Software Requirements Specification

for

LivingPlus

Prepared by

Muhammad Sibtain – 22i-0887

Abdullah Shakir – 22i-1138

Aqib Mehmood– 22p-9430

Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the requirements for the LivingPlus: A hostel and rental system, version 1.0. It details the system's functional and non-functional requirements, intended to provide a comprehensive description of what the software should do and the constraints under which it should operate. This SRS covers the complete scope of the LivingPlus system, focusing on facilitating efficient hostel and rental management processes.

1.2 Document Conventions

This SRS follows the IEEE Std 830-1998 format for Software Requirements Specifications. Priorities for requirements are explicitly stated (High, Moderate, Low) within each section. Each requirement is assigned a unique identifier (e.g., REQ-1, REQ-NF-1) for traceability.

1.3 Intended Audience and Reading Suggestions

This document is intended for various stakeholders, including:

- *Developers: To understand what the system should do.*
- *Project Managers: To plan and manage the development process.*
- *Hostel Owners and Tenants: To validate that the system meets their needs.*
- *Testers: To create test cases for verifying the system.*
- *Documentation Writers: To create user manuals and help materials.*

1.4 Product Scope

The LivingPlus system is a digital platform designed to streamline hostel and rental management. It aims to replace inefficient and manual processes, providing a centralized system for hostel owners and property managers to:

Register and manage hostel profiles.
View and manage room availability.
Manage tenant applications.
Handle tenant complaints and feedback.
Track payments and generate invoices.
Communicate with tenants.

The system also provides features for tenants to search for properties, manage their accounts, and communicate with landlords. The successful implementation of LivingPlus will improve operational efficiency, reduce errors, and enhance the overall tenant experience.

1.5 References

GitHub Repository: Living-Plus - https://github.com/ChaudaryAbdullah/Living-Plus

2. Overall Description

2.1 Product Perspective

The LivingPlus system is a new, self-contained product aimed at replacing inefficient and manual hostel and rental management processes. It is not a follow-on to an existing system but rather a solution designed to address the challenges faced by hostel owners, property managers, and tenants in managing rental properties. The system will provide a digital platform to streamline operations and enhance the tenant experience

2.2 Product Functions

The LivingPlus system will provide the following major functions:

- *Property Management:*
  - *Register and manage hostel profiles*
  - *View and manage available rooms & occupancy status.*
- *Property Search & Filters: Allow tenants to search for properties using various criteria.*
- *Application Management:*
  - *User Registration & Profile Management.*
  - *Approve tenant applications.*
  - *Booking & Viewing Requests.*
- *Tenant Management:*
  - *Assign rooms.*
  - *View tenant profiles.*
  - *Handle tenant complaints and feedback.*
- *Financial Management:*
  - *Track rental payments and dues.*
  - *Generate and send invoices.*
- *Communication Tools:*
  - *Send notifications & announcements to tenants and owners*
  - *Enable chat between owners and tenants.*

2.3 User Classes and Characteristics

The LivingPlus system will primarily serve the following user classes:

- *Hostel Owners/Property Managers: These users need to manage their properties efficiently, including listing hostels, managing rooms, handling applications, tracking payments, and communicating with tenants. They are expected to be moderately tech-savvy.*

- *Tenants: These users will use the system to search for properties, create accounts, manage their profiles, submit applications, communicate with landlords, and make payments. They are expected to be comfortable using web-based applications.*

2.4 Operating Environment

The LivingPlus system is intended to operate as a web-based application. It should be compatible with common web browsers such as Chrome, Firefox, and Safari. The system will require a web server to host the application and a database to store data.

2.5 Assumptions and Dependencies

The following assumptions and dependencies exist for the project:

- *It is assumed that users have access to a computer and internet connection.*
- *The system's success depends on the consistent availability of the web server and database.*
- *The project depends on the team's ability to effectively use the selected web technologies.*

3. External Interface Requirements

3.1 User Interfaces

The LivingPlus system will primarily utilize web-based user interfaces. These interfaces should adhere to modern web design principles, emphasizing clarity, consistency, and ease of use.

- *The system will provide separate interfaces for hostel owners/property managers and tenants, tailored to their specific functions.*
- *Standard UI components such as navigation menus, buttons, forms, and tables will be used consistently throughout the application.*
- *The user interface will be designed to be responsive, ensuring usability across various screen sizes and devices (desktops, laptops, tablets, and smartphones).*
- *Clear error messages and validation will be provided to guide users and prevent incorrect data entry.*
- *Accessibility guidelines (e.g., WCAG) will be considered to ensure the system is usable by people with disabilities.*
- *A separate user interface specification document will provide detailed designs and mockups of the user interfaces.*

3.2 Hardware Interfaces

The LivingPlus system is primarily a web-based application and does not have specific or direct hardware interface requirements beyond standard computing hardware.

3.3 Software Interfaces

The LivingPlus system will interact with the following software components:

- *Web Browser: The system will be accessed through standard web browsers (e.g., Chrome, Firefox, Safari).*

- *Operating System: The system is platform-independent from the client-side, as it is web-based. The server-side will require a compatible OS (e.g., Linux, Windows) to host the server.*
- *React: A JavaScript library for building the user interface. React will be used to create interactive and dynamic front-end components.*
- *Express.js: A Node.js web application framework. Express.js will be used to build the server-side application and handle routing, middleware, and API endpoints.*
- *Node.js: A JavaScript runtime environment. Node.js will be used to execute the Express.js server-side application.*
- *MongoDB: A NoSQL database will be used to store and retrieve system data, including user accounts, property listings, and application information.*
- *Mongoose: An Object Data Modeling (ODM) library for MongoDB and Node.js. It will be used to interact with the MongoDB database.*
- *Web Server: Node.js with Express will handle HTTP requests.*
- *Git: Git will be used for version control.*

3.4 Communications Interfaces

The LivingPlus system will utilize the following communications interfaces:

- *HTTP/HTTPS: The system will use HTTP/HTTPS for communication between the web browser and the web server. HTTPS will be used to ensure secure communication, especially when handling sensitive data like user credentials and payment information.*
- *Email: The system will use email for sending notifications, alerts, and invoices to users.*
- *Web Sockets: Real-time communication features, such as the chat between owners and tenants, may utilize Web Sockets for efficient, bi-directional communication.*
- *Data Formats: Data will be exchanged in standard formats such as JSON.*
- *Communication Security: Encryption protocols will be used to secure data transmission*

4. System Functional Requirements

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Hostel Registration

4.1.1 Description and Priority

This feature allows hostel owners to register their hostel profiles on the platform. This is a high-priority feature as it is essential for owners to list their properties.

Priority: High

4.1.2 Stimulus/Response Sequences

- ***User Action:*** *Hostel owner navigates to the "Register Hostel" page.*
- ***System Response:*** *The system displays the hostel registration form.*
- ***User Action:*** *Hostel owner fills in the required details and submits the form.*
- ***System Response:*** *The system validates the input.*

- ***System Response:*** *If valid, the system saves the hostel profile and displays a confirmation message.*
- ***System Response:*** *If invalid, the system displays validation errors and prevents submission.*

4.1.3 Functional Requirements

REQ-1: The system shall provide a user interface for hostel owners to register their hostel profiles.
REQ-2: The system shall provide fields for hostel details, including hostel name, address, capacity, and amenities.
REQ-3: The system shall validate all required fields before submission.
REQ-4: The system shall save the hostel profile upon successful submission.
REQ-5: The system shall display a confirmation message to the user after successful registration.
REQ-6: The system shall make the registered hostel visible in the owner's property list.
REQ-7: The system shall prevent submission if required fields are missing and display appropriate validation errors.

4.2 Room Management

4.1.1 Description and Priority

This feature enables hostel owners to view and update room availability and occupancy status. This is a high-priority feature, as it allows owners to efficiently manage their room inventory.
Priority: High

4.1.2 Stimulus/Response Sequences

- *User Action: Hostel owner navigates to the "Manage Rooms" page.*
- *System Response: The system displays a list of rooms with their occupancy status.*
- *User Action: Hostel owner updates the availability status of a room.*
- *System Response: The system updates the room status accordingly.*

4.1.3 Functional Requirements

REQ-8: The system shall provide a user interface for hostel owners to view a list of rooms.

REQ-9: The system shall display the current occupancy status of each room.

REQ-10: The system shall allow hostel owners to update the availability status of a room (e.g., from "Occupied" to "Available").

REQ-11: The system shall update the room status successfully upon request.

4.3 Application Management

4.1.1 Description and Priority

This feature allows hostel owners to review and approve tenant applications. This is a low-priority feature.

Priority: Low

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner views pending tenant applications.
- System Response: The system displays the applications.
- User Action: Hostel owner approves an application.
- System Response: The system changes the application status to "Approved" and notifies the tenant.
- User Action: Hostel owner rejects an application.
- System Response: The system changes the application status to "Rejected" and notifies the tenant.

4.1.3 Functional Requirements

REQ-12: The system shall provide a user interface for hostel owners to view pending tenant applications.
REQ-13: The system shall allow hostel owners to approve a tenant application.
REQ-14: The system shall change the application status to "Approved" when an application is approved.
REQ-15: The system shall notify the tenant of the application approval.
REQ-16: The system shall allow hostel owners to reject a tenant application.
REQ-17: The system shall change the application status to "Rejected" when an application is rejected.
REQ-18: The system shall notify the tenant of the application rejection

4.4 Tenant Profile Management

4.1.1 Description and Priority

This feature enables hostel owners to view tenant profiles and lease details. This feature is of moderate priority.

Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner views a tenant's profile.
- System Response: The system displays the tenant's personal information and lease details.

4.1.3 Functional Requirements

REQ-19: The system shall provide a user interface for hostel owners to view tenant profiles.

REQ-20: The system shall display tenant personal information and lease details.

4.5 Complaint Management

4.1.1 Description and Priority

This feature allows hostel owners to receive and address tenant complaints. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner views a tenant complaint.
- System Response: The system displays the complaint details.
- User Action: Hostel owner updates the complaint status or adds a resolution note.
- System Response: The system saves the changes.

