SOEN390 - Team Design Project

Architecture Description of Condo360 Architecture for Condo360

Version 3

By

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

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

Contents

1 Introduction

1.1.1 Summary:

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

1.1.2 Vision

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

1.1.3 Scope

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

1.2 Supplementary information

Date of Issue: 19/01/2024

Status: Draft

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

Reviewers: Triet Pham - Teaching Assistant

Approving Authority: Triet Pham - Teaching Assistant

Issuing Organization: Concordia University

1.2.1 Context:

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

1.2.2 Glossary:

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

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

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

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

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

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

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

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

References:

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

1.3 Other information

1.3.1 System or Architecture Overview:

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

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

This Architecture Description (AD) provides a detailed breakdown of the Condo Management Systems, elucidating its core functionalities, additional features, and bonus features as specified in the project requirements. It is organized into sections covering system overview, architectural views and models, supplementary information, and other pertinent details. Readers can navigate through the document to gain a comprehensive understanding of the system's architecture, design decisions, and rationale.

1.3.3 Results of Architecture Evaluations:

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

1.3.4 Rationale for key decisions

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

Trade-offs in Architectural Decisions:

a) HTML vs. NEXT:

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

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

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

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

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

2 Stakeholders and concerns

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

2.1: Stakeholders

2.1.1User

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

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

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

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

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

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

2.1.2 Software Provider Organization

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

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

Project Managers: Individuals overseeing the project development.

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

2.2: Concerns

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

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

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

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

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

3 Viewpoints+

3.1 Condo Information Management Viewpoint

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

3.2 Overview

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

3.3 Concerns and stakeholders

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

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

3.3.2 Typical stakeholders

Stakeholders:

