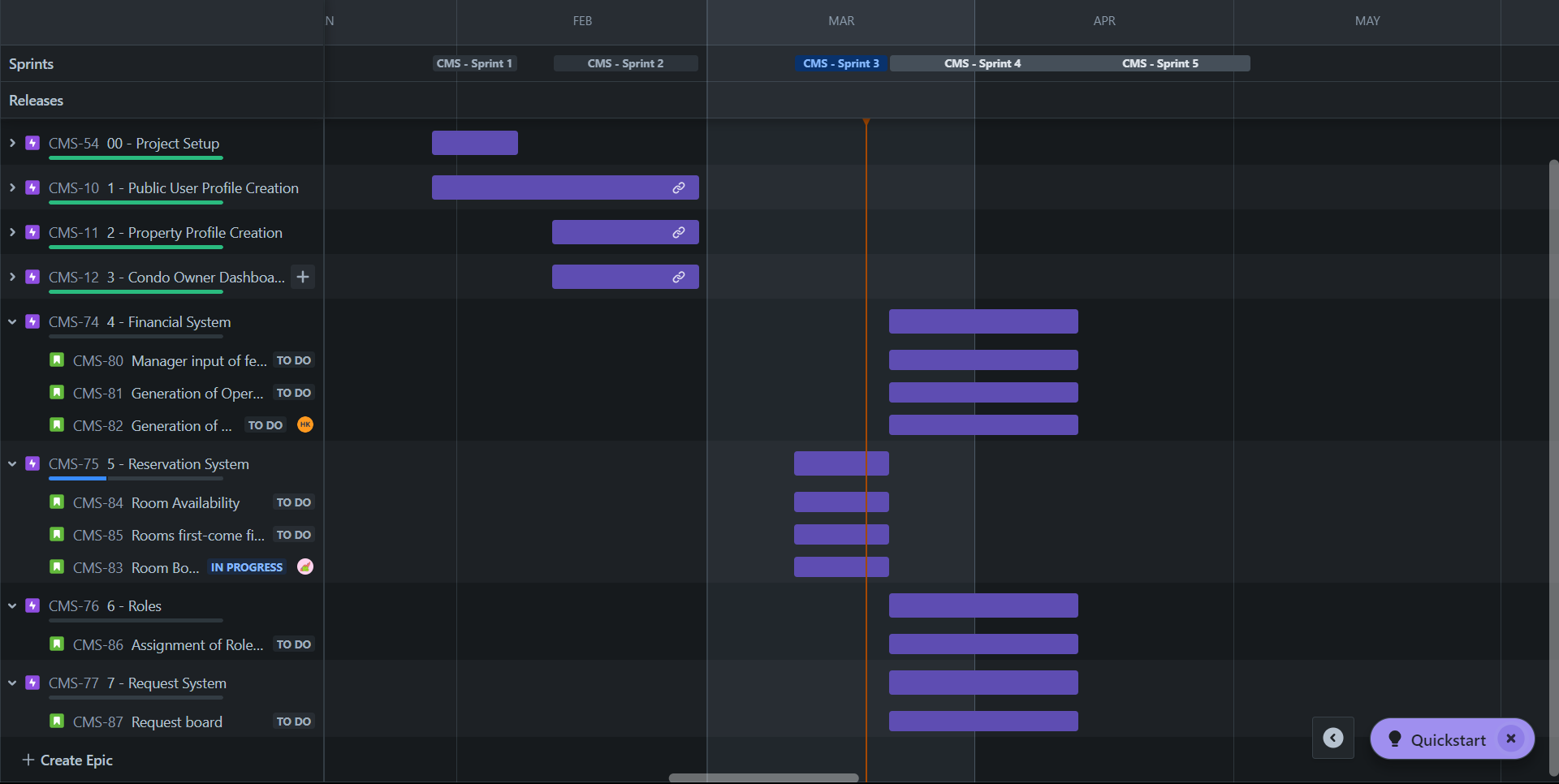
Diving deep into each user story:

1. “Manager Input of Fees”: The manager of the respective condo management company enters the condo fee per square foot, per parking spot for each unit.
2. “Generation of operational budget”, and “Generation of annual report”: these two user stories fall under the same roof. The first user story consists of creating a financial system for the condo management company so the manager can keep track of all fees and costs. The second user story consists of generating an annual report describing the fees collected in a year, so the manager can view the yearly gross income.
3. “Assignment of Roles to Employees”: The manager of the condo management company assigns roles to different employees based on their needs. In addition, managers can create new roles and add employees.
4. “Request Board”: condo owners can submit a request for moving in/out, intercom changes, reporting a violation, reporting a deficiency and/or asking any questions so that necessary action and assistance can be taken by the management.
5. “Notification Board”: User can follow the latest activities by accessing a notification page.
6. “Discussion Board”: User can communicate with others within the community.
7. “Event Board”: User can organize events and invite other occupants and can also accept or reject invites for other events.
8. “Offers Board”: The manager can list coupons/offers for unit owners or rental users for a specific property.

| **User Story ID** | **User Story Points**  **(USP)** | **Priority** | **Status** |
| --- | --- | --- | --- |
| Manager Input of Fees (CMS-80) | 4 | High | TODO |
| Generation of operational budget (CMS-81) | 4 | High | TODO |
| Generation of annual report (CMS-82) | 5 | Medium | TODO |
| Assignment of Roles to Employees (CMS-86) | 4 | High | TODO |
| Request Board (CMS-87) | 4 | High | TODO |
| Notification Board (CMS-103) | 3 | Medium | TODO |
| Discussion Board (CMS-104) | 4 | High | TODO |
| Event Board (CMS-105) | 4 | High | TODO |
| Offers Board (CMS-106) | 4 | Medium | TODO |
| **Total USP** | 36 |  |  |

*The user story points were chosen based on 2 factors: difficulty of implementation and importance of implementation (how important it is to have this feature).*

Future deployment plans: Microsoft Azure



*bar graph that shows our team’s progress*

## 

## 8.4 Sprint #5 Release Plan

Generally speaking, the tasks that will be accomplished during Sprint 4 are related to:

· Request board, notification board, discussion board, event board, and offers board.

· Cross-Platform access.

· Multilingual support.

· External account authentication.

Diving deep into each user story:

1) “Request Board”: condo owners can submit a request for moving in/out, intercom changes, reporting a violation, reporting a deficiency and/or asking any questions so that necessary action and assistance can be taken by the management.

2) “Notification Board”: User can follow the latest activities by accessing a notification page.

3) “Discussion Board”: User can communicate with others within the community.

4) “Event Board”: User can organize events and invite other occupants and can also accept or reject invites for other events.

5) “Offers Board”: The manager can list coupons/offers for unit owners or rental users for a specific property.

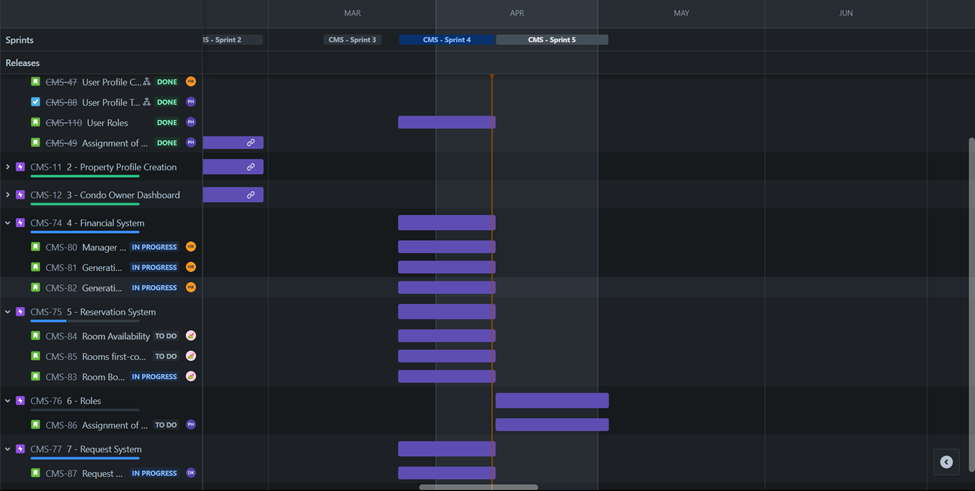
6) “Cross-Platform access”: User can access the system on Android, iOS, Linux, MacOS, and Windows so that he has the freedom to use any device.

7) “Multilingual support”: User has the option to use the app in English and at least one other language so that the app can be more accessible to everyone.

8) “External account authentication”: User has the option to log in using his Gmail account or other such email account so that he can access the app seamlessly without creating a new set of credentials.

| **User Story ID** | **User Story Points**  **(USP)** | **Priority** | **Status** |
| --- | --- | --- | --- |
| Notification Board (CMS-103) | 3 | Medium | TODO |
| Discussion Board (CMS-104) | 4 | High | TODO |
| Event Board (CMS-105) | 4 | High | TODO |
| Offers Board (CMS-106) | 4 | Medium | TODO |
| Cross-Platform access (CMS-107) | 3 | Medium | TODO |
| Multilingual support (CMS-108) | 2 | Low | TODO |
| External account authentication (CMS-109) | 2 | Low | TODO |
| **Total USP** | 22 |  |  |

*The user story points were chosen based on 2 factors: difficulty of implementation and importance of implementation (how important it is to have this feature).*

Future deployment plans: Microsoft Azure  
  


*bar graph that shows our team’s progress*

# 9 UI Prototypes

## 9.1 Sprint #2 UI Prototype

User Story 1

- US-1.1

As a public user, I want to insert the registration key to access the application content.

## 

**Claiming Condo page**

Steps:

1- After signing up or logging in (without having entered the key before), the user is automatically taken to this page.

2- The user enters the registration key they received from their condo management company, and presses on “claim condo”.

3- The user is taken to the condo’s dashboard if the registration key is correct. Otherwise, they will have to enter the key again, until it is the correct one.

