Group Project

**TCP2201 - Object Oriented Analysis and Design
By <<Group 6>>**

Cyberjaya Online Rental Management System

Lecture Session : TC2V
Lecture Name : WK Soo

Prepared by

Name	Student ID	E-mail
Liew Jiann Shen	1191100556	1191100556@student.mmu.edu.my
Adam Koh Jia Le	1191101858	1191101858@student.mmu.edu.my
Soon Jie Kang	1201301760	1201301760@student.mmu.edu.my
Teoh Yee Chien	1201300833	1201300833@student.mmu.edu.my

Table of Contents

1.Introduction

The assignment aims to create the Cyberjaya Online Rental Management System which is an online tool for owner, property agent and potential tenant in Cyberjaya to rent out or to rent a property.

2.Assumption

1. Admin can edit all property details but property owners or property agents can also edit the details but after the property is approved by the admin they can only edit the price.
2. A property can have many potential tenants apply to rent at the same time but only one potential tenant can rent the property.
3. During registration for new users the username and phone number must be unique in the database for the user to be successfully registered.
4. New property entered into the system must have a unique address in order for it to be added into the system.
5. Potential tenants can rent multiple properties at the same time but property can only have one potential tenant.
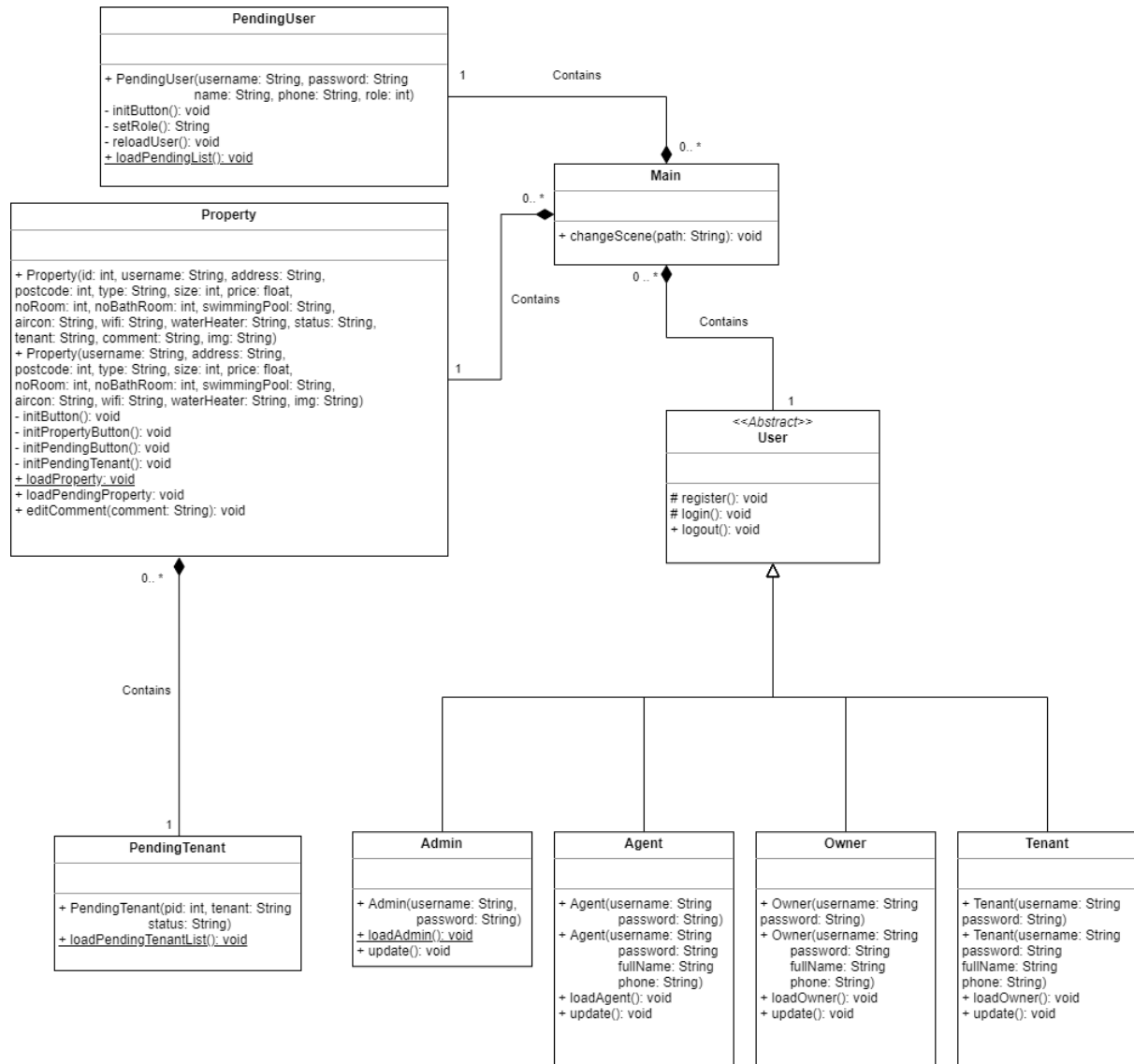
3. Program Design

