Software Requirements and Design Document

For

Living+

Prepared by:

Muhammad Sibtain 22i-0887

Abdullah Shakir 22i-1138

Maaz Khan 22i-2125

23-11-2024

Table of Contents

1. Introduction

1.1 Purpose

The purpose of the LIVING+ project is to provide a streamlined platform for hostel and rental management, catering to users searching for accommodations and property owners managing properties. The document outlines the system's requirements, objectives, and design considerations.

1.2 Product Scope

LIVING+ is a comprehensive solution enabling users to discover, book, and manage accommodations. It focuses on enhancing user control over shared living arrangements through advanced roommate selection features. Property owners benefit from an intuitive dashboard for efficient management.

1.3 Title

LIVING+: Streamlined Hostel and Rental Management System

This project aims to transform the hostel and rental experience by providing a seamless platform for discovering, booking, and managing accommodations. It offers an immediate solution to the challenges of finding suitable living arrangements and managing properties efficiently, ensuring personalized roommate selection and enhanced property management for tenants and owners alike.

1.4 Objectives

- Simplify property discovery and comparison.
- Enhance the user experience through personalized roommate matching.
- Offer property owners tools for streamlined management.
- Ensure transparency and convenience in transactions and operations.

1.5 Problem Statement

Traditional accommodation systems lack personalization and efficient management tools, leading to tenant dissatisfaction and operational inefficiencies. LIVING+ addresses these issues by introducing roommate selection features, streamlined booking processes, and a robust property management interface.

2. Overall Description

2.1 Product Perspective

LIVING+ is a new solution aimed at replacing manual methods of hostel and rental management. It integrates advanced roommate matching, property browsing, and comprehensive management tools into a single platform.

2.2 Product Functions

- **Search and Filter Properties:** Users can search based on location, amenities, price, and availability.
- **Booking and Payment:** Secure accommodations with integrated payment options.
- **Property Management Dashboard:** Tools for property owners to manage rooms, tenants, and maintenance.
- **User Reviews and Ratings:** Feedback system for tenants and property owners.

2.3 List of Use cases

1. Register A Hostel
2. Approve Applicants
3. Monitor Tenant Actions
4. Manage Maintenance
5. Make Eviction
6. Allocate Parking Slot
7. Add Menu
8. Request Parking Slot
9. Make Payment
10. Choose A Rental
11. Give Feedback
12. View Properties
13. Manage Meals
14. Calculate Rent
15. Notifications
16. Due Date Notifications

2.4 Extended Use cases

Use Case#1: (M. Sibtain 22i-0887)

Register A Hostel:

Use Case Name	Register A Hostel
Scope	LIVING+ System
Level	User-goal level
Primary Actor	Hostel/Rental Owner
Stakeholders & Interests	**Hostel/Rental Owner:** Wants an easy and efficient way to list their properties. **Admin:** Ensures that all hostels and rentals meet platform standards.
Preconditions	Hostel/Rental Owner must have an account.
Postconditions	The Hostel/rental is successfully registered and visible to potential tenants.
Main Success Scenario	**Hostel/Rental Owner** 1. The owner logs in to the system. 2. The owner selects "Register a Hostel" from the dashboard. 3. The owner submits the form. **System** 2 The system prompts for property details (location, amenities, pricing). 4 The system validates the submission. 5 The hostel or rental is added to the listing.
Extensions	**3a**: If the property information is incomplete, the system prompts the owner to fill in missing details. **5a**: If the submission fails, the system shows an error message, and the owner can retry.

Use Case#2: (M. Sibtain 22i-0887)

Approve Applicants:

Use Case Name	Approve Applicants
Scope	LIVING+ System
Level	User-goal level
Primary Actor	Hostel/Rental Owner
Stakeholders & Interests	**Hostel/Rental Owner:** Wants to verify tenant details before approval. **Tenant:** Wants quick and smooth approval for room bookings.
Preconditions	Tenant must apply for a room, and the owner must receive the application.
Postconditions	The tenant's application is either approved or rejected.
Main Success Scenario	**Hostel/Rental Owner** / **System** Hostel/Rental Owner: 1. The owner logs in to the system. System: 2. The system displays pending tenant applications. Hostel/Rental Owner: 3. The owner reviews the tenant's details Hostel/Rental Owner: 4 The owner clicks "Approve" or "Reject." System: 5.The system sends tenant a notification.
Extensions	**3a**: If the tenant's details are incomplete, the owner can request additional information. **4a**: If the application is rejected, the system prompts the owner to provide a reason.

Main Success Scenario:

Hostel/Rental Owner	System
1. The owner logs in to the system.	
	2. The system displays pending tenant applications.
3. The owner reviews the tenant's details	
4 The owner clicks "Approve" or "Reject."	
	5.The system sends tenant a notification.

Use Case#3: (M. Sibtain 22i-0887)

Monitor Tenant Actions:

Use Case Name	Monitor Tenant Actions
Scope	LIVING+ System
Level	User-goal level
Primary Actor	Hostel/Rental Owner
Stakeholders & Interests	**Hostel/Rental Owner:** Wants to manage existing tenants by adding costs (e.g., rent, utilities), issuing fines, and monitoring tenant activities. **Tenant**: Hopes for clear and transparent management regarding additional charges and fines.
Preconditions	The tenant must be living in the hostel/rental unit.
Postconditions	The tenant's account is updated with the new charges, fines, or other management actions.
Main Success Scenario	(see below)
Extensions	**3a**: If the tenant disputes the charges, the owner can review and update the charges.. **5a**: If the system fails to process the charge, an error message is shown, and the owner retries the operation. **5b**: The tenant can appeal fines through the system.

Main Success Scenario:

Hostel/Rental Owner	**System**
1. The owner logs in to the system.	
	2.The system displays the list of current tenants.
3. The owner selects a tenant and chooses actions such as: • Add rent charges. • Issue a fine (e.g., for rule violations). • Add utility or maintenance costs.	
	4.The System updates the tenant's account with the new information. 5.System sends tenant a notification.

Use Case#4: (M. Sibtain 22i-0887)

Manage Maintenance:

Use Case Name	Manage Maintenance	
Scope	LIVING+ System	
Level	User-goal level	
Primary Actor	Maintenance Staff	
Stakeholders & Interests	**Hostel/Rental Owner**: Wants maintenance issues resolved quickly & keeps tenants happy. **Tenant:** Wants maintenance issues resolved quickly. **Maintenance Staff:** Needs to receive tasks and update the status of work.	
Preconditions	Maintenance tasks must be assigned.	
Postconditions	The maintenance request is completed or escalated.	
Main Success Scenario	**Hostel/Rental Owner**	**System**
	1. The Maintenance staff logs in to the system.	
		2. The system displays maintenance tasks.
	3. The staff updates the task status (e.g., in progress, completed). 4. The staff add costs for new equipment etc.	
		5. The system sends tenant/owner the bill.
Extensions	**3a**: If a task cannot be completed, the staff can escalate the issue to the admin.	

Use Case#5: (M. Sibtain 22i-0887)

Eviction:

Use Case Name	Eviction
Scope	LIVING+ System
Level	User-goal level
Primary Actor	Hostel/Rental Owner
Stakeholders & Interests	**Hostel/Rental Owner:** Wants to evict tenants who violate rules or fail to pay rent. **Tenant:** Hopes to avoid eviction by resolving issues.
Preconditions	The tenant must be in violation of the rental agreement or complaint lodged by other tenants.
Postconditions	The tenant is either evicted, or the issue is resolved.
Main Success Scenario	**Hostel/Rental Owner** 1. The owner reviews the tenant's issue & history. 2. The owner sends a warning or starts the eviction process. **System** 3. The tenant is notified of the action. 4. The tenant either resolves the issue or is evicted. .
Extensions	**4a**: If the tenant resolves the issue, the eviction process is halted.

Use Case#6: (Abdullah Shakir 22i-1138)

Allocate Parking Slot:

Use Case Name	**Allocate Parking Slot**
Scope	Living+
Level	User-goal level
Primary Actor	Owner
Stakeholders & Interests	**Tenant**: Needs a parking slot for their vehicle. **Owner**: Wants to allocate parking slots efficiently. **Security Guard**: Guarantees security and get payment.
Preconditions	Tenant has requested a slot, and availability has been confirmed.
Postconditions	Tenant receives confirmation of the parking slot allocation.
Main Success Scenario	**Owner** 1. Owner reviews the tenant's request. 2. Owner looks for the pending dues of the tenant 3. Owner allocates an available parking slot. **System** 4. System allocates a slot and notifies the tenant.
Extensions	1a. If the tenant has fulfilled his requirements then the system notifies the tenant that they already reached their limit (upgrade your package). 2a. If the tenant has pending dues then system notifies him for due clearance. 4a. If no slots are available, the system informs the tenant of unavailability and adds them to the waiting list.

Use Case#7: (Abdullah Shakir 22i-1138)

Add Menu:

Use Case Name	**Add Menu**
Scope	Living+
Level	User-goal level
Primary Actor	Owner
Stakeholders & Interests	**Tenant**: Expects a clear and updated menu. **Owner**: Wants to add new items to the menu for tenants. **Chef**: Cooks food according to the menu.
Preconditions	The owner is logged into the system and has access to menu management functions.
Postconditions	The new menu item is added successfully, and tenants can view the updated menu.
Main Success Scenario	**Owner**: 1. Owner selects the option to add a new menu item. 3. Owner provides the required information. **System**: 2. System prompts for details 4. System saves the new menu to the database. 5. The system displays the updated menu for the tenants.
Extensions	3a. If the details provided are incomplete, the system prompts the owner to fill in the missing information.

Main Success Scenario:

Owner	**System**
1. Owner selects the option to add a new menu item.	
	2. System prompts for details
3. Owner provides the required information.	
	4. System saves the new menu to the database.
	5. The system displays the updated menu for the tenants.

Use Case#8: (Abdullah Shakir 22i-1138)

Request Parking Slot:

Use Case Name	**Request Parking Slot**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Needs to park their vehicle in the allotted space. **Owner**: Needs to ensure that parking slots are managed properly. **Security Guard**: Guarantees security and get payment. **Tax department**: collects tax
Preconditions	Tenant must have a valid rental or hostel contract. Parking slots are available.
Postconditions	Tenant is allocated a parking slot, or notified of unavailability.
Main Success Scenario	**Tenant** 1. Tenant request for parking slot. 4. Tenant parks their vehicle. **System** 2. System checks for the requirements of the tenant's application and checks for the pending dues. 3. System allocates a slot and notifies the tenant.
Extensions	2a. If the tenant has pending dues then system notifies him for due clearance. 2b. If the tenant has fulfilled his requirements then the system notifies the tenant that they already reached their limit (upgrade your package). 3a. If no slots are available, the system informs the tenant of unavailability and adds them to the waiting list.

Use Case#9: (Abdullah Shakir 22i-1138)

Make Payment:

Use Case Name	**Make Payment**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wants to pay rent or other charges conveniently. **Owner**: Wants to track tenant payments and ensure timely processing.
Preconditions	The tenant has an outstanding balance, and the system is connected to payment services.
Postconditions	The payment is successfully processed, and both the tenant and owner are notified of the completed transaction.
Main Success Scenario	
Extensions	4a. If payment details are invalid, the system prompts the tenant to re-enter valid details. 6a. If the payment fails, the system notifies the tenant and provides troubleshooting options.

Tenant	**System**
1. Tenant logs into the system and views the outstanding balance.	
2. Tenant selects the option to make a payment.	
	3. The system provides payment options (credit card, online transfer, etc.).
4. Tenant enters payment details and confirms the transaction.	
	5. The system processes the payment.
	6. The system updates the payment status and sends a receipt to the tenant.
	7. Owner is notified of the payment.

Use Case#10: (Abdullah Shakir 22i-1138)

Rent a Rental:

Use Case Name	**Rent a Rental**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wants to rent a rental in the hostel. **Owner**: Needs to manage rental allocation and tenant details.
Preconditions	Rentals are available, and the tenant has registered or logged into the system.
Postconditions	The rental is successfully rented to the tenant, and both tenant and owner are notified.
Main Success Scenario	**Tenant** 1. Tenant logs into the system. 2. Tenant searches for available rentals. 4. Tenant selects a rental to rent. 5. Tenant agrees to the terms and conditions. **System** 3. The system displays available rentals with details (price, size, amenities). 7. The system confirms the selection and processes the booking. 8. Owner is notified of the new rental.
Extensions	3a. If no rentals are available, the system informs the tenant and offers a waitlist option. 5a. If the tenant does not agree to the terms, the process is terminated.

Use Case#11: (Abdullah Shakir 22i-1138)

Provide Feedback on Rental:

Use Case Name	**Provide Feedback on Rentals**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wants to provide feedback about the rental experience. **Owner**: Wants to gather feedback for service improvement and record tenant satisfaction.
Preconditions	The tenant has rented a rental, and the feedback option is available in the system.
Postconditions	The feedback is submitted successfully and is available for review by the owner.
Main Success Scenario	**Tenant** / **System** 1. Tenant logs into the system and selects the option to provide feedback. 2. The system presents a feedback form with fields for ratings (e.g., cleanliness, service, facilities) and comments. 3. Tenant fills out the form and submits the feedback. 4. The system stores the feedback and acknowledges submission. 5. Owner receives a notification that feedback has been submitted.
Extensions	3a. If mandatory fields are not filled, the system prompts the tenant to complete the form before submitting.