4.1.3 Functional Requirements

REQ-21: The system shall provide a user interface for hostel owners to view tenant complaints.
REQ-22: The system shall allow hostel owners to update the status of a complaint.
REQ-23: The system shall allow hostel owners to add a resolution note to a complaint.

4.6 Feedback Management

4.1.1 Description and Priority

This feature allows hostel owners to collect tenant feedback. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner accesses the feedback section.
- System Response: The system displays all submitted feedback with ratings and comments.

4.1.3 Functional Requirements

REQ-24: The system shall provide a user interface for hostel owners to view tenant feedback.
REQ-25: The system shall display all submitted feedback, including ratings and comments.

4.7 Parking Management

4.1.1 Description and Priority

This feature enables hostel owners to review and approve tenant parking requests. This feature is of moderate priority.

Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Tenant submits a parking request with vehicle details.
- System Response: The system stores the request.
- User Action: Hostel owner reviews the parking request.
- System Response: The system displays the request details.
- User Action: Hostel owner approves or rejects the request.
- System Response: The system updates the request status.
- System Response: If approved, the system assigns a parking slot and notifies the tenant.

4.1.3 Functional Requirements

REQ-26: The system shall provide a user interface for tenants to submit parking requests.
REQ-27: The system shall allow tenants to enter vehicle details (e.g., make, model, license plate) when submitting a parking request.
REQ-28: The system shall provide a user interface for hostel owners to review parking requests.
REQ-29: The system shall display the request details, including tenant information and vehicle details.
REQ-30: The system shall allow hostel owners to approve or reject parking requests.
REQ-31: The system shall update the request status accordingly when a request is approved or rejected.
REQ-32: The system shall automatically assign an available parking slot based on predefined allocation rules.
REQ-33: The system shall notify the tenant with the assigned slot details and any relevant instructions upon approval.

4.8 Payment Tracking

4.1.1 Description and Priority

This feature allows hostel owners to track rental payments. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner opens the payments section.
- System Response: The system displays a list of tenants with their payment status and due amounts.

4.1.3 Functional Requirements

REQ-34: The system shall provide a user interface for hostel owners to view rental transactions.
REQ-35: The system shall display a list of tenants with their payment status and due amounts.

4.9 Invoice Generation

4.1.1 Description and Priority
This feature allows hostel owners to generate and send invoices. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner generates an invoice.
- System Response: The system creates an invoice with accurate billing details and sends it to the tenant.

4.1.3 Functional Requirements
REQ-36: The system shall allow hostel owners to generate invoices.
REQ-37: The generated invoice shall contain accurate billing details.
REQ-38: The system shall send the generated invoice to the tenant's registered email.

4.10 Notification Management

4.1.1 Description and Priority
This feature enables hostel owners to send notifications and announcements to tenants. This is a low-priority feature.

Priority: Low

4.1.2 Stimulus/Response Sequences

- User Action: Hostel owner drafts and sends a notification.
- System Response: The system sends the notification to all tenants

4.1.3 Functional Requirements
REQ-39: The system shall allow hostel owners to draft and send notifications.
REQ-40: The system shall ensure that all tenants receive the notification.

4.11 Tenant Account Management

4.1.1 Description and Priority
This feature allows tenants to create accounts and manage their profiles. This feature is of moderate priority.

Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Tenant enters valid information on the registration page and submits.
- System Response: The system creates a tenant account.
- User Action: Tenant updates profile details.
- System Response: The system updates the tenant's profile.
- User Action: Tenant requests a password reset.
- System Response: The system sends a reset link or OTP.

4.1.3 Functional Requirements

REQ-41: The system shall provide a user interface for tenants to create an account.
REQ-42: The system shall allow tenants to enter a valid email or phone number and complete the registration form.
REQ-43: The system shall successfully create a tenant account upon valid registration.
REQ-44: The system shall provide a user interface for tenants to update their profile details.
REQ-45: The system shall allow tenants to update their name, contact information, or preferences.
REQ-46: The system shall successfully update tenant profile details upon saving changes.
REQ-47: The system shall provide a user interface for tenants to reset their password.
REQ-48: The system shall allow tenants to request a password reset by providing their registered email or phone number.
REQ-49: The system shall send a reset link or OTP to allow tenants to set a new password.

4.12 Property Search

4.1.1 Description and Priority

This feature allows tenants to search for rental properties using filters. This feature is of moderate priority.

Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Tenant enters a keyword to search for properties.
- System Response: The system displays relevant properties.
- User Action: Tenant applies filters to refine the search.
- System Response: The system updates the search results based on the filters.
- User Action: Tenant sorts the search results.
- System Response: The system rearranges the results based on the sorting option.

4.1.3 Functional Requirements
REQ-50: The system shall provide a user interface for tenants to search for properties.
REQ-51: The system shall allow tenants to search for properties using keywords.
REQ-52: The system shall display relevant properties matching the keyword.

4.13 Property Viewing

4.1.1 Description and Priority
This feature allows tenants to request property viewings or book units online. This feature is of moderate priority.
Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Tenant requests a property tour.
- System Response: The system sends the request to the landlord.
- System Response: The system notifies the tenant of the confirmation or rejection of the tour request.
- System Response: The system sends reminders for scheduled viewings.
- User Action: Tenant cancels or reschedules a viewing.
- System Response: The system updates the booking and notifies the landlord

4.1.3 Functional Requirements
REQ-53: The system shall allow tenants to request a property tour (either in-person or virtual).
REQ-54: The system shall send the tour request to the landlord for approval.
REQ-55: The system shall notify the tenant of the confirmation or rejection of their tour request.

4.14 Communication

4.1.1 Description and Priority
This feature enables communication between tenants and landlords. This feature is of moderate priority.

Priority: Moderate

4.1.2 Stimulus/Response Sequences

- User Action: Tenant initiates a chat with the landlord.
- System Response: The system allows sending and receiving messages.
- System Response: The system notifies tenants of new messages.

4.1.3 Functional Requirements
REQ-56: The system shall allow tenants to initiate a chat with the landlord.

REQ-57: The system shall allow tenants to send and receive messages within the platform.
REQ-58: The system shall notify tenants of new messages from landlords.

4.15 Feedback and Reviews

4.1.1 Description and Priority

This feature allows tenants to rate and review their rental experience. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Tenant provides a star rating and writes a review.
- System Response: The system stores the rating and review.
- System Response: The system displays reviews on the property listing.

4.1.3 Functional Requirements

REQ-59: The system shall allow tenants to provide a star rating (1-5) for a property.
REQ-60: The system shall allow tenants to write a review for a property.
REQ-61: The system shall display submitted reviews and ratings on the property listing.

4.16 Parking Requests (Tenant)

4.1.1 Description and Priority

This feature allows tenants to request a parking slot. This is a high-priority feature.

Priority: High

4.1.2 Stimulus/Response Sequences

- User Action: Tenant submits a parking request with vehicle details.
- System Response: The system records the request.

4.1.3 Functional Requirements

REQ-62: The system shall allow tenants to enter vehicle details (e.g., make, model, license plate) when submitting a parking request.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable

design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

REQ-NF-7: The system shall ensure that user data (e.g., personal information, payment details) is stored securely to prevent unauthorized access.

5.3 Security Requirements

REQ-NF-8: The system shall use secure password storage techniques (e.g., hashing with salt) to protect user passwords.
REQ-NF-10: The system shall implement user roles and permissions to restrict access to sensitive data and functions (e.g., only hostel owners can manage their property listings).

5.4 Software Quality Attributes

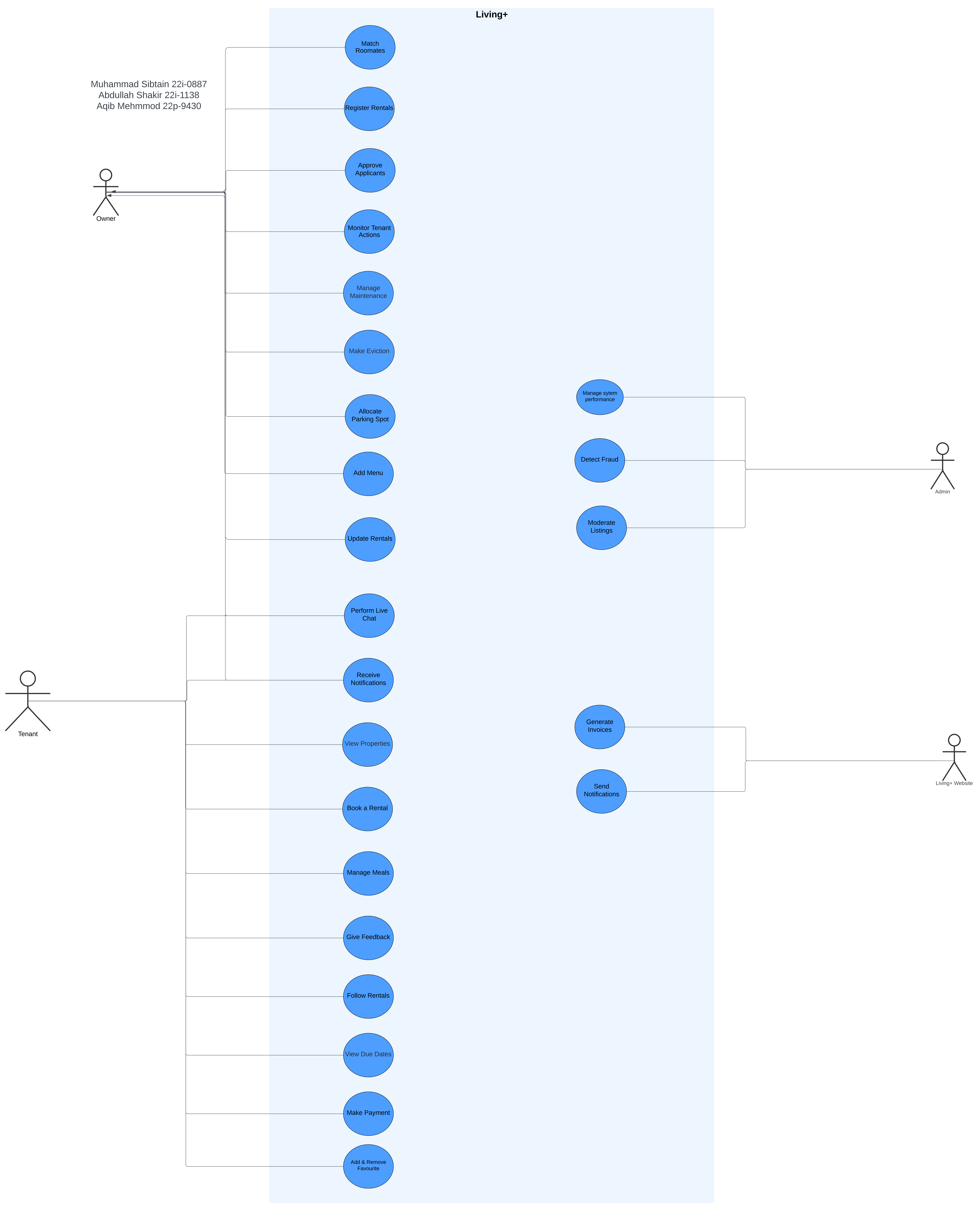
REQ-NF-1: The system shall have a user-friendly interface that is easy to navigate and understand for both hostel owners and tenants.
REQ-NF-2: The system shall provide clear and helpful error messages to guide users in case of problems.
REQ-NF-3: The system shall be designed in a modular way to facilitate future maintenance and updates.
REQ-NF-4: The system should be able to function across different web browsers (Chrome, Firefox, Safari)