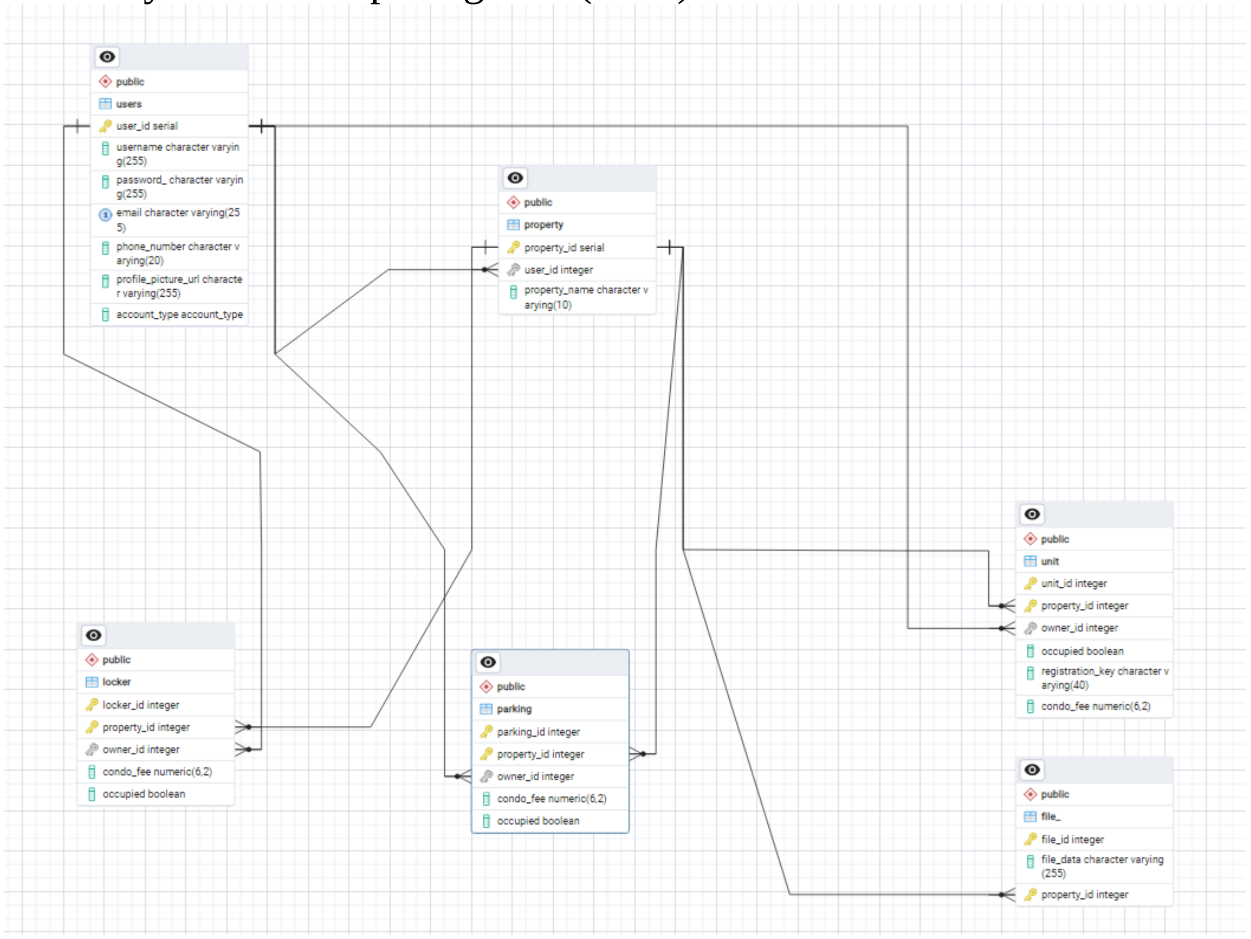
- Property Managers

- Residents

- Administrative Staff

3.4 Model kinds+

3.5 Entity Relationship Diagrams (ERD)



3.5.1 ERD conventions

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

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

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

3.6 Operations on views

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

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

  Define relationships

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

Determine Relationships and Cardinalities

Create draft

- **interpretation methods**:

  Overview:

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

  Entities are denoted by rectangles

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

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

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

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

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

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

  Ensure Cardinality Alignment

  Validate Constraints

  Review Notation Consistency

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

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

  Transform ERD attributes into class attributes, preserving data characteristics.

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

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

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

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

3.7 Correspondence rules

Correspondence with Use Case Diagrams:

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

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

Correspondence with Class Diagrams:

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

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

Correspondence with Sequence Diagrams:

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

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

3.9 Sources

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

Entity–relationship model - Wikipedia

ERD Tool — pgAdmin 4 8.3 documentation

4 Views+

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

4.1 Logical Viewpoint

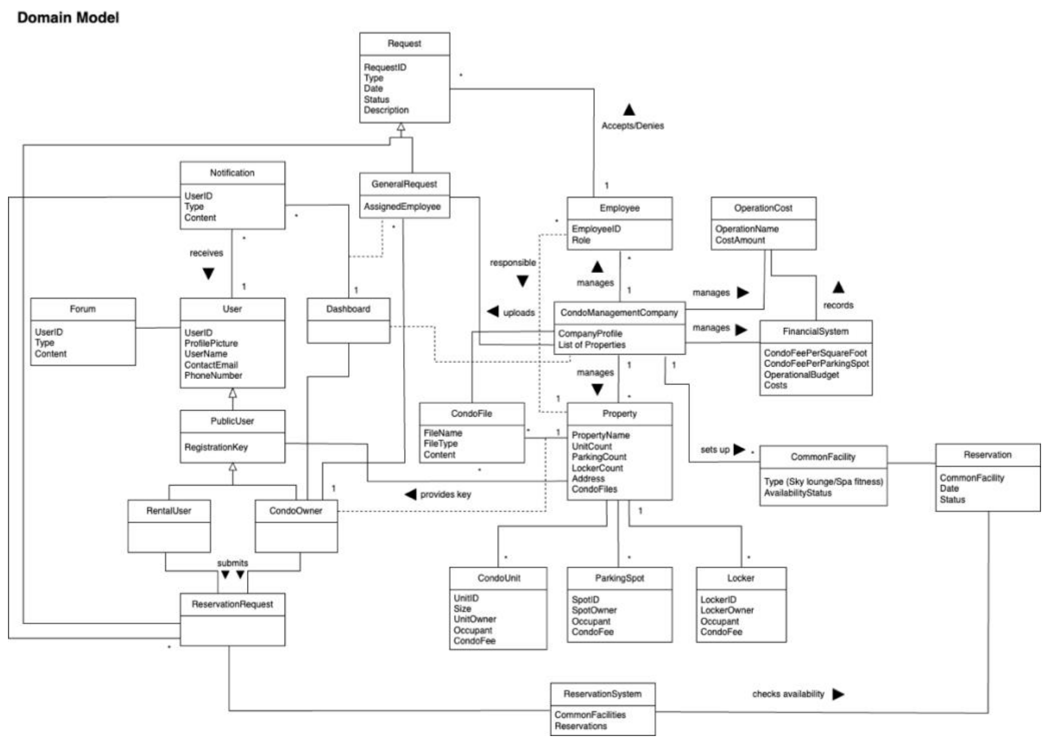
Overview: The Logical Viewpoint of the Condo360 System (CMS) provides a comprehensive view of the software's core conceptual foundation and operational functionality. It defines the system's essential structure, portraying how key entities such as condo owners, properties, and management companies interact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