User Story 2

- US-2.1

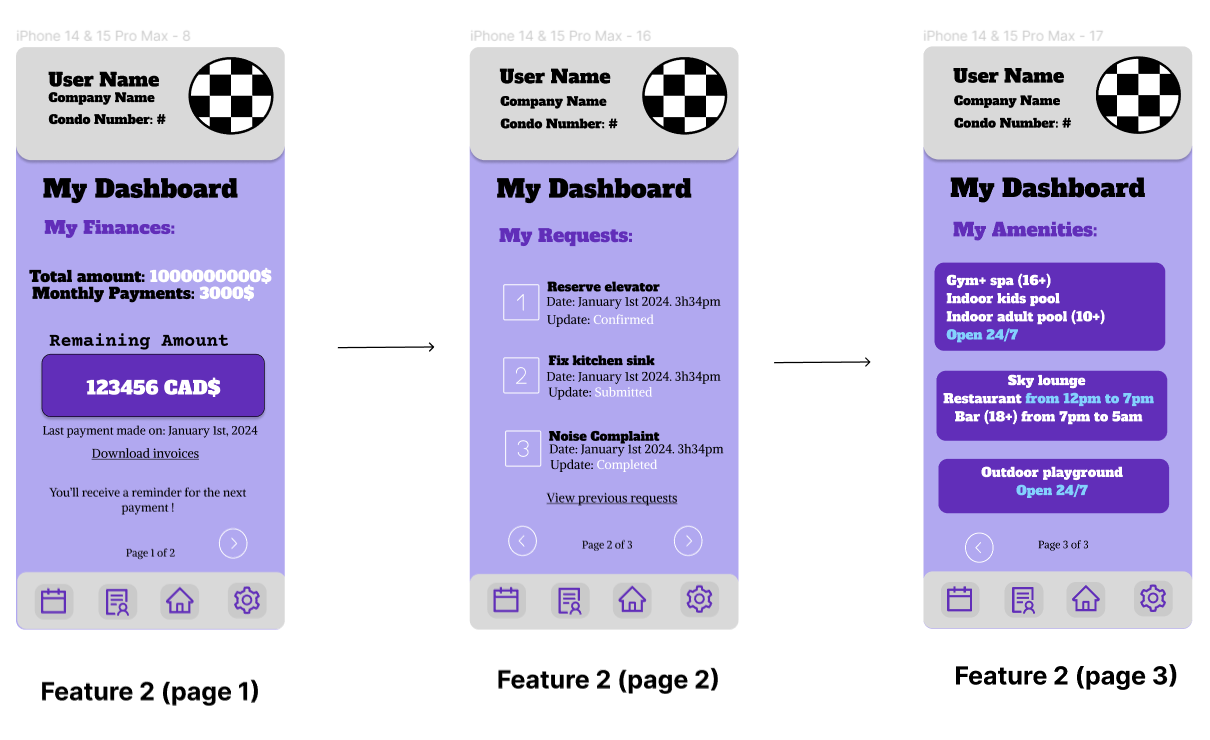
As a condo owner, I want to check my finances: the required monthly payments, download the past invoices and check any additional fees

- US-2.2

As a condo owner, I want to see the updates concerning my submitted requests

- US-2.3

As a condo owner, I want to see the available amenities and their opening hours



**Condo dashboard combining user stories 2.1, 2.2, 2.3**

Steps:

1- To land on the first page, the user is either new to the app and has just inputted the registration key, or the user is not new to the app and has clicked on the home button on the footer.

2- On the first page, the user can see the total price of the condo, the monthly payments due, and the remaining amount from the total price of the house. The user can also download invoices from past payments by clicking on the “download invoices” link, and they can see when the date of the last payment was made. Finally, the user will receive a reminder of the next due payment. To navigate to the next page, the user clicks on the white arrow on the bottom right of the page.

3- On the second page, the user can see all the submitted requests, the updates concerning each one, as well as the date each one was submitted. If the user has made more than 3 requests, they can click on “view previous requests” to see the rest of the requests.

4- The user can go back to the previous page using the left white arrow at the bottom of the page, or they can go to the next page by using the right white arrow at the bottom of the page.

5- On the third page, the user can see the amenities that are accessible to them, as well as the opening hours and age restrictions. The user can go back to the previous page by clicking the white arrow on the bottom left of the page.

User Story 3

- US-3.1

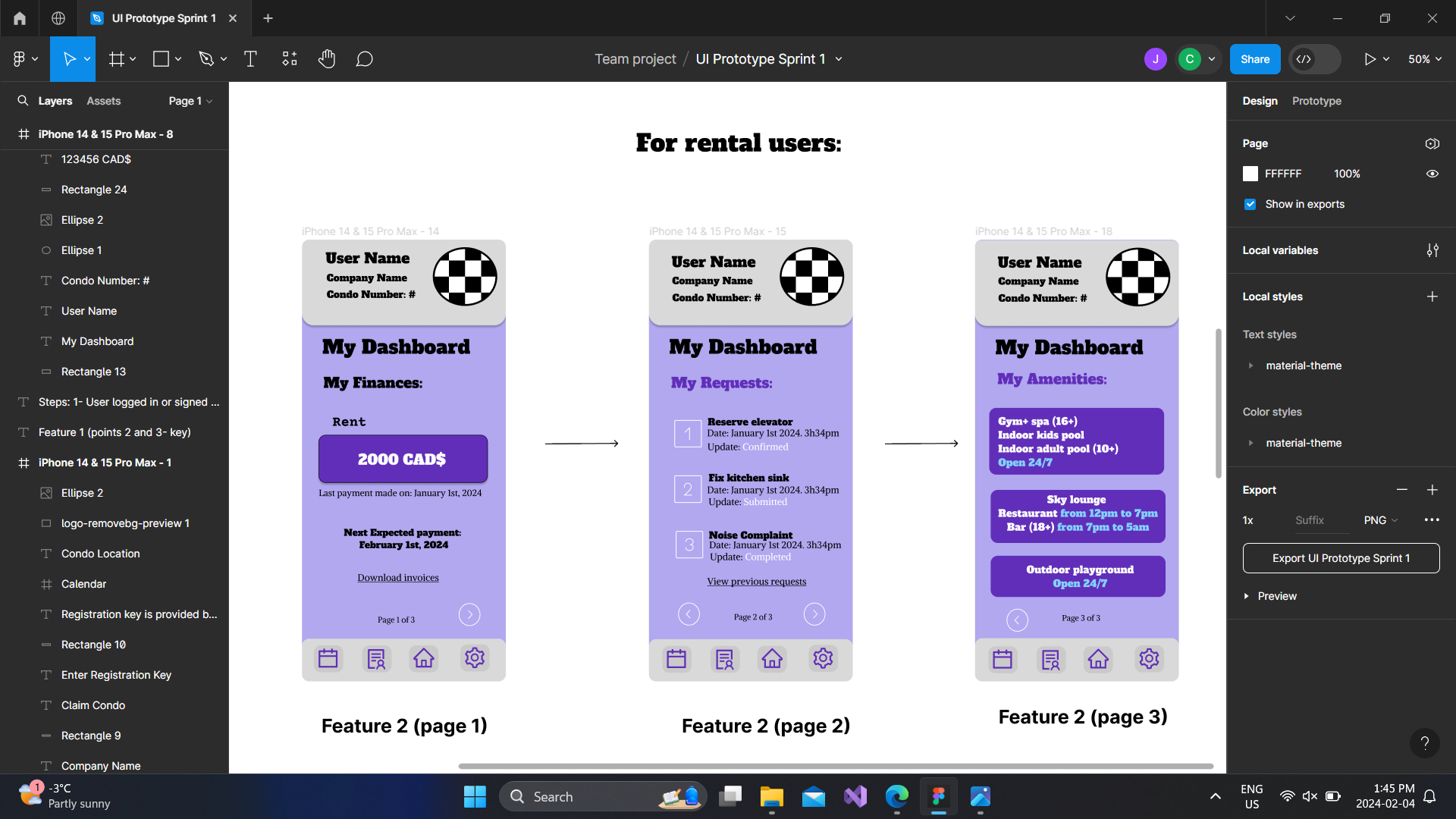
As a rental user, I want to check my finances: the rent, past payments, and the date of the next expected payment (which is on the first of the month).

- US-3.2

As a rental user, I want to see the updates concerning my submitted requests

- US-3.3

As a rental user, I want to see the available amenities and their opening hours



**Condo dashboard combining user stories 3.1, 3.2, 3.3**

**(pages 2 and 3 are identical for condo owners and rental users)**

Steps:

1- To land on the first page, the user is either new to the app and has just input the registration key, or the user is not new to the app and has clicked on the home button on the footer.

2- On the first page, the user can find the rent, the date of the last payment made, and the date of the next expected payment, which is on the first day of the month. The user can also download invoices for past payments by clicking on “download invoices." To navigate to the next page, the user clicks on the white arrow on the bottom right of the page.

3- On the second page, the user can see all the submitted requests, the updates concerning each one, as well as the date each one was submitted. If the user has made more than 3 requests, they can click on “view previous requests” to see the rest of the requests.

4-The user can go back to the previous page using the left white arrow at the bottom of the page, or they can go to the next page by using the right white arrow at the bottom of the page.

5- On the third page, the user can see the amenities that are accessible to them, as well as the opening hours and age restrictions. The user can go back to the previous page by clicking the white arrow on the bottom left of the page.

# 

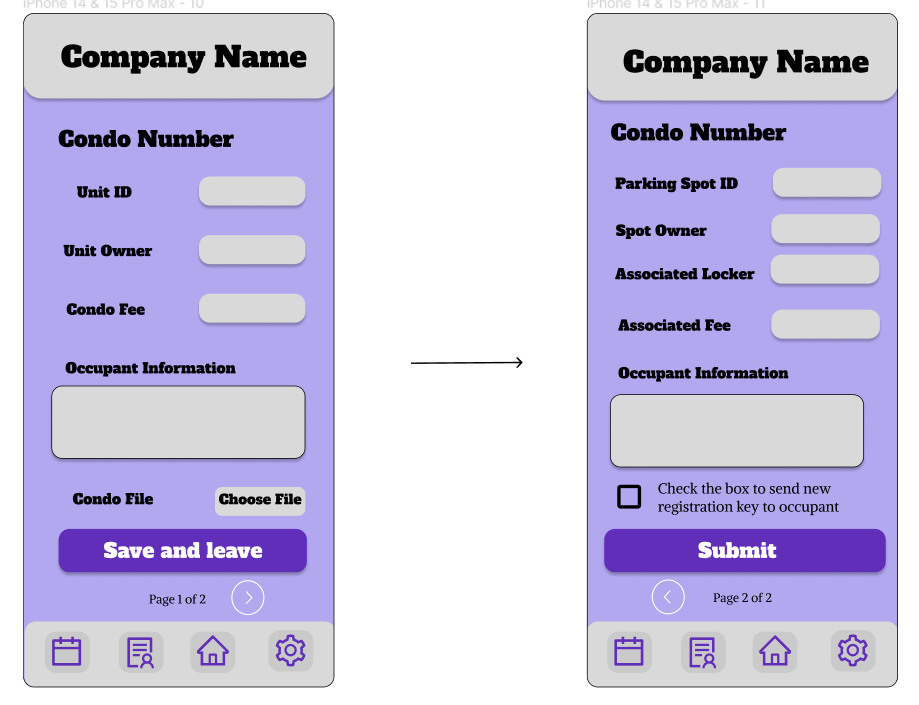
User Story 4

- US-4.1

As a condo management company, I want to be able to register a condo with all its information, from the unit size to the dedicated parking spot.

- US-4.2

As a condo management company, I want to be able to send a registration key to the occupant of the corresponding condo.



**Condo, parking, and locker info combining user stories 4.1, 4.2**

Steps:

1- The user searches for a specific property, if available, or creates a new one.

2- The user adds a file containing information about the condo and fills in the required fields (add or edit).

3- The user clicks on “save and leave” and goes back to the properties page automatically, or the user clicks on the arrow button to go to the next page.

4- Once on the next page, the user has the option to go back to the previous page or press “submit” to save all changes and go back to the properties page.

5- if the user checks the box, the occupant of the corresponding condo will receive a new registration key.

## 

## 9.2 Sprint #3 UI Prototype

User Story 5

As a manager in a condo management company, I want to enter the condo fee per square foot, per parking spot for each unit in a property, so the financial system within the condo management system can generate the condo fee for each unit and present it condo owners, making it easy for them to view their total condo fees.

A screenshot of a test

Description automatically generated

**Manager Input of Fees**

Steps:

1. User inputs the fees per square foot for the condos, and fees per parking spot.
2. User can select a condo ID to test and see the expected total amount.
3. User clicks on “save and leave” and saves the new fees and goes back to home page.