5.5 Business Rules

REQ-NF-5: The system shall enforce that only registered hostel owners can create and manage hostel listings.
REQ-NF-6: The system shall ensure that tenants can only submit applications for properties that are currently available.
REQ-NF-7: The system shall define the rules for allocating parking spots (e.g., first-come, first-served, priority for certain tenants).

Usecase Diagram

Muhammad Sibtain 22i-0887
Abdullah Shakir 22i-1138
Aqib Mehmmod 22p-9430



6. User Stories

Story ID: 1

Story Title: Register a Hostel

User Story:
As a Hostel Owner, I want to register my hostel profile, so that I can list my property on the platform.

Importance:
High

Estimate:
M

Acceptance Criteria:

Scenario: Successful Registration

- **Given:** Given I am logged into the system
- **When:** I navigate to the "Register Hostel" page and fill in the required details
- **Then:** the system should save the hostel profile and display a confirmation message
- **And:** the hostel should be visible in my property list

Scenario: Missing Required Information

- **Given:** I am on the "Register Hostel" page
- **When:** I submit the form without filling in required fields
- **Then:** the system should show validation errors and prevent submission

Type:
☐Search
☐Workflow
☒Manage Data
☐Payment
☐Report/View

Story ID:2 **Story Title:** View & Update	Importance: High
User Story: As a Hostel Owner, I want to view and update available rooms and current occupancy status, so that I can manage room availability efficiently.	Estimate: L
Acceptance Criteria: **Scenario: View Room List** • **Given:** Given I am on the “Manage Rooms” page • **When:** I view the list of rooms • **Then:** I should see the current occupancy status of each room **Scenario: Update Room Availability** • **Given:** a room is listed as "Occupied" • **When:** I set it to "Available" • **Then:** the room status should update successfully	Type: ☐Search ☐Workflow ☒Manage Data ☐Payment ☒Report/View

Story ID: 3

Story Title: Review Applicants

User Story:
As a Hostel Owner, I want to review and approve tenant applications, so that I can fill my hostel rooms with suitable tenants.

Importance:
Low

Estimate:
L

Acceptance Criteria:

Scenario: Approve Application

- **Given:** there are pending tenant applications
- **When:** I review an application and click “Approve”
- **Then:** the application status should change to “Approved”
- **And:** the tenant should receive a confirmation notification

Scenario 2: Reject Application

- **Given:** there are pending tenant applications
- **When:** I review an application and click “Reject”
- **Then:** the application status should change to “Rejected”
- **And:** the tenant should receive a rejection notification

Type:
☐Search
☐Workflow
☒Manage Data
☐Payment
☐Report/View

Story ID: 4 **Story Title:** View Tenant Profiles **User Story:** As a Hostel Owner, I want to view tenant profiles and lease details, so that I can manage tenant information efficiently.	Importance: Moderate Estimate: XL
Acceptance Criteria: • **Given:** there are tenants in the hostel • **When:** I open a tenant's profile • **Then:** I should see their personal information and lease details	Type: ☐Search ☐Workflow ☐Manage Data ☐Payment ☒Report/View

Story ID: 5	**Story Title:** Address Tenant
User Story: As a Hostel Owner, I want to receive and address tenant complaints, so that I can improve their living experience.	Importance: High Estimate: XL
Acceptance Criteria: • **Given:** a tenant has submitted a complaint • **When:** I view the complaint • **Then:** I should be able to update its status and add a resolution note	Type: ☐Search ☒Workflow ☐Manage Data ☐Payment ☐Report/View

Story ID: 6	**Story Title:** Collect Tenant Feedback
User Story: As a Hostel Owner, I want to collect tenant feedback, so that I can enhance the quality of my hostel services.	Importance: High Estimate: XL
Acceptance Criteria: • **Given:** tenants have provided feedback • **When:** I access the feedback section • **Then:** I should see all submitted feedback with ratings and comments	Type: ☐Search ☒Workflow ☐Manage Data ☐Payment ☐Report/View

Story ID: 7

Story Title: Accept Parking

User Story:
As a Hostel Owner, I want to review and approve tenant parking requests so that I can allocate parking spaces efficiently and ensure proper management of available slots.

Importance:
Moderate

Estimate:
M

Acceptance Criteria:

Scenario 1: Landlord reviews a parking request

- Given a tenant has submitted a parking request
- When the landlord navigates to the parking request management section
- Then the system should display the request details, including tenant information and vehicle details

Scenario 2: Landlord approves or rejects the request

- Given the landlord is reviewing a parking request
- When they choose to approve or reject the request
- Then the system should update the request status accordingly

Scenario 3: System assigns and notifies the tenant of approval

- Given the landlord has approved the parking request
- When a parking slot is allocated
- Then the system should notify the tenant with the assigned slot details and any relevant instructions

Type:
☐Search
☐Workflow
☐Manage Data
☐Payment
☒Report/View

Story ID: 8

Story Title: Track Payments

User Story:
As a Hostel Owner, I want to track rental payments, so that I know which tenants have paid and who has pending dues.

Importance:
High

Estimate:
XL

Acceptance Criteria:

- **Given:** there are rental transactions
- **When:** I open the payments section
- **Then:** I should see a list of tenants with their payment status and due amounts

Type:
☐Search
☐Workflow
☐Manage Data
☒Payment
☒Report/View

Story ID: 9

Story Title: Generate Invoices

User Story:
As a Hostel Owner, I want to generate and send invoices, so that tenants receive accurate billing information.

Importance:
High

Estimate:
XL

Acceptance Criteria:

- **Given:** a tenant's rent is due
- **When:** I generate an invoice
- **Then:** the invoice should contain accurate billing details and be sent to the tenant's registered email

Type:
☐Search
☐Workflow
☐Manage Data
☒Payment
☐Report/View

Story ID: 10

Story Title: Send Notifications

User Story:
As a Hostel Owner, I want to send notifications and announcements to tenants, so that I can keep them informed about important updates and events.

Importance:
Low

Estimate:
L

Acceptance Criteria:

- **Given:** I have an important announcement
- **When:** I draft and send a notification
- **Then:** all tenants should receive the notification

Type:
☐Search
☒Workflow
☐Manage Data
☐Payment
☐Report/View

Story ID: 11

Story Title: Create Tenant

User Story:
As a tenant, I want to create an account and manage my profile so that I can save my preferences and interact with the platform.

Importance:
Moderate

Estimate:
S

Acceptance Criteria:

Scenario 1: Tenant creates an account

- **Given:** the tenant is on the registration page
- **When:** they enter a valid email or phone number and complete the registration form
- **Then:** their account should be successfully created

Scenario 2: Tenant updates profile details

- **Given:** the tenant is logged into their account
- **When:** they update their name, contact info, or preferences and save changes
- **Then:** the system should successfully update their profile

Scenario 3: Tenant resets password

- **Given:** the tenant has forgotten their password
- **When:** they request a password reset and provide their registered email or phone number
- **Then:** the system should send a reset link or OTP to allow them to set a new password

Type:
☐Search
☐Workflow
☒Manage Data
☐Payment
☐Report/View

Story ID: 12

Story Title: Search properties

Importance: Moderate

Estimate: M

User Story:
As a tenant, I want to search for rental properties using filters like location, price, facilities and availability so that I can quickly find options that match my needs.

Type:
☒Search
☐Workflow
☐Manage Data
☐Payment
☐Report/View

Acceptance Criteria:

Scenario 1: Tenant searches for properties using keywords

- **Given** the tenant is on the property search page
- **When** they enter a keyword related to a property (e.g., "hostel near university")
- **Then** the system should display relevant properties matching the keyword

Scenario 2: Tenant applies filters to refine search

- **Given** the tenant is on the search results page
- **When** they apply filters such as location, price range, property type, or facilities
- **Then** the system should update the search results based on the selected filters

Scenario 3: System returns relevant search results

- **Given** the tenant has entered a keyword or applied filters
- **When** they submit the search request
- **Then** the system should return a list of properties that match the criteria

Scenario 4: Tenant sorts search results

- **Given** the tenant has a list of search results
- **When** they select a sorting option (e.g., by price, rating, or proximity)
- **Then** the system should rearrange the results accordingly

Story ID: 13

Story Title: View Properties

User Story:
As a tenant, I want to request a property viewing or book a unit online so that I can inspect the property before committing to a rental.