Use Case#12: Maaz Khan (22i-2125)

View Properties (Rental):

Use Case Name	**View Properties (Rental)**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wants to find suitable rental accommodations based on preferences. **Owner**:Wants to attract potential tenants and wants the rental service work perfectly.
Preconditions	Properties must be listed on the platform.
Postconditions	• Tenant is able to view the list of rental properties and can compare them based on preferences (price, location, amenities). • Tenant has the option to inquire or reserve a property
Main Success Scenario	**Tenant** 1. User logs in and navigates to the "View Properties" section. 2. User applies filters to find suitable properties. **System** 3. System retrieves properties based on applied filters. 4. System displays the results in a list format with detailed information.
Extensions	2a. If no properties match the criteria, the system notifies the user and offers to modify the filters or get notified when matching properties are available. 4a. If a property is unavailable for the chosen dates, the system displays availability options or adds the user to a notification list.

Use Case#13: Maaz Khan (22i-2125)

Manage Meals:

Use Case Name	**Manage Meals**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wants to subscribe for to a meal plan provided by the rental . **Owner**: Offers meal plans and needs a way to manage meal preferences and payments. **Meal Service Provider**: Prepares and provides meals for the tenants.
Preconditions	• Tenant must be registered in the respective rental. • Meal plans must be available for selection.
Postconditions	• Tenant successfully subscribes to or cancels a meal plan.
Main Success Scenario	**Tenant** / **System** Tenant: 1. Tenant navigates to "Manage Meals" and selects a meal plan. System: 2. System displays available meal plans with pricing and options. System: 3. System records the tenant's selection and sends it to the meal service provider.
Extensions	2a. If no meal plans are available, the system notifies the tenant and offers updates when they become available.

Main Success Scenario:

Tenant	**System**
1. Tenant navigates to "Manage Meals" and selects a meal plan.	
	2. System displays available meal plans with pricing and options. 3. System records the tenant's selection and sends it to the meal service provider.

Use Case#14: Maaz Khan (22i-2125)

Calculate Rent:

Use Case Name	**Calculate Rent**
Scope	Living+
Level	User-goal level
Primary Actor	Hostel/Rental Owner
Stakeholders & Interests	**Hostel/Rental Owner:** Wants to easily calculate rent for each tenant, considering any additional costs (e.g., utilities, fines, maintenance fees) and ensure rent payments are accurate and up to date. **Tenant:** Wants a transparent and accurate rent calculation that reflects agreed terms and any additional costs. **Admin:** Ensures system calculates and displays accurate rent information according to the policies of the hostel.
Preconditions	The tenant must be residing in the hostel or rental property. The tenant's rent agreement must be present in the system, including rent amount, payment terms, and any applicable charges or discounts.
Postconditions	The system calculates the total rent for the tenant, including additional charges. The tenant's account is updated with the final rent amount for the upcoming billing period. The tenant receives a notification about the rent calculation.

Main Success Scenario	**Tenant**	**System**
	1. The owner logs into the system. 2. The owner selects the option "Calculate Rent" from the dashboard.	
		3.The system displays a list of current tenants.
	4. The owner selects a tenant to calculate rent for.	
		5. The system retrieves the tenant's base rent from the rental agreement. 6. The system adds any additional costs (e.g., utilities, maintenance fees, fines). 7. The system applies any discounts or adjustments

		8. The system calculates the total rent due for the tenant.
	9. The owner confirms the calculation.	
		10. The system updates the tenant’s account with the rent amount.
Extensions	**6a.** If additional costs have not been added yet (e.g., utility costs), the system prompts the owner to input those values before proceeding with the rent calculation. **9a.** If the owner detects any incorrect details (e.g., incorrect base rent), the owner can manually edit the rent details and recalculate	

Use Case#15: Maaz Khan (22i-2125)

Follow Rentals (Notify Availability):

Use Case Name	**Follow Rentals (Notify Availability)**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Wishes to receive notifications about specific rental properties. **Owner**: Wants to ensure properties are booked promptly when available.
Preconditions	Properties must be listed on the platform.
Postconditions	Tenant is notified when the selected rental becomes available.
Main Success Scenario	**Tenant** / **System** 1. Tenant opts for the "Follow Availability" options. 2. System records the follow request and monitors the property's availability. 3. When the property becomes available, the system sends a notification to the tenant via email or app notification. 4. The user receives notification regarding updates.
Extensions	3a. If the property becomes unavailable due to another booking, the system informs the tenant and suggests similar available properties

Main Success Scenario:

Tenant	System
1. Tenant opts for the "Follow Availability" options.	
	2. System records the follow request and monitors the property's availability.
	3. When the property becomes available, the system sends a notification to the tenant via email or app notification.
4. The user receives notification regarding updates.	

Use Case#16: Maaz Khan (22i-2125)

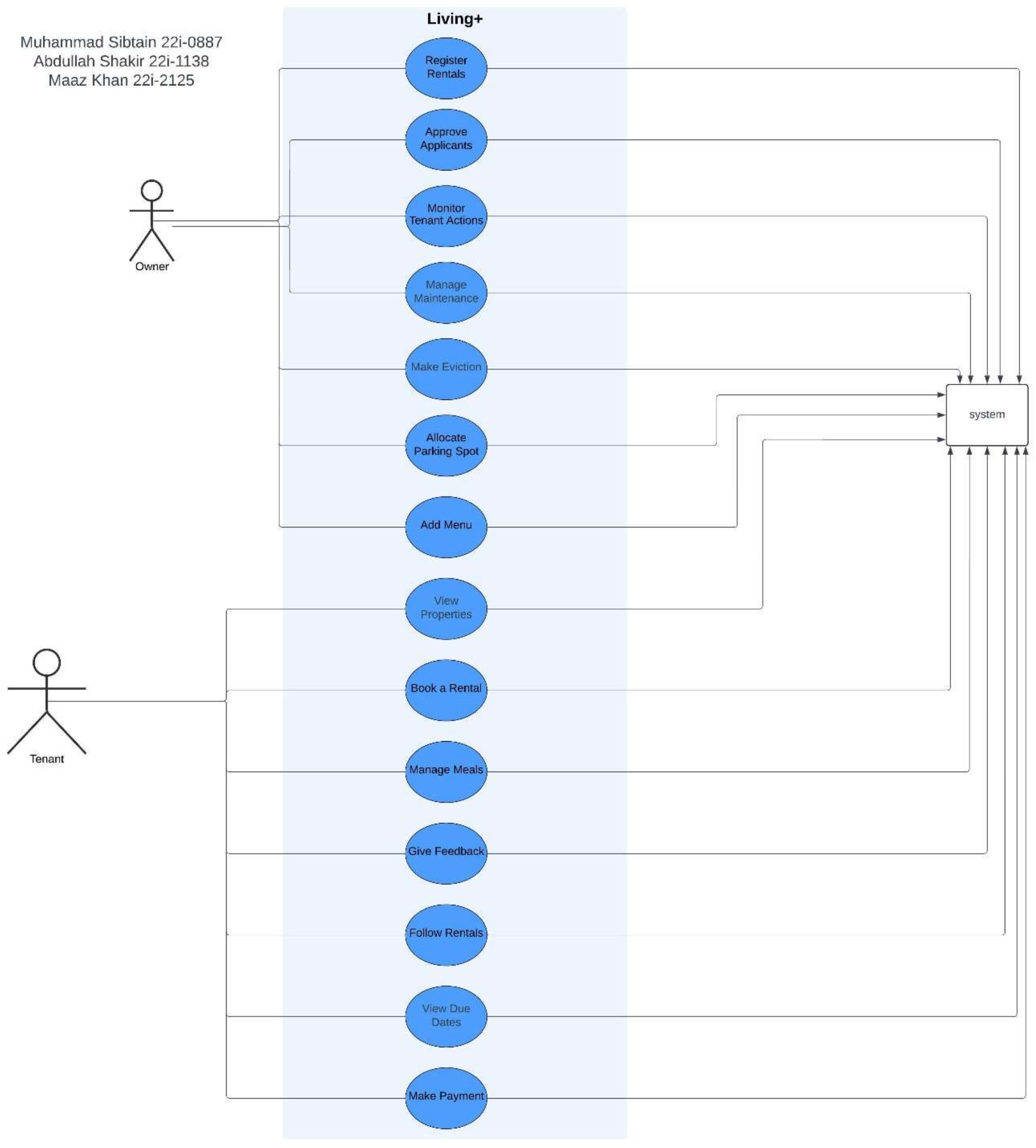
View Due Date Notification:

Use Case Name	**View Due Date Notification**
Scope	Living+
Level	User-goal level
Primary Actor	Tenant
Stakeholders & Interests	**Tenant**: Needs to be reminded of rent payment deadlines to avoid late fees. **Owner**: Wants to ensure timely payment from tenants.
Preconditions	Tenant must have a valid contract and rent payment schedule.
Postconditions	Tenant is notified of upcoming rent payment deadlines.
Main Success Scenario	see below

Tenant	**System**
1. Tenant logs in and goes to the notification section.	
	2. System checks the tenant's payment schedule and identifies upcoming due dates.
	3. System sends notifications to the tenant reminding them of the due date.
4. The tenant is able to see the updates in the section.	

Extensions	2a. If the tenant has a history of late payments, the system sends an additional early reminder. 3a. If the tenant has already paid, the system updates their payment status and cancels the notification.

Usecase Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

1. **Concurrent User Support:**

- The system should support up to **50 concurrent users** during peak usage times without noticeable performance degradation.

2. Response Time:

- Search queries for properties or roommates should return results within **5 seconds**.
- Booking or transaction processing should complete within **10 seconds**.

3. Data Handling:

- The system should be able to handle a database of **up to 5,000 records** (e.g., properties, users, reviews) without performance degradation.

4. Availability:

- The system should be operational at least **95% of the time** during testing, with planned downtime for maintenance.

5. Scalability:

- The system should support a **10% increase** in user traffic and data size without requiring major architectural changes.

6. Error Recovery:

- The system should recover from minor errors (e.g., failed transactions or interruptions) within **30 seconds** and log the errors for troubleshooting.

7. Backup:

- All data should be backed up automatically once every **24 hours** to ensure data integrity during unexpected failures.

8. Throughput:

- The system should process at least **10 bookings or transactions per minute** under normal conditions.

3.2 Safety Requirements

- The system must have safeguards to prevent data loss during unexpected shutdowns or hardware failures.
- Backups of all data must occur daily to ensure data integrity.
- Error-handling mechanisms should provide clear recovery options to users in case of system failures.

3.3 Security Requirements

- Regular security audits should be conducted to identify vulnerabilities.
- Access control must be role-based, ensuring that users can only access data relevant to their roles.
- All login attempts and major actions must be logged for security monitoring.

3.4 Software Quality Attributes

- **Adaptability:** The platform should be flexible to incorporate new features like additional payment gateways or advanced filters.
- **Usability:** The interface must be intuitive and easy to use, requiring no more than two training sessions for property owners.
- **Maintainability:** The codebase should be modular and well-documented to support future updates with minimal effort.
- **Reliability:** The system should recover automatically from minor errors to ensure consistent service.
- **Scalability:** The platform must handle a 50% increase in traffic without significant performance degradation.
- **Interoperability:** The platform should integrate seamlessly with third-party services like Google Maps and payment gateways.

3.5 Business Rules

Only verified users can post or book properties.

Property owners must respond to booking requests within 48 hours, or the request will expire.

Transactions and bookings are subject to a service fee, outlined transparently at the time of booking.

Reviews can only be posted by users after a completed stay.

Properties with unresolved maintenance issues cannot be listed as available.

3.6 Operating Environment

1. Hardware Requirements:

- **Processor:** Minimum Intel Core i3 (or equivalent) processor.
- **RAM:** At least 4 GB (8 GB recommended for better performance).
- **Storage:** Minimum 500 MB free disk space for application installation and database.

2. Operating System:

- Windows 10/11 (64-bit) or macOS 10.14+ (if cross-platform support is implemented).

3. Software Requirements:

- **Java Runtime Environment (JRE):** JDK 11 or later, compatible with JavaFX.
- **Database Management System:** MySQL Community Server 8.0 running on localhost.
- **Integrated Development Environment (IDE):** IntelliJ IDEA, Eclipse, or NetBeans (for development and debugging).

4. Database Configuration:

- MySQL database hosted locally with connection credentials securely stored in the application configuration file.
- Default MySQL port (3306) or custom port configuration as per user setup.

5. Networking Requirements:

- Local network access to connect the JavaFX application with the MySQL server on localhost.