Object Oriented concepts used:
1. Inheritance. User extends to Admin, Agent, Owner and Tenant
2. Composition. Property -> PendingTenant
3. Composition. PendingTenant -> Tenant
4. Implements. Fxml controller implements initializable
5. Encapsulation. Data from admin, agent, owner and tenant class are using get and set methods
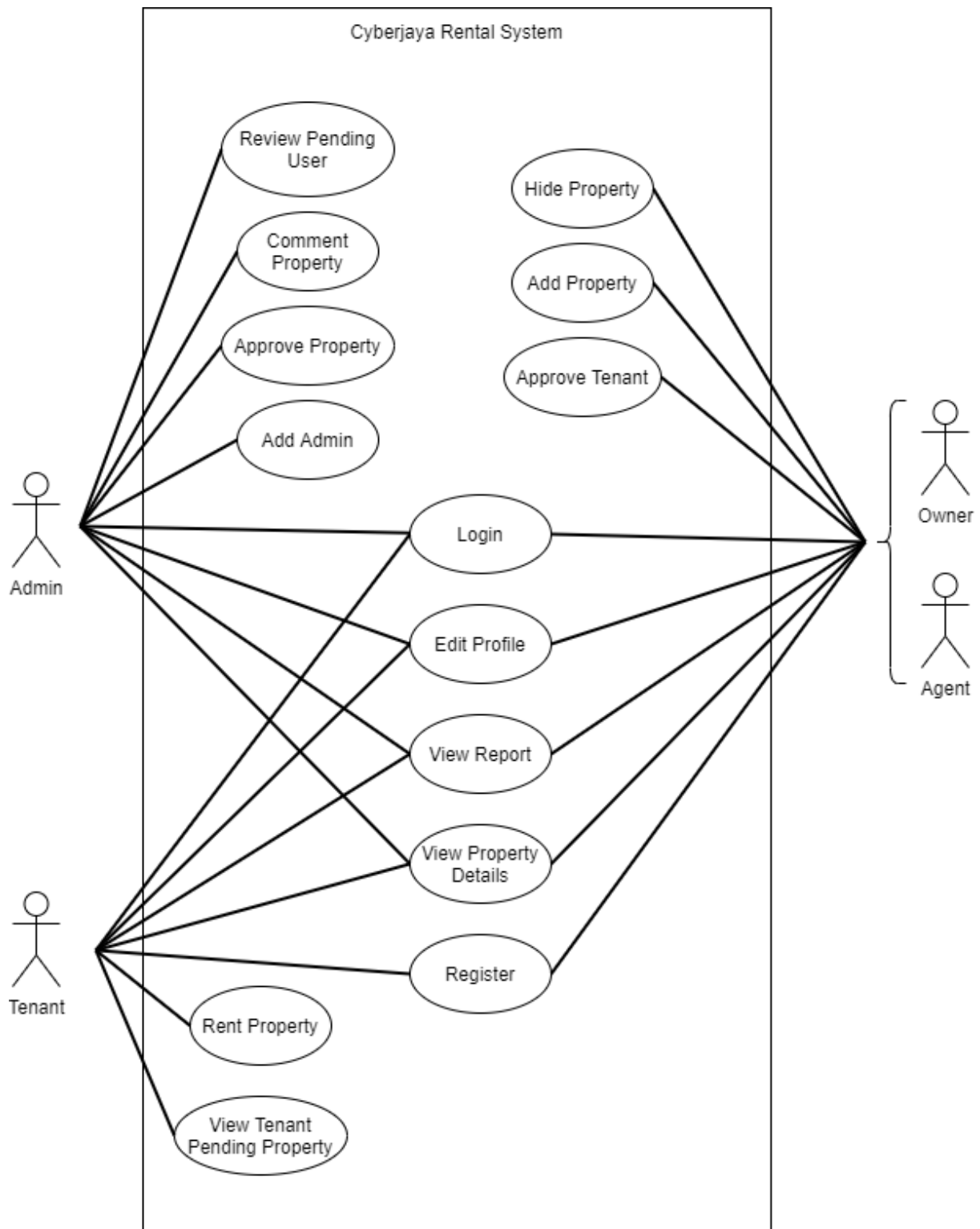
Design Pattern Used:
1. Model-View-Controller for the implementation of javafx with fxml and controller.
2. Singleton design pattern. The data is stored into the static linked list of respective user data: admin, property agent, property owner and potential tenant so that there is only one instance of itself during runtime and provides a local point of access to it.
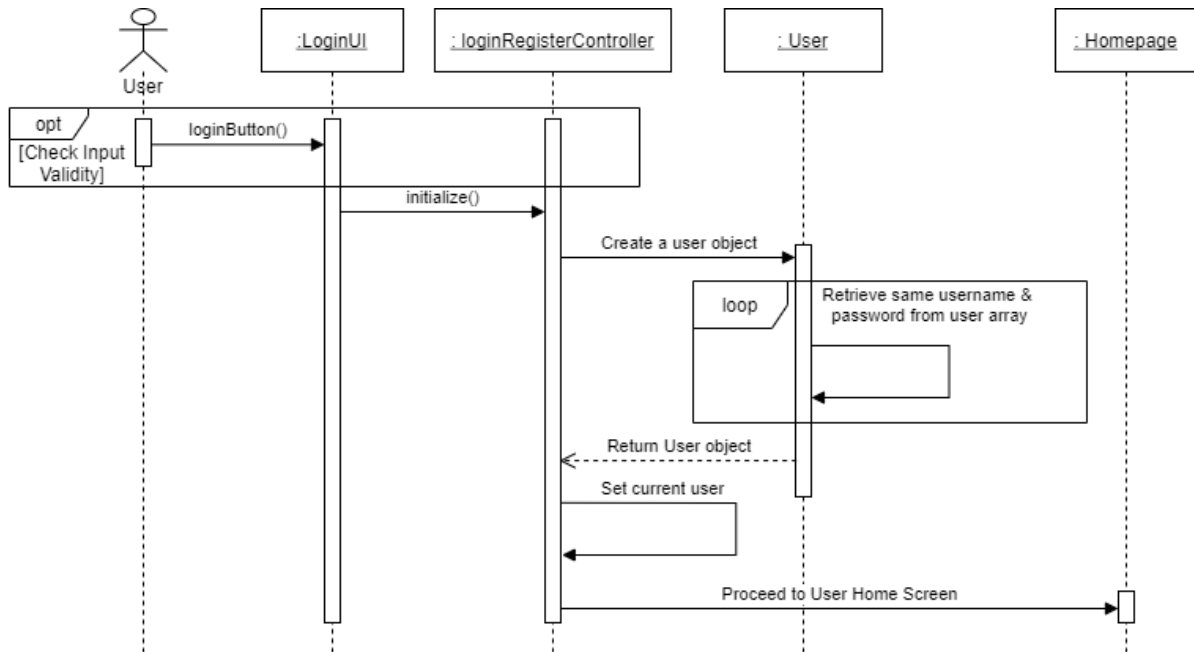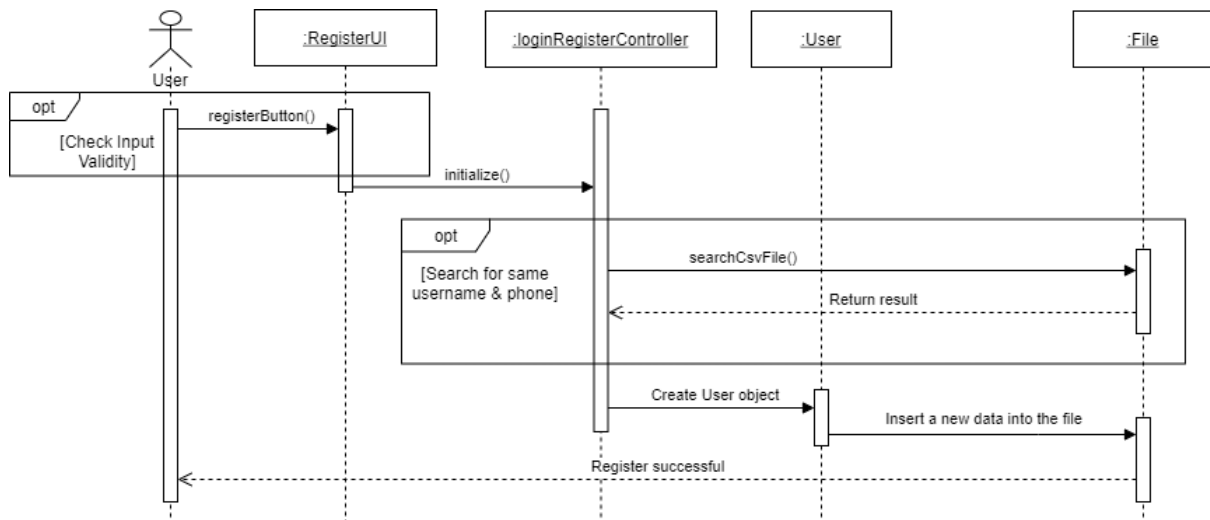
4.UML Diagram