Importance:
Moderate

Estimate:
M

Acceptance Criteria:

Scenario 1: Tenant requests an in-person or virtual property tour

- **Given** the tenant is on the property listing page
- **When** they request a property tour (either in-person or virtual)
- **Then** the system should send the request to the landlord for approval

Scenario 2: Tenant receives confirmation or rejection from the landlord

- **Given** the tenant has requested a property tour
- **When** the landlord approves or rejects the request
- **Then** the system should notify the tenant of the confirmation or rejection
-

Scenario 3: System sends notifications for scheduled viewings

- **Given** the tenant has an approved property viewing
- **When** the scheduled time approaches
- **Then** the system should send reminders to both the tenant and the landlord

Scenario 4: Tenant cancels or reschedules a viewing

- **Given** the tenant has a scheduled property tour
- **When** they choose to cancel or reschedule the appointment
- **Then** the system should update the booking and notify the landlord of the changes

Type:
☐Search
☐Workflow
☐Manage Data
☐Payment
☒Report/View

Story ID: 14	Story Title: Communication

As a tenant, I want to communicate with landlords through the platform's messaging system so that I can clarify details and negotiate terms securely.

Importance: Moderate

Estimate: L

Acceptance Criteria:

Scenario 1: Tenant sends and receives messages within the platform

- **Given** the tenant is logged into their account and viewing a property listing
- **When** they initiate a chat with the landlord
- **Then** the system should allow them to send and receive messages

Scenario 2: Tenant receives notifications for new messages

- **Given** the tenant has an active conversation with a landlord
- **When** the landlord sends a message
- **Then** the system should notify the tenant of the new message

Scenario 3: Chat supports text, file attachments, and read receipts

- **Given** the tenant is in a chat conversation
- **When** they send a text message or attach a file (e.g., ID proof, rental agreement)
- **Then** the system should successfully deliver the message and display read receipts when the landlord views it

Scenario 4: Messages are secure and encrypted

- **Given** the tenant is using the chat feature
- **When** they send or receive a message
- **Then** the system should ensure all messages are encrypted and securely stored

Type:
☐Search
☒Workflow
☐Manage Data
☐Payment
☐Report/View

Story ID: 15	Story Title: Give Feedback
As a tenant, I want to rate and review my rental experience so that other tenants can make informed decisions based on real feedback.	Importance: High Estimate: L

Scenario 1: Tenant leaves a star rating and writes a review

- **Given** the tenant has completed their rental stay
- **When** they navigate to the review section of the property
- **Then** the system should allow them to provide a star rating (1-5) and write a review

Scenario 2: Reviews are visible on property listings

- **Given** the tenant has submitted a review
- **When** other users view the property listing
- **Then** the system should display the submitted review and rating

Scenario 3: Tenant can only review properties they have rented

- **Given** the tenant is logged into their account
- **When** they attempt to submit a review for a property they have not rented
- **Then** the system should prevent them from submitting the review and display an error message

Type:
☐Search
☐Workflow
☐Manage Data
☐Payment
☒Report/View

Story ID: 16

Story Title: Request Parking

Importance: High

As a tenant, I want to request a parking slot through the platform so that I can secure a designated parking space for my vehicle.

Estimate: L

Scenario 1: Tenant submits a parking request with vehicle details

- Given the tenant needs a parking slot
- When they navigate to the parking request section
- Then the system should allow them to enter vehicle details (e.g., make, model, license plate) and submit a parking request

Scenario 2: System assigns a parking slot to the tenant

- Given the tenant has submitted a parking request
- When the request is processed
- Then the system should automatically assign an available parking slot based on predefined allocation rules

Scenario 3: Tenant receives confirmation of parking slot allocation

- Given the parking request has been approved
- When a parking slot is assigned
- Then the system should notify the tenant with parking details, including slot number and any relevant instructions

Type:
☐Search
☐Workflow
☐Manage Data
☐Payment
☒Report/View

Story ID: 17 Story Title: Receive Notifications As a tenant, I want to receive notifications for new property listings, landlord responses, payment reminders, and maintenance updates so that I stay informed about important events.	Importance: High Estimate: XL
Scenario 1: Tenant receives real-time notifications for new property listings that match their preferences • **Given** the tenant has set their property preferences • **When** a new listing that matches their criteria is added • **Then** the system should send a real-time notification to the tenant **Scenario 2: Tenant receives booking confirmations or cancellations** • **Given** the tenant has requested a property viewing or booking • **When** the landlord approves or cancels the request • **Then** the system should send a notification updating the tenant **Scenario 3: Tenant receives notifications for new messages from landlords** • **Given** the tenant has ongoing communication with a landlord • **When** the landlord sends a new message • **Then** the system should send a real-time notification to the tenant **Scenario 4: Tenant receives payment reminders** • **Given** the tenant has an upcoming or overdue payment • **When** the due date approaches or passes • **Then** the system should send a payment reminder notification **Scenario 5: Notifications can be received through in-app alerts** • **Given** the tenant is logged into the platform • **When** a new notification is triggered • **Then** the system should display an in-app alert	Type: ☐Search ☐Workflow ☒Manage Data ☐Payment ☐Report/View

Scenario 6: Tenant can manage notification preferences • **Given** the tenant wants to customize their notifications • **When** they access the notification settings • **Then** the system should allow them to enable or disable specific notifications	

7. Team Agreement

Team Members: Muhammad Sibtain (22i-0887), Abdullah Shakir (22i-1138), Aqib Mehmood (22p-9430)

Roles:

- **Product Owner:** Muhammad Sibtain - Responsible for defining the product backlog, prioritizing user stories, and representing the stakeholders.
- **Scrum Master:** Abdullah Shakir - Responsible for facilitating Scrum ceremonies, removing impediments.
- **Development Teams:** All team members - Responsible for developing and testing the product.

Methods of Communication:

- Whatsapp will be used for daily communication, quick questions and real time updates.
- Email will be used for formal announcements, document sharing.
- Phone calls will be reserved for urgent matters requiring immediate attention.

Communication Response Time:

- Whatsapp messages will be responded to within 1 hour.
- Emails will be acknowledged within 24 hours.
- Phone calls for urgent matters will be answered immediately.

Meeting Attendance:

- Daily stand-up meetings will be held every morning before first lecture.
- Sprint planning, sprint review, and sprint retrospective meetings will be held face-to-face in University Library.

Running Meetings:

- Daily stand-ups will be limited to 10 to 15 minutes.
- Sprint planning, review, and retrospective meetings will be scheduled for a duration appropriate for the sprint.
- The Scrum Master will facilitate all Scrum ceremonies and ensure meetings stay on track.
- Meeting notes/minutes will be taken by a rotating team member and shared on the Whatsapp group after the meeting.

Meeting Preparation:

- For daily stand-ups, team members will prepare a brief update on their progress, any blockers, and their plan for the day.
- For sprint planning, team members will review the product backlog and estimate user story points.
- For sprint review, team members will prepare a demo of their work.
- For sprint retrospective, team members will prepare to discuss what went well, what could have been improved, and action items for the next sprint.

Version Control:

- We will use GitHub for version control.
- Only working, tested code will be committed to the repository.
- Commit messages will be clear, concise, and follow the format: "[Task ID] - [Brief Description of Changes]".
- Pull requests will be reviewed by at least one other team member before merging.

Division of Work:

- The Product Owner will prioritize the product backlog.
- During sprint planning, the Scrum Master will facilitate task assignment based on team member skills and availability.
- All team members will participate in task estimation and assignment.

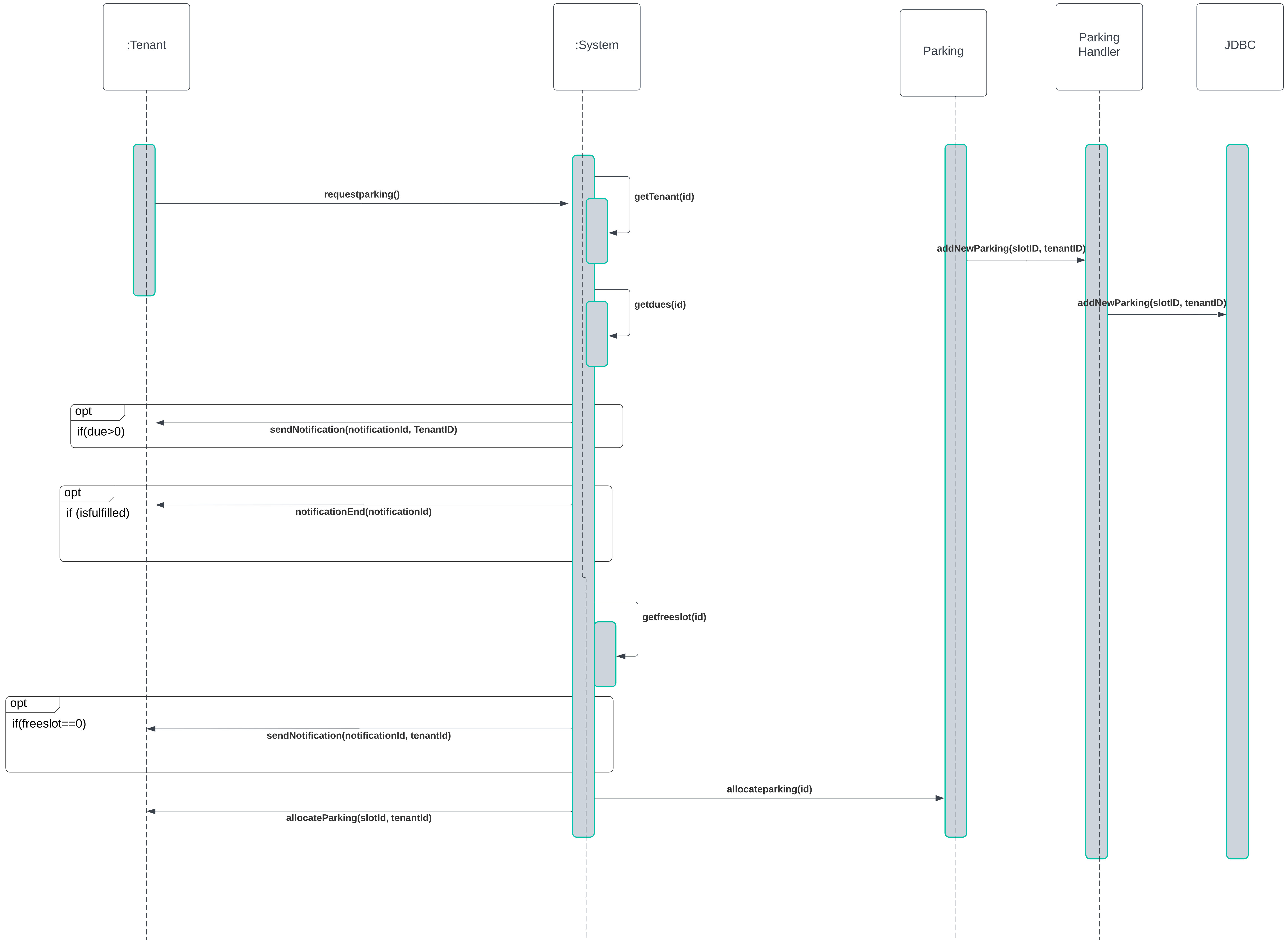
Submitting Assignments:

- Code reviews will be conducted via GitHub pull requests.
- All code and documentation will be submitted by the deadlines agreed upon during sprint planning.
- The Scrum Master will verify that all required deliverables have been submitted.