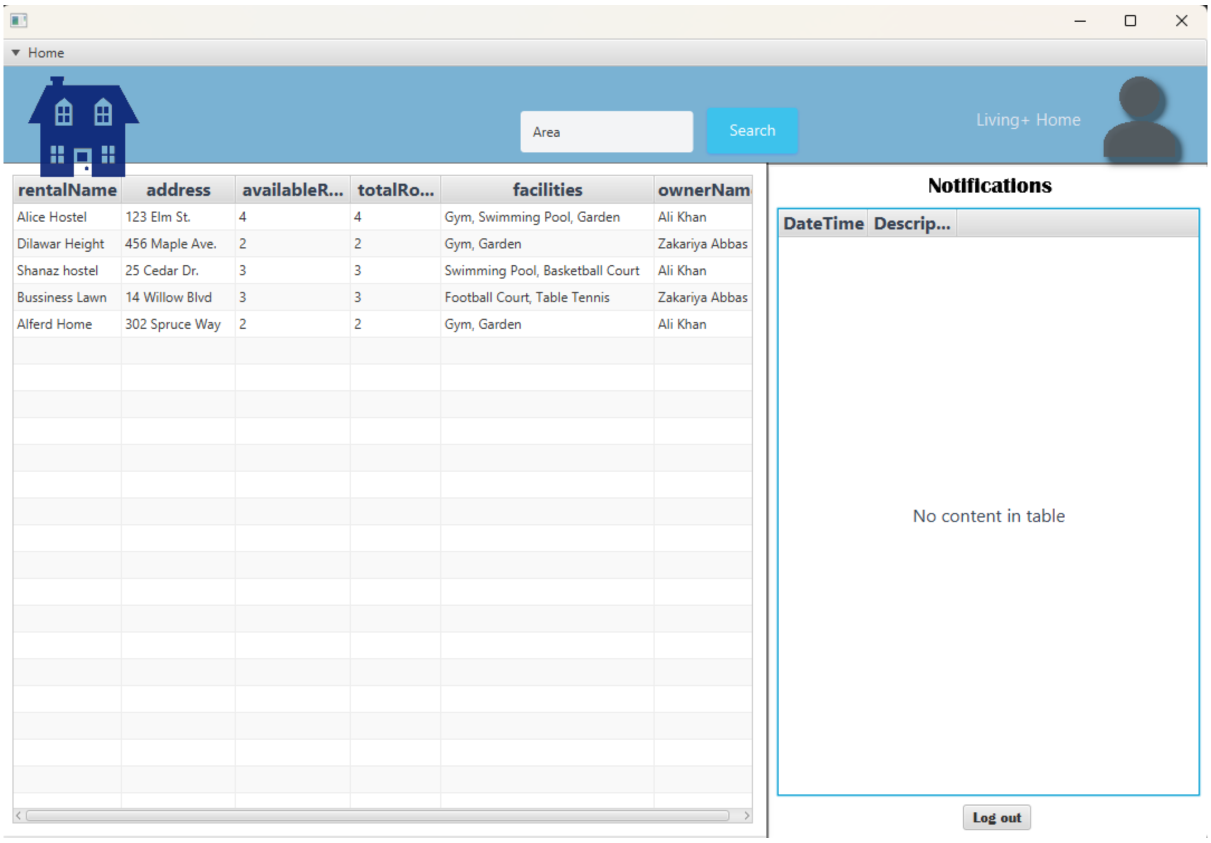
6. User Interface Compatibility:

- The application should be optimized for a minimum screen resolution of **1366x768**.
- Supports mouse and keyboard for interaction; touch support optional.

7. Additional Requirements:

- **Libraries:** JavaFX SDK, MySQL Connector/J for Java database connectivity.
- **Security:** Localhost database should be secured with a username and password to prevent unauthorized access.
- **Testing Environment:** Application should be tested in a single-user environment with localhost configurations.

3.7 User Interfaces

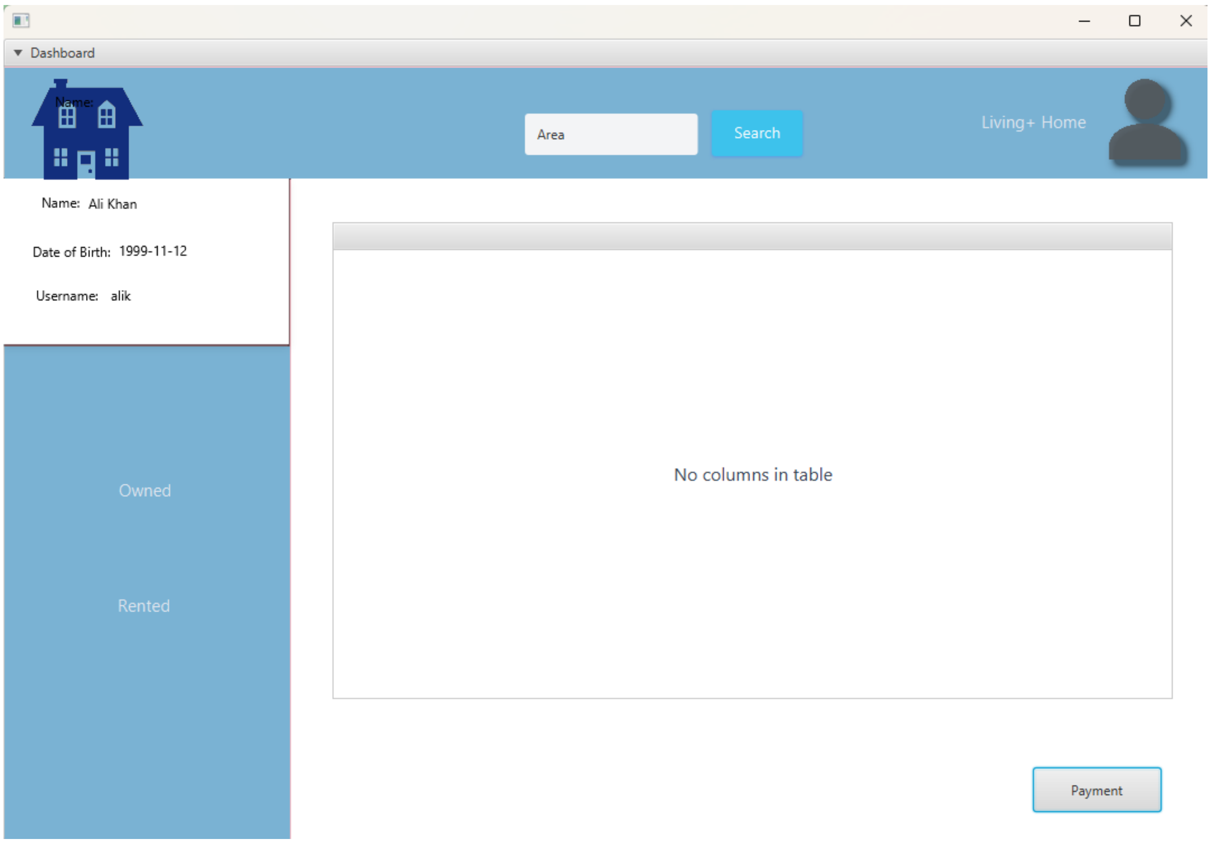


rentalName	address	availableR...	totalRo...	facilities	ownerNam
Alice Hostel	123 Elm St.	4	4	Gym, Swimming Pool, Garden	Ali Khan
Dilawar Height	456 Maple Ave.	2	2	Gym, Garden	Zakariya Abbas
Shanaz hostel	25 Cedar Dr.	3	3	Swimming Pool, Basketball Court	Ali Khan
Bussiness Lawn	14 Willow Blvd	3	3	Football Court, Table Tennis	Zakariya Abbas
Alferd Home	302 Spruce Way	2	2	Gym, Garden	Ali Khan

DateTime	Descrip...

No content in table

Description: User Homepage where it gets information of available rentals & notification. Logo in the top right corner takes user to the Dashboard.



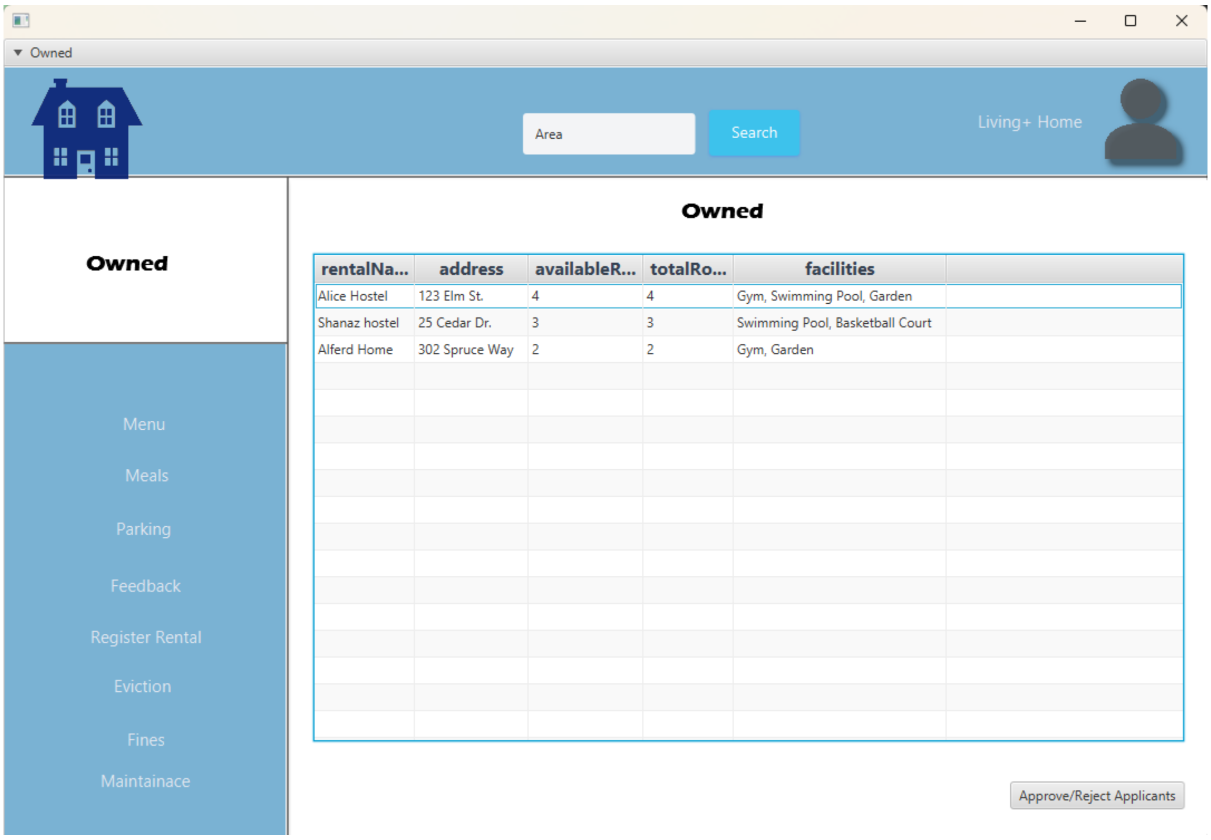
Name: Ali Khan

Date of Birth: 1999-11-12

Username: alik

No columns in table

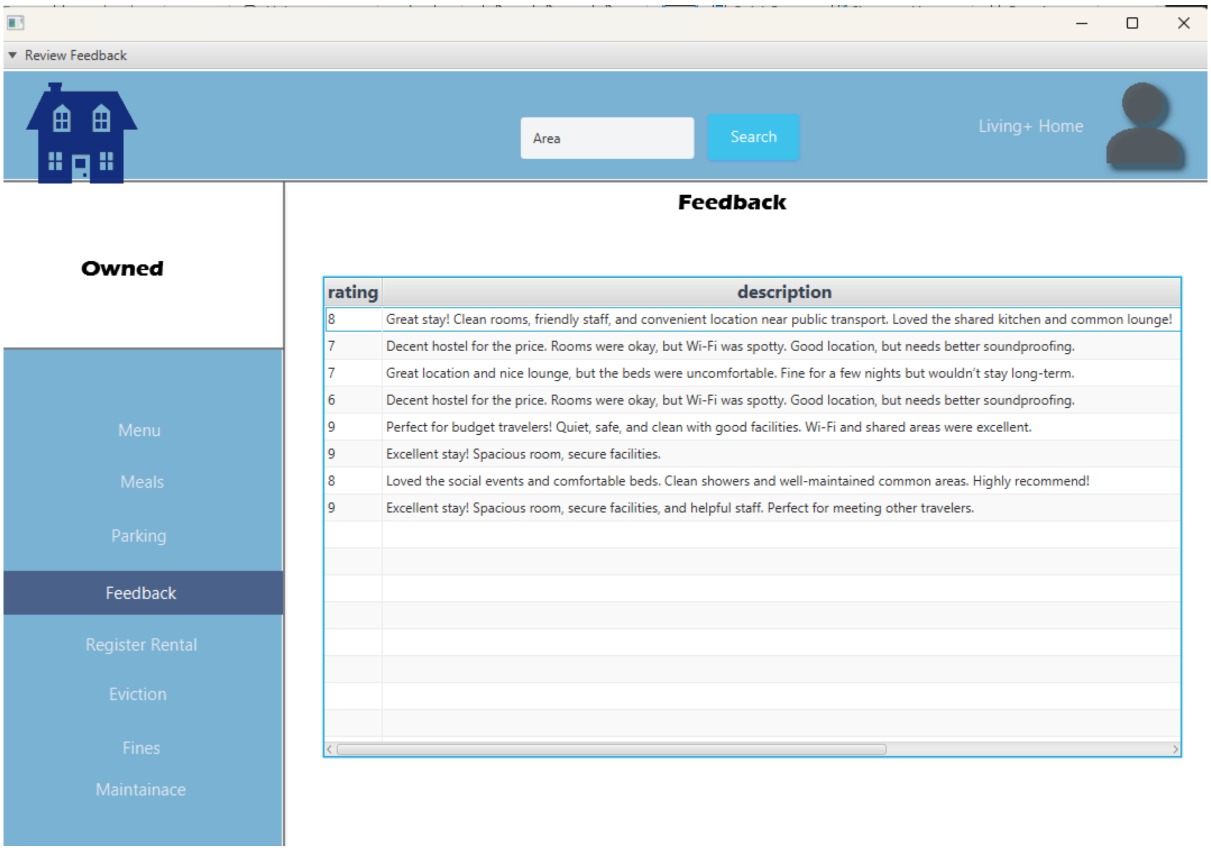
Description: User has option to proceed as owner or an renter.