PendingUser

+ PendingUser(username: String, password: String
                   name: String, phone: String, role: int)
- initButton(): void
- setRole(): String
- reloadUser(): void
+ loadPendingList(): void

Property

+ Property(id: int, username: String, address: String,
postcode: int, type: String, size: int, price: float,
noRoom: int, noBathRoom: int, swimmingPool: String,
aircon: String, wifi: String, waterHeater: String, status: String,
tenant: String, comment: String, img: String)
+ Property(username: String, address: String,
postcode: int, type: String, size: int, price: float,
noRoom: int, noBathRoom: int, swimmingPool: String,
aircon: String, wifi: String, waterHeater: String, img: String)
- initButton(): void
- initPropertyButton(): void
- initPendingButton(): void
- initPendingTenant(): void
+ loadProperty: void
+ loadPendingProperty: void
+ editComment(comment: String): void

Main

+ changeScene(path: String): void

**<<Abstract>>
User**

register(): void
login(): void
+ logout(): void

1   Contains   0.. *

0.. *

Contains

1

0.. *

Contains

1

0.. *

Contains

1

PendingTenant

+ PendingTenant(pid: int, tenant: String
           status: String)
+ loadPendingTenantList(): void

Admin

+ Admin(username: String,
    password: String)
+ loadAdmin(): void
+ update(): void

Agent

+ Agent(username: String
    password: String)
+ Agent(username: String
    password: String
    fullName: String
    phone: String)
+ loadAgent(): void
+ update(): void

Owner

+ Owner(username: String
password: String)
+ Owner(username: String
    password: String
    fullName: String
    phone: String)
+ loadOwner(): void
+ update(): void

Tenant

+ Tenant(username: String
password: String)
+ Tenant(username: String
password: String
fullName: String
phone: String)
+ loadOwner(): void
+ update(): void

5.Use Case Diagram

Cyberjaya Rental System

- Review Pending User
- Comment Property
- Approve Property
- Add Admin
- Hide Property
- Add Property
- Approve Tenant
- Login
- Edit Profile
- View Report
- View Property Details
- Register
- Rent Property
- View Tenant Pending Property

Admin

Tenant

Owner

Agent

6.Sequence Diagram

6.1. User Login

Users login into the system using username and password.

