

House Rent management system

SWE Final Project – Group 6



TEam Members

*Luwam Weldemichael (# 615160)*

*Md Rashedul Bari (# 615217)*

*Sanjaya Jayasooriya (#615730)*

*Sumayya Jahan (#614628)*

Submitted to

**Prof. Obinna Kalu**

Table Content Page Number

[House Rental Management System 3](#_Toc127037703)

[Introduction 3](#_Toc127037704)

[Highlights 4](#_Toc127037705)

[Methodology 5](#_Toc127037706)

[Code Repository 6](#_Toc127037707)

[Code/Folder Structure. 8](#_Toc127037708)

[Logging 10](#_Toc127037709)

[Open API endpoint for searching properties 12](#_Toc127037710)

[Functionality 14](#_Toc127037711)

[Guest User Behavior: 14](#_Toc127037712)

[Legal Entities List: 14](#_Toc127037713)

[Properties List View: 15](#_Toc127037714)

[Registration Page: 15](#_Toc127037715)

[Login Screen: 15](#_Toc127037716)

[Landlords: 16](#_Toc127037717)

[Administrators: 21](#_Toc127037718)

[ User Management: 21](#_Toc127037719)

[ Legal Entities Management: 21](#_Toc127037720)

[Diagrams 24](#_Toc127037721)

# House Rental Management System

## Introduction

House Rental Management System is a web-based platform where house owners (Landlords) and customers (tenants) can exchange information effectively and inexpensively. This system provides a user-friendly interface, satisfying the needs of the consumers.

It allows admins to manage Landlords and all other users of the system. It also allows owners and customers to register. Landlords are the user who own the property and renting it out to tenants.

The system allows the Landlord to upload all the details of the property such as the number of rooms, price, location, and ensures the possibility to manage the property. Management actions include updating the property details and updating the availability. The system also provides the possibility for searching the properties based on criteria such as location, number of rooms etc. A prospective tenant (Guest user) can start the journey by searching for a property as per the need where the user gets the option to search as well through the portal. The Guest users will have limited access to the portal like search and display properties however, they will be able to filter properties by price, property type, number of rooms, and location.

When the Guest User is interested in a property, prior to the Submission of an application, the Guest User should become a Registered User in the system.

Registered users only can apply for properties through the system after logging in into the system. The Landlord gets the notification once an inquiry is made. Then the landlord communicated with the interested user (prospective tenant) and prepare the contract. Once both parties come to a mutual understanding, the contract is signed to complete the process.

The high-level requirement document can be found in the following GitHub location:

<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/documents/requirements/ProblemStatement.md>

## Highlights

* [Atlassian Jira](#Jira)
* [Github Actions](#Github_Actions)
* [H2 in-memory db for Unit tests](#H2_DB)
* [Open API endpoint for “properties” search](#_Open_API_endpoint)
* [Usage of fragments with Java reflection](#fragments)
* [Logging](#logging)
* [Common error page](#common_error_page)

## Methodology

The implementation was carried out using the Agile Scrum methodology. To follow the standard and best practices, team uses Atlassian Jira tool (Jira Agile plugin) to plan and track the progress throughout the lifecycle. A snapshot of it is shown below which is accessible through the link below as well.

<https://cs-425-student-project.atlassian.net/jira/software/c/projects/SP/boards/1/roadmap>

Graphical user interface

Description automatically generated with medium confidence

To summarize this agile practice, the project was planned for 4 incremental releases in consecutive 4 sprints (one week each).

## Code Repository

A GitHub public repository is used for team collaboration, managing the code, and preserving all the design documents. The project is accessible on GitHub using the link below.

<https://github.com/cs-425-student-project-miu-edu/student-project>

As part of the standard practice, we made a bare repository public project, and all the team members (collaborators) forked the project in their personal repo. Thus, the team ensured to review and merge each other's contribution to maintain standards.

Additionally, a GitHub action was created to run the CI/CD pipeline activities to test the changes before pushing those to the main repo. This pipeline executes ***maven verify*** command to make sure the code does not break, and the tests are carried out successfully. Thus, we ensure possible standard practices to avoid side effects.

A glimpse of the CI/CD pipeline is shown below, and live link is shared below as well.

Live link: <https://github.com/cs-425-student-project-miu-edu/student-project/actions>

A couple of screenshots of GitHub Action view with test results:

Graphical user interface, text, application, email

Description automatically generated

Detailed outcome of a test cycle can be analyzed in greater detail as shown in the screenshot below.

Graphical user interface, text

Description automatically generated

Though MySQL Database has been used for the project, the Spring test cycles are executed using H2 in-memory database and through GitHub Actions (CI/CD pipeline) as shown below.

Text

Description automatically generated

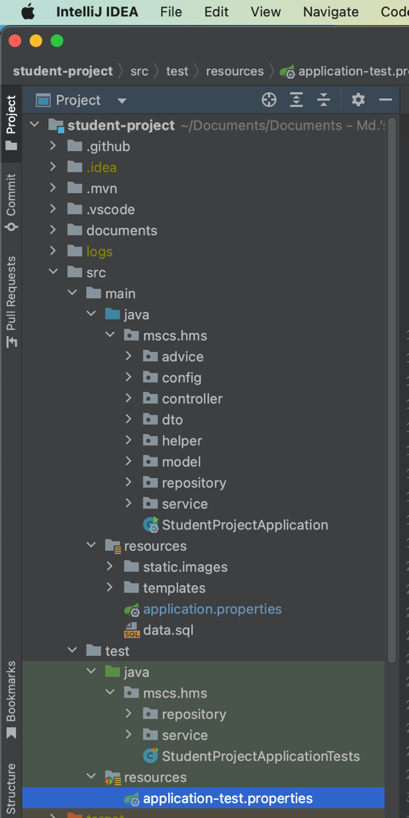
A separate test profile has been configured to use H2 database as shown below to achieve the above.