Owned

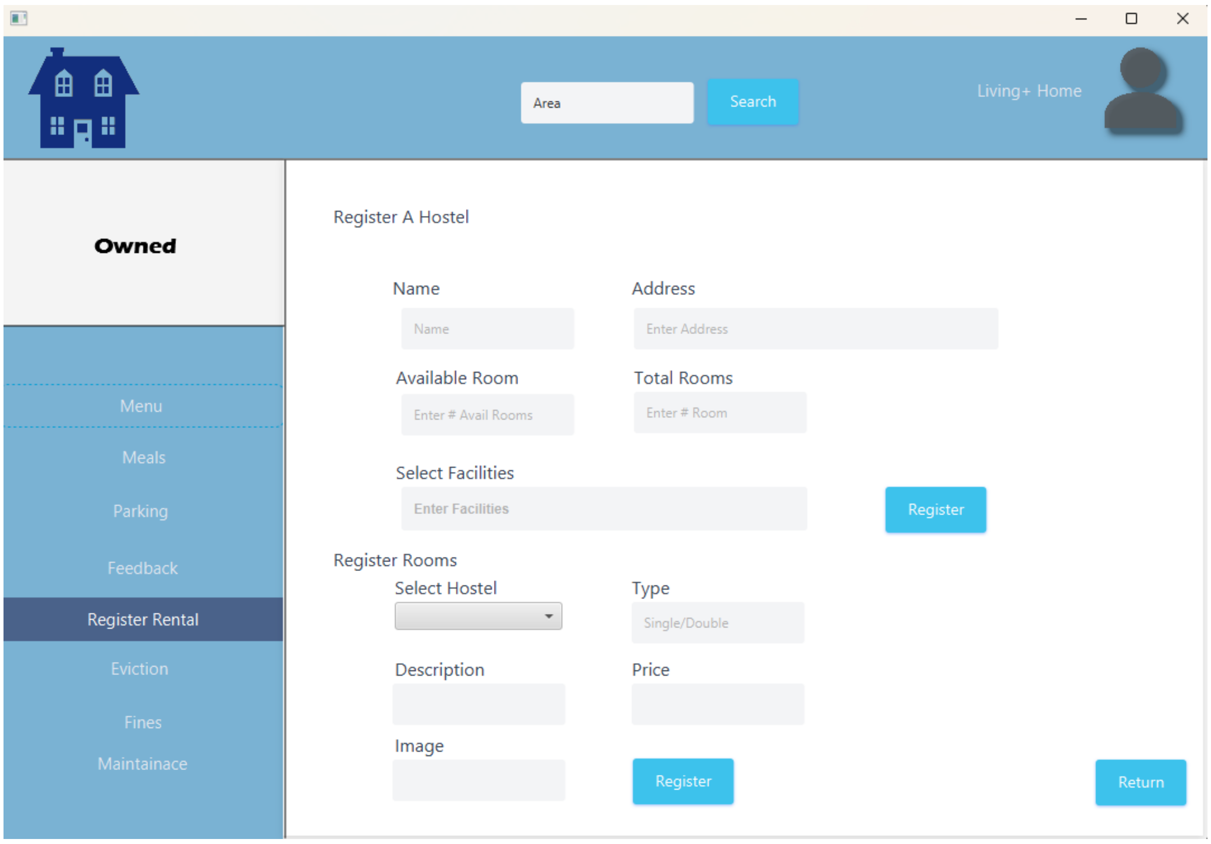
rentalNa...	address	availableR...	totalRo...	facilities
Alice Hostel	123 Elm St.	4	4	Gym, Swimming Pool, Garden
Shanaz hostel	25 Cedar Dr.	3	3	Swimming Pool, Basketball Court
Alferd Home	302 Spruce Way	2	2	Gym, Garden

Description: Owner Home shows hostels owned by the owner. And other tabs to navigate through the options.

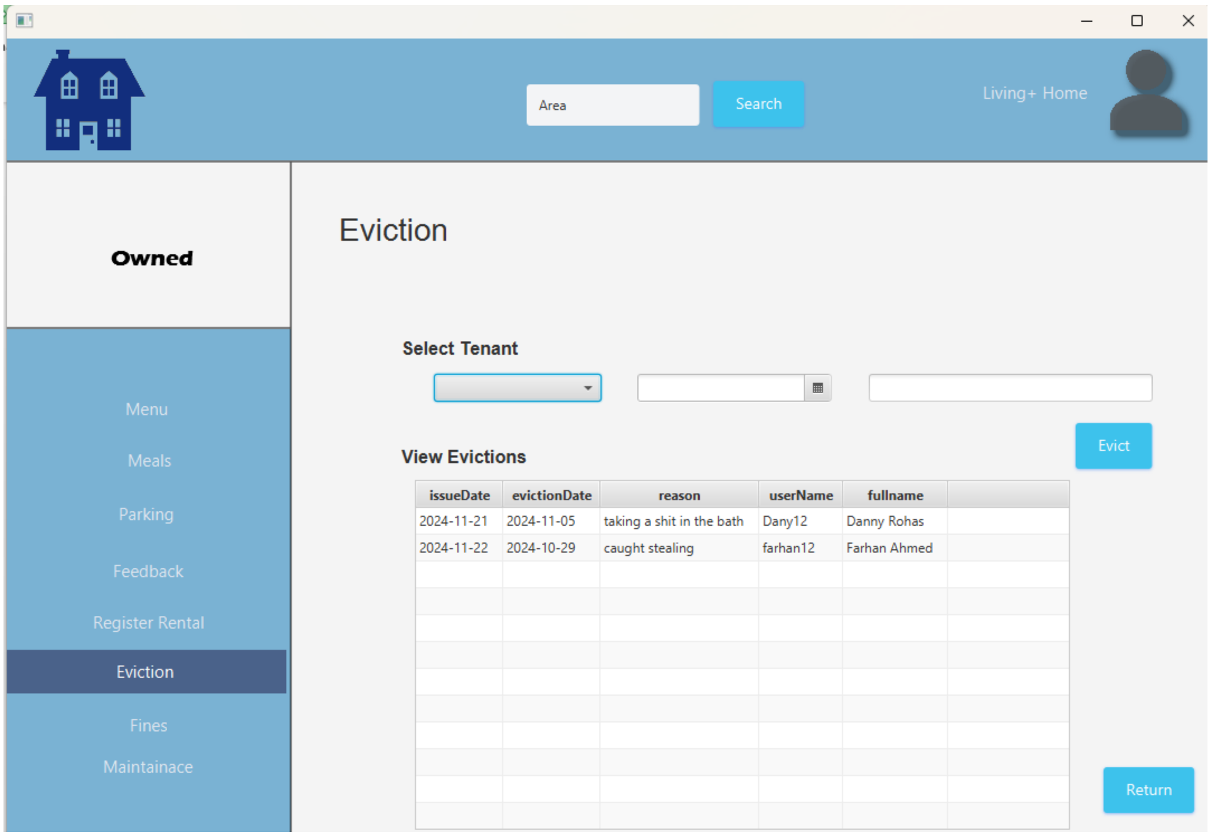


rating	description
8	Great stay! Clean rooms, friendly staff, and convenient location near public transport. Loved the shared kitchen and common lounge!
7	Decent hostel for the price. Rooms were okay, but Wi-Fi was spotty. Good location, but needs better soundproofing.
7	Great location and nice lounge, but the beds were uncomfortable. Fine for a few nights but wouldn't stay long-term.
6	Decent hostel for the price. Rooms were okay, but Wi-Fi was spotty. Good location, but needs better soundproofing.
9	Perfect for budget travelers! Quiet, safe, and clean with good facilities. Wi-Fi and shared areas were excellent.
9	Excellent stay! Spacious room, secure facilities.
8	Loved the social events and comfortable beds. Clean showers and well-maintained common areas. Highly recommend!
9	Excellent stay! Spacious room, secure facilities, and helpful staff. Perfect for meeting other travelers.

Description: Shows all reviews on from the renters.

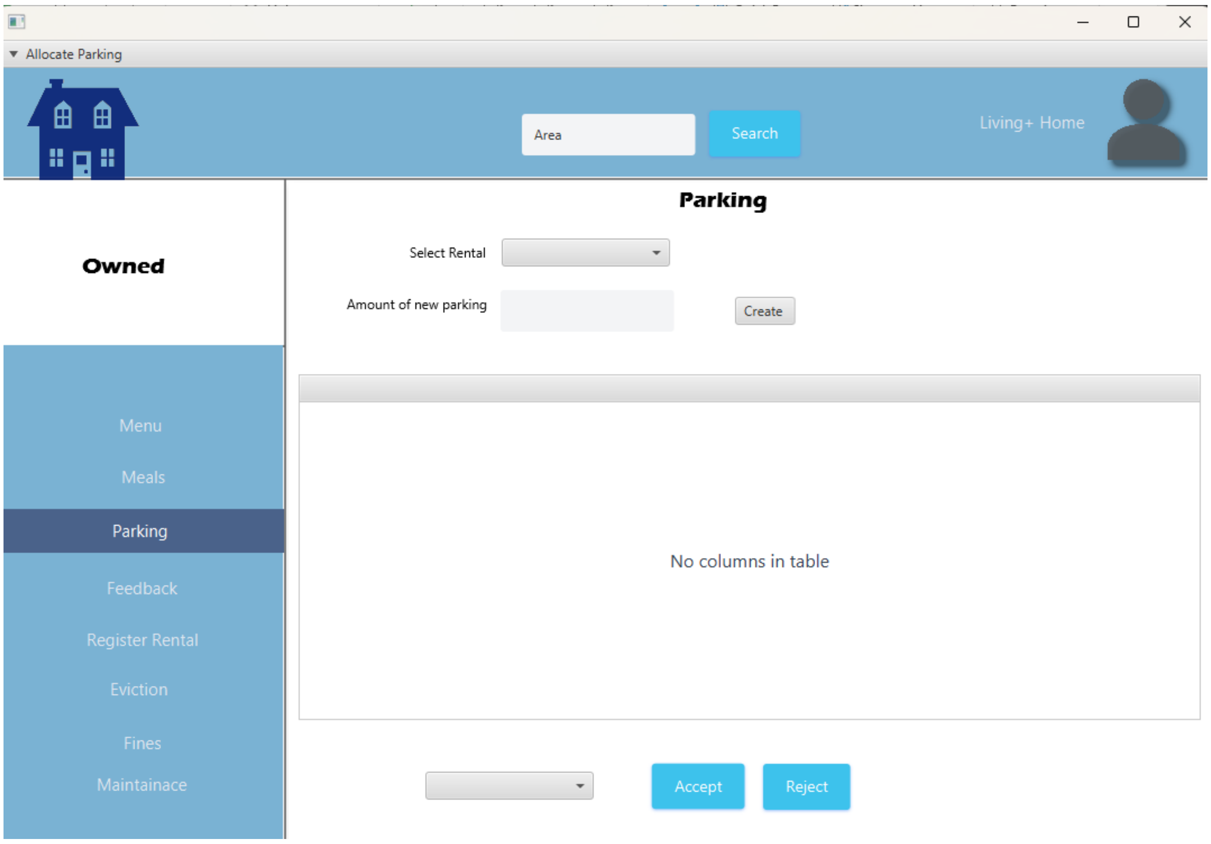


Description: Allows owner to register new hostel and add new rooms.

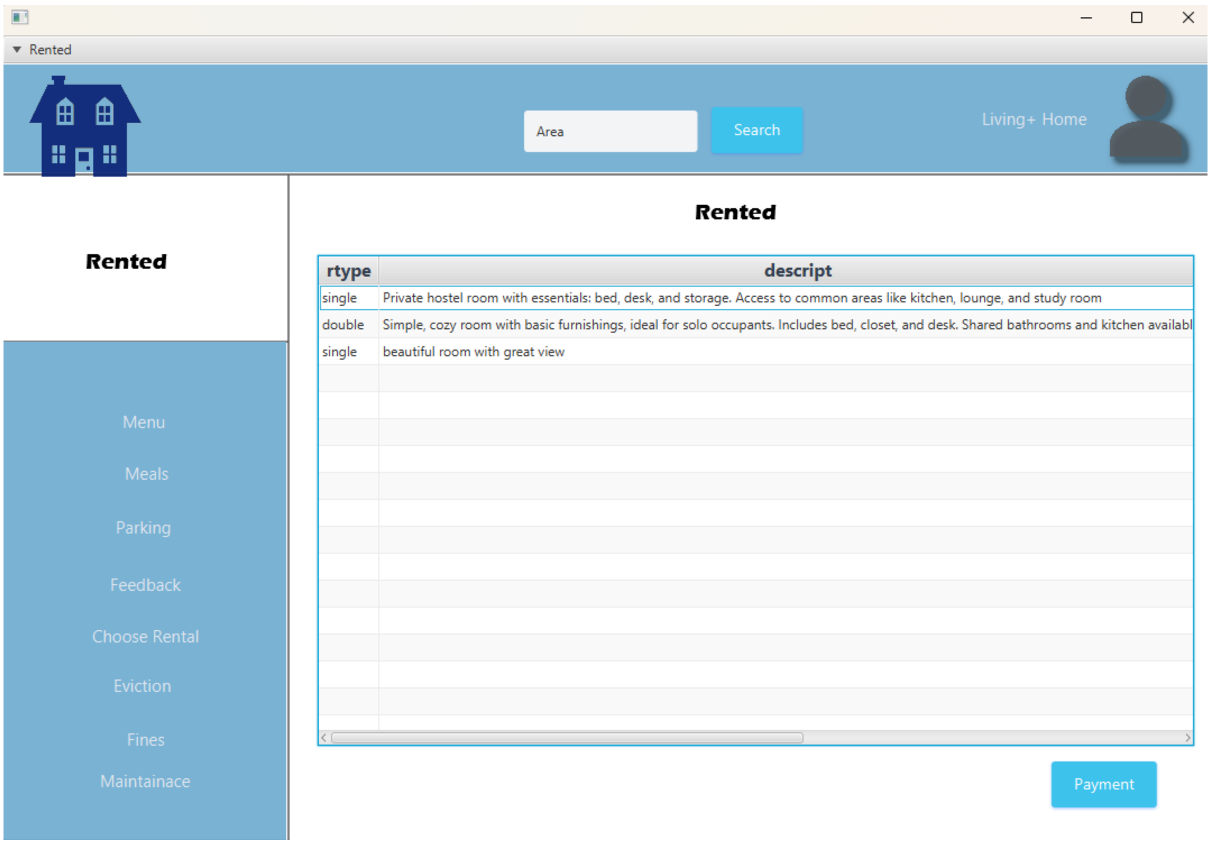


issueDate	evictionDate	reason	userName	fullname
2024-11-21	2024-11-05	taking a shit in the bath	Dany12	Danny Rohas
2024-11-22	2024-10-29	caught stealing	farhan12	Farhan Ahmed

Description: Owner can evict any its tenants.