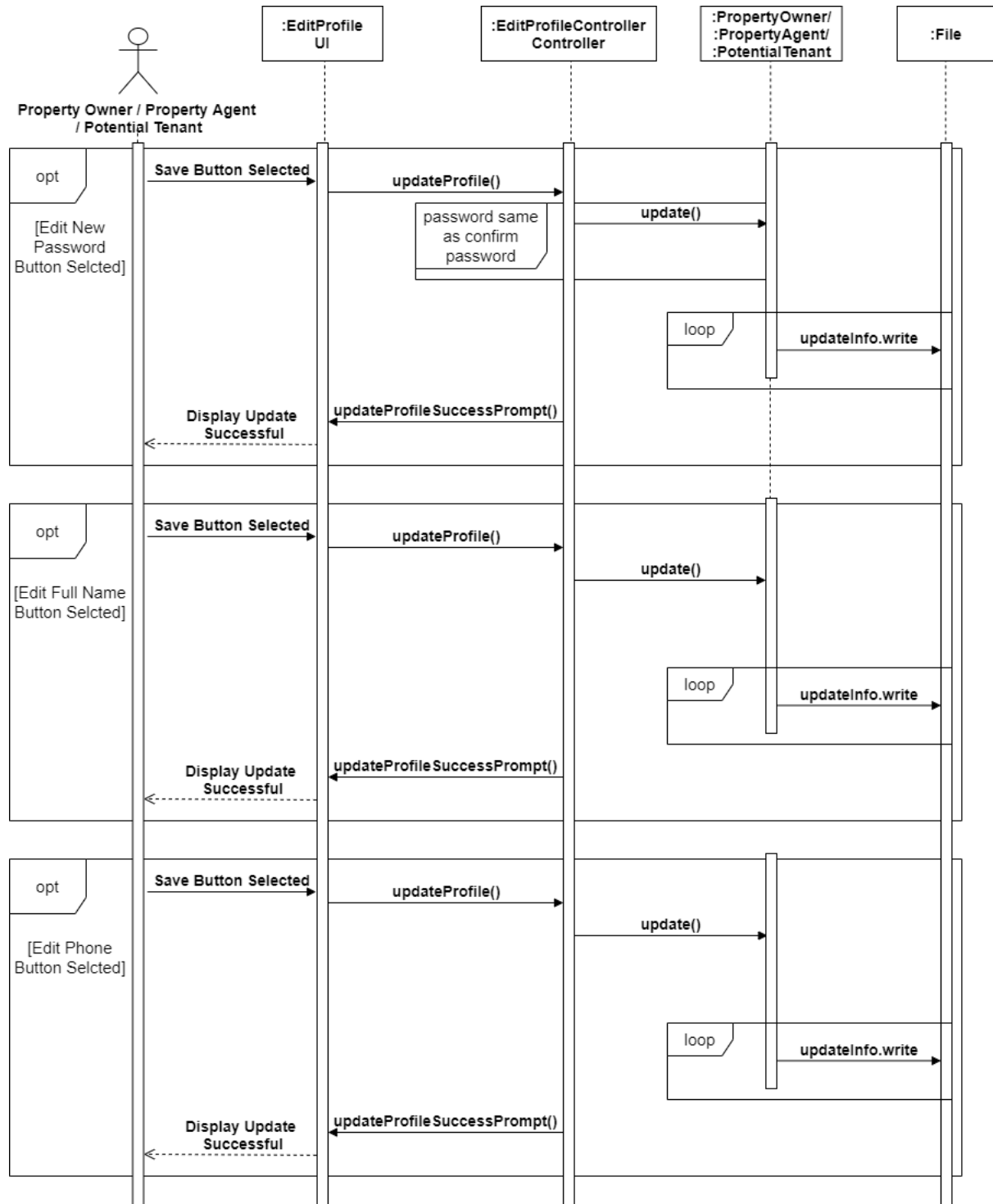
6.2. User Register

User register into system to get approval from the admin

6.3. Admin Update Details

Admin update the password for their own account

6.4. Property Owner/Property Agent/ Potential Tenant Update Details

Property owner/property agent/ potential tenant update their own account password, name and phone number

6.5. View Report

User view report listing of all properties based on the user account admin, property agent, property owner or potential tenant.
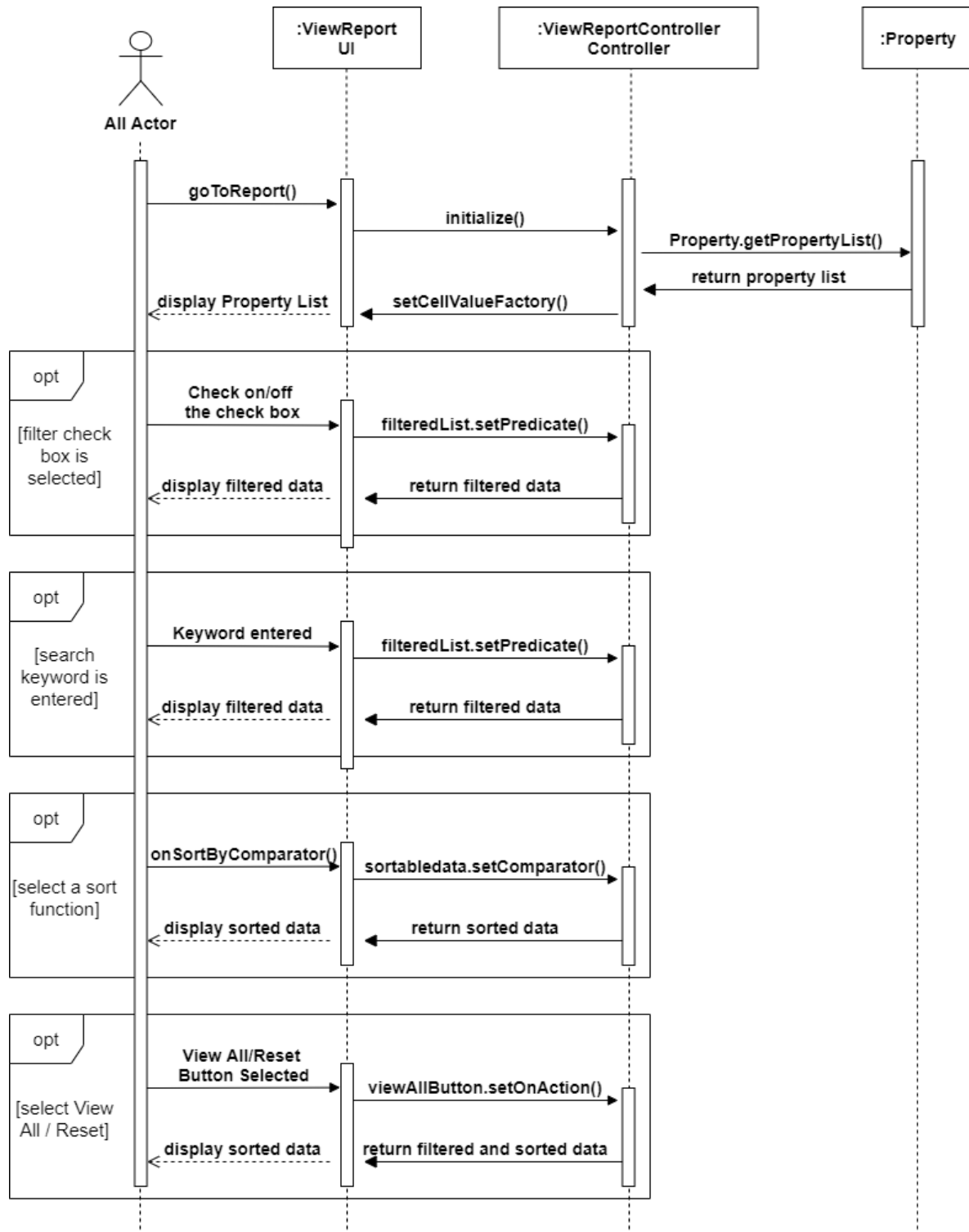
6.6. View Property Details