Concerns and stakeholders

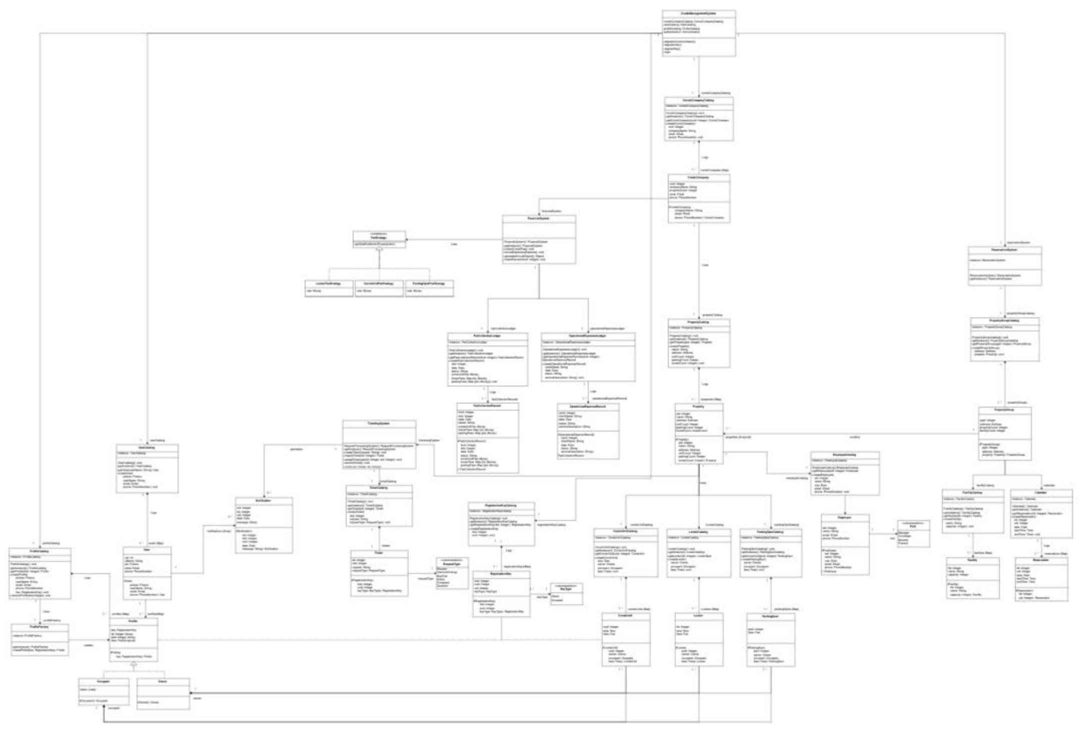
- **Concerns**:
  - **System Interaction**: Understanding how different user roles (condo owners, tenants, management staff) interact with the CMS to perform their tasks is crucial. This includes navigating the interface, accessing property information, making payments, and submitting service requests.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

Model kinds: UML Class Diagrams, Domain Models.