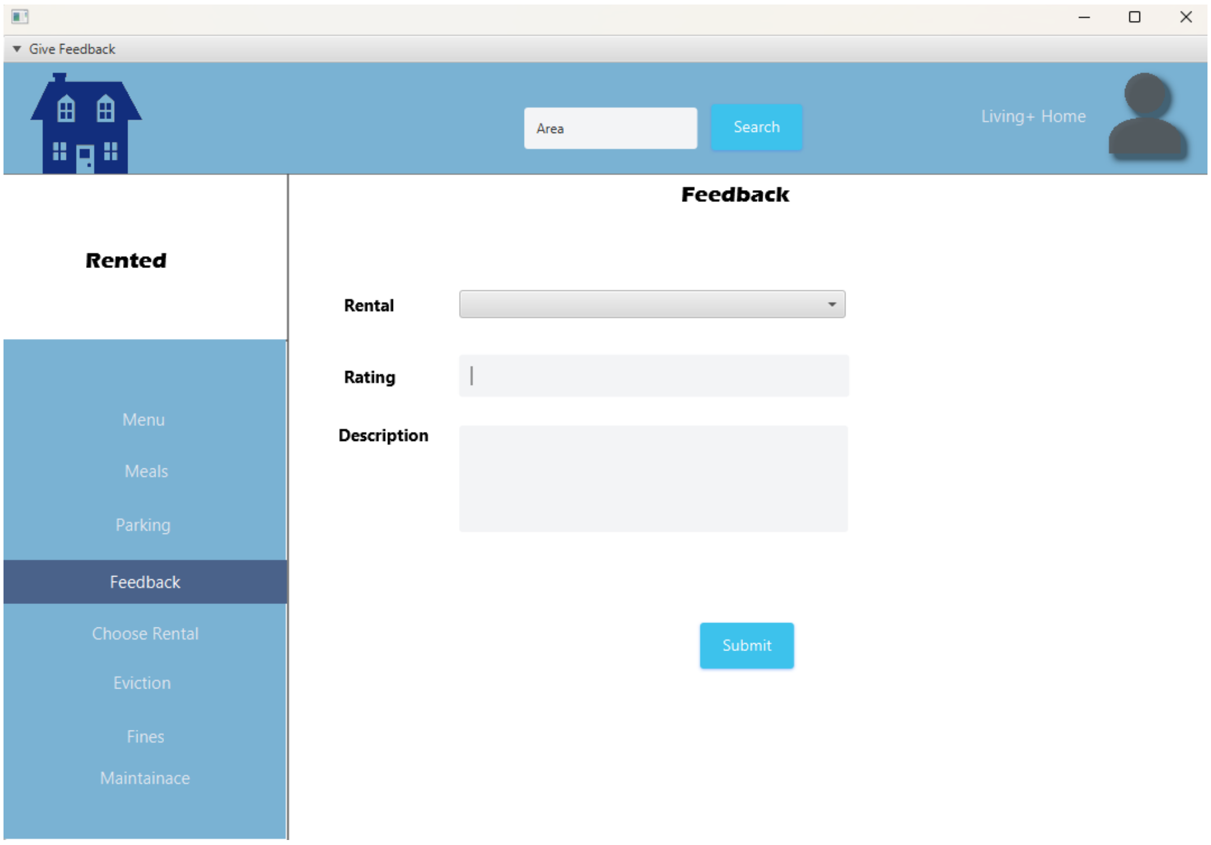
Description: Owner can add new parking slots for its hostels.



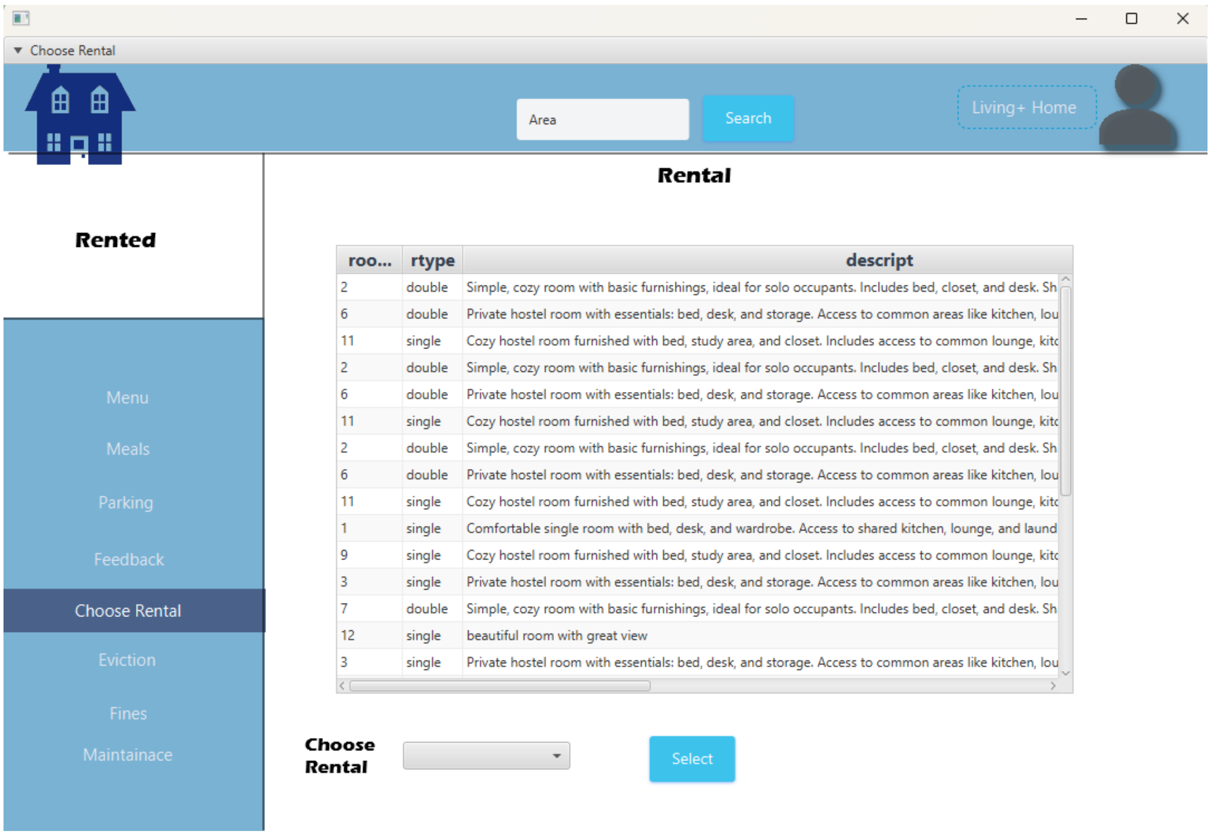
Rented

rtype	descript
single	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lounge, and study room
double	Simple, cozy room with basic furnishings, ideal for solo occupants. Includes bed, closet, and desk. Shared bathrooms and kitchen availabl
single	beautiful room with great view

Description: Renters homepage shows its rental information.



Description: tenant can provide anonymous feedback.

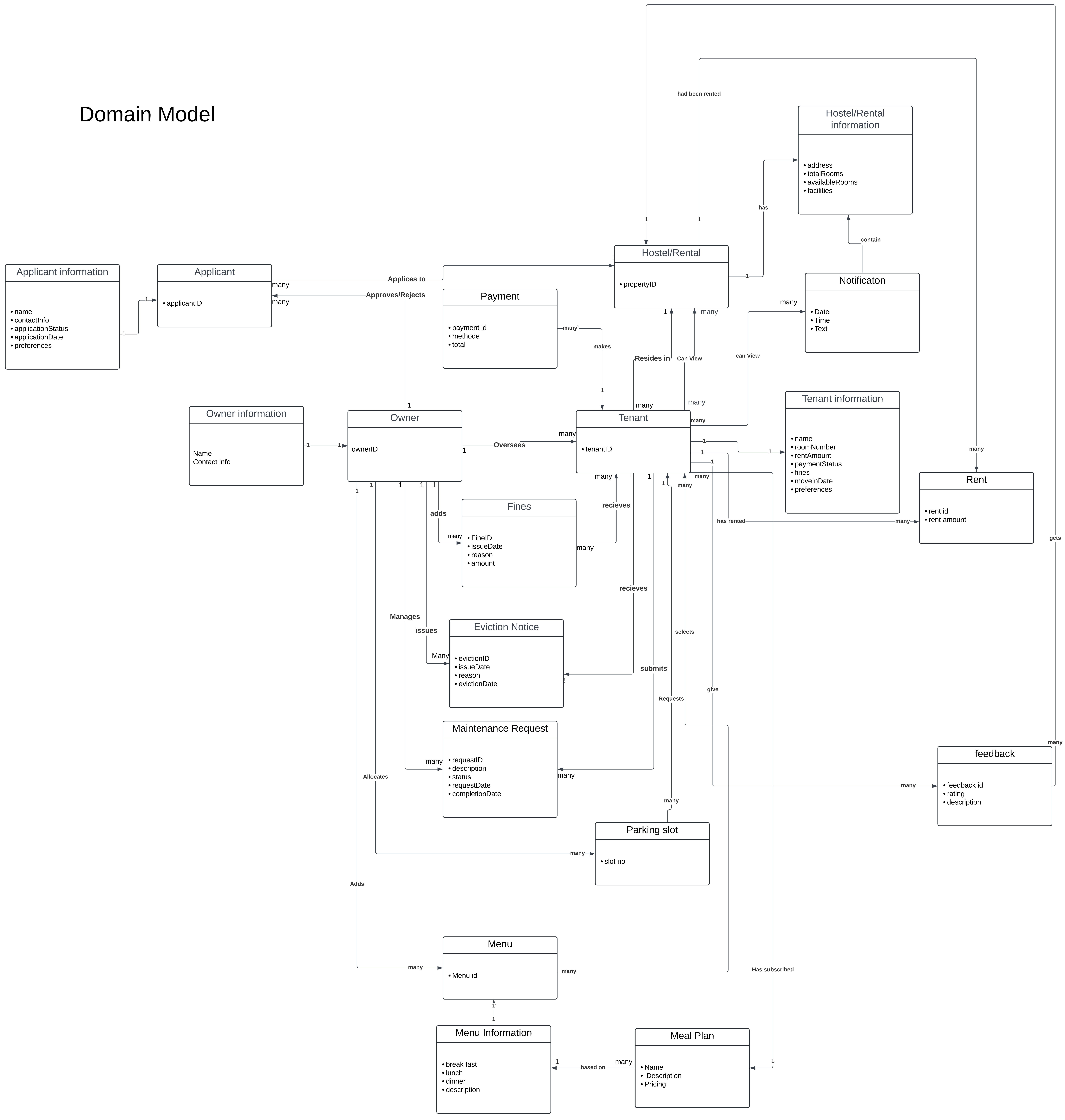


Rental

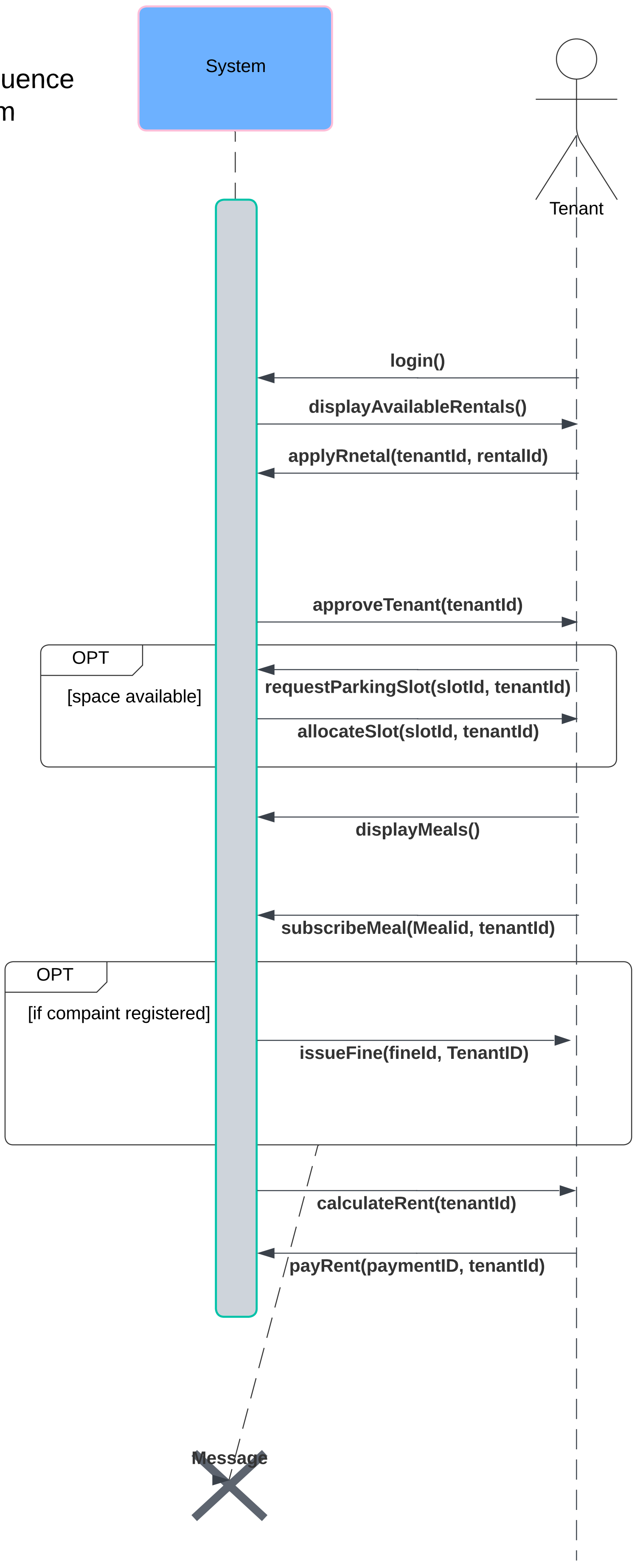
roo...	rtype	descript
2	double	Simple, cozy room with basic furnishings, ideal for solo occupants. Includes bed, closet, and desk. Sh
6	double	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lou
11	single	Cozy hostel room furnished with bed, study area, and closet. Includes access to common lounge, kitc
2	double	Simple, cozy room with basic furnishings, ideal for solo occupants. Includes bed, closet, and desk. Sh
6	double	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lou
11	single	Cozy hostel room furnished with bed, study area, and closet. Includes access to common lounge, kitc
2	double	Simple, cozy room with basic furnishings, ideal for solo occupants. Includes bed, closet, and desk. Sh
6	double	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lou
11	single	Cozy hostel room furnished with bed, study area, and closet. Includes access to common lounge, kitc
1	single	Comfortable single room with bed, desk, and wardrobe. Access to shared kitchen, lounge, and laund
9	single	Cozy hostel room furnished with bed, study area, and closet. Includes access to common lounge, kitc
3	single	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lou
7	double	Simple, cozy room with basic furnishings, ideal for solo occupants. Includes bed, closet, and desk. Sh
12	single	beautiful room with great view
3	single	Private hostel room with essentials: bed, desk, and storage. Access to common areas like kitchen, lou

Description: tenant can choose from variety of different options to Rent.