A screenshot of a computer

Description automatically generated with medium confidence

## Code/Folder Structure.

A standard Spring MVC web-based project structure was created using Spring-Boot initializer portal. Along with this basic project structure, project team has maintained logical separation through proper package structure to main the application internal layering as shown below.



The UI logic uses fragments extensively. The layout is drawn making use of Java reflection. A lot of sections are dynamically populated through reflection and proper component structuring which can be visualized through the links below.

Graphical user interface

Description automatically generated with medium confidence

<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/src/main/java/mscs/hms/controller/util/ViewFieldUtil.java>

<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/src/main/resources/templates/fragments/entity_list.html>

<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/src/main/resources/templates/fragments/entity_edit.html>

## Logging

Both Console and File based Logging mechanisms are enabled to easily identify the errors and all other information as shown below where Console login in the default behavior of Spring Boot applications.

Text

Description automatically generated

A global error handler is designed and implemented to prevent leakages of stack traces.

Graphical user interface, text, application

Description automatically generated

## Open API endpoint for searching properties

Swagger endpoint was introduced to make it easier for the integrations to find the metadata about the properties search endpoint. Thus, the system is ready to have all the upcoming APIs to be documented effortlessly.

Graphical user interface, application, email

Description automatically generated

Functionalities and UI Views

## Functionality

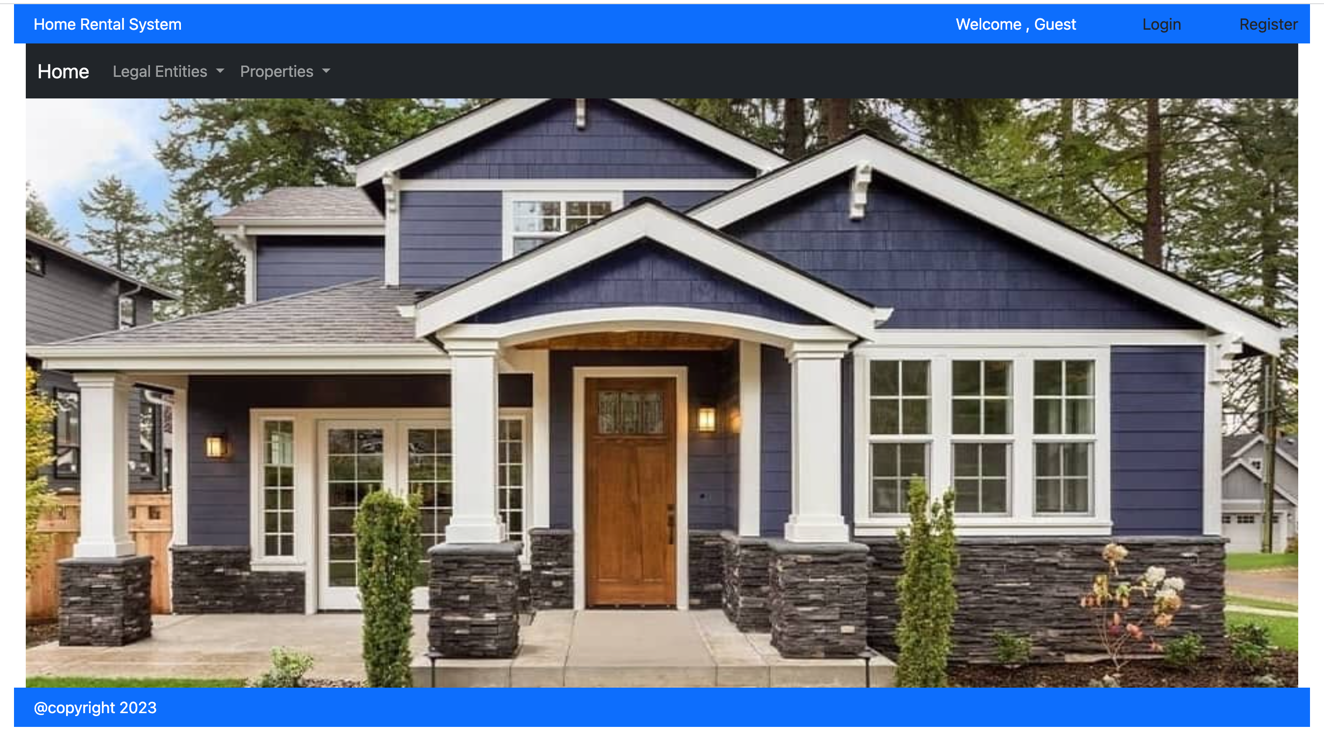
Default username/password of the system generated admin user is,

***admin/admin***

* All the list view screens have pagination and search.

### **Guest User Behavior:**

When the guest user accesses the portal, the user will see the below homepage with limited functionality.