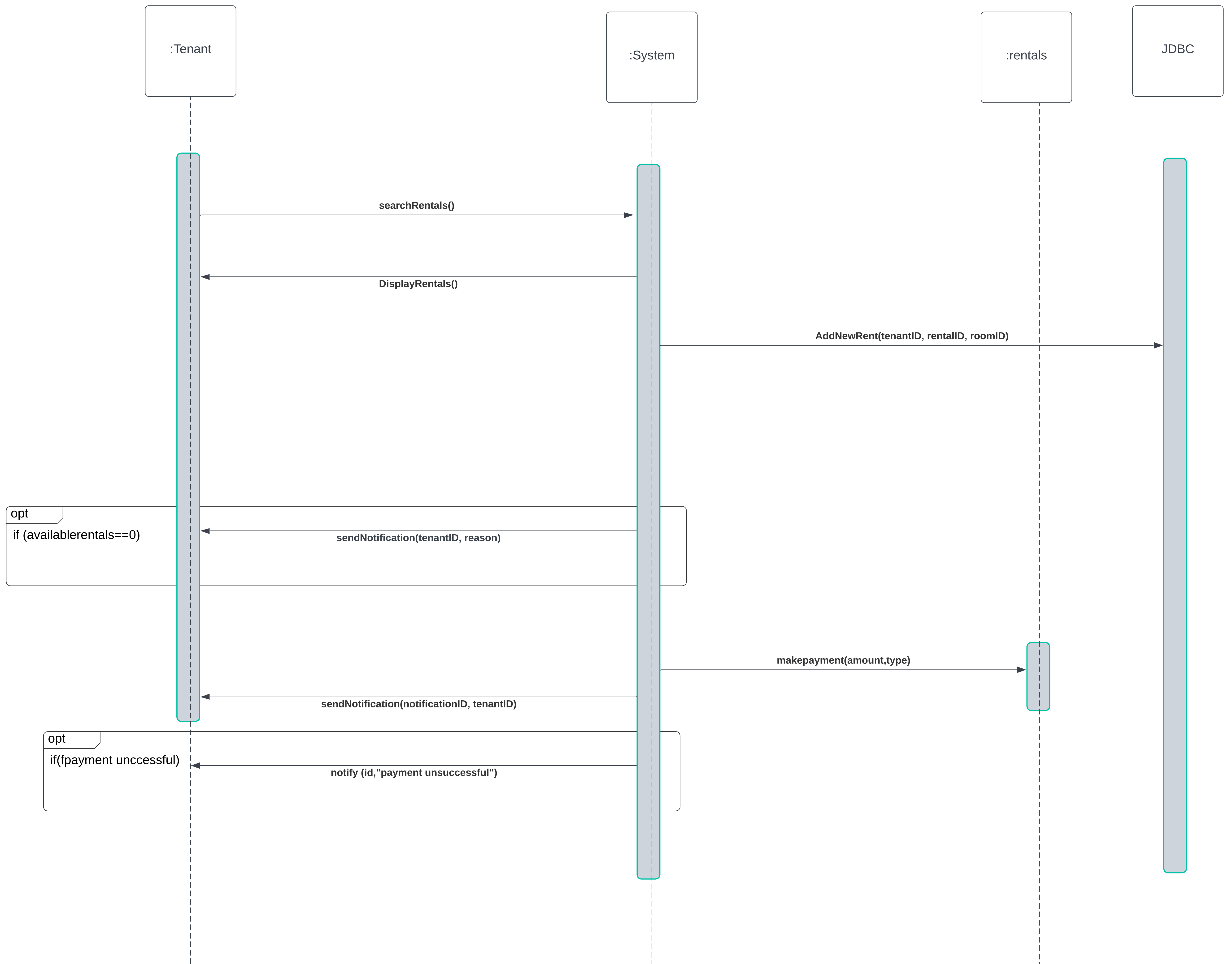
Contingency Planning:

- If a team member is unavailable due to illness or other unforeseen circumstances, their tasks will be redistributed among the remaining team members.
- If a team member consistently misses meetings or fails to meet deadlines, the Scrum Master will address the issue with the team member and, if necessary, with the instructor.
- If a team member drops out of the team, the team will inform the instructor immediately and work with the instructor to find a solution.

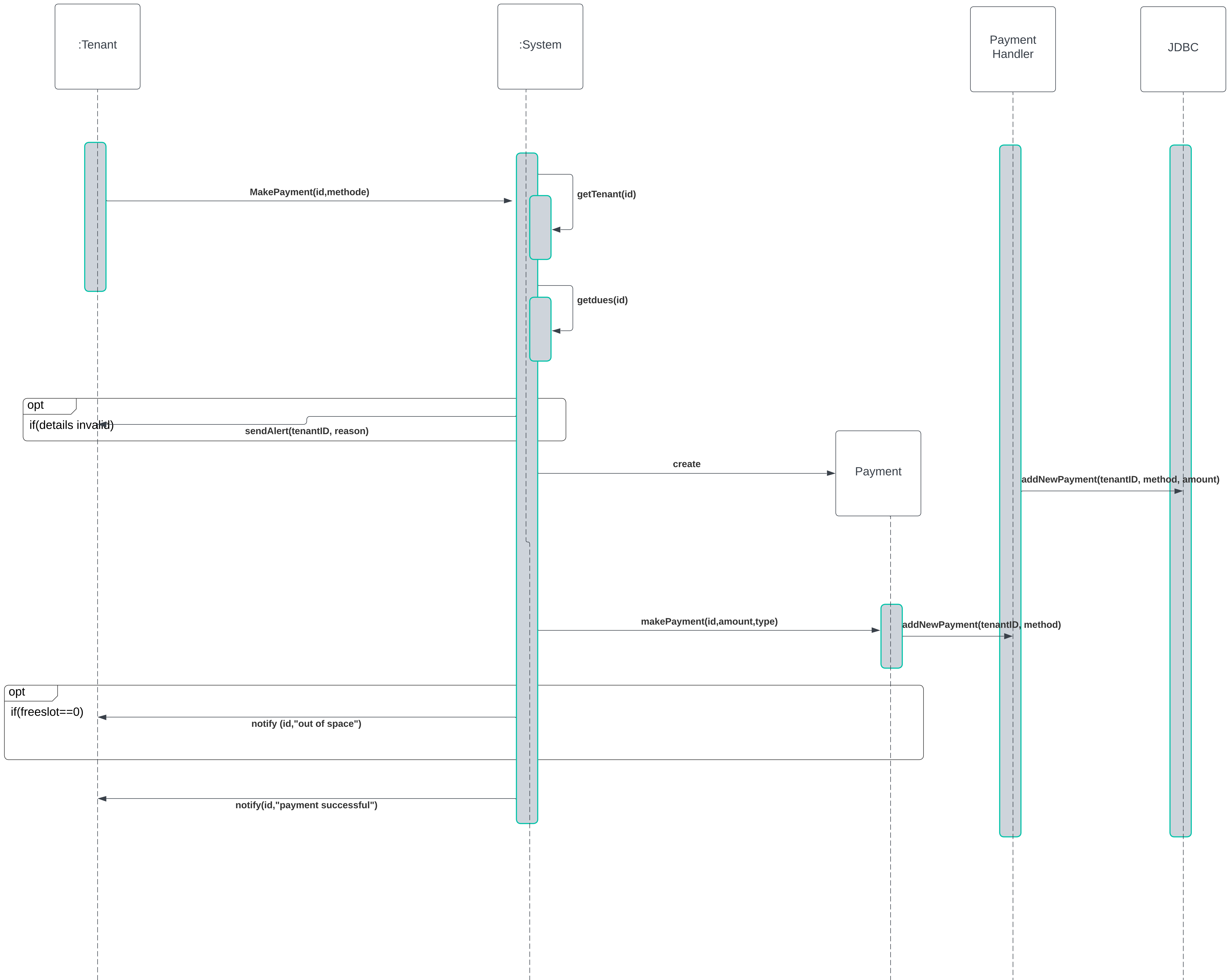
Request Parking


:Tenant
:System
Parking
Parking Handler
JDBC
requestparking()
getTenant(id)
addNewParking(slotID, tenantID)
getdues(id)
addNewParking(slotID, tenantID)
opt
if(due>0)
sendNotification(notificationId, TenantID)
opt
if (isfulfilled)
notificationEnd(notificationId)
getfreeslot(id)
opt
if(freeslot==0)
sendNotification(notificationId, tenantId)
allocateparking(id)
allocateParking(slotId, tenantId)