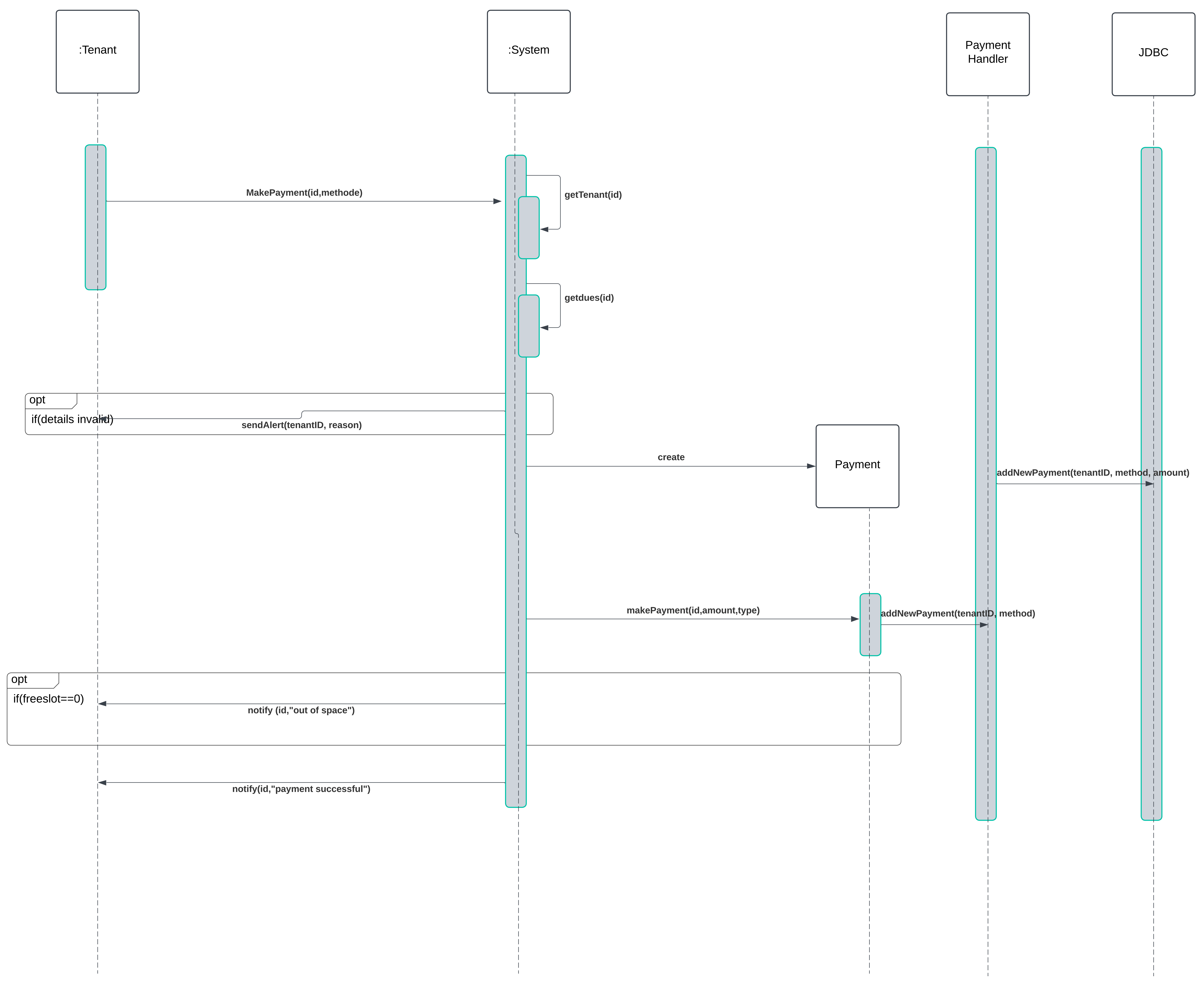
Domain Model



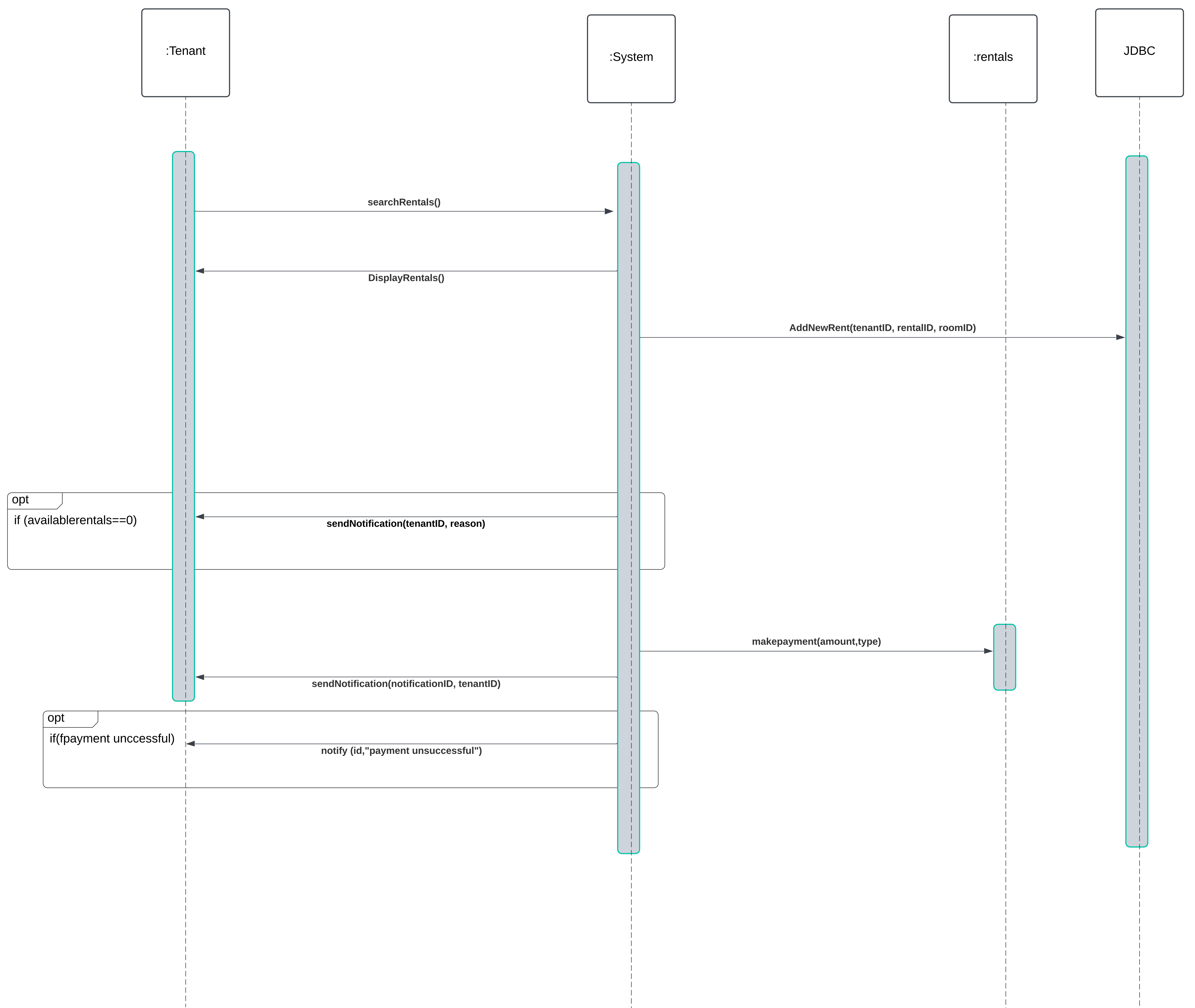
System Sequence Diagram


System
Tenant
login()
displayAvailableRentals()
applyRnetal(tenantId, rentalId)
approveTenant(tenantId)
OPT
[space available]
requestParkingSlot(slotId, tenantId)
allocateSlot(slotId, tenantId)
displayMeals()
subscribeMeal(Mealid, tenantId)
OPT
[if compaint registered]
issueFine(fineId, TenantID)
calculateRent(tenantId)
payRent(paymentID, tenantId)
Message

Make Payment

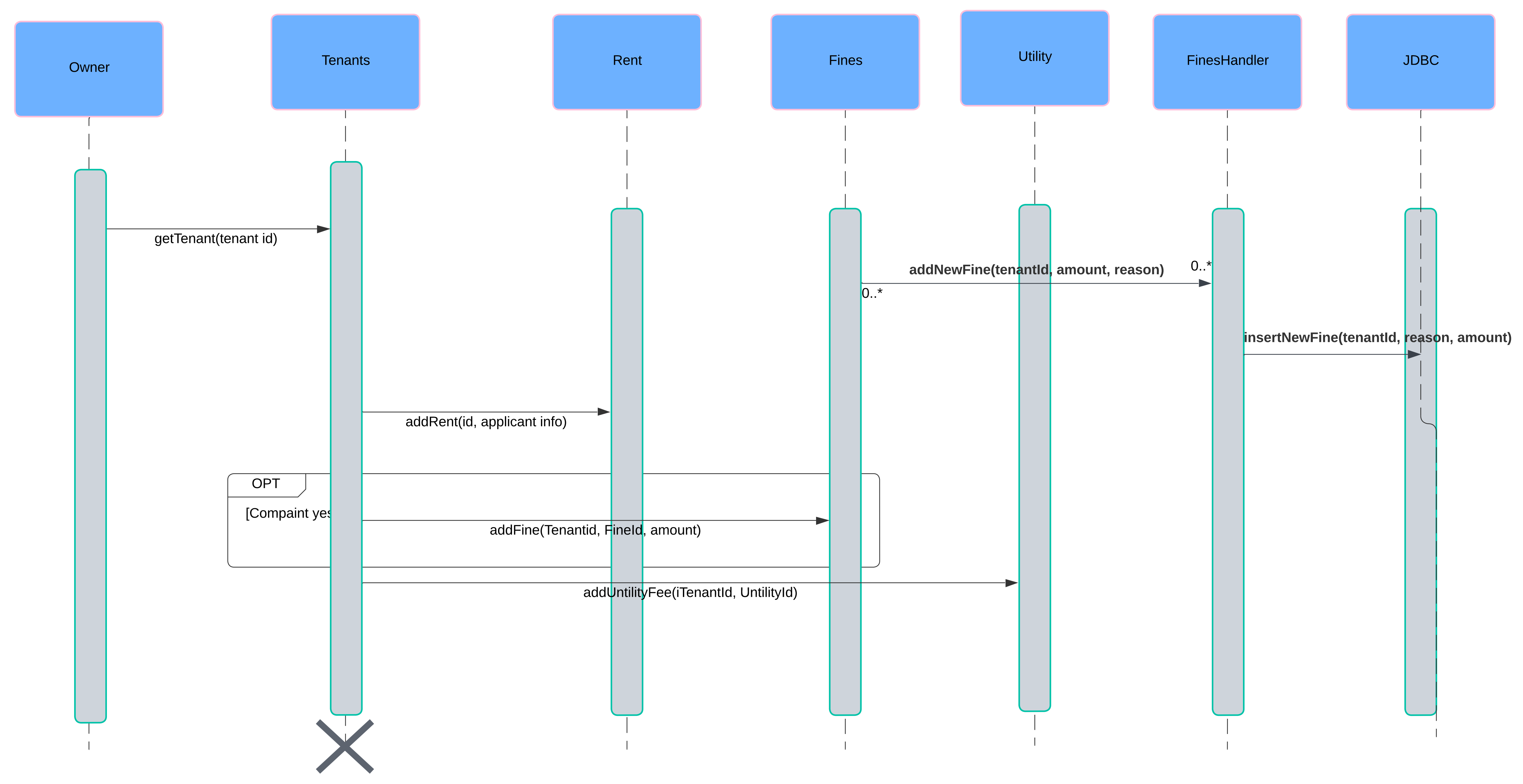

:Tenant
:System
Payment Handler
JDBC
MakePayment(id,methode)
getTenant(id)
getdues(id)
opt
if(details invalid)
sendAlert(tenantID, reason)
create
Payment
addNewPayment(tenantID, method, amount)
makePayment(id,amount,type)
addNewPayment(tenantID, method)
opt
if(freeslot==0)
notify (id,"out of space")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