User Story 6

As a manager in a condo management company, I want the financial system to generate an operational budget, so I can keep track of all fees and costs.

A purple and white card

Description automatically generated

**Generation of Operational Budget**

Steps:

1. Users can check the total amount collected from condos.
2. User can check the total amount paid for operations costs.
3. User clicks on the refresh icon to see the latest updates.
4. User can go to the next page to see more details from the annual report.

User Story 7

As a manager in a condo management company, I want the financial system to generate an annual report that describes all condo fees collected in a year, so I can easily view my yearly gross income.

A screenshot of a phone

Description automatically generated

**Generation of Annual Report**

Steps:

1. User selects a year to see the annual report.
2. User can go to the previous page to check the current collected amount and operation costs so far.

# 

User Story 8 and 9

- US-8

As a user who wants to book a room, I would like to be able to reserve additional rooms provided by the condo management in a calendar-like interface so that I may have a better stay.

- US-9

As a user who wants to book a room, I would like for the reservation to be first one come first one serve, so that I can be assured of my reservation.

Screens screenshot of a phone

Description automatically generated

**Room booking and first-come first-serve rooms.**

Steps:

1. User can select the room he wants to book.
2. User can select on the calendar the date he wants.
3. User clicks on the next page to select a time.
4. User clicks on “Book Now” to confirm his booking.

User Story 10

As a user who wants to book a room, I would like the system to show the availability of common facilities, so that I may plan more services or receive better suggestions.

A screenshot of a room available

Description automatically generated

**Generation of Annual Report**

Steps:

1. User can check the availability on the calendar for future bookings. (marked dates are already booked)
2. User can click on “Go to Booking” to reserve a room.

## 

## 

## 9.3 Sprint #4 UI Prototype

User Story 11

As a manager in a condo management company, I want to be able to define different roles for employees, so that I can assign specific responsibilities related to property management.

A screenshot of a phone

Description automatically generated

**Assignment of Roles to Employees Users**

Steps:

1. User choose a specific property.
2. User selects a role for each employee from a list of available roles.
3. User can add an employee to work within a property from available employees.
4. User can add new roles.
5. User clicks on “save and leave” and saves the new roles and newly added employees and goes back to home page.

User Story 12

As a condo owner, I want to submit a request for moving in/out, intercom changes, reporting a violation, reporting a deficiency and/or asking any questions so that necessary action and assistance can be taken by the management. If applicable a corresponding employee needs to be assigned.

A screenshot of a purple and black login form

Description automatically generated

**Request Board**

Steps:

1. User selects a type of request to submit.
2. User writes a message for more clarification.
3. User clicks on submit to send the request and goes back to the home page.

User Story 13

As a user, I want to have access to a notification page where I can see the latest activities submitted or assigned requests.

A screenshot of a phone

Description automatically generated

**Notification Board**

Steps:

1. User can view the latest activities.
2. User can refresh the notification page to view any new activity.
3. User clicks on “Back to HomePage” to go back.

# 

User Story 14

As a user, I want to post and reply to new discussions, so that I can share my thoughts, questions, and experiences within the community.

A screenshot of a chat

Description automatically generated

**Discussion Board**

Steps:

1. User can read and send messages on the discussion board with the rest of the members.
2. User can send voice messages and attachments.
3. User clicks on “Back to HomePage” to go back.

User Story 15

As a user, I want to organize events and invite other occupants to attend, as well as view events and accept/decline invitations so that I can facilitate community engagement and collaboration.

A screenshot of a computer

Description automatically generated

**Event Board**

Steps: part1

1. User can click on each event to see more info.
2. User can accept or reject each event invitation.
3. User can click on Create Even to organize his own event.
4. User click on “Back to HomePage” to go back.

Steps: part2

1. User can select a date on the calendar for the event.
2. User can provide more info about the event.
3. User can enter occupants name and send them invitations.
4. User clicks on “Create Event” to create and save the event.
5. User click on “Back to HomePage” to go back.

User Story 16

As a manager in a condo management company, I want to list coupons/offers that are visible to unit owners or rental users for a specific property, so I can provide exclusive benefits to the community.

A screenshot of a coupon

Description automatically generated

**Offers Board**

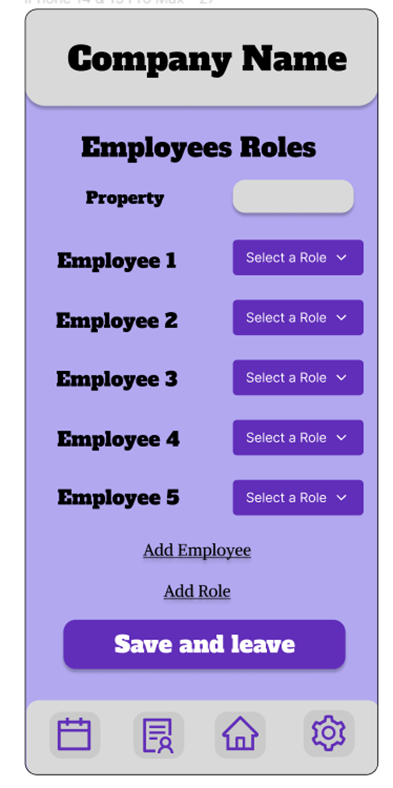
Steps:

1. User can enter the unit ID to reach the owner or the renter.
2. User can enter benefits for each unit ID.
3. User can add extra benefits.
4. User add a message for the owner/renter in addition to the benefits.
5. User can click on the toggle button to enable and disable visibility of the benefits.
6. User click on “Save and Leave” to save the changes and leave the page.

## 9.4 Sprint #5 UI Prototype

User Story 11

As a manager in a condo management company, I want to be able to define different roles for employees, so that I can assign specific responsibilities related to property management.



**Assignment of Roles to Employees Users**

Steps:

1. User choose a specific property.

2. User selects a role for each employee from a list of available roles.

3. User can add an employee to work within a property from available employees.

4. User can add new roles.

5. User clicks on “save and leave” and saves the new roles and newly added employees and goes back to home page.

User Story 14

As a user, I want to post and reply to new discussions, so that I can share my thoughts, questions, and experiences within the community.



**Discussion Board**

Steps:

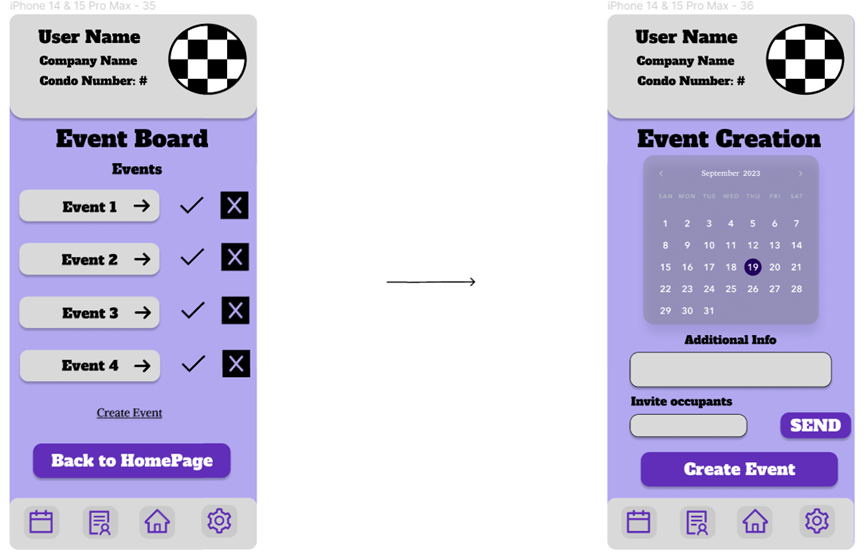
1. User can read and send messages on the discussion board with the rest of the members.

2. User can send voice messages and attachments.

3. User clicks on “Back to HomePage” to go back.

User Story 15

As a user, I want to organize events and invite other occupants to attend, as well as view events and accept/decline invitations so that I can facilitate community engagement and collaboration.



**Event Board**

Steps: part1

1. User can click on each event to see more info.

2. User can accept or reject each event invitation.

3. User can click on Create Even to organize his own event.

4. User click on “Back to HomePage” to go back.

Steps: part2

1. User can select a date on the calendar for the event.

2. User can provide more info about the event.

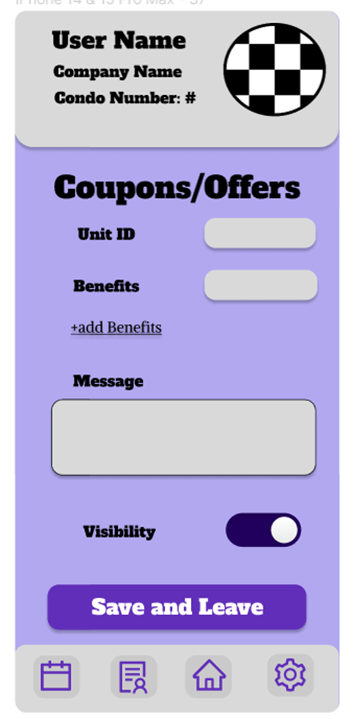
3. User can enter occupants name and send them invitations.

4. User clicks on “Create Event” to create and save the event.

5. User click on “Back to HomePage” to go back.

User Story 16

As a manager in a condo management company, I want to list coupons/offers that are visible to unit owners or rental users for a specific property, so I can provide exclusive benefits to the community.



**Offers Board**

Steps:

1. User can enter the unit ID to reach the owner or the renter.

2. User can enter benefits for each unit ID.

3. User can add extra benefits.

4. User add a message for the owner/renter in addition to the benefits.

5. User can click on the toggle button to enable and disable visibility of the benefits.

6. User click on “Save and Leave” to save the changes and leave the page.

# 

# 10 Testing Plans

## 10.1 Testing Plan for Sprint #2

1. Testing approach

During the development of the system, we will be writing tests at all levels:

1. Unit tests
2. Integration tests
3. System tests
4. Unit testing

Unit testing refers to the smallest testable part of software. The objective of unit testing is to verify that each unit of the software performs as designed.

1. Integration testing

The goal of integration testing is to test multiple components of a system together, focusing on the interactions between them instead of testing the system.

1. System Testing

System testing is a level of software testing that evaluates the complete and integrated software system. The purpose of system testing is to verify that the entire system functions according to the specified requirements.

To execute these tests, we must first create a form of automation. In our case, a Continuous Integration (CI) pipeline will be used to trigger automatically upon new commits or merges to the codebase. This will help fetch the latest code, build the software, run automated tests at various levels (unit, integration, and system), and generate reports on test results and code coverage. Examples of CI tools are Gitlab CI/CD, Azure DevOps, GitHub Actions, Jenkins, and Travis CI.

We will adopt a comprehensive testing approach that includes both automated unit testing and manual acceptance testing. Unit tests will focus on validating individual components and functionalities of the software, while acceptance tests will ensure that the overall system meets the specified requirements and user expectations. Future research will be conducted to decide which tools we will be using for automated and manual testing.

1. Testing tool

Best C# testing frameworks research in Sprint#1:

Adapted from JUnit, NUnit is an open-source unit testing framework for the .NET framework, and it is widely used in C# development. This framework would allow us to write and execute unit tests in a structured and organized manner. It also has fast testing and execution speeds. While NUnit does have documentation and resources available, it may not be as beginner-friendly as other testing frameworks for team members who haven’t used it in past projects.

xUnit.Net is an open-source testing framework that is based on the .NET framework.  It is popular for its intuitive terminology and structure for writing tests, making it easy for developers to understand and use. While xUnit.net has documentation and resources available, it may not be as extensive or well-established as NUnit.