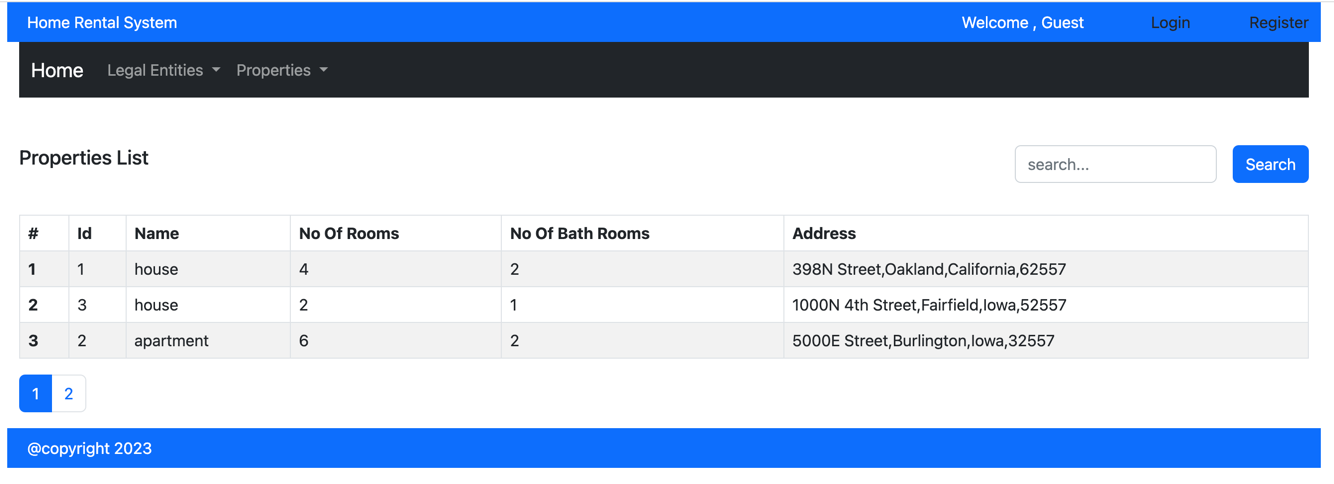
### **Legal Entities List**:

This page helps all the users to find the companies or the Individual registering their assets for rental.



### **Properties List View:**

Properties pages helps the landlords to manage their assets and the prospective tenants to look for available house or apartments for rent.



### **Registration Page:**

When registering, the user will be assigned the Guest role. Later an Administrator changes the roles by assigning the user to a landlord or a Tenant role defined in the system.

Graphical user interface, application

Description automatically generated

### **Login Screen:**

This view helps registered user to login to the system to enjoy all the registered user functionalities as per the roles and authority

Graphical user interface, application

Description automatically generated

### Landlords:

Property owners are recognized here in the system as Landlords, and they can manage their properties after login to the portal as shown below.

A screenshot of a house

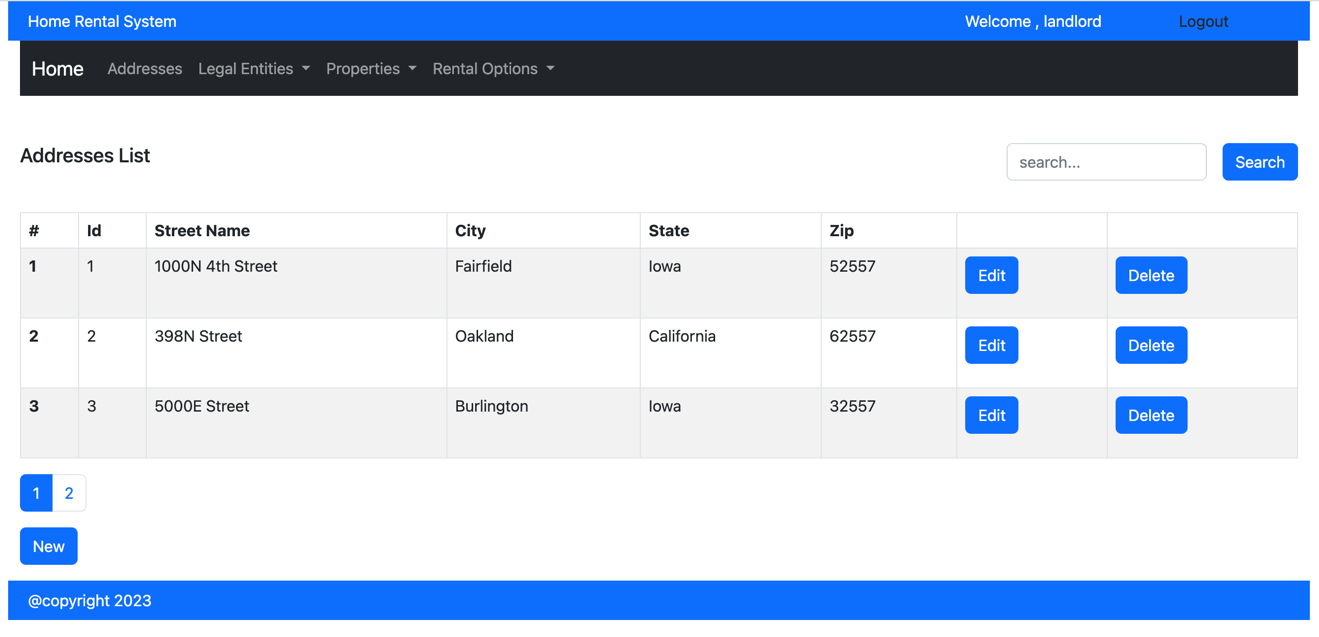
Description automatically generated with low confidence

Manage Addresses:

The address section is to assign a designated address to all the properties. Admins and Landlord type users can manage (view, edit, add and delete) the property addresses.