Rent a Rental

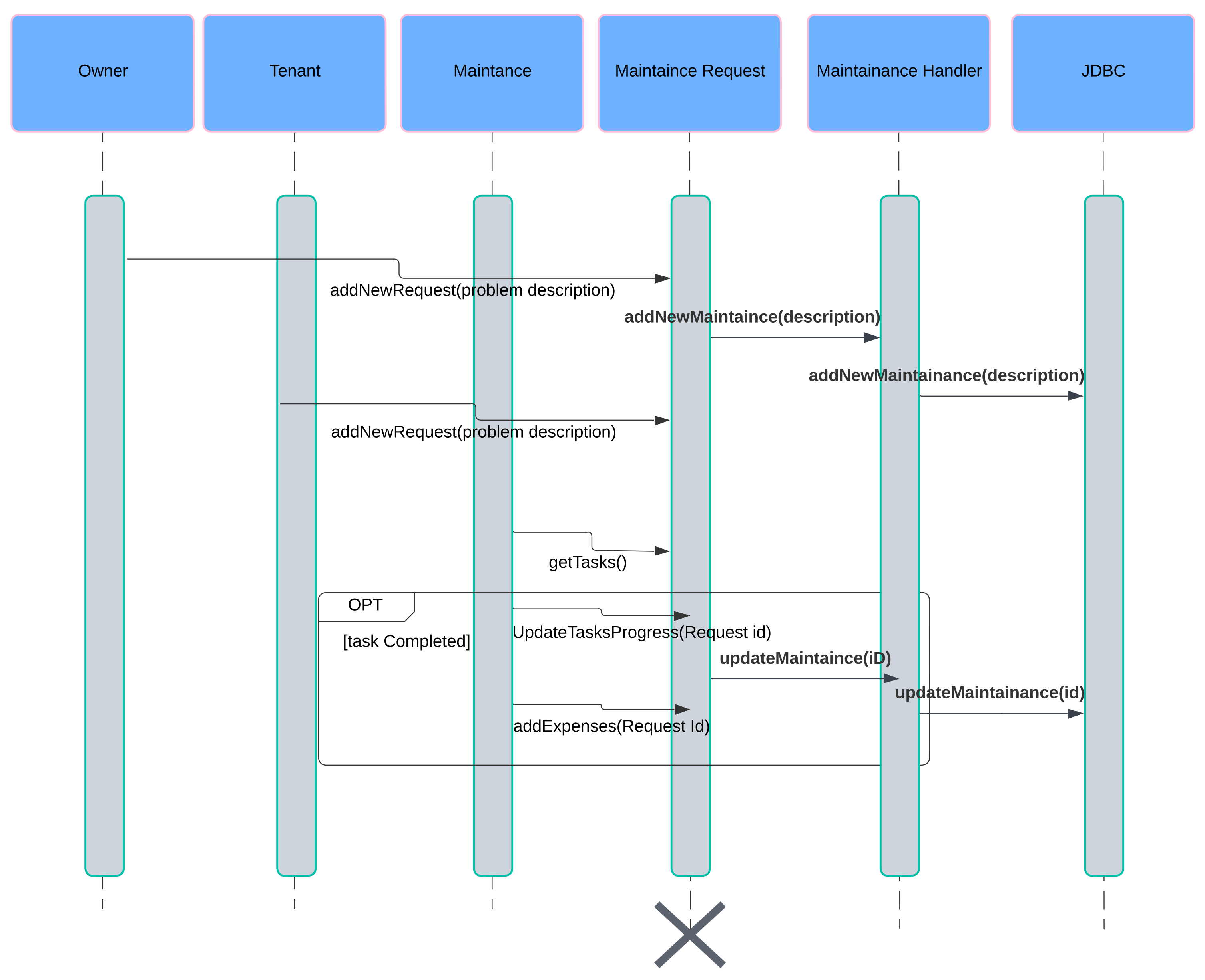

:Tenant
:System
:rentals
JDBC
searchRentals()
DisplayRentals()
AddNewRent(tenantID, rentalID, roomID)
opt
if (availablerentals==0)
sendNotification(tenantID, reason)
makepayment(amount,type)
sendNotification(notificationID, tenantID)
opt
if(fpayment unccessful)
notify (id,"payment unsuccessful")

Action Sequence

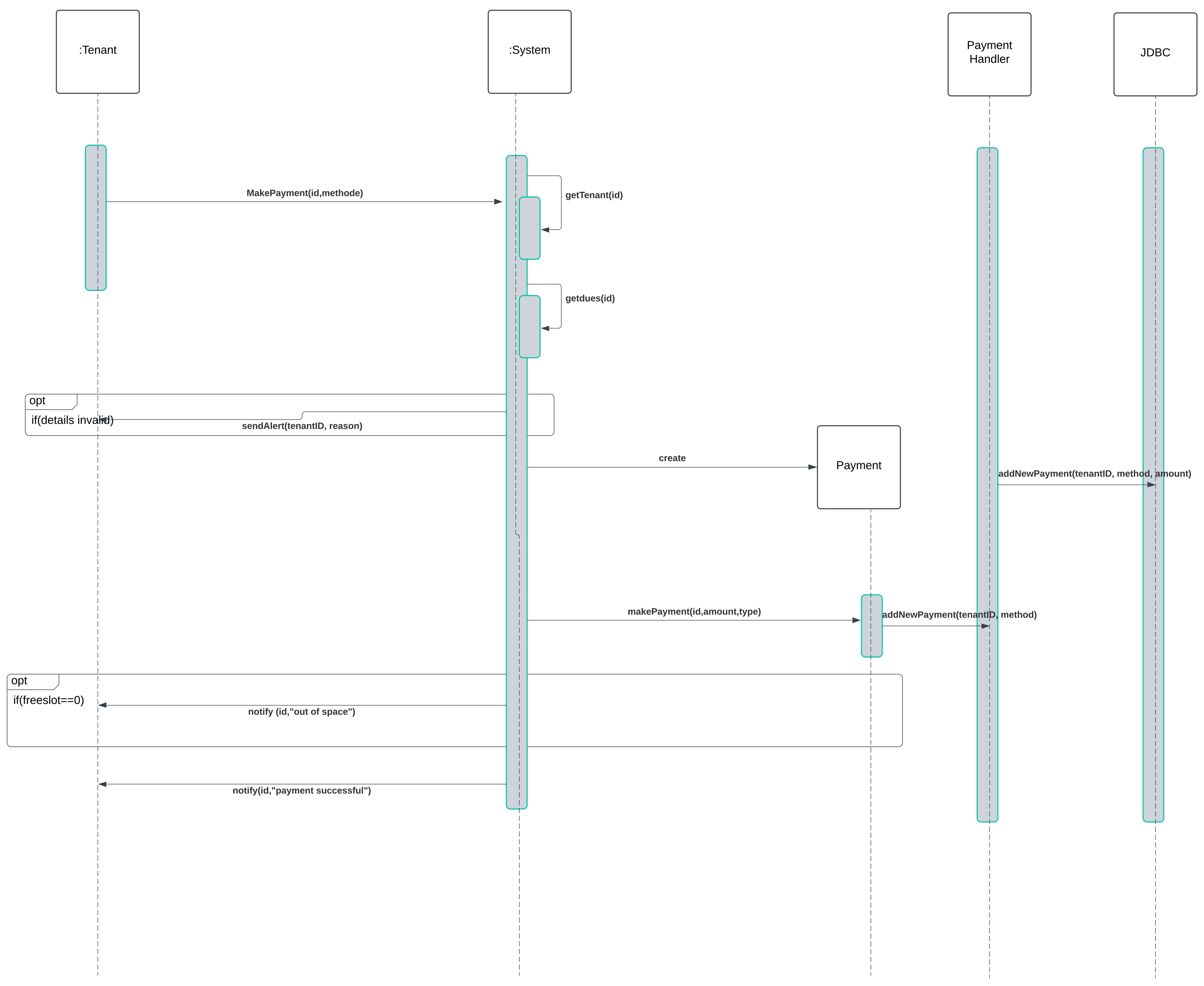
Moniter Tenants


Owner
Tenants
Rent
Fines
Utility
FinesHandler
JDBC
getTenant(tenant id)
addNewFine(tenantId, amount, reason)
0..*
0..*
insertNewFine(tenantId, reason, amount)
addRent(id, applicant info)
OPT
[Compaint yes
addFine(Tenantid, FineId, amount)
addUntilityFee(iTenantId, UntilityId)

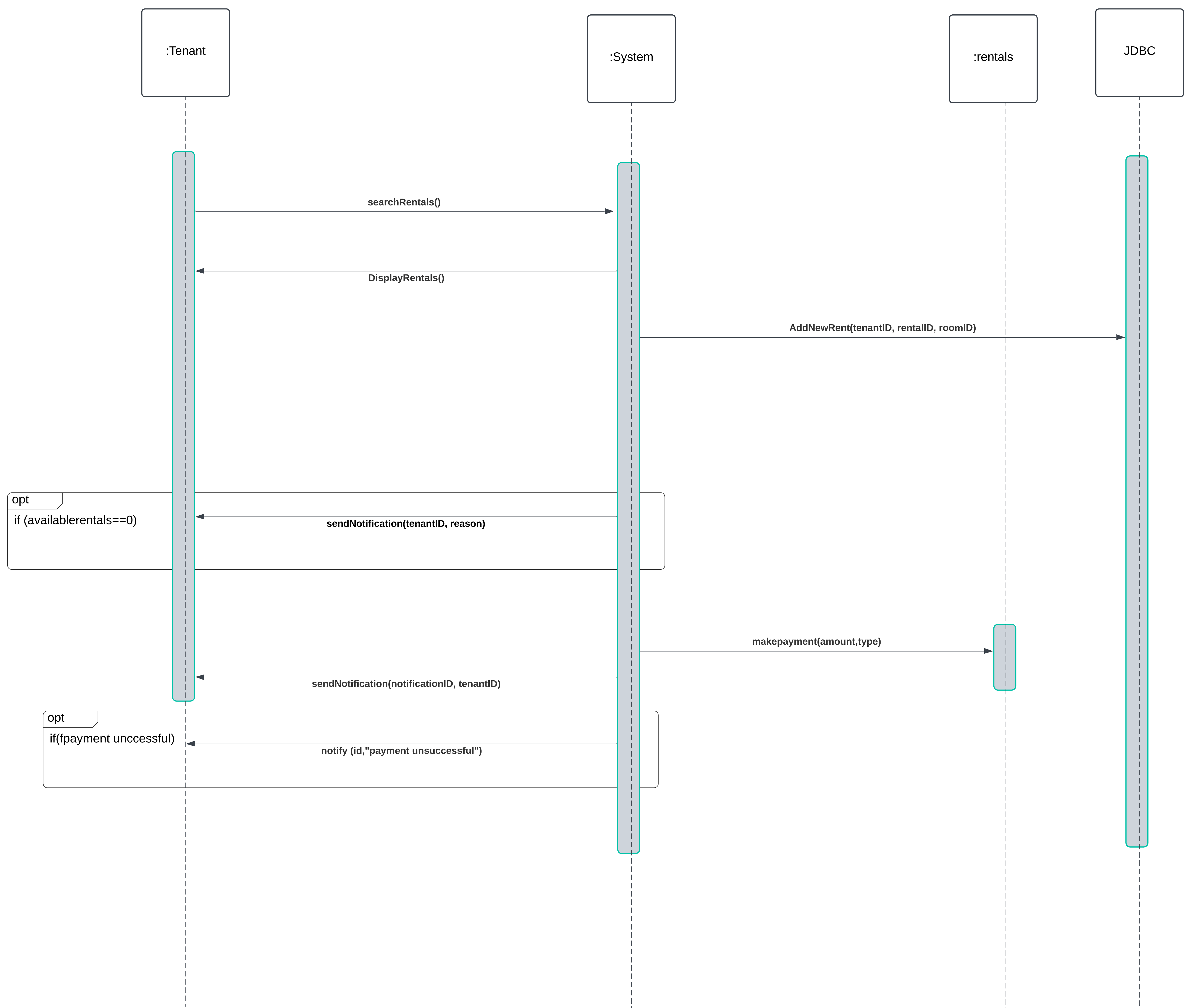
Manage Maintainance


Owner
Tenant
Maintance
Maintaince Request
Maintainance Handler
JDBC
addNewRequest(problem description)
addNewMaintaince(description)
addNewMaintainance(description)
addNewRequest(problem description)
getTasks()
OPT
[task Completed]
UpdateTasksProgress(Request id)
updateMaintaince(iD)
updateMaintainance(id)
addExpenses(Request Id)