MSTest (Microsoft Unit Testing Framework for .NET) is a unit testing framework developed by Microsoft for testing .NET applications. It is tightly integrated with Visual Studio and offers simplified test creation, execution, and debugging. Since MSTest is included with Visual Studio, there’s no need to install additional packages or frameworks to start writing and executing tests. However, since it is primarily designed for Windows, it has limited cross-platform support. Furthermore, MSTest may be less extensible and more challenging to integrate with other tools or extend its functionality. 

For sprint#2, we will continue using NUnit. NUnit provides seamless integration with the Uno Platform and offers robust features for writing and executing unit tests for our C# codebase.

<https://avatars.githubusercontent.com/u/2678858?s=280&v=4>

1. Testing Metrics and Coverage

We aim to achieve a minimum test coverage of 80% for our codebase by the end of Sprint #2. Test coverage will be measured using Visual Studio's built-in code coverage analysis tools.

We will track the number of defects identified during testing per thousand lines of code (KLOC) to assess the overall quality of our software.

We will monitor the time taken to execute our automated test suite to identify any performance bottlenecks or inefficiencies in our testing process.

1. Acceptance tests

Acceptance tests will be based on the acceptance criteria outlined in the user stories planned for Sprint #2. These criteria define the specific behaviors and functionalities that must be present in the software to meet user expectations. Once these acceptance tests are passed, we will be able to cross out

This testing plan outlines our approach, tools, metrics, and acceptance testing strategy for Sprint #2. By following this plan, we aim to ensure the quality, reliability, and compliance of our software with the specified requirements and user expectations.

## 10.2 Testing Plan for Sprint #3

1. Testing approach

During the development of the system, we will be writing test in all levels:

1. Unit tests
2. Integration tests
3. System tests
4. Unit testing

Unit testing refers to the smallest testable part of a software. The objective of unit testing is to verify that each unit of the software performs as designed.

1. Integration testing

The goal of integration testing is to test multiple components of a system together, focusing on the interactions between them instead of testing the system.

1. System Testing

System testing is a level of software testing that evaluates the complete and integrated software system. The purpose of system testing is to verify that the entire system functions according to the specified requirements.

To execute these tests, we must first create a form of automation. In our case, a Continuous Integration (CI) pipeline will be used to trigger automatically upon new commits or merges to the codebase. This will help fetch the latest code, builds the software, run automated tests at various levels (unit, integration, system), and generate reports on test results and code coverage. Examples of CI tools are Gitlab CI/CD, Azure DevOps, Github Actions, Jenkins, and Travis CI.

We will adopt a comprehensive testing approach that includes both automated unit testing and manual acceptance testing. Unit tests will focus on validating individual components and functionalities of the software, while acceptance tests will ensure that the overall system meets the specified requirements and user expectations. To determine which tool suits our project the best, we will be trying automated unit testing and manual acceptance testing on Travis CI, Jenkins and Github Actions in sprint #3.

1. Testing tool

Best C# testing frameworks research (conducted in Sprint#1):

Adapted from JUnit, NUnit is an open-source unit testing framework for the .NET framework, and it is widely used in C# development. This framework would allow us to write and execute unit tests in a structured and organized manner. It also has fast testing execution speeds. While NUnit does have documentation and resources available, it may not be as beginner friendly as other other testing frameworks for team members that haven’t used it in past projects.

xUnit.Net is an open-source testing framework that is based on the .NET framework.  It is popular for its intuitive terminology and structure for writing test, making it easy for developers to understand and use. While xUnit.net has documentation and resources available, it may not be as extensive or well-established as NUnit.

MSTest (Microsoft Unit testing framework for .NET) is a unit testing framework developed by Microsoft for testing .NET applications. It is tightly integrated with Visual Studio and offers simplified test creation, execution and debugging. Since MSTest is included with Visual Studio, there’s no need to install additional packages or frameworks to start writing and executing tests. However, since it is primarily designed for Windows, it has limited cross-platform support. Furthermore, MSTest may be less extensible and more challenging to integrate with other tools or extend its functionality.

Choice of testing frameworks

The testing for the backend specifically for sprint #2 was done using xUnit.net for multiple reasons: A black and white logo

Description automatically generated

1. Integration with Visual Studio:

xUnit.Net seamlessly integrates with Visual Studio, providing a familiar and efficient testing environment for team members. This integration streamlines the testing workflow, allowing us to write, run, and debug tests within the same IDE we use for development.

1. Rich Feature Set:

xUnit.Net offers a rich feature set for unit testing, including support for parameterized tests, test categorization, and extensibility through custom assertions and test runners. These features would allow us to write expressive and comprehensive tests that a multitude of scenarios.

1. Clear and Readable Syntax:

The xUnit.Net framework provides a clear and readable syntax for writing tests, making it easier for developers to understand the test cases, identify failures, and maintain tests over time. This clarity enhances the maintainability and comprehensibility of the test suite, facilitating collaboration among our team.

xUnit.Net will continue to be used to ensure the quality and reliability of backend code during future sprints.

As for the front-end, unit testing has been applied to all pages.

1. Testing Metrics and Coverage

We aim to achieve 80% code coverage for both front-end and back-end the end of Sprint #3. Test coverage will be measured using Visual Studio's built-in code coverage analysis tools. In addition to test coverage, we will track the following metrics to assess the overall quality of our software:

1. Code Duplication:

We will monitor the percentage of duplicated code within our database using tools such as Visual Studio’s Code Clone Analysis. By minimizing code duplication, we aim to improve maintainability and reduce the risk of introducing bugs.

1. Cyclomatic Complexity:

We will measure the cyclomatic complexity of our methods and classes using a tool called ReSharper. High cyclomatic complexity can indicate code that is difficult to understand and maintain, increasing the likelihood of defects.

1. Coupling:

We will assess the level of coupling between modules and components in our system using a static code analysis tool like ReSharper. High coupling can lead to increased dependencies and fragility, making our codebase more prone to defects and harder to refactor.

In addition to these metrics, we will also monitor the following aspects of our testing process:

1. Defect Density:

We will track the number of defects identified during testing per thousand lines of code (KLOC) to measure the effectiveness of our testing efforts and identify areas of the codebase that require additional testing coverage.

1. Test Execution Time:

We will monitor the time taken to execute our automated test suite using xUnit.Net. By identifying any performance inefficiencies in our testing process, we can optimize our testing strategy for the following sprints to ensure timely feedback on code changes.

1. Acceptance tests

For sprint #3, acceptance tests will be based on the acceptance criteria outlined in the user stories that were updated during sprint #2. These criteria define the specific behaviors and functionalities that must be present in the software to meet user expectations. Once these acceptance tests are passed, we will be able to confidently assert that the software meets the defined requirements and is ready for release.

To ensure consistency and clarity in our acceptance test approach, we will be using the following user acceptance testing template offered by Jira, a platform significantly used by our team to track our progress throughout the development cycle.

1. Summary: A brief overview of the test scenario or objective.
2. Description: Detailed information about the test scenario, including relevant context or background.
3. Test steps: Sequential steps to be executed to verify the functionality or behavior under test.
4. Test Case ID: A unique identifier for tracking and referencing the test case.
5. Status: Indicates whether the test case is currently in progress, completed, or failed.
6. Expected Results: Expected outcomes or behaviors that indicate the successful completion of the test.
7. Actual Results: The actual outcomes observed during test execution, which will be compared against the expected results to determine the test outcome.

In addition to these components, we will be adding a component of our own:

1. Pass/Fail Criteria: Criteria for determining whether the test has passed or failed based on the comparison of actual and expected results.

By following this template, we ensure comprehensive coverage of all relevant aspects of the testing process, facilitating clear communication and documentation of test scenarios and outcomes.

Conclusion

This testing plan outlines our approach, tools, metrics, and acceptance testing strategy for sprint #3. By following this plan, we aim to ensure the quality, reliability, and compliance of our software with the specified requirements and user expectations.

## 10.3 Testing Plan for Sprint #4

1. Testing approach

During the development of the system, we will be writing test in all levels:

1. Unit tests
2. Integration tests
3. System tests
4. Unit testing

Unit testing refers to the smallest testable part of a software. The objective of unit testing is to verify that each unit of the software performs as designed.

1. Integration testing

The goal of integration testing is to test multiple components of a system together, focusing on the interactions between them instead of testing the system.

1. System Testing

System testing is a level of software testing that evaluates the complete and integrated software system. The purpose of system testing is to verify that the entire system functions according to the specified requirements.

To execute these tests, we must first create a form of automation. In our case, a Continuous Integration (CI) pipeline will be used to trigger automatically upon new commits or merges to the codebase. This will help fetch the latest code, builds the software, run automated tests at various levels (unit, integration, system), and generate reports on test results and code coverage. Examples of CI tools are Gitlab CI/CD, Azure DevOps, Github Actions, Jenkins, and Travis CI.

We will adopt a comprehensive testing approach that includes both automated unit testing and manual acceptance testing. Unit tests will focus on validating individual components and functionalities of the software, while acceptance tests will ensure that the overall system meets the specified requirements and user expectations. To determine which tool suits our project the best, we will be trying automated unit testing and manual acceptance testing on Travis CI, Jenkins and Github Actions in sprint #4.

1. Testing tool

Best C# testing frameworks research (conducted in Sprint#1):

Adapted from JUnit, NUnit is an open-source unit testing framework for the .NET framework, and it is widely used in C# development. This framework would allow us to write and execute unit tests in a structured and organized manner. It also has fast testing execution speeds. While NUnit does have documentation and resources available, it may not be as beginner friendly as other other testing frameworks for team members that haven’t used it in past projects.

xUnit.Net is an open-source testing framework that is based on the .NET framework.  It is popular for its intuitive terminology and structure for writing test, making it easy for developers to understand and use. While xUnit.net has documentation and resources available, it may not be as extensive or well-established as NUnit.

MSTest (Microsoft Unit testing framework for .NET) is a unit testing framework developed by Microsoft for testing .NET applications. It is tightly integrated with Visual Studio and offers simplified test creation, execution and debugging. Since MSTest is included with Visual Studio, there’s no need to install additional packages or frameworks to start writing and executing tests. However, since it is primarily designed for Windows, it has limited cross-platform support. Furthermore, MSTest may be less extensible and more challenging to integrate with other tools or extend its functionality.

Choice of testing frameworks

The testing for the backend specifically for sprint #2 was done using xUnit.net for multiple reasons: A black and white logo

Description automatically generated

1. Integration with Visual Studio:

xUnit.Net seamlessly integrates with Visual Studio, providing a familiar and efficient testing environment for team members. This integration streamlines the testing workflow, allowing us to write, run, and debug tests within the same IDE we use for development.

1. Rich Feature Set:

xUnit.Net offers a rich feature set for unit testing, including support for parameterized tests, test categorization, and extensibility through custom assertions and test runners. These features would allow us to write expressive and comprehensive tests that a multitude of scenarios.