- **Domain Models**



- **UML Class Diagrams**



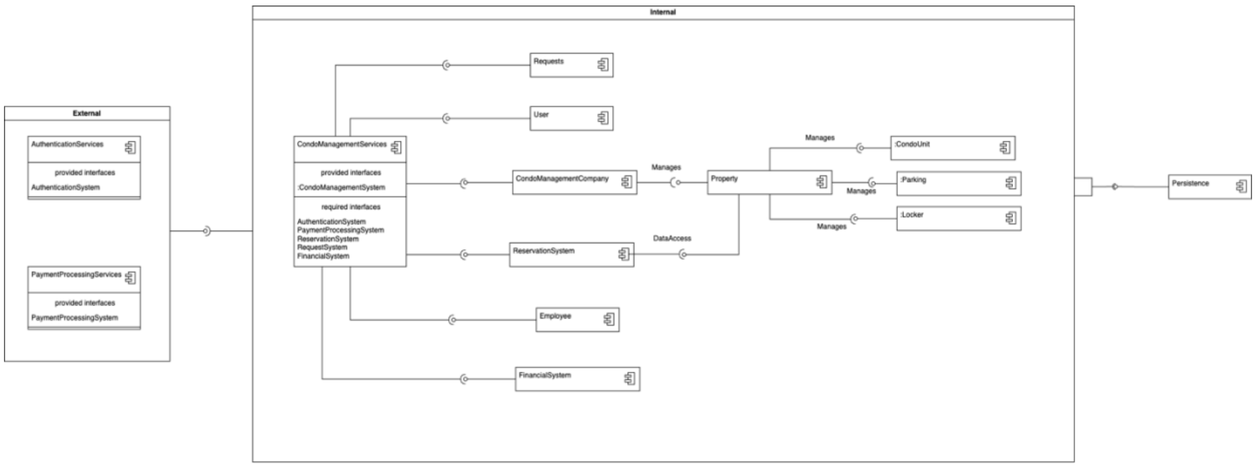
4.2 Development Viewpoint

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

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

Model Kinds: ComponentDiagram

- ○ **Component Diagram**



Concerns and Stakeholders

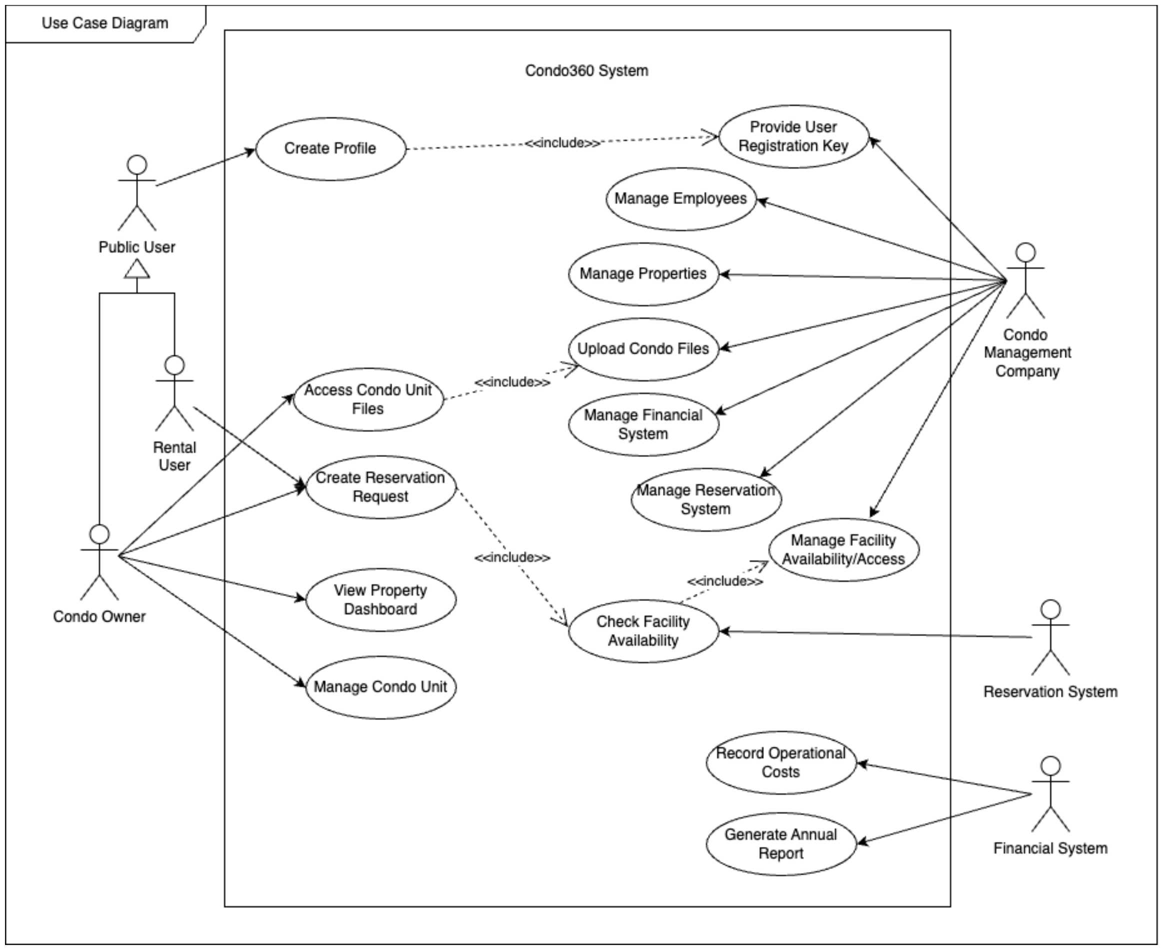
- **Concerns**:
  - **Modularity**: Ensuring the system is divided into well-defined, loosely coupled components that can be developed, tested, and deployed independently.
  - **Dependencies**: Identifying and managing dependencies between components to minimize coupling and facilitate easier integration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