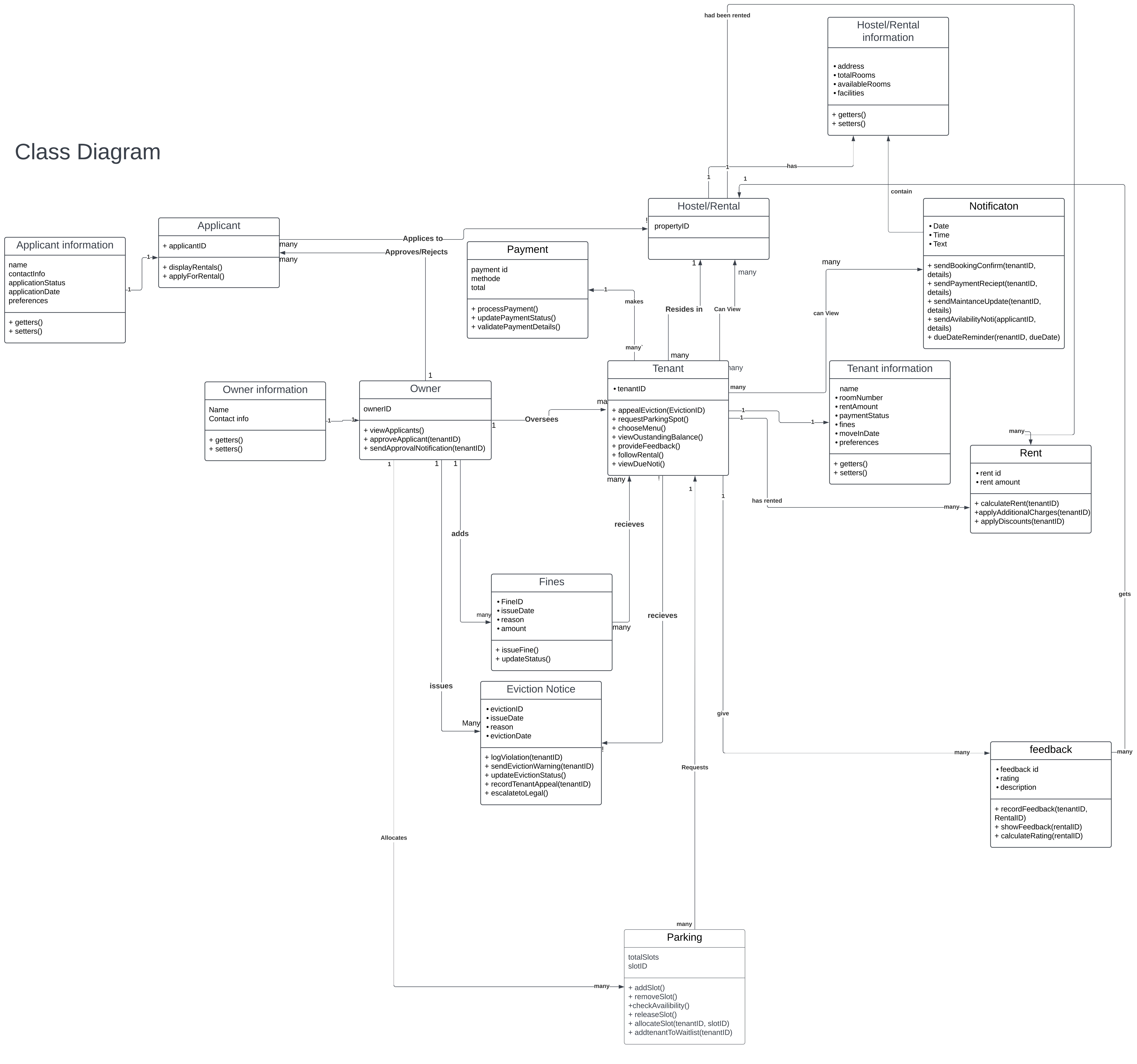
Rent a Rental


:Tenant
:System
:rentals
JDBC
searchRentals()
DisplayRentals()
AddNewRent(tenantID, rentalID, roomID)
opt
if (availablerentals==0)
sendNotification(tenantID, reason)
makepayment(amount,type)
sendNotification(notificationID, tenantID)
opt
if(fpayment unccessful)
notify (id,"payment unsuccessful")

Make Payment


:Tenant
:System
Payment Handler
JDBC
MakePayment(id,methode)
getTenant(id)
getdues(id)
opt
if(details invalid)
sendAlert(tenantID, reason)
create
Payment
addNewPayment(tenantID, method, amount)
makePayment(id,amount,type)
addNewPayment(tenantID, method)
opt
if(freeslot==0)
notify (id,"out of space")
notify(id,"payment successful")

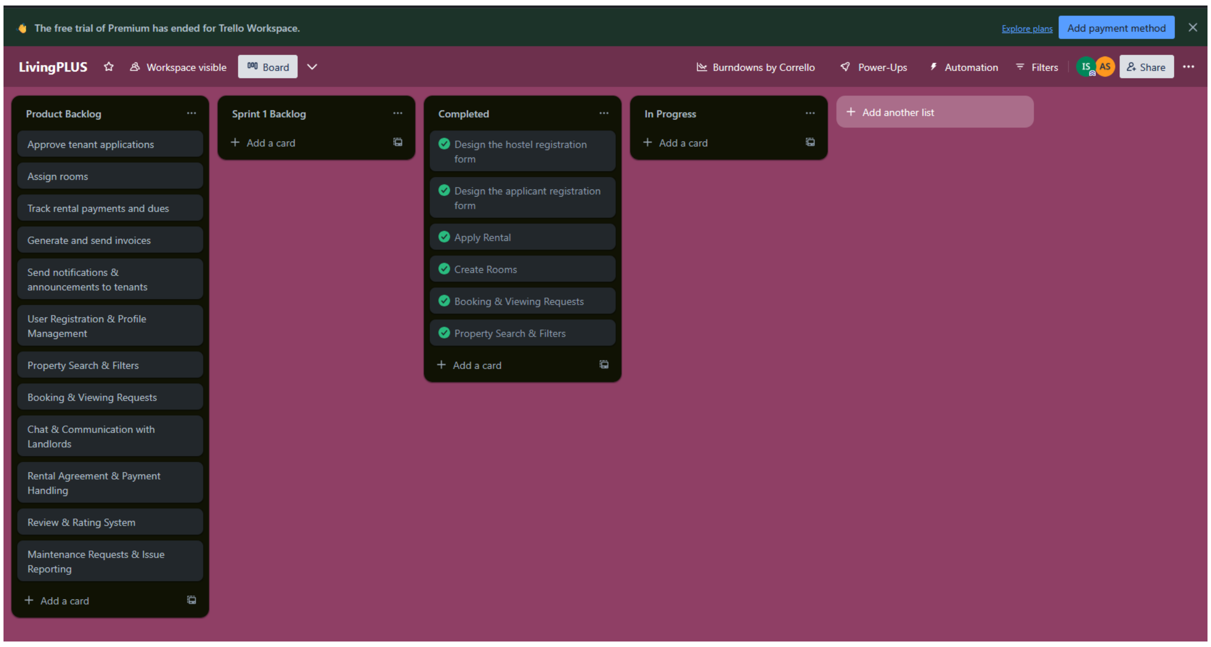
Class Diagram



Trello:

Sprint 1:

Backlog:


The free trial of Premium has ended for Trello Workspace. Explore plans Add payment method

LivingPLUS Workspace visible Board Burndowns by Corrello Power-Ups Automation Filters Share

Product Backlog
- Approve tenant applications
- Assign rooms
- Track rental payments and dues
- Generate and send invoices
- Send notifications & announcements to tenants
- User Registration & Profile Management
- Property Search & Filters
- Booking & Viewing Requests
- Chat & Communication with Landlords
- Rental Agreement & Payment Handling
- Review & Rating System
- Maintenance Requests & Issue Reporting
- Add a card

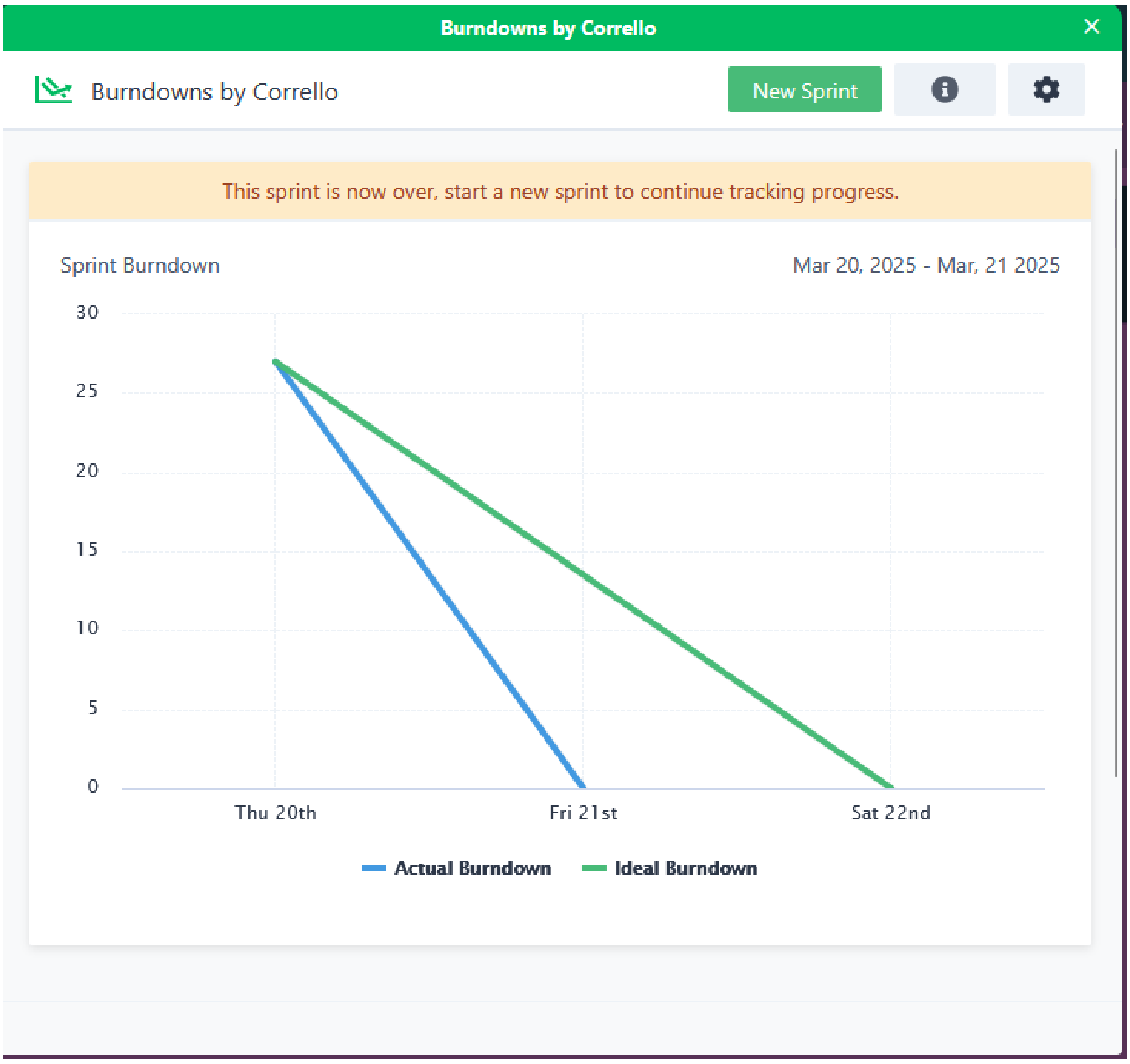
Sprint 1 Backlog
- Add a card

Completed
- ✓ Design the hostel registration form
- ✓ Design the applicant registration form
- ✓ Apply Rental
- ✓ Create Rooms
- ✓ Booking & Viewing Requests
- ✓ Property Search & Filters
- Add a card