1. Clear and Readable Syntax:

The xUnit.Net framework provides a clear and readable syntax for writing tests, making it easier for developers to understand the test cases, identify failures, and maintain tests over time. This clarity enhances the maintainability and comprehensibility of the test suite, facilitating collaboration among our team.

xUnit.Net will continue to be used to ensure the quality and reliability of backend code during future sprints.

As for the front-end, unit testing hasn’t been applied yet but will most definitely be applied in sprint 4.

1. Testing Metrics and Coverage

we have achieved 85% code coverage for the back end during sprint 3, and we aim to achieve at least 80% during sprint 4.

Test coverage will be measured using Visual Studio's built-in code coverage analysis tools. In addition to test coverage, we will track the following metrics to assess the overall quality of our software:

1. Code Duplication:

We will monitor the percentage of duplicated code within our database using tools such as Visual Studio’s Code Clone Analysis. By minimizing code duplication, we aim to improve maintainability and reduce the risk of introducing bugs.

1. Cyclomatic Complexity:

We will measure the cyclomatic complexity of our methods and classes using a tool called ReSharper. High cyclomatic complexity can indicate code that is difficult to understand and maintain, increasing the likelihood of defects.

1. Coupling:

We will assess the level of coupling between modules and components in our system using a static code analysis tool like ReSharper. High coupling can lead to increased dependencies and fragility, making our codebase more prone to defects and harder to refactor.

In addition to these metrics, we will also monitor the following aspects of our testing process:

1. Defect Density:

We will track the number of defects identified during testing per thousand lines of code (KLOC) to measure the effectiveness of our testing efforts and identify areas of the codebase that require additional testing coverage.

1. Test Execution Time:

We will monitor the time taken to execute our automated test suite using xUnit.Net. By identifying any performance inefficiencies in our testing process, we can optimize our testing strategy for the following sprints to ensure timely feedback on code changes.

1. Acceptance tests

For sprint #4, acceptance tests will be based on the acceptance criteria outlined in the user stories that were updated during sprint #3. These criteria define the specific behaviors and functionalities that must be present in the software to meet user expectations. Once these acceptance tests are passed, we will be able to confidently assert that the software meets the defined requirements and is ready for release. To ensure consistency and clarity in our acceptance test approach, we will be using the following user acceptance testing template offered by Jira, a platform significantly used by our team to track our progress throughout the development cycle. Here is an example:

1. Summary: Creating a new user through the SoftCondo Application UI
2. Description: A new user should be able to click on a Sign Up button and fill out the necessary fields to create a new account. Having done so, the user clicks on “Submit” button that adds the new account to the database and logs the user into this new account, redirecting to the user profile page.
3. Test steps:

1- From the main page click on “Sign up”

2- Fill out the following fields:

-First name

-Last name

-Email

-Password

-Confirm password

3- Click on “Sign Up” button

4- The page redirects the user to the new account’s profile page

1. Test Case ID: AT-1.
2. Status: Passing
3. Expected Results:

-The new account is saved to the database

-The user is redirected to the user profile page

1. Actual Results:

-The new account is saved to the database

-The user is redirected to the user profile page

In addition to these components, we will be adding a component of our own:

1. Pass/Fail Criteria:

Pass: only if expected results are equal to actual results

Fail: any other outcome

By following this template, we ensure comprehensive coverage of all relevant aspects of the testing process, facilitating clear communication and documentation of test scenarios and outcomes.

Conclusion

This testing plan outlines our approach, tools, metrics, and acceptance testing strategy for sprint #4. By following this plan, we aim to ensure the quality, reliability, and compliance of our software with the specified requirements and user expectations.

## 10.4 Testing Plan for Sprint #5

Testing approach

During the development of the system, we will be writing test in all levels:

1. Unit testing

Unit testing refers to the smallest testable part of a software. The objective of unit testing is to verify that each unit of the software performs as designed.

1. Integration testing

The goal of integration testing is to test multiple components of a system together, focusing on the interactions between them instead of testing the system.

1. System Testing

System testing is a level of software testing that evaluates the complete and integrated software system. The purpose of system testing is to verify that the entire system functions according to the specified requirements.

To execute these tests, we must first create a form of automation. In our case, a Continuous Integration (CI) pipeline will be used to trigger automatically upon new commits or merges to the codebase. This will help fetch the latest code, builds the software, run automated tests at various levels (unit, integration, system), and generate reports on test results and code coverage. Examples of CI tools are Gitlab CI/CD, Azure DevOps, Github Actions, Jenkins, and Travis CI.

We will adopt a comprehensive testing approach that includes both automated unit testing and manual acceptance testing. Unit tests will focus on validating individual components and functionalities of the software, while acceptance tests will ensure that the overall system meets the specified requirements and user expectations. To determine which tool suits our project the best, we will be trying automated unit testing and manual acceptance testing on Travis CI, Jenkins and Github Actions in sprint #5.

Testing tool

Best C# testing frameworks research (conducted in Sprint#1):

Adapted from JUnit, NUnit is an open-source unit testing framework for the .NET framework, and it is widely used in C# development. This framework would allow us to write and execute unit tests in a structured and organized manner. It also has fast testing execution speeds. While NUnit does have documentation and resources available, it may not be as beginner friendly as other other testing frameworks for team members that haven’t used it in past projects.

xUnit.Net is an open-source testing framework that is based on the .NET framework.  It is popular for its intuitive terminology and structure for writing test, making it easy for developers to understand and use. While xUnit.net has documentation and resources available, it may not be as extensive or well-established as NUnit.

MSTest (Microsoft Unit testing framework for .NET) is a unit testing framework developed by Microsoft for testing .NET applications. It is tightly integrated with Visual Studio and offers simplified test creation, execution and debugging. Since MSTest is included with Visual Studio, there’s no need to install additional packages or frameworks to start writing and executing tests. However, since it is primarily designed for Windows, it has limited cross-platform support. Furthermore, MSTest may be less extensible and more challenging to integrate with other tools or extend its functionality.

Choice of testing frameworks

The testing for the backend specifically for all sprints was done using xUnit.net for multiple reasons: A black and white logo

Description automatically generated

1. **Integration with Visual Studio:**

xUnit.Net seamlessly integrates with Visual Studio, providing a familiar and efficient testing environment for team members. This integration streamlines the testing workflow, allowing us to write, run, and debug tests within the same IDE we use for development.

1. **Rich Feature Set:**

xUnit.Net offers a rich feature set for unit testing, including support for parameterized tests, test categorization, and extensibility through custom assertions and test runners. These features would allow us to write expressive and comprehensive tests that a multitude of scenarios.

1. **Clear and Readable Syntax:**

The xUnit.Net framework provides a clear and readable syntax for writing tests, making it easier for developers to understand the test cases, identify failures, and maintain tests over time. This clarity enhances the maintainability and comprehensibility of the test suite, facilitating collaboration among our team.

xUnit.Net will continue to be used to ensure the quality and reliability of backend code during future sprints.

Testing Metrics and Coverage

In all our sprints, we aim to achieve at least 80%. Test coverage is measured using Visual Studio's built-in code coverage analysis tools. In addition to test coverage, we will track the following metrics to assess the overall quality of our software:

1. **Code Duplication:**

We will monitor the percentage of duplicated code within our database using tools such as Visual Studio’s Code Clone Analysis. By minimizing code duplication, we aim to improve maintainability and reduce the risk of introducing bugs.

1. **Cyclomatic Complexity:**

We will measure the cyclomatic complexity of our methods and classes using a tool called ReSharper. High cyclomatic complexity can indicate code that is difficult to understand and maintain, increasing the likelihood of defects.

1. **Coupling:**

We will assess the level of coupling between modules and components in our system using a static code analysis tool like ReSharper. High coupling can lead to increased dependencies and fragility, making our codebase more prone to defects and harder to refactor.

In addition to these metrics, we will also monitor the following aspects of our testing process:

1. **Defect Density:**

We will track the number of defects identified during testing per thousand lines of code (KLOC) to measure the effectiveness of our testing efforts and identify areas of the codebase that require additional testing coverage.

1. **Test Execution Time:**

We will monitor the time taken to execute our automated test suite using xUnit.Net. By identifying any performance inefficiencies in our testing process, we can optimize our testing strategy for the following sprints to ensure timely feedback on code changes.

Acceptance tests

For sprint #5, acceptance tests will be based on the acceptance criteria outlined in the user stories that were updated during sprint #4. These criteria define the specific behaviors and functionalities that must be present in the software to meet user expectations. Once these acceptance tests are passed, we will be able to confidently assert that the software meets the defined requirements and is ready for release. To ensure consistency and clarity in our acceptance test approach, we will be using the following user acceptance testing template offered by Jira, a platform significantly used by our team to track our progress throughout the development cycle. Here is an example:

AT#2- CMS-80: Manager input of fees

| Related User Stories | CMS-81: Generation of Operational Budget  CMS-82: Generation of Annual Report |
| --- | --- |
| Related System Tests | N/A |
| Acceptance Criteria | 1. User logs in to the condo management system as an authorized employee 2. User is redirected to his Company Dashboard 3. User clicks on the desired section on the Company Dashboard 4. User selects the specific unit for which fee rates need to be entered 5. User enters condo fee per square foot and per parking spot for a selected unit 6. User clicks on the ‘Generate Condo Fee’ button 7. The page displays the entered condo fee for the unit that will also be displayed for condo owners |
| Label | Closed |
| Signed Off |  |

AT#3- CMS-81: Generation of Operational Budget

| Related User Stories | CMS-80: CMS-80: Manager input of fees  CMS-82: Generation of Annual Report |
| --- | --- |
| Related System Tests | N/A |
| Acceptance Criteria | 1. User logs in to the condo management system as an authorized employee 2. User is redirected to the Company Dashboard 3. User clicks on the section ‘Finances’ on the Company Dashboard User 4. User is redirected to a page showing the overview of the ‘Financial System’ 5. The page displays all the fees and costs the manager needs to keep track of |
| Label | Closed |
| Signed Off |  |

AT#3- CMS-82: Generation of Annual Report

| Related User Stories | CMS-80: CMS-80: Manager input of fees  CMS-81: Generation of Operational Budget |
| --- | --- |
| Related System Tests | N/A |
| Acceptance Criteria | 1. User logs in to the condo management system as an authorized employee 2. User is redirected to the Company Dashboard 3. User clicks on the section ‘Finances’ on the Company Dashboard 4. User is redirected to a page showing the overview of the ‘Financial System’ 5. The page displays all the fees and costs the manager needs to keep track of 6. User selects which year he wishes to have an annual report for 7. User clicks on the ‘Generate Annual Report’ button 8. A popup window alerts the user that the annual report was downloaded successfully |
| Label | Closed |
| Signed Off |  |

By following this template, we ensure comprehensive coverage of all relevant aspects of the testing process, facilitating clear communication and documentation of test scenarios and outcomes.

Conclusion

This testing plan outlines our approach, tools, metrics, and acceptance testing strategy for sprint #5. By following this plan, we aim to ensure the quality, reliability, and compliance of our software with the specified requirements and user expectations.