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

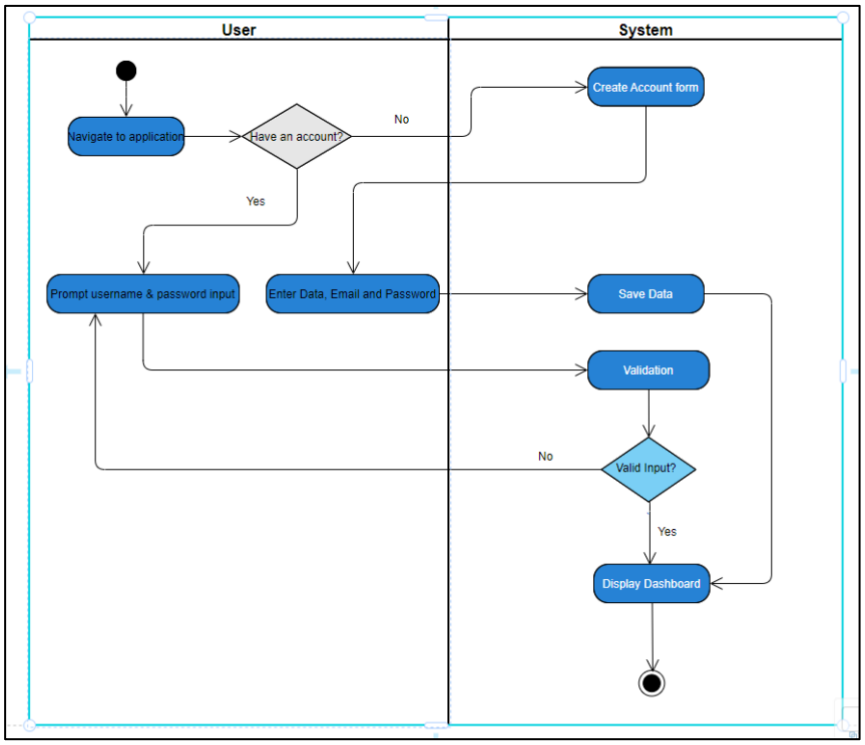
4.3 Process Viewpoint

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

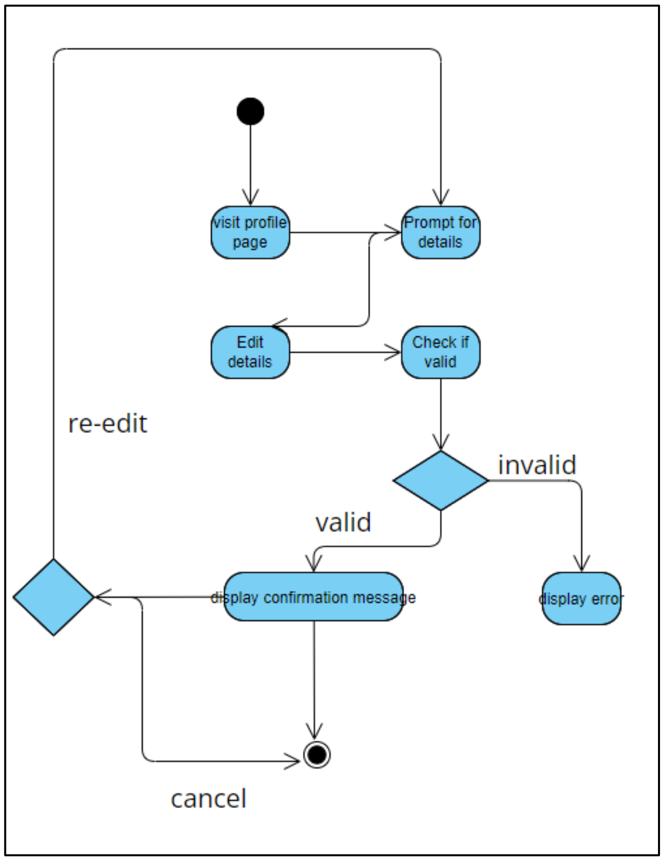
Model Kinds: UseCase Diagram, Activity Diagrams