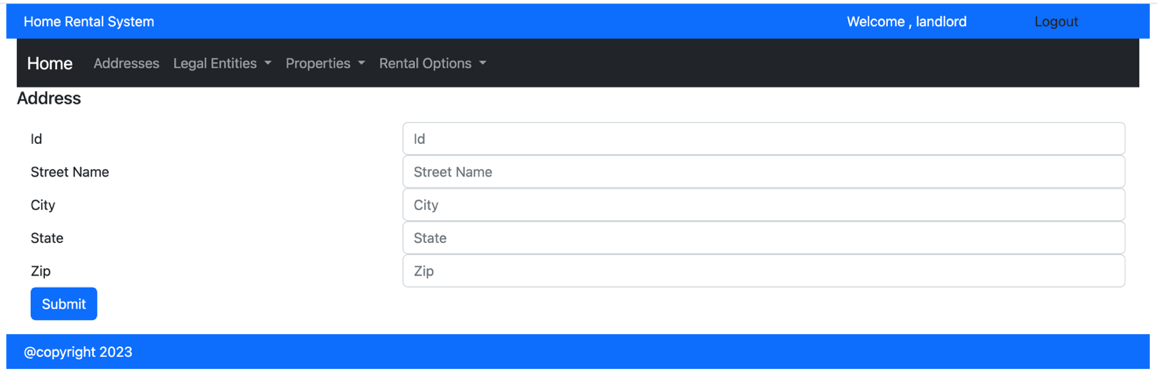
* **Address List Page:**

This page helps Admins and Landlords to view addresses list of the assets.



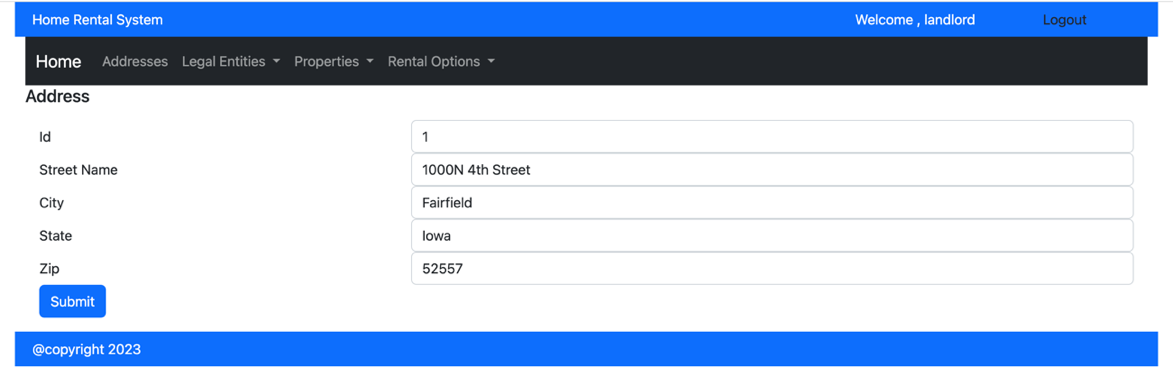
* **Adding New Address:**

Add address feature helps to register new addresses by the Landlords. Admin can do that on behalf for the Landlords as well.



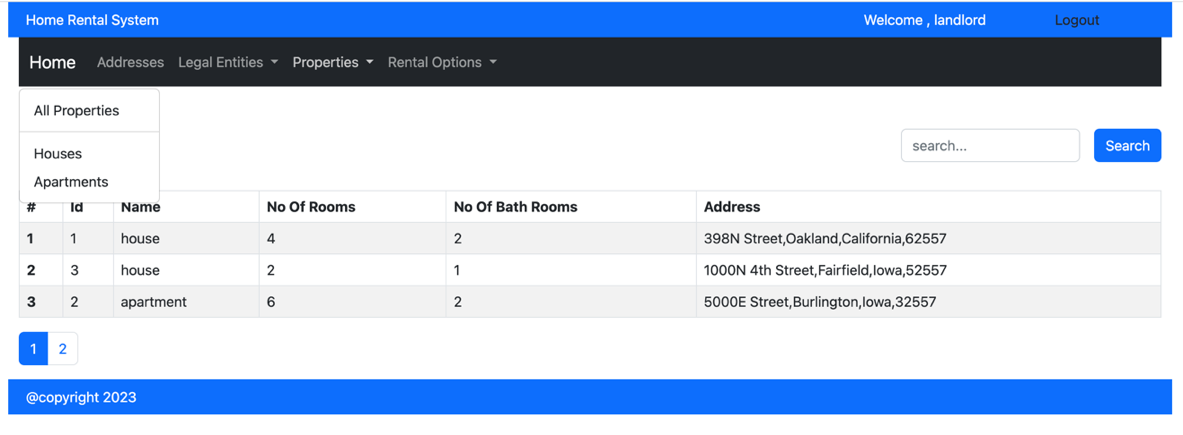
* **Editing Existing Address:**

Edit view is just an additional feature to manage the asset addresses better.