# 

# 

# 11- Code Management

## 11.1 Software Implementation of planned user stories

| **Phase 2** | [CMS-47](https://soen390-cms.atlassian.net/browse/CMS-47?atlOrigin=eyJpIjoiODBhNTE0ODAxOWUzNDU5OWFiZGIxN2I5OGMwYjA3OTEiLCJwIjoiaiJ9): User Profile Creation  Start date: 2024/01/29  End date: 2024/02/28  Effort: 5  [CMS-49](https://soen390-cms.atlassian.net/browse/CMS-49?atlOrigin=eyJpIjoiY2UyZDk5YzU4ZjdmNDhiNGJjZTg1NjBhMThhOTJjOTYiLCJwIjoiaiJ9): Assignment of Registration Keys  Start date:2024/02/12  End date: 2024/03/04  Effort: 2  [CMS-48](https://soen390-cms.atlassian.net/browse/CMS-48?atlOrigin=eyJpIjoiMjUzYzI0OTAzMmU4NGRlMGJmNmRiODc5MmJlZDAzZjYiLCJwIjoiaiJ9): Property Profile Creation  Start date: 2024/02/12  End date: 2024/03/04  Effort: 4  [CMS-52](https://soen390-cms.atlassian.net/browse/CMS-52?atlOrigin=eyJpIjoiMjI2NjdhYTNkOGQ4NDUyZGI1NmUzYzViNjkxOTUwY2UiLCJwIjoiaiJ9): Condo File Upload  Start date:2024/02/12  End date: 2024/03/04  Effort: 4  [CMS-53](https://soen390-cms.atlassian.net/browse/CMS-53?atlOrigin=eyJpIjoiMWM3OWJmNjdmODE2NGVhOWI3NzgyYmIwZDE0NjgwZDMiLCJwIjoiaiJ9): Condo Unit Management  Start date:2024/02/12  End date: 2024/03/04  Effort: 4  [CMS-78](https://soen390-cms.atlassian.net/browse/CMS-78?atlOrigin=eyJpIjoiNmI1ZjU4ODBjMmE2NDM5Mzg0YmRmYjgzMjJjNzI0N2IiLCJwIjoiaiJ9): Adding Parking Spots and Lockers to a Property Profile  Start date:2024/02/12  End date: 2024/03/04  Effort: 3  [CMS-79](https://soen390-cms.atlassian.net/browse/CMS-79?atlOrigin=eyJpIjoiZjc3N2U3MTlkYzFmNGEzMGIzNWEyODRkNDE0MTk3NGYiLCJwIjoiaiJ9): Send Registration Keys from Condo Unit to Owner/Renter  Start date:2024/02/12  End date: 2024/03/04  Effort: 2    [CMS-51](https://soen390-cms.atlassian.net/browse/CMS-51?atlOrigin=eyJpIjoiMDg0MjdmOWFkNTkyNDZkZjg3YWE4YjBiNjQ2ODQyMmUiLCJwIjoiaiJ9): Condo owner dashboard  Start date:2024/02/12  End date: 2024/03/04  Effort: 6  Total Effort: 30 |
| --- | --- |
| **Phase 3** | No new features completed, only unit tests implemented |
| **Phase 4** | [CMS-110](https://soen390-cms.atlassian.net/browse/CMS-110): User Roles  Start date:2024/03/22  End date:2024/04/12  Effort: 3  [CMS-80](https://soen390-cms.atlassian.net/browse/CMS-80?atlOrigin=eyJpIjoiNjlkZGVhODhkN2YxNGI4ZDk5MjEwY2IwNDczZmY4YmQiLCJwIjoiaiJ9): Manager input of fees  Start date:2024/03/22  End date:2024/04/12  Effort: 5  [CMS-81](https://soen390-cms.atlassian.net/browse/CMS-81?atlOrigin=eyJpIjoiMTVlYjdmNDNkMjJjNDI4M2IyZjczYmZkOGRkZDg0NGIiLCJwIjoiaiJ9): Generation of Operational Budget  Start date:2024/03/22  End date:2024/04/12  Effort: 5  [CMS-82](https://soen390-cms.atlassian.net/browse/CMS-82?atlOrigin=eyJpIjoiZjE2YWVmNGVhYTA0NGZiODk0MjdiOTkwMWI4Yjc4YTkiLCJwIjoiaiJ9): Generation of Annual Report  Start date:2024/03/22  End date: 2024/04/12  Effort: 4  [CMS-83](https://soen390-cms.atlassian.net/browse/CMS-83?atlOrigin=eyJpIjoiMTg5NDg4ODQ3ODYzNDA2Y2I1MzdiYmM4ODRmYzkxZWYiLCJwIjoiaiJ9): Room Booking  Start date:2024/03/22  End date: 2024/04/12  Effort: 6  Total Effort: 23 |
| **Phase 5** | [CMS-84](https://soen390-cms.atlassian.net/browse/CMS-84?atlOrigin=eyJpIjoiNTg2MzU5NWVmNmQ2NDA4MTkwMzdlNDlmMTVjMTNiN2EiLCJwIjoiaiJ9): Room Availability  Start date:2024/03/22  End date: 2024/05/01  Effort: 5  [CMS-85](https://soen390-cms.atlassian.net/browse/CMS-85?atlOrigin=eyJpIjoiYjNmMTBkZGI0ZjRkNGFjNzg4NWE3YTU2NGY0MDhhYWQiLCJwIjoiaiJ9): Rooms first-come-first-serve  Start date:2024/03/22  End date: 2024/05/01  Effort: 3  [CMS-87](https://soen390-cms.atlassian.net/browse/CMS-87?atlOrigin=eyJpIjoiNGY2ZjFiMmU3YTRiNGJhY2EyMjlmM2Y2MGIwNWRkZDEiLCJwIjoiaiJ9) Request board  Start date: 2024/03/22  End date: 2024/05/01  Effort: 6  [CMS-103](https://soen390-cms.atlassian.net/browse/CMS-103?atlOrigin=eyJpIjoiZGI0NDJjZDBhNTg5NDdhNzk0YzhlYzg5MmQ4Mzk5ZDYiLCJwIjoiaiJ9) Notification Board  Start date: 2024/04/12  End date: 2024/05/01  Effort: 5  [CMS-86](https://soen390-cms.atlassian.net/browse/CMS-86?atlOrigin=eyJpIjoiYThhNWYzZTA0NDAwNGViN2IzMDI1ZGZhOTlhODY5ZDIiLCJwIjoiaiJ9) Assignment of Roles to Employee Users - **Optional**  Start date: 2024/03/22  End date: 2024/05/01  Effort: 4  [CMS-87](https://soen390-cms.atlassian.net/browse/CMS-87?atlOrigin=eyJpIjoiNGY2ZjFiMmU3YTRiNGJhY2EyMjlmM2Y2MGIwNWRkZDEiLCJwIjoiaiJ9) Request board  Start date: 2024/04/12  End date: 2024/05/01  Effort: 6  Total Effort: 29 |

* The total project effort is 76
* Estimated team velocity is 15 points per iteration

## 11.2 Bug reports/fixing

| **Bug ID** | 112 | | |
| --- | --- | --- | --- |
| **Originator** | HICHAM KITAZ | Email:hishamo-hhk@hotmail.com | Signature: hisham-kitaz |
| **Submit Date** | February 25, 2024 | | |
| **Summary** | Data fields go blank after saving edit | | |
| **Severity** | minor | | |
| **Product** | Progressive web application. | | |
| **Version** | 1.1 | | |
| **Platform** | [PC] | | |
| **OS** | [Windows] | | |
| **Browser** | chrome | | |

## 

## 11.3 Bug reports fixing techniques

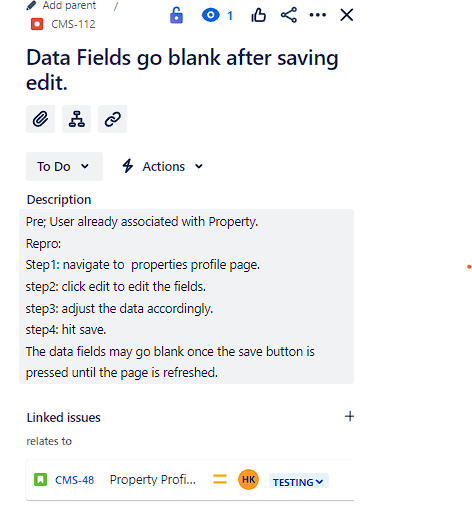
To streamline the defect management process for our condo management system project, we will implement a concise and effective strategy based on the structured approach outlined on the guru99.com website, tailored to our specific needs. Here's how we will manage and fix bugs throughout our project lifecycle:

* Discovery: Our project team will proactively identify defects through thorough testing before they become apparent to end users. Defects recognized by developers will be accepted for further action.
* Categorization: Defects will be categorized based on priority (Critical, High, Medium, Low) to help developers prioritize their tasks efficiently. This ensures critical issues that could severely impact user experience are addressed first.
* Resolution: Assigned developers will fix defects based on the assigned priority. This process includes:
  + Assigning the defect to a specific developer or technician.
  + Scheduling the fix according to its priority.
  + Fixing the defect and tracking the progress.
  + Reporting the resolution back to the project management team.
* Verification: Post-resolution, our testing team will verify that defects have been properly fixed and that no new issues have arisen due to the fixes.
* Closure: Once a defect has been resolved and verified, it will be marked as closed. Any unresolved issues will be sent back to the development team for re-evaluation.
* Defect Reporting: We will prepare and share regular defect reports with the management team to provide updates on the defect management process and current defect status. This facilitates better communication, tracking, and management of defects.

By adhering to this structured defect management process, we aim to minimize the impact of defects on our condo management system project, ensuring a high-quality product and satisfaction for our end users.

## 11.4 Defect Tracking Tool for Software Testing

For managing defects and user stories in our software testing process, we have chosen to use JIRA. This decision is rooted in JIRA's comprehensive capabilities as a defect-tracking tool, which offers an intuitive and robust platform for efficiently managing and tracking bugs. JIRA's flexibility allows us to customize workflows, fields, and reporting to suit our specific project needs, making it an ideal choice for our team. Moreover, its integration with a wide range of development tools streamlines the tracking of issues from discovery through to resolution. JIRA's user-friendly interface and powerful features not only enhance our team's productivity but also ensure transparency and effective communication throughout the project lifecycle. By adopting JIRA, we are equipped to maintain high-quality standards in our software development process, ensuring that all issues are addressed promptly and thoroughly.



# 

## 11.5 Code Review Tools

When it comes to reviewing code for our project, there are several methods we employ to ensure quality and collaboration among our team members. Here are some examples:

1. **Interactive Review Sessions:** An approach we frequently use is live code reviews that allow real-time feedback and discussion. This makes it easy to address any issues or questions as they arise instead of postponing them and creating a delay in our development. Although this approach is usually conducted in person, we have adapted this method for our distributed team using tools such as Visual Studio Live Share.
2. **Pair programming Sessions**: Another approach our team has utilized is pair programming sessions where we pair up team members for code reviews. These sessions involved two members working together at a single workstation, with one writing code and the other providing immediate feedback. While this approach can be time-intensive, we’ve found it to be highly effective for knowledge sharing and fostering collaboration within our team.
3. **Utilizing Code Review Tools:** We have also leveraged specialized software tools to streamline the code review process. The main tool our team has been using is GitHub’s integrated code review tool within its pull request feature. This tool allows us to analyze the differences in code, provide inline comments, and track the history of changes.
4. **Annotating source code before the review:** In our progress, we’ve implemented a practice of annotating our source code before initiating the review process. This approach has proven to be valuable in guiding us through different changes made, providing clear pointers to the files that require attention, and offering insights behind the modification. Furthermore, it prompts us to catch and address additional errors before the peer review even begins. This proactive error detection has led to a significant decrease in defect density, as we’re able to solve issues earlier in the development cycle. Overall, the integration of annotations into our review process has contributed to improved code quality, smoother reviews, and a more efficient workflow within our project.