In Progress
- Add a card

Add another list

Burndown Chart:


Burndowns by Corrello

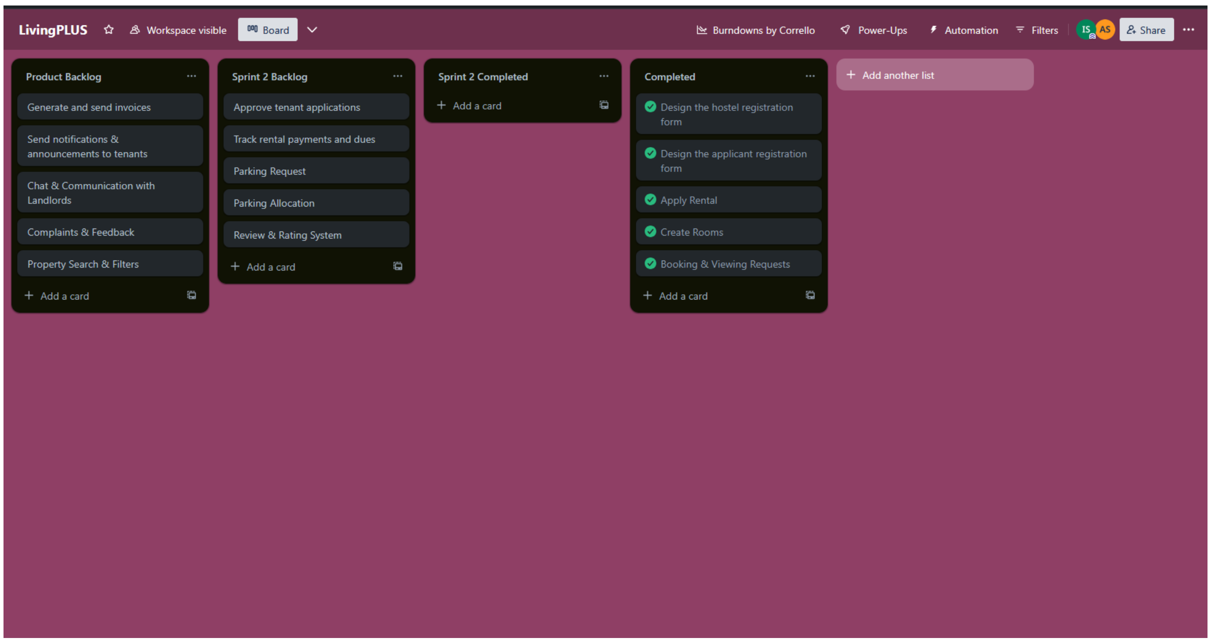
This sprint is now over, start a new sprint to continue tracking progress.

Sprint Burndown

Mar 20, 2025 - Mar, 21 2025

Sprint 2:

Sprint Start:



LivingPLUS

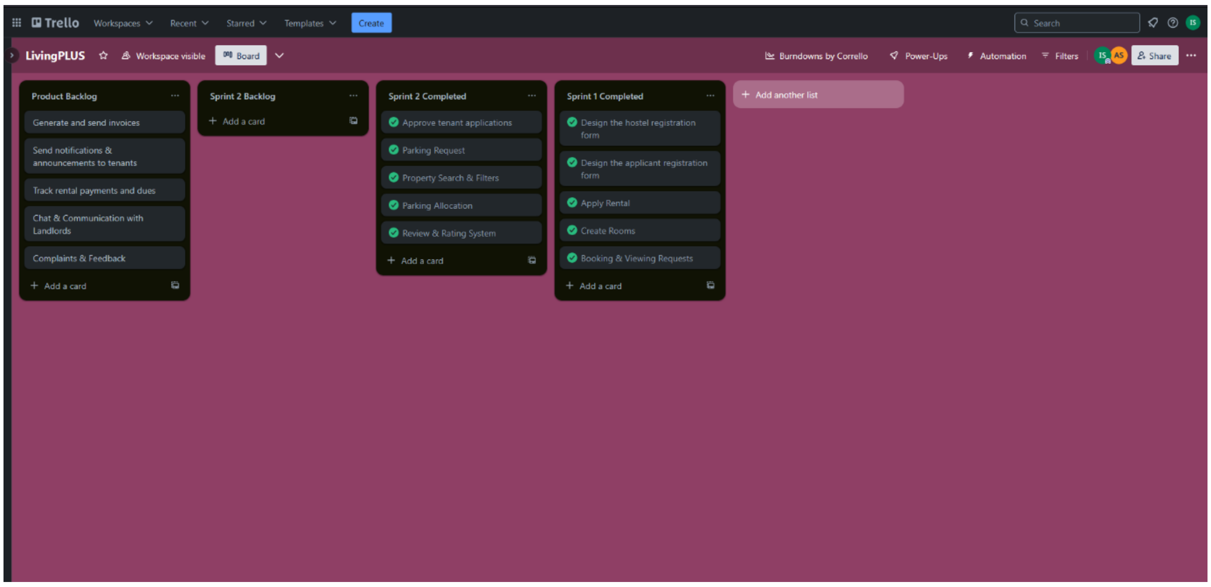
Product Backlog
- Generate and send invoices
- Send notifications & announcements to tenants
- Chat & Communication with Landlords
- Complaints & Feedback
- Property Search & Filters

Sprint 2 Backlog
- Approve tenant applications
- Track rental payments and dues
- Parking Request
- Parking Allocation
- Review & Rating System

Sprint 2 Completed

Completed
- ✓ Design the hostel registration form
- ✓ Design the applicant registration form
- ✓ Apply Rental
- ✓ Create Rooms
- ✓ Booking & Viewing Requests

Sprint End:



LivingPLUS

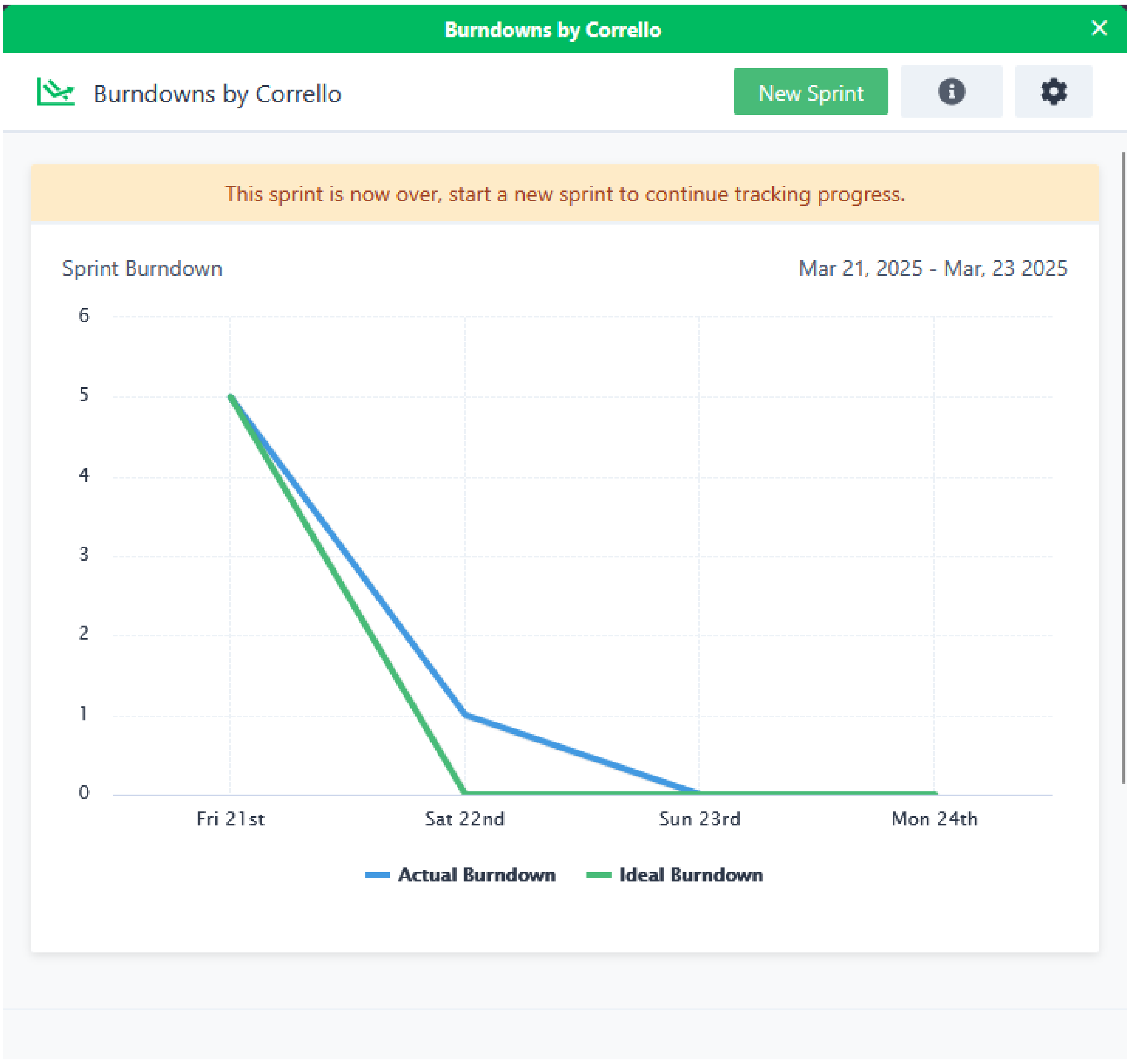
Product Backlog
- Generate and send invoices
- Send notifications & announcements to tenants
- Track rental payments and dues
- Chat & Communication with Landlords
- Complaints & Feedback