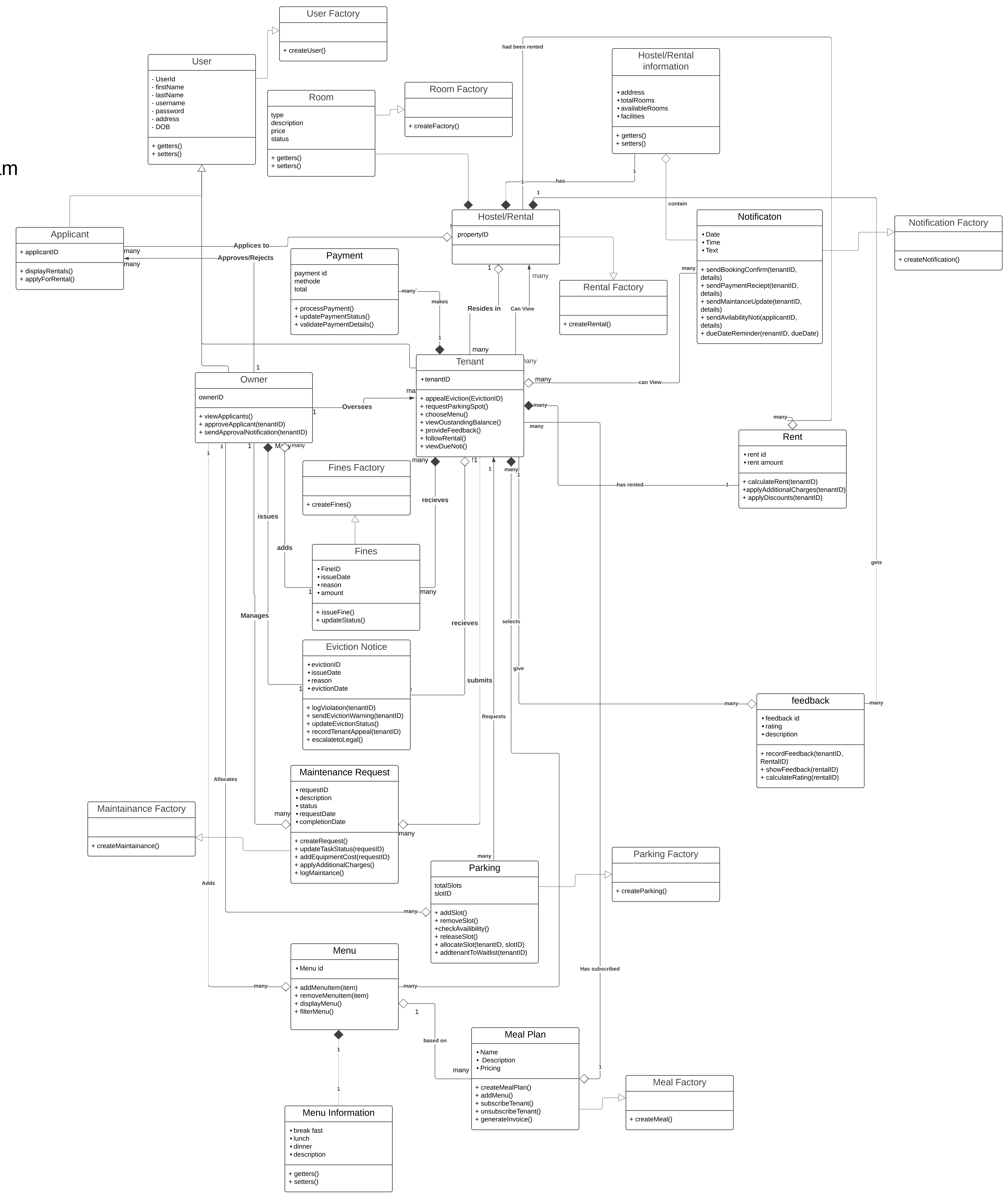
Make Payment


: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")

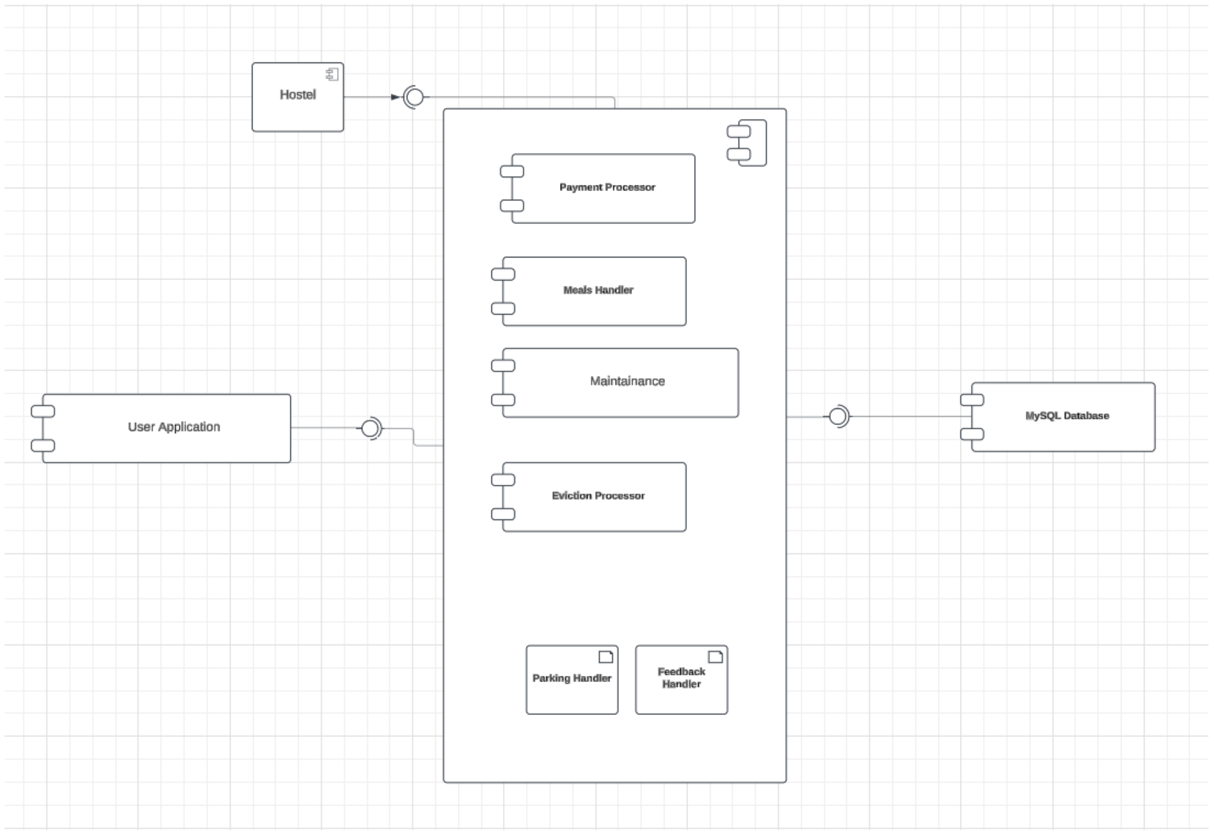
Rent a Rental


: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")

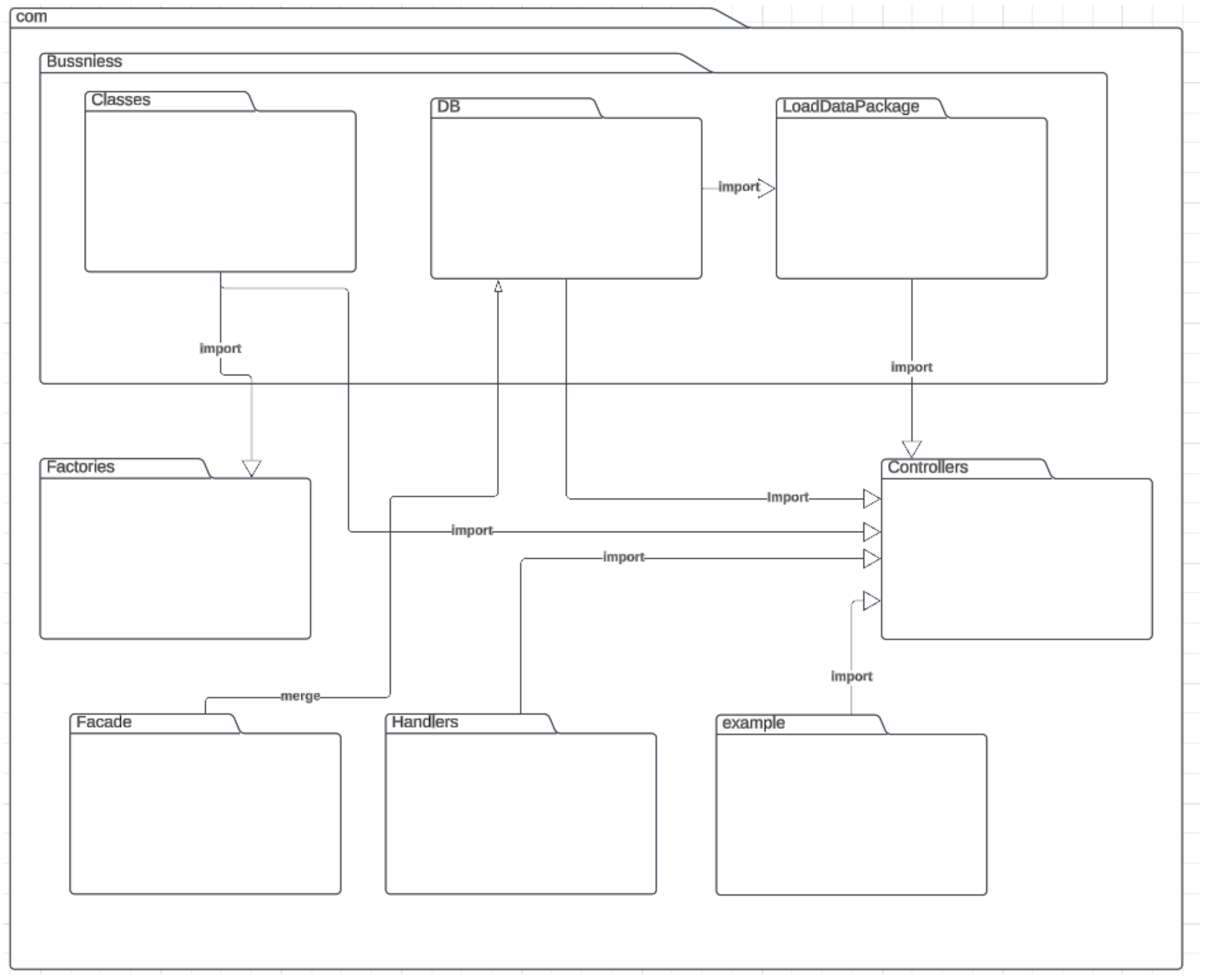
Class Diagram



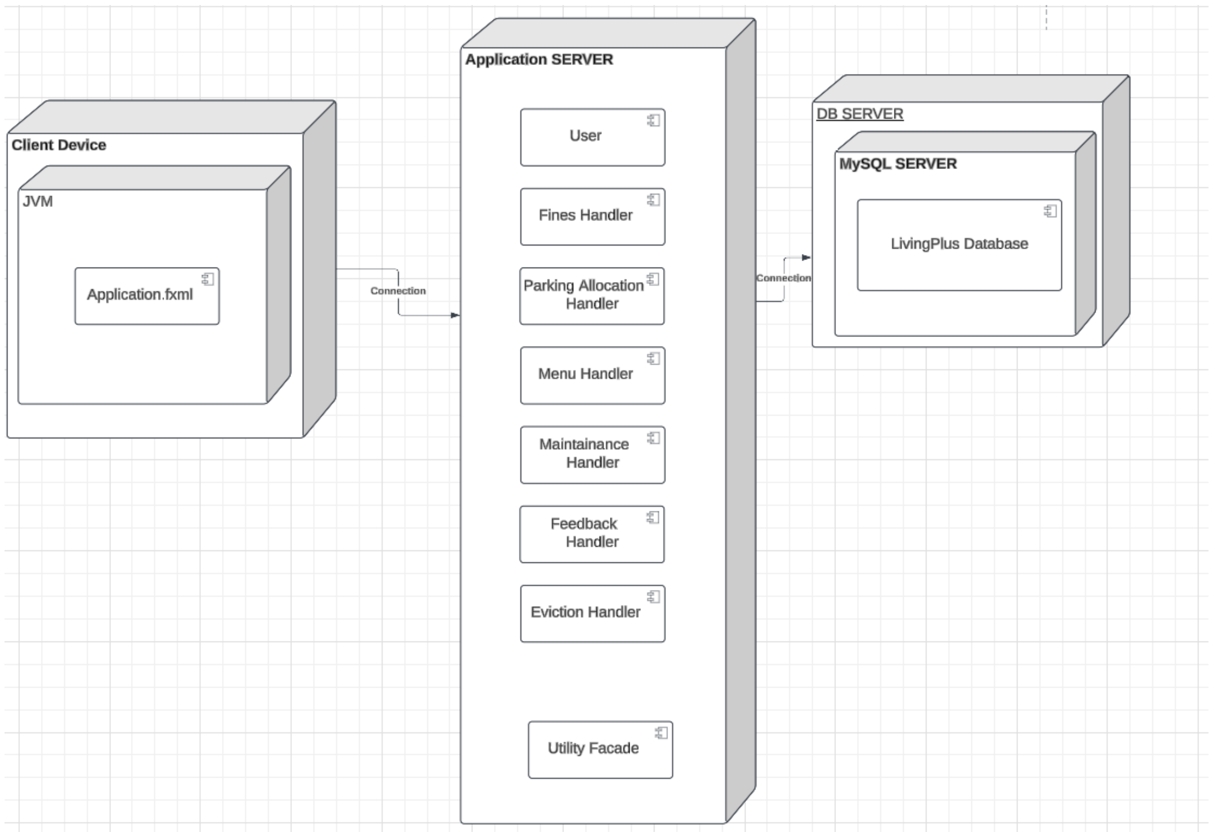
8. Component Diagram



9. Package Diagram



10. Deployment Diagram


Client Device
JVM
Application.fxml
Connection
Application SERVER
User
Fines Handler
Parking Allocation Handler
Menu Handler
Maintainance Handler
Feedback Handler
Eviction Handler
Utility Facade
Connection
DB SERVER
MySQL SERVER
LivingPlus Database