## 11.6 Correct use of design patterns

The need for better cross-platform compatibility, maintainability, and others played a role in the change of approach from Uno Platform to React for the front end of the Condo Management System (CMS). On the Uno platform, though, it mostly brought along maintainability challenges, not to mention smooth operation on all the targeted platforms. On the flip side, React has a strong ecosystem and boasts broad platform support as a Progressive Web App (PWA) with a very strong community. This switch necessitated a reevaluation and modification of the design patterns used in the development process.

### **11.6.1 Modified Design Patterns**

As for the architecture of the front end, the CMS was rather tidy in placing its codebase. This is to standardize the way components, containers, assets, src, and tests are put, mostly to represent very much a normal composition of a React project, hence keeping it at a high level of modularity, reusability, and maintainability.

* **components**: It is the smallest UI unit of the application, constituting the rendering of the visual parts of the application. They should be designed reusable and encapsulated in a modular manner so that even the styling and logic could be reused. Containers: Things that take care of how some other things work, and provide data and behavior to exist for presentational or other container components.
* **assets:** These are directories of files that include images, fonts, global style sheets, and others that form part of the visual and functional aspects of the CMS.
* **src:** The root source folder that will provide the structure for the project, housing all the components, containers, services, and utilities responsible for application logic.
* **tests:** Unit and integration tests for every piece of your application, to make sure they are working with their intended behavior in isolation and in combination with other pieces.

At the same time, this architectural pattern is complemented by the general stylesheet for common styles and the relevant styles to components that live inside the folder of the component. In such a way, it allows the customization of every component separately, but at the same time, the views throughout the application are the same in every separate instance of the component.

### **11.6.2 Backend Architecture**

The backend involves the simplified version of the Model-View-Controller (MVC) architectural framework to pave the way for efficiency and ease of development. It leverages ASP.NET capability for building Web APIs with the following clear separation of concerns.

* **Model**: Defines the data structure and is manipulated by services.
* **View:** In this light, the "view" is simply the JSON response presented by the API consumed by the front end.
* **Controller**: It is an intermediary between frontend requests and backend responses. Every entity has an entity-specific controller that, in turn, calls the appropriate service.
* **Service:** This namespace contains business logic, and Entity Framework Core mediates interaction with the database to be able to abstract the access data layer from the controllers. This architecture reduces the number of classes in the backend structure. It suggests removing the classes such as Data Transfer Objects (DTOs) and the repository class, urging some kind of direct interaction of controllers and services with the database. Dependency injection in the back end causes many dependencies to be taken care of to keep components in a loose-coupling relationship.

### **11.6.3 Design patterns**

In the development of the CMS Api, the Singleton design pattern was applied to ensure the management of system time is centralized through a single SystemTime object. This choice is justified by the need for a consistent and synchronized timekeeping mechanism across the entire backend. The Singleton pattern is crucial in this context because it guarantees that there is only one instance of the SystemTime class, which avoids potential conflicts or discrepancies in time data management, essential for operations such as scheduled balance updates.

# 

# 

# 

# 

Pros of this approach include:

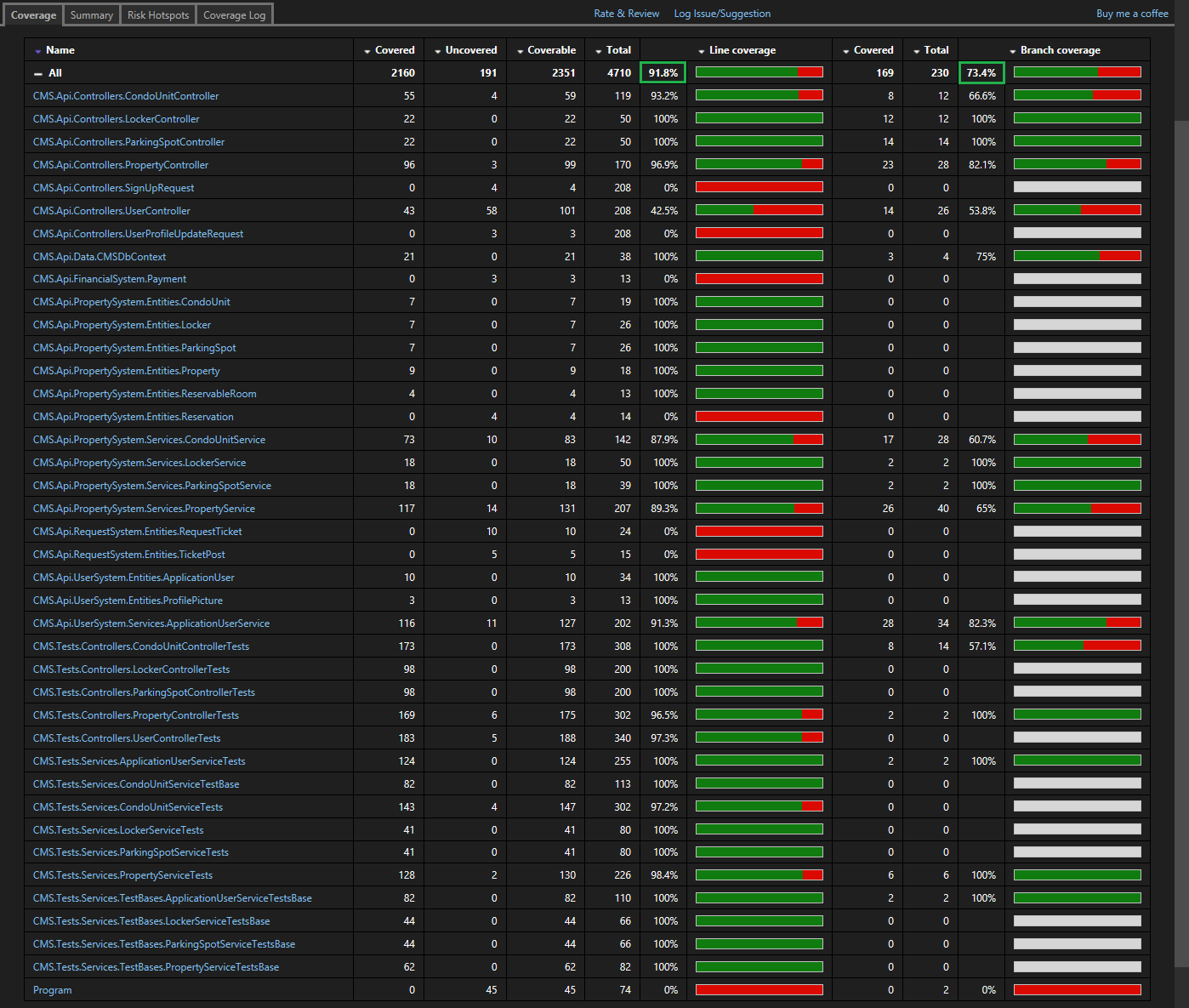
* Consistency: All parts of the system refer to the same instance of the SystemTime, ensuring consistent behavior and time calculations across different modules.
* Resource Efficiency: By maintaining a single instance, the system minimizes resource usage compared to having multiple timekeeping objects.

Cons, however, include:

* Reduced Flexibility: The use of the Singleton pattern can make it harder to modify the timekeeping strategy as the system evolves, as changes to the SystemTime class affect all parts of the application.
* Potential for Tight Coupling: With many parts of the application depending on this single instance, it can lead to tight coupling, which may complicate further development and testing.

# 

## 11.7 Code coverage

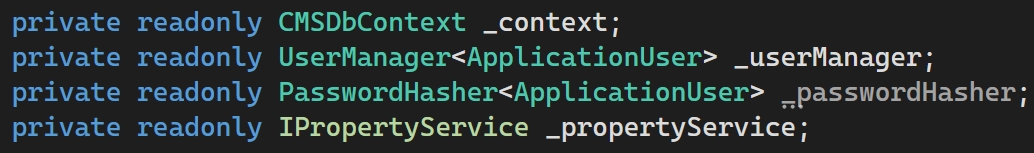
We switched code coverage tools for sprint 3 from the *coverlet* package used previously to the *Fine Code Coverage* extension for Visual Studio. This provides the team with a better visual of which classes have/haven’t been tested and to what degree. Given the shorter duration of this sprint, our focus was purely to catch up on unit testing which was lacking in previous sprints. With this done, our line coverage is now at 91.8%.

## 

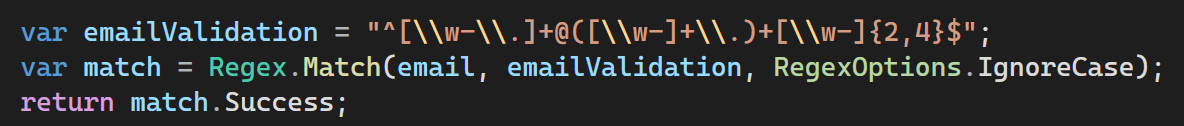
## 11.8 Respect to code conventions

With the majority of the code being in C#, we do our best to adhere to [Microsoft’s C# Code Conventions](https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions) and [naming conventions](https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names). For example, all variables in a defined entity are named using PascalCasing:



In comparison, instances that are using dependency injection begin with an underscore:

While regular instances are in camelCase:



# 

## 11.9 Design quality (number of classes/packages, size, coupling, cohesion)

For assessing and enhancing the design quality of our software—specifically focusing on the number of classes/packages, size, coupling, and cohesion—we opted for Understand by SciTools. Understand is a comprehensive and customizable integrated development environment tailored for static code analysis, equipped with a vast array of visualizations, documentation, and metric tools. Its selection was motivated by the need for an in-depth understanding of our codebase, enabling us to identify and address design flaws systematically. The tool’s capability to generate detailed metrics and visual representations of code structure helps us monitor and improve aspects such as class/package organization, code size, and the relationships between different components, aiming for a well-structured, maintainable, and efficiently decoupled architecture. By leveraging Understand, we can pinpoint areas that require refactoring or optimization, ensuring our code adheres to best practices in software design for enhanced cohesion and reduced coupling, ultimately leading to a more robust and scalable software product.