Sprint 2 Backlog

Sprint 2 Completed
- ✓ Approve tenant applications
- ✓ Parking Request
- ✓ Property Search & Filters
- ✓ Parking Allocation
- ✓ Review & Rating System

Sprint 1 Completed
- ✓ Design the hostel registration form
- ✓ Design the applicant registration form
- ✓ Apply Rental
- ✓ Create Rooms
- ✓ Booking & Viewing Requests

Burndown Chart:



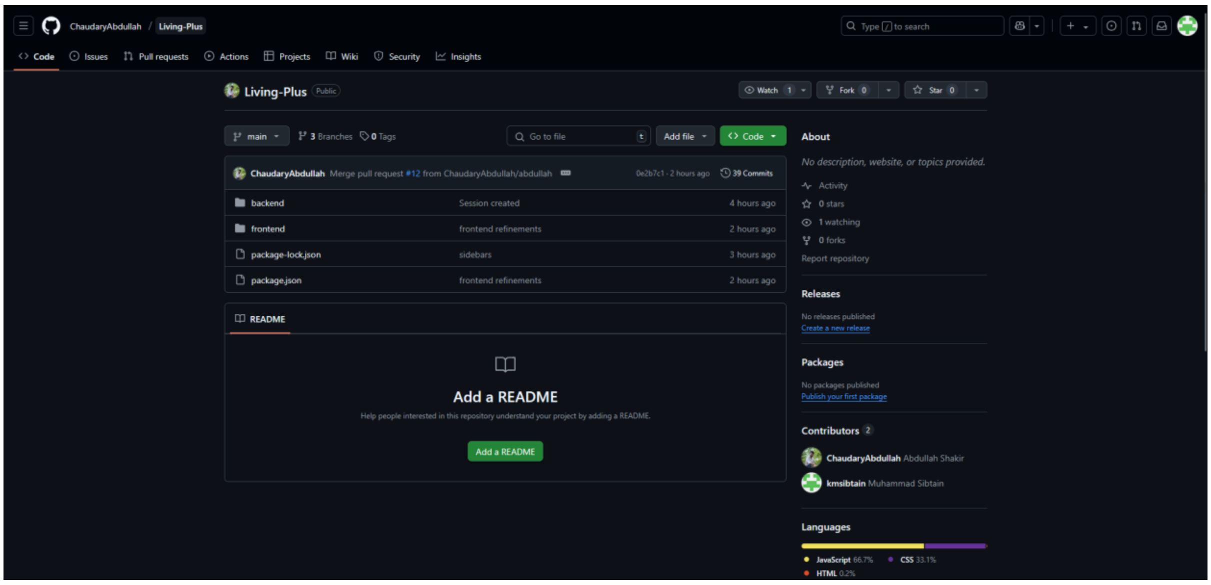
This sprint is now over, start a new sprint to continue tracking progress.

GitHub:

Link:

https://github.com/ChaudaryAbdullah/Living-Plus

Screenshot:



Living-Plus Public

3 Branches 0 Tags

ChaudaryAbdullah Merge pull request #12 from ChaudaryAbdullah/abdullah 0e2b7c1 · 2 hours ago 39 Commits

backend	Session created	4 hours ago
frontend	frontend refinements	2 hours ago
package-lock.json	sidebars	3 hours ago
package.json	frontend refinements	2 hours ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks
Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

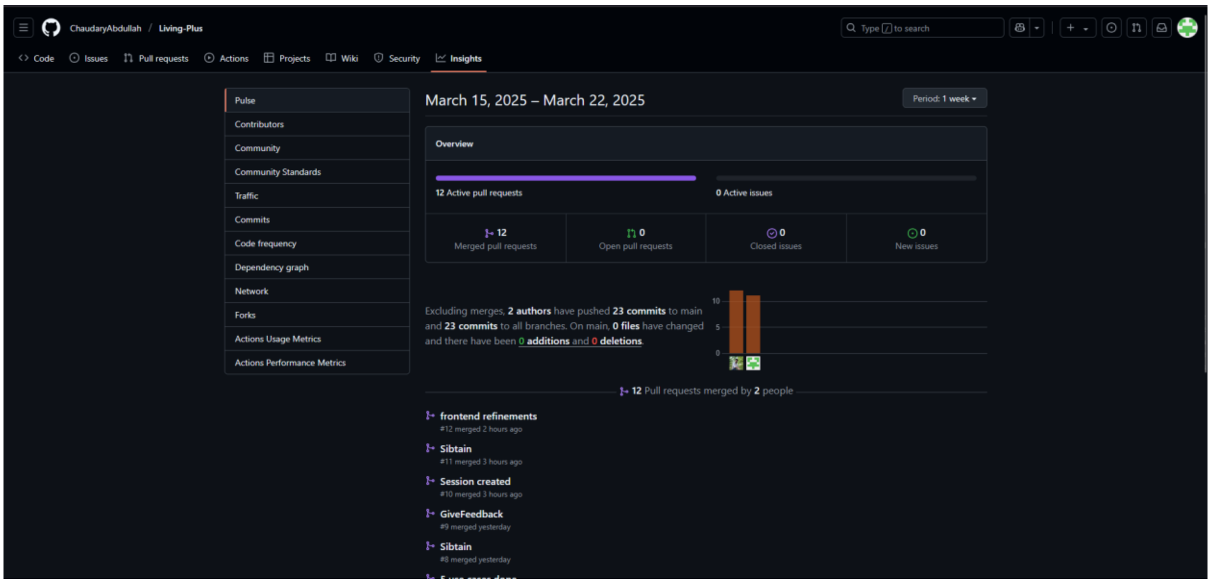
Contributors 2

ChaudaryAbdullah Abdullah Shakir
kmsibtain Muhammad Sibtain

Languages

JavaScript 66.7% CSS 33.1% HTML 0.2%

Insights:



March 15, 2025 – March 22, 2025

Period: 1 week

Overview

12 Active pull requests
0 Active issues

12 Merged pull requests	0 Open pull requests	0 Closed issues	0 New issues

Excluding merges, **2 authors** have pushed **23 commits** to main and **23 commits** to all branches. On main, **0 files** have changed and there have been 0 **additions** and 0 **deletions**.

12 Pull requests merged by **2** people

frontend refinements
#12 merged 2 hours ago

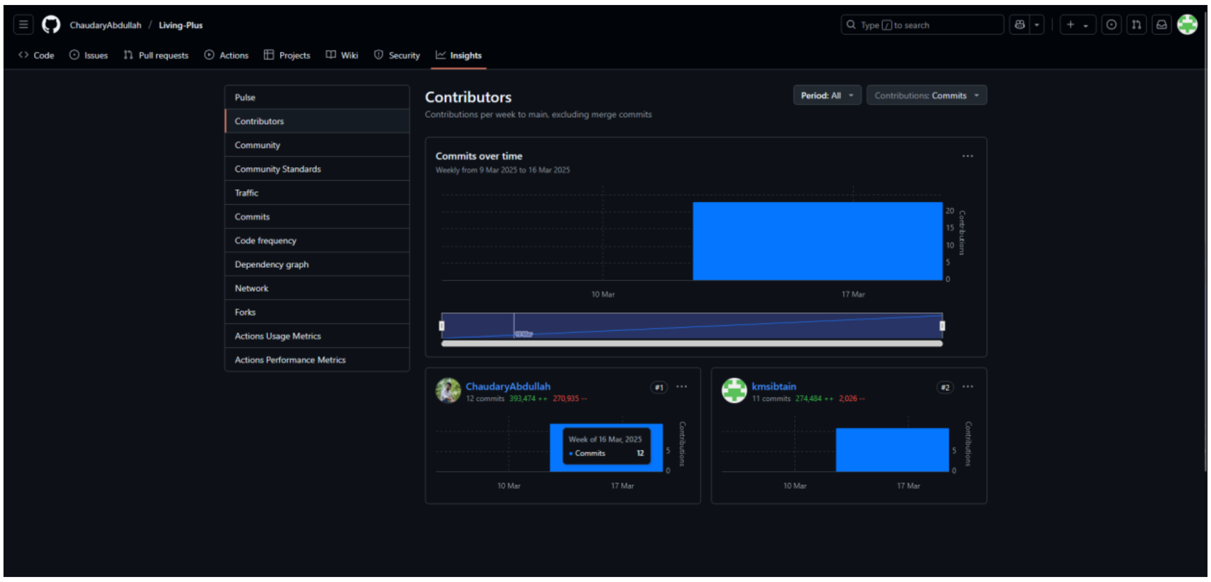
Sibtain
#11 merged 3 hours ago

Session created
#10 merged 3 hours ago

GiveFeedback
#9 merged yesterday

Sibtain
#8 merged yesterday

Contributions:


ChaudaryAbdullah / Living-Plus

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights

Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Actions Usage Metrics
Actions Performance Metrics

Contributors
Contributions per week to main, excluding merge commits

Period: All · Contributions: Commits

Commits over time
Weekly from 9 Mar 2025 to 16 Mar 2025

ChaudaryAbdullah #1
12 commits 393,474 ++ 270,935 --

Week of 16 Mar, 2025
• Commits 12

kmsibtain #2
11 commits 274,484 ++ 2,026 --