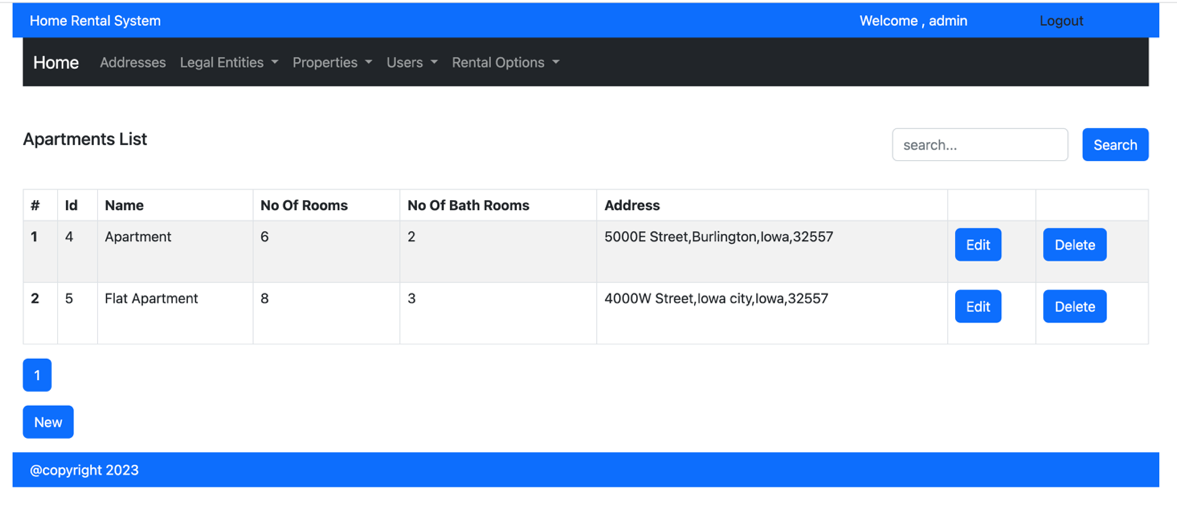
**Manage Properties:**

In the portal, two types of properties are considered: Houses and Apartments. Both the type has its individual management sections as shown below:



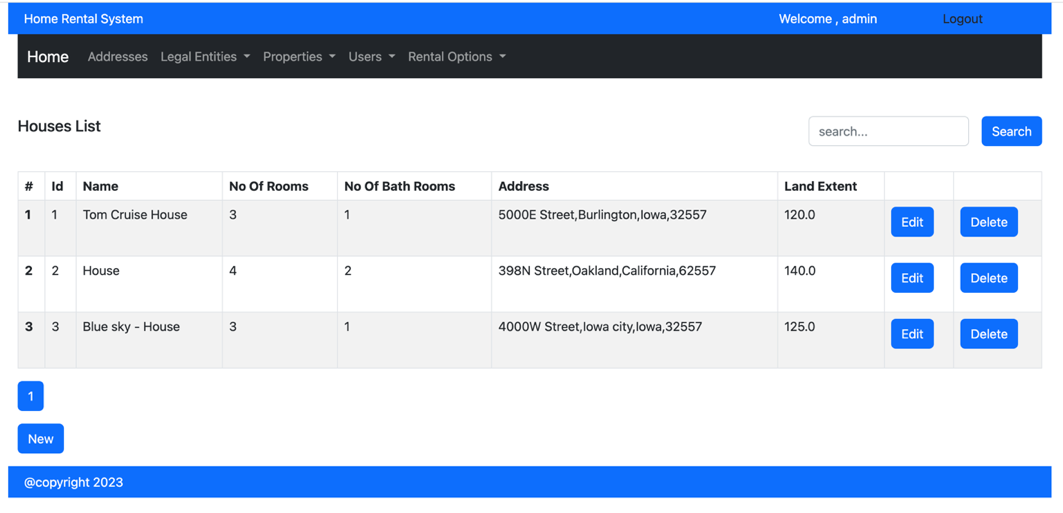
* **Apartments List:**

This page lists all the apartments and their summary information.



* **Houses List:**

This page lists all the houses and their summary information.



**Rental Management:**

Landlords can view Inquiries, Rent Applications, and create rental agreements. Graphical user interface, text, application, Word