## 11.10 Quality of source code documentation

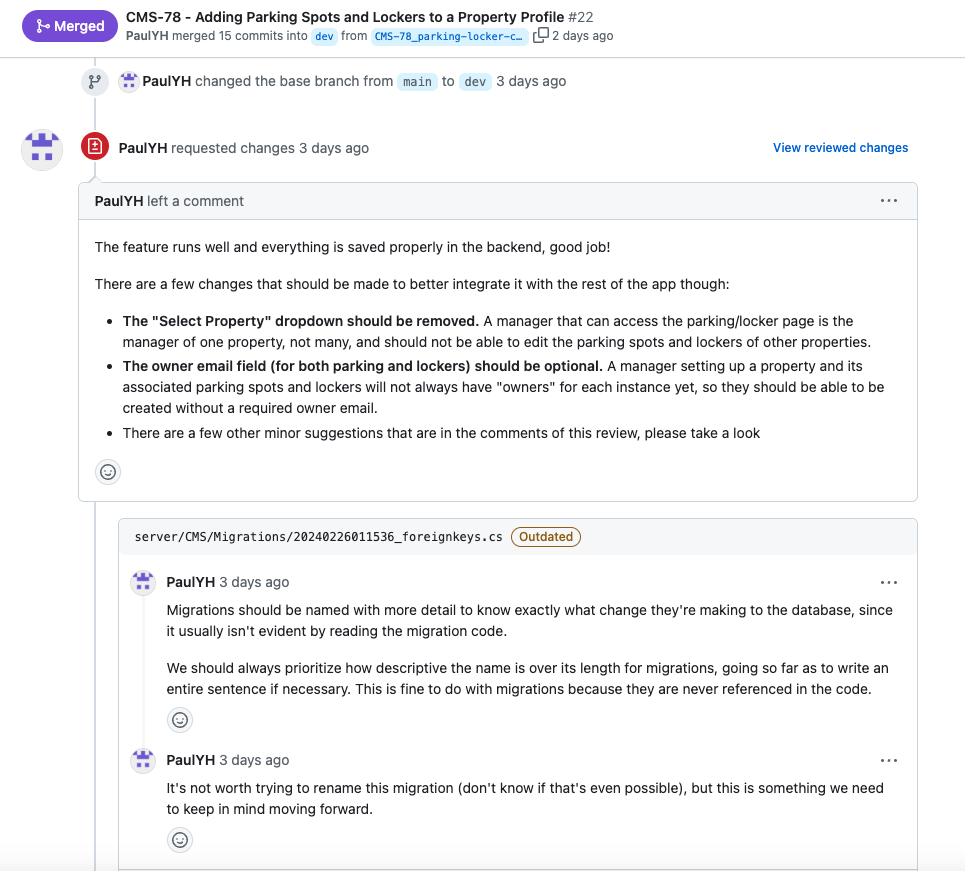
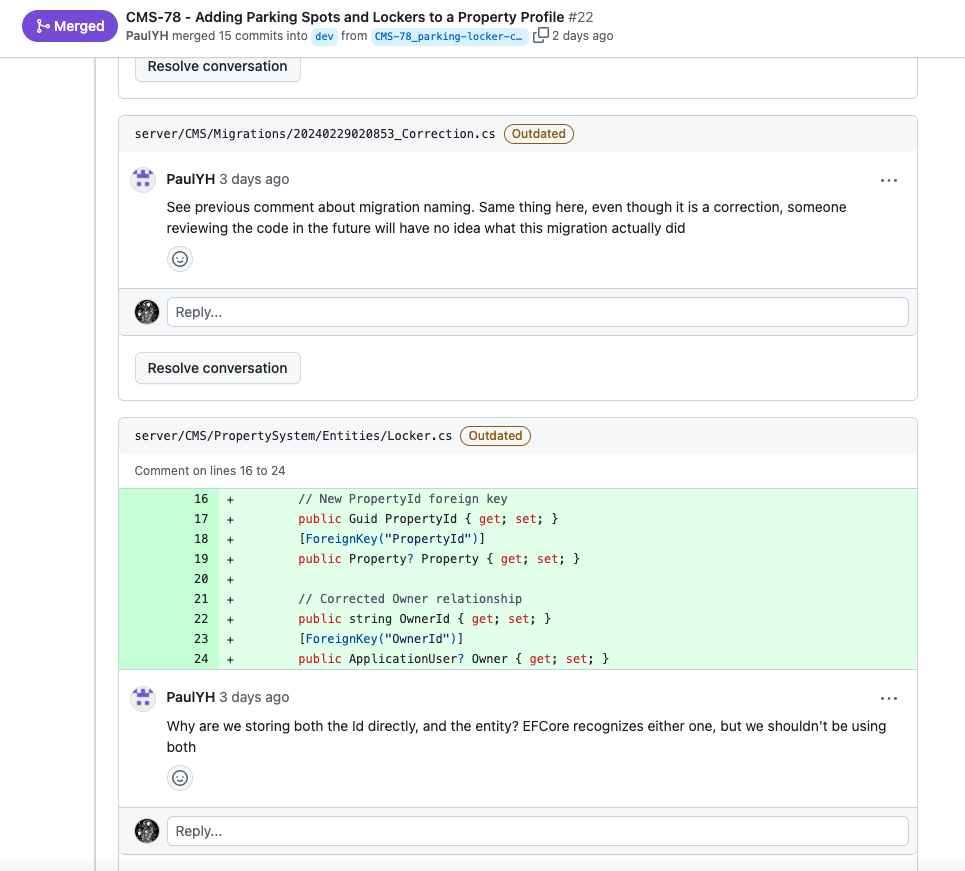
### 11.10.1 Importance of Quality Source Code Documentation

* Improving readability: Should the need to refactor code come up, analyzing and changing commented code will be far easier for any of us than if it did not have comments. It is easier to navigate through the program when clear comments give context, explanations, and insights into the purpose of individual components.
* Development acceleration: This happens when there is thorough documentation because programmers can give more effort to writing and improving code rather than interpreting it. Achieving project deadlines and producing high-quality software, results in greater productivity and faster iteration cycles.
* Collaboration: By making sure that all team members have an equal understanding of the application, thorough documentation will increase the cooperation between members of the team. By this we can achieve better collaboration and communication, which will improve the effectiveness of development procedures.

### 11.10.2 Practices we used for Source Code Documentation on GitHub

1. Use of descriptive comments: We wrote comments that describe each code function and what needs to be improved, or if everything was good with the code. We Stayed free of ambiguity and unnecessary technical terms by using language that is straightforward and brief. When writing comments, we kept the viewpoint of someone who isn't familiar with the codebase in mind.
2. Using Pull Request reviews: To perform thorough code reviews that take into account the documentation's quality, we made use of GitHub's pull request (PR) capability. Encourage team members to evaluate each other's documentation and code, offering helpful feedback and suggestions for enhancement.
3. Consistent commenting: We made sure that all comments were written in a way that they are easy to read and can be understood by all team members. This is done through maintaining consistent commenting throughout the codes.

*Examples on next page*



## 11.11 Refactoring activity documented in commit messages

In general, we agreed to use atomic commits, which means we make a commit for every substantial change. That means our commits have a fairly small and defined scope and purpose. For example, when adding a service class for one of our entities, we would make a commit for each method that we would add, unless the class is fairly small in which case we would permit ourselves to add the entire class in a single commit. In terms of refactoring, whenever we would change a class's name or purpose, there would be a detailed explanation in our commit message that explains what the class used to do and what it does now.

| [CMS-119](https://soen390-cms.atlassian.net/browse/CMS-119)  Refactor front-end calls to use ReactQuery | Changed basic fetch API calls to more robust and less error-prone calls using React Query npm package |
| --- | --- |
| [CMS-120](https://soen390-cms.atlassian.net/browse/CMS-120)  Refactor front-end elements to use NextUI components | Changed HTML elements, primarily buttons, to use the NextUI component library |

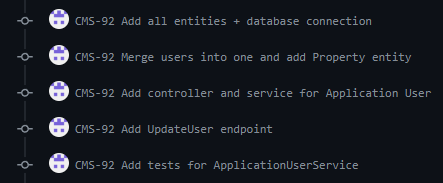
According to Refactoring Guru:

-The refactoring technique for [CMS-119](https://soen390-cms.atlassian.net/browse/CMS-119) is [Replace Method with Method Object](https://refactoring.guru/replace-method-with-method-object). We’re encapsulating the fetching API logic into a more robust structure (provided by React Query), instead of directly using basic fetch API. This encapsulated logic helps in better organization and maintenance.

-The refactoring technique for [CMS-120](https://soen390-cms.atlassian.net/browse/CMS-120) is [Replace Conditional with Polymorphism](https://refactoring.guru/replace-conditional-with-polymorphism). By changing HTML elements to NextUI components, we’re replacing conditional rendering of different types of buttons with polymorphism. Each button type in the NextUI library can have its own behavior and appearance, allowing for cleaner and more maintainable code compared to managing different conditions for each type of button.

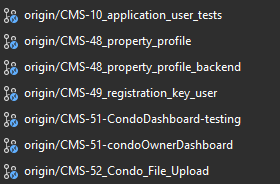
## 11.12 Quality/detail of commit messages

We have done our best to write clear and concise commit messages to convey exactly what is added/modified in a given commit. At the head of each commit message, we include the issue key (CMS-#) which allows Jira to link the commit to the appropriate issue. We also, the majority of the time, write the commits in the present tense as is the convention.



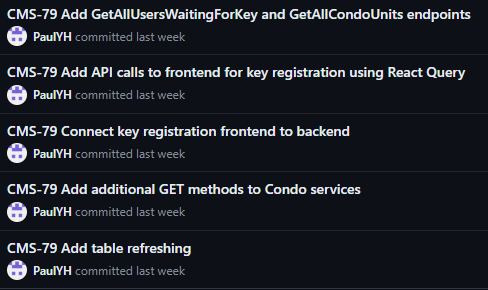
## 11.13 Use of feature branches

We are extensively using feature branches, ensuring that they are named properly to reflect the user story that each aims to implement. At the head of the branch name is the issue key (CMS-#) which allows Jira to link the branch to the appropriate issue.



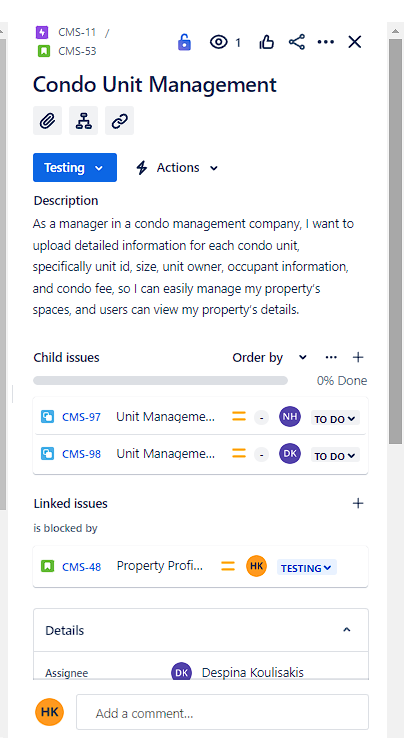
## 11.14 Atomic commits

During Sprint 2 of our project, we adopted the atomic commit methodology to enhance our version control practices. We continued this in Sprint 3. By embracing this approach, we ensure that our commits are granular, focusing on making small, self-contained changes. This means that for every two or three functions added or modified, we proceed with a commit. This strategy not only makes our commit history more intuitive and manageable but also simplifies the process of identifying and reverting specific changes if necessary. Implementing atomic commits has significantly improved the quality and clarity of our codebase, making it easier for the team to collaborate and track progress efficiently.



## 11.15 Linking of commits to bug reports/features





We have linked our GitHub repositories to JIRA, User stories are tracked through CMS-# where # is replaced with the number of the user story.

## 11.16 Coding guides

* React.JS Guide: <https://react.dev/blog/2023/03/16/introducing-react-dev>
* React.JS Documentation: <https://react.dev/reference/react>
* Node.JS Documentation: <https://nodejs.org/docs/latest/api/>
* ASP.NET Guide: <https://learn.microsoft.com/en-us/aspnet/tutorials>
* ASP.NET Documentation: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0>
* Entity Framework Documentation: <https://learn.microsoft.com/en-us/ef/>
* Microsoft SQL Server Documentation: <https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>