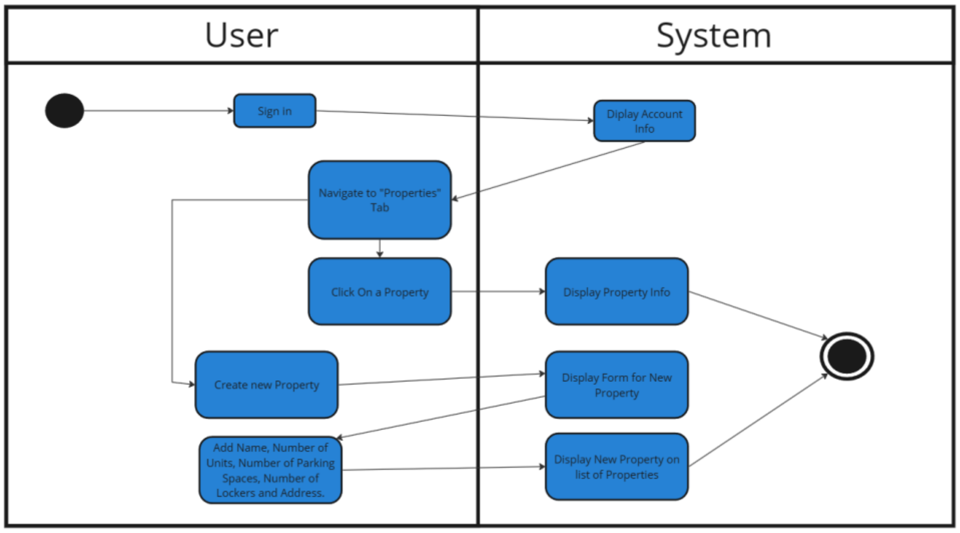
Sign Up/Login Activity Diagram:



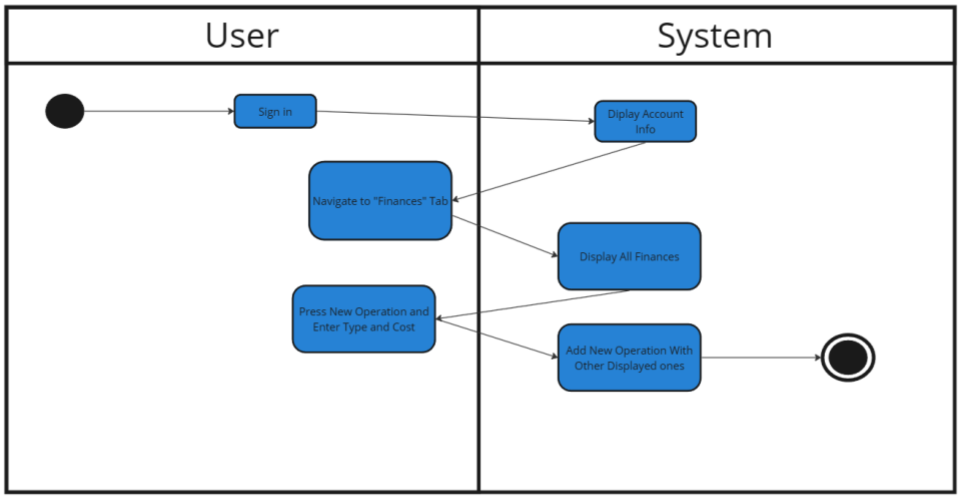
Profile Activity Diagram:



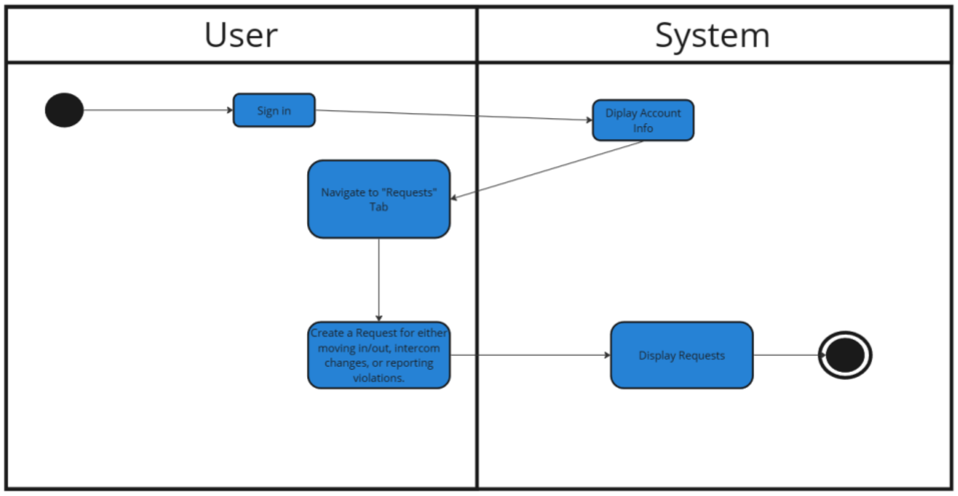
Property Activity Diagram:


User
System
Sign in
Diplay Account Info
Navigate to "Properties" Tab
Click On a Property
Display Property Info
Create new Property
Display Form for New Property
Add Name, Number of Units, Number of Parking Spaces, Number of Lockers and Address.
Display New Property on list of Properties

Finances Activity Diagram:


User
System
Sign in
Diplay Account Info
Navigate to "Finances" Tab
Display All Finances
Press New Operation and Enter Type and Cost
Add New Operation With Other Displayed ones

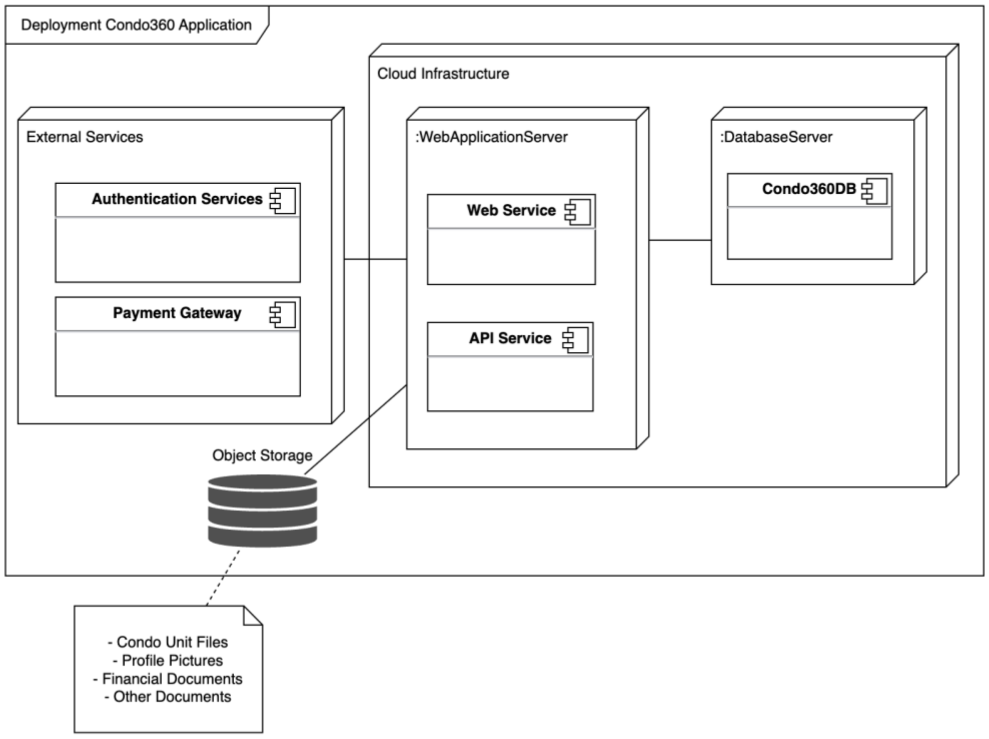
Requests Activity Diagram:


User
System
Sign in
Diplay Account Info
Navigate to "Requests" Tab
Create a Request for either moving in/out, intercom changes, or reporting violations.
Display Requests

4.4 Physical Viewpoint (Deployment Viewpoint)

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

Model Kinds: Deployment Diagram



5 Consistency and correspondences

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

5.2 Correspondences in the AD

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

5.3 Correspondence rules

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

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

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

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

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

Bibliography

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

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

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

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

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

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

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