Description automatically generated

* **Rental Application List View**

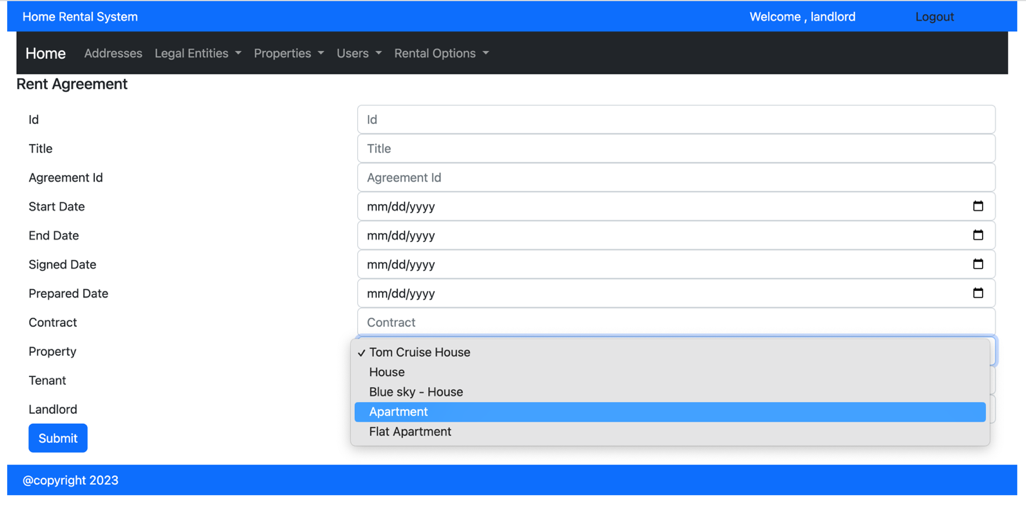
Prospective Tenants can create new applications using this section.

Graphical user interface, text

Description automatically generated

* **Rental Agreements:**

Landlord creates rental agreements based on the inquiries and applications processed earlier. This can be considered as the last step of the rental signing process.



* **Agreement List View**:

All the agreements will be available in this section to review and process as needed.

Graphical user interface, application, Word

Description automatically generated

**Tenants**:

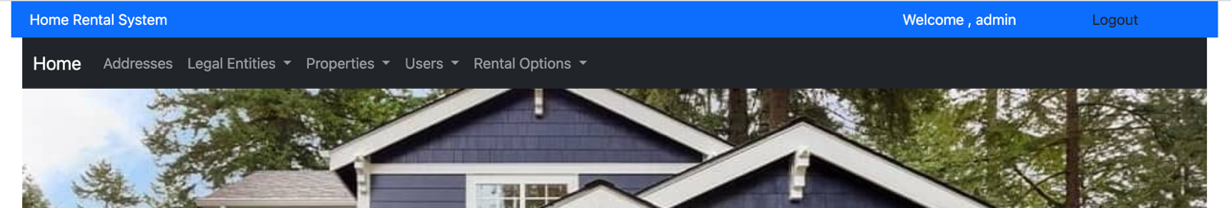
Tenants have the same options as admins and Landlords, but cannot edit/delete properties, addresses and Rental Agreements

Graphical user interface

Description automatically generated with medium confidence

### **Administrators**:

Administrators has all the access which the Guests, Tenants and Landlords have. Additionally, the admin users are responsible for managing the registered users and legal entities.



### **User Management:**

When a user is registered, the admin should create a user based on a Legal Entity. The legal entities are converted to Landlords or Tenants as per the request.

### **Legal Entities Management:**

The registered user needs to be first connected to a Legal Entity. A legal entity can be a Company or Person.

Graphical user interface, text, application

Description automatically generated

* **Legal Entities List View:**

This view shows all the legal entities to be managed by admin user.

Graphical user interface, text, application, website

Description automatically generated

The below screenshot shows the menu options available for the admin user to create Landlords, and Tenants and assigning them with a Legal Entity using edit button.

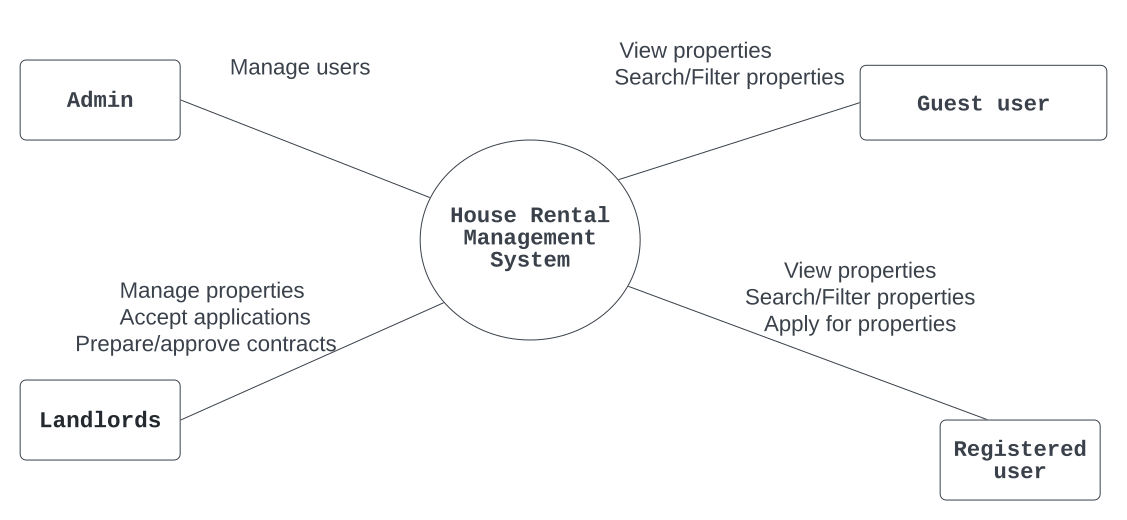
Graphical user interface, text, application, website