Users view the specific property details.
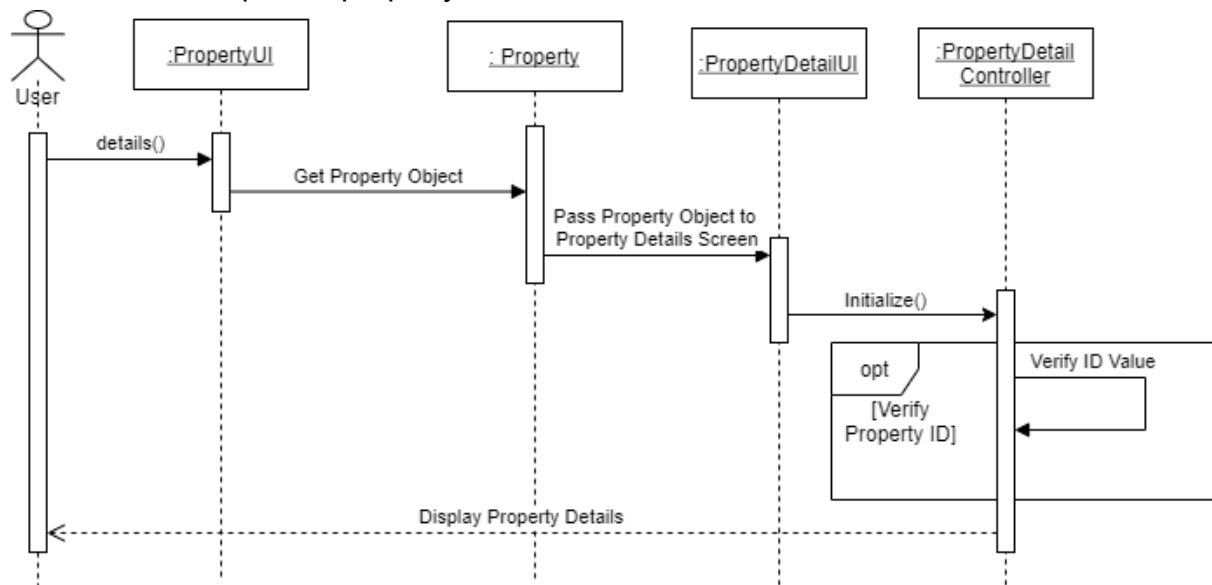
6.7. Delete property

Admin/Agent/Owner delete property.
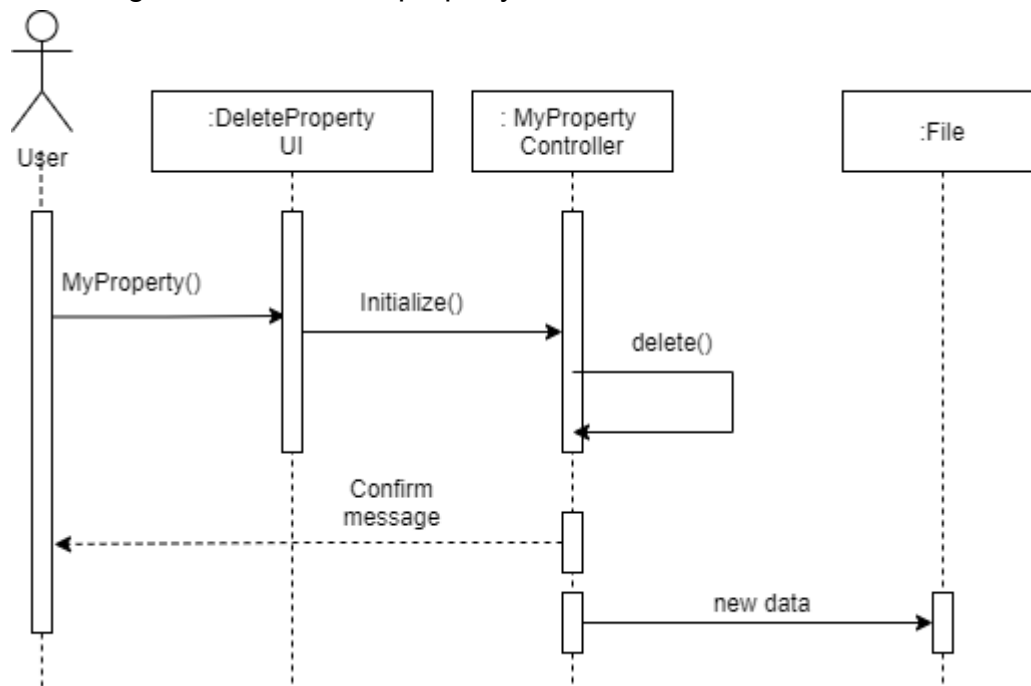
6.8.   Edit property

Admin/Agent/Owner edit property details.
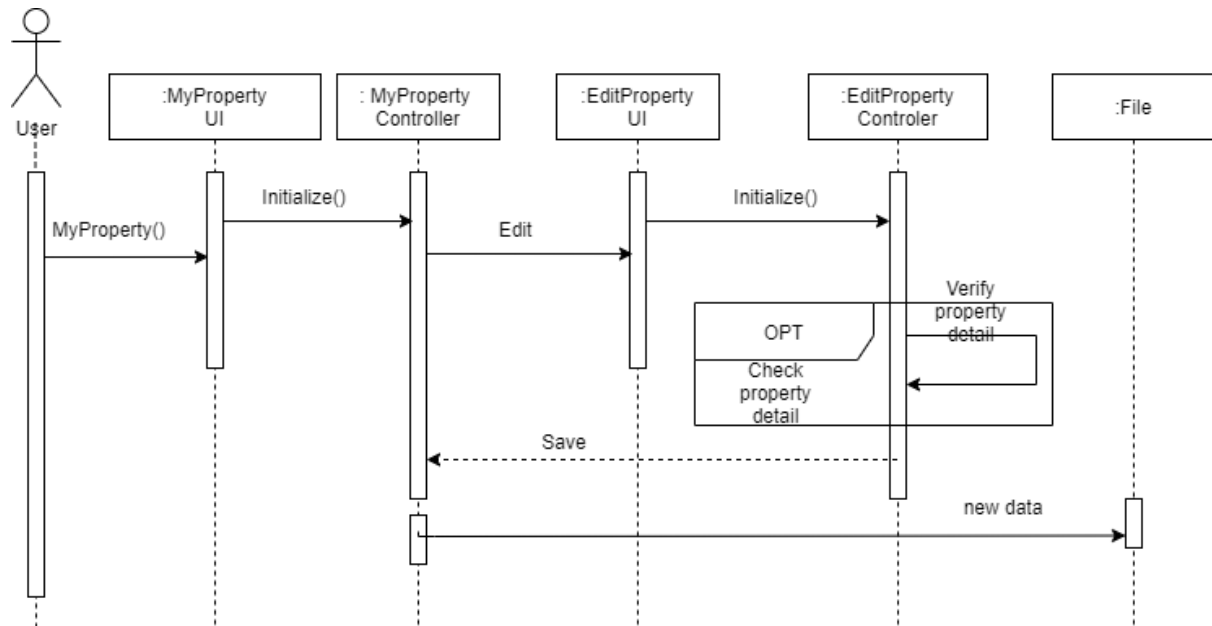
6.9.   Kick tenant

Admin/Agent/Owner kicks the tenant from property.
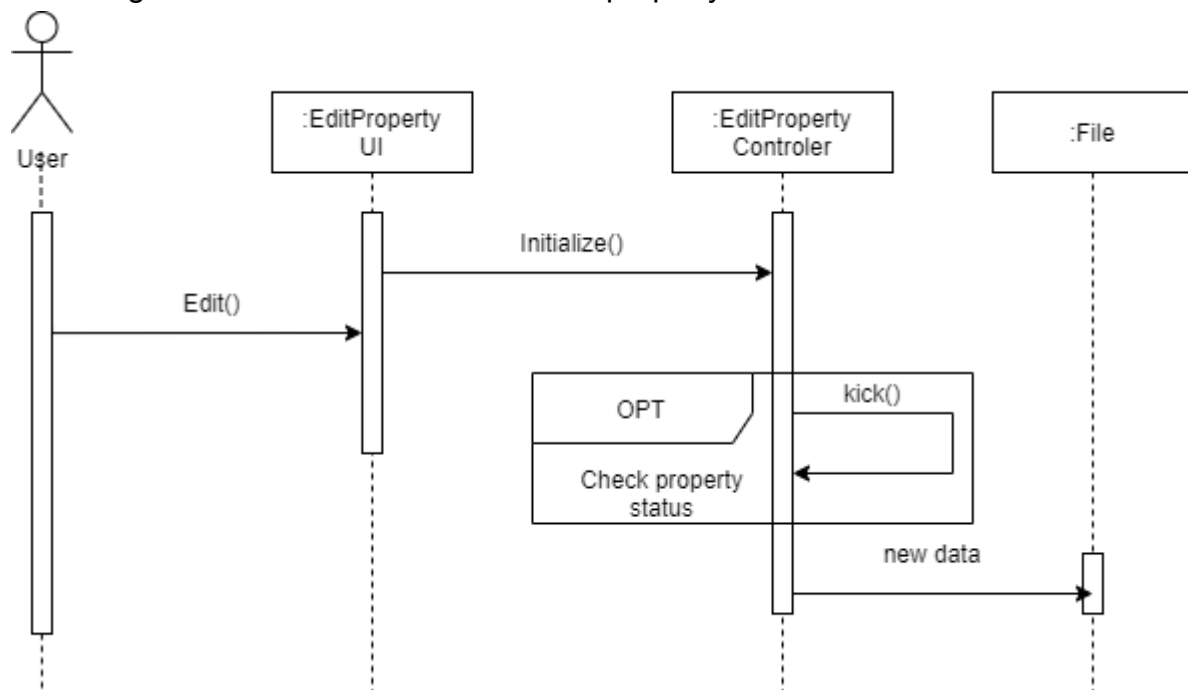
6.10. Admin Approve Property

Admin approves property of agent/owner.
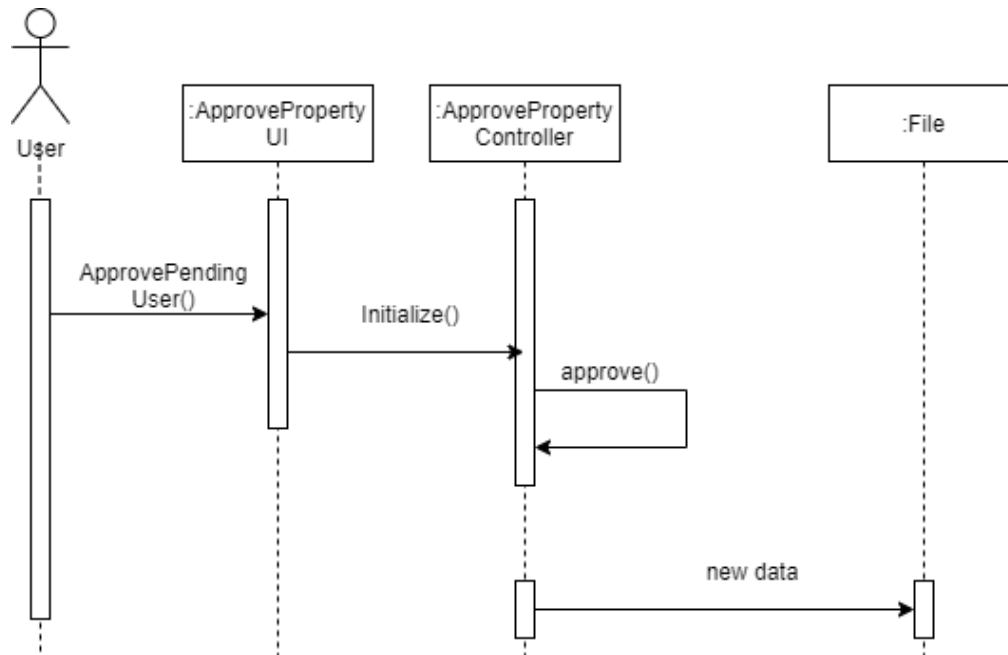
6.11. Admin Approve Pending User

Admin approve pending user such as property owner, property agent or potential tenant to be added into the system
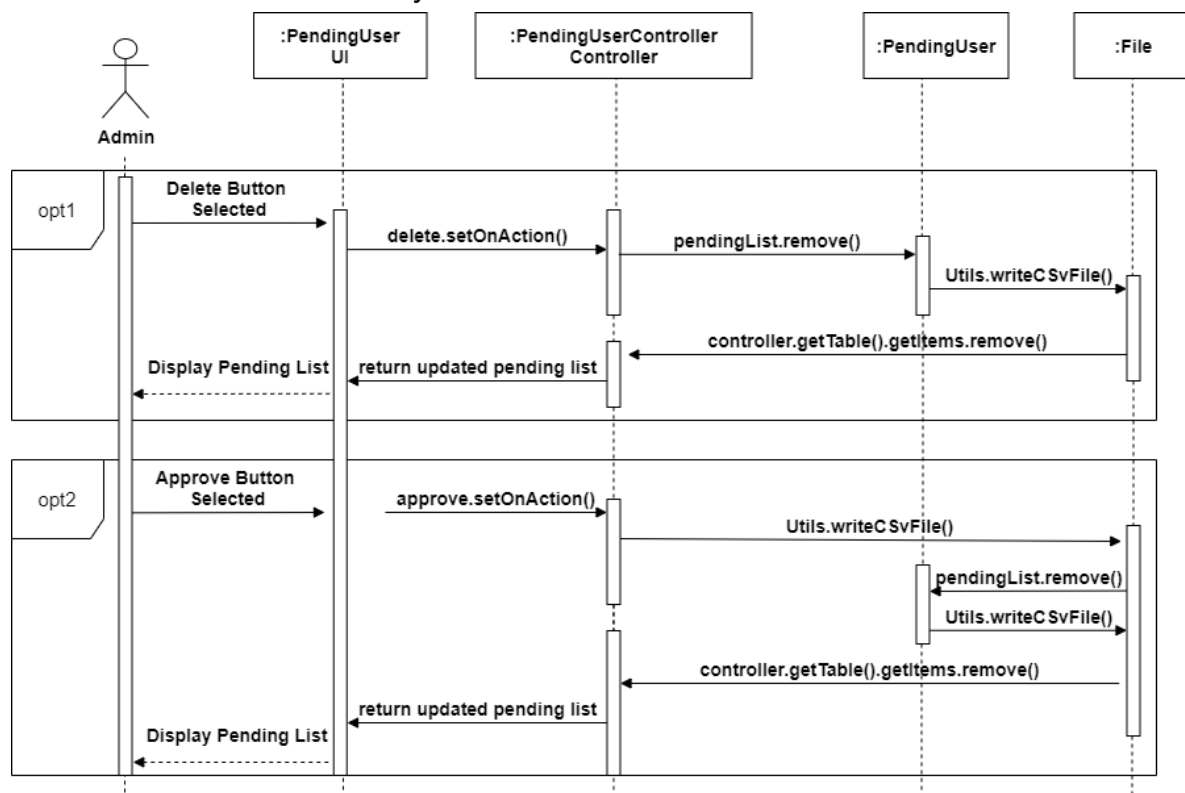
6.12. Admin Comment Property

Admin gives comment to property.
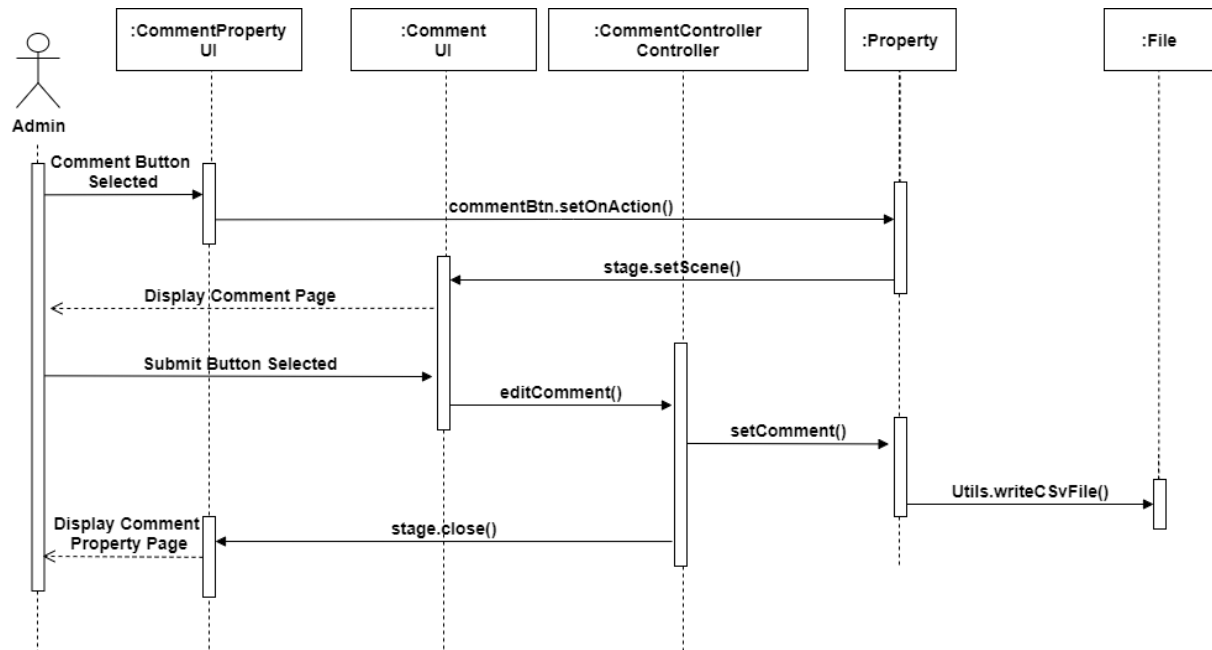
6.13. Admin Add Admin

Admin can add more admin accounts into the system.
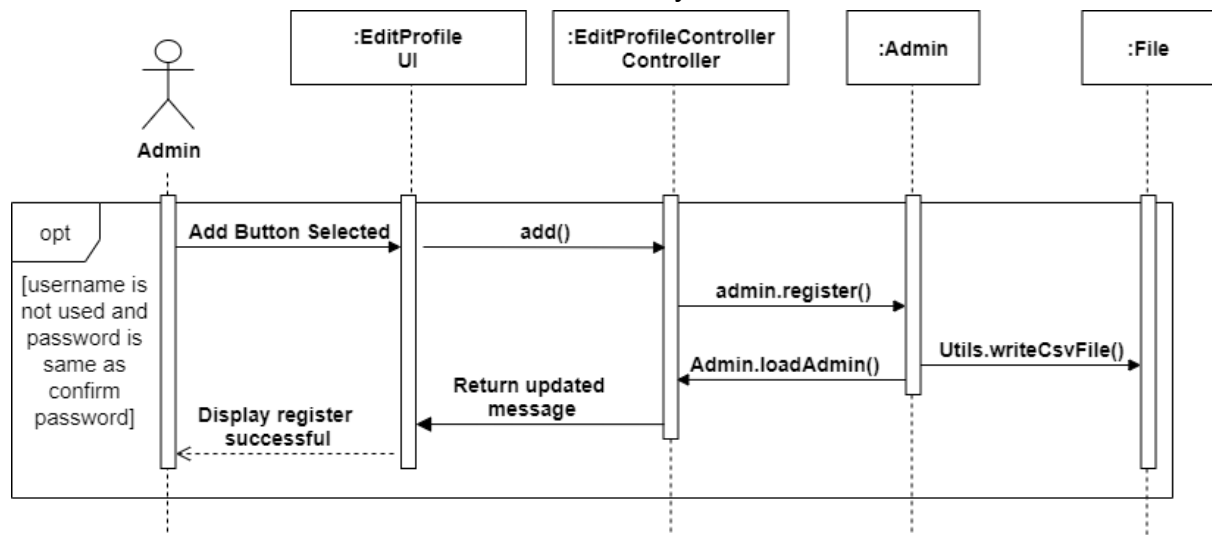
6.14.   Agent/Owner Hide Property

Agent/owner changes the status of the property into hidden.
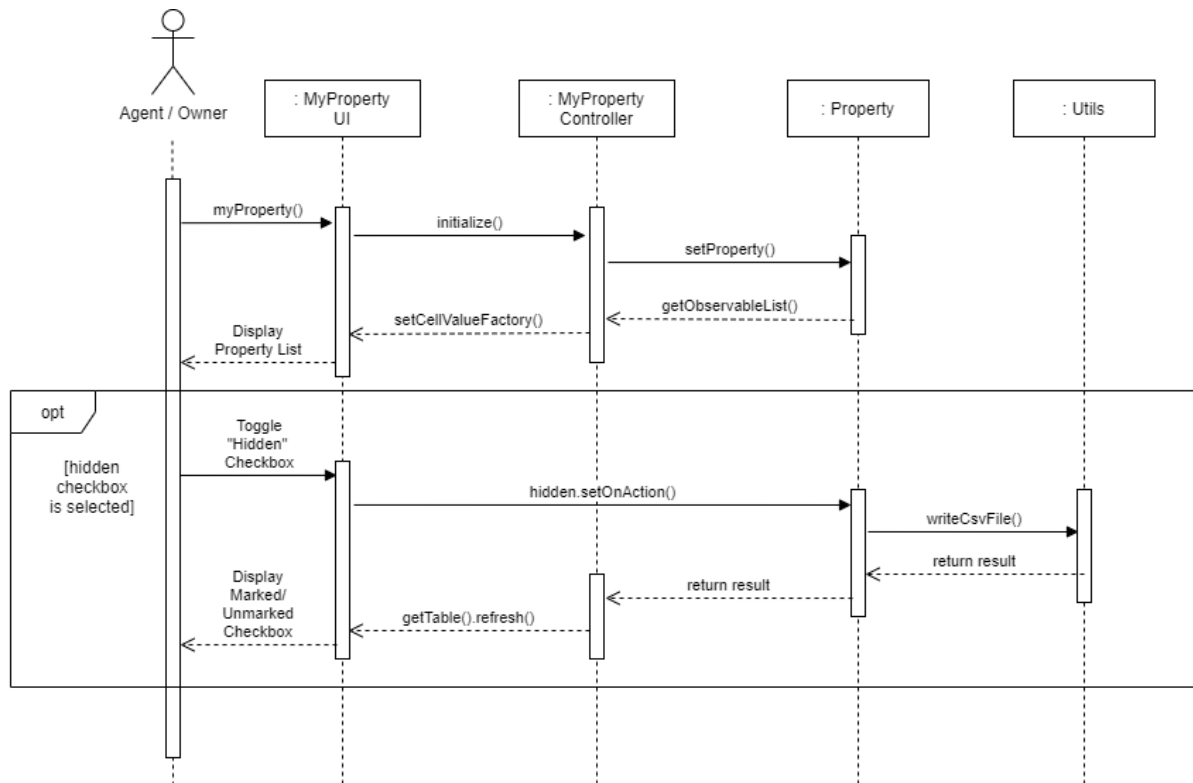
6.15.   Agent/Owner Add Property

Agent/owner adds new property.