Description automatically generated

System Designs and Diagrams

## Diagrams

* Context Diagram



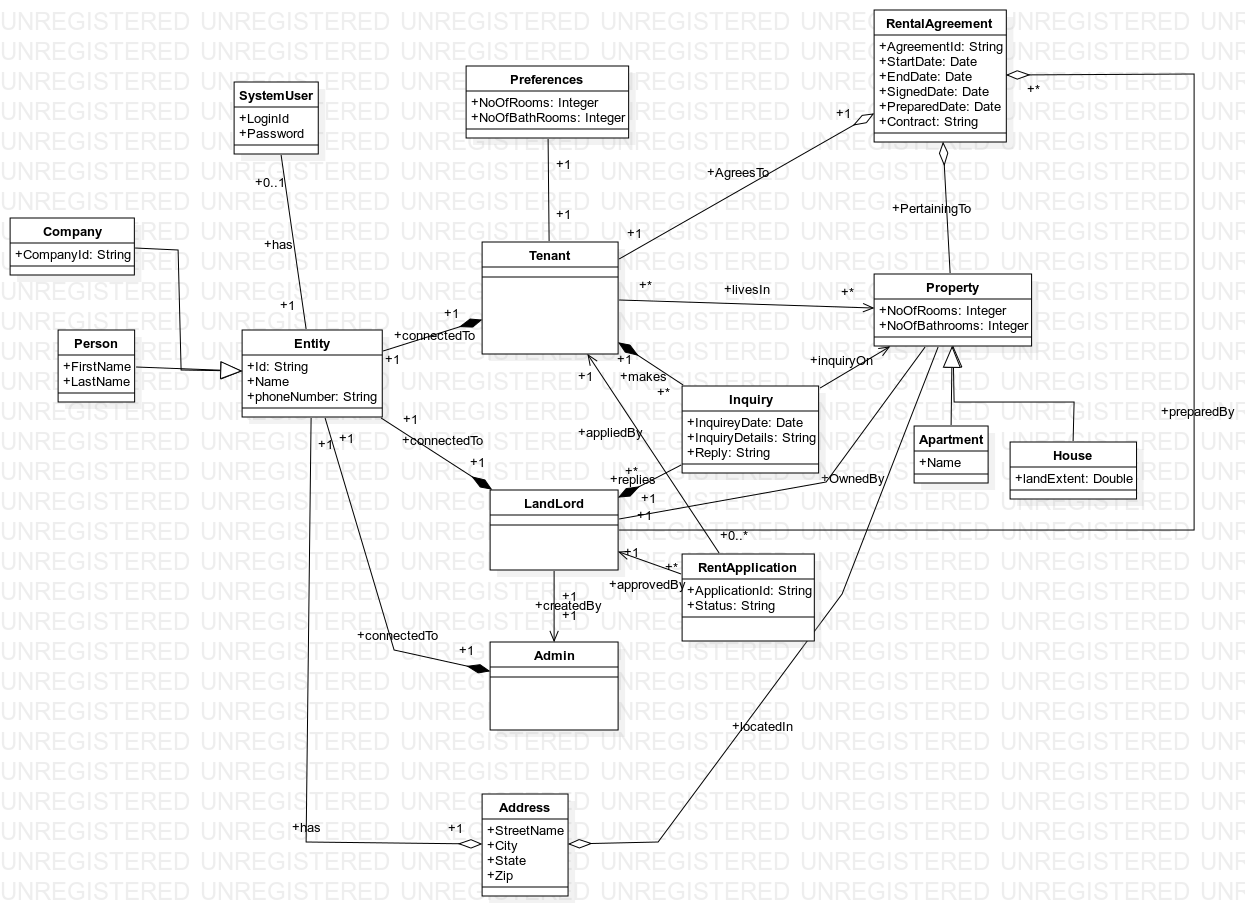
<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/documents/diagrams/ContextDiagram.svg>

* Use Case Diagram



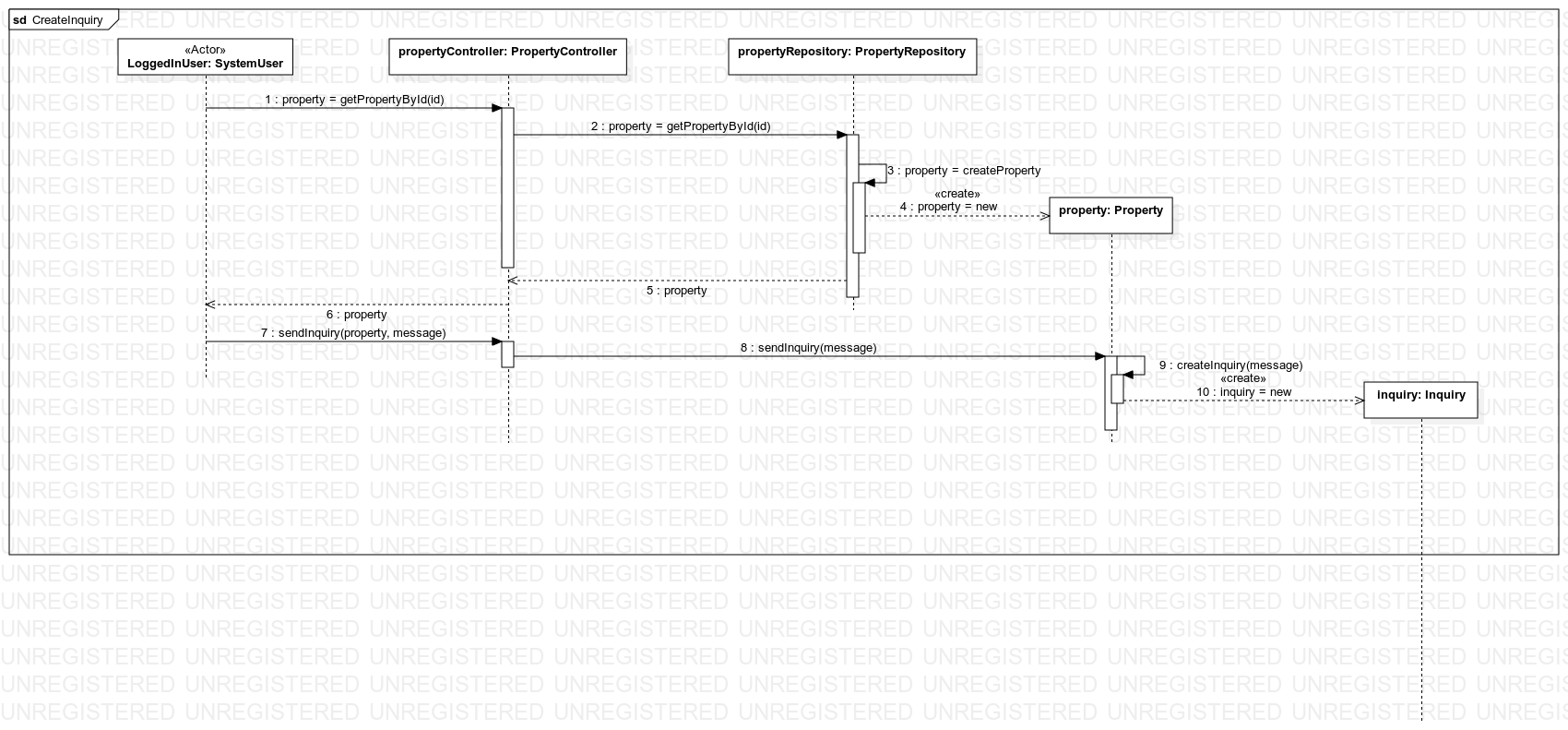
<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/documents/diagrams/UseCaseDiagram.svg>

* Class Diagram



<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/documents/diagrams/ClassDiagram.svg>

* Sequence Diagram



Create Inquiry



Registered User Search Property



Respond to Inquiry

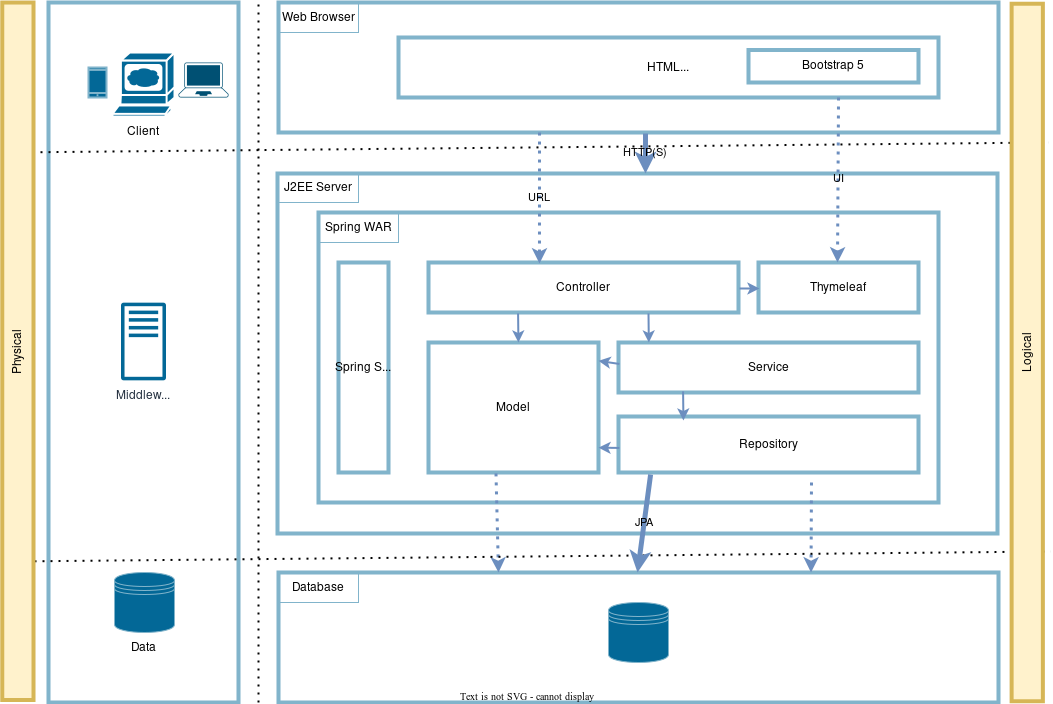


Submit Rent Application

All these diagrams are preserved and stored in GitHub in the following link:

<https://github.com/cs-425-student-project-miu-edu/student-project/tree/main/documents/diagrams/sequenceDiagrams>

* Architecture Diagram



<https://github.com/cs-425-student-project-miu-edu/student-project/blob/main/documents/diagrams/Architecture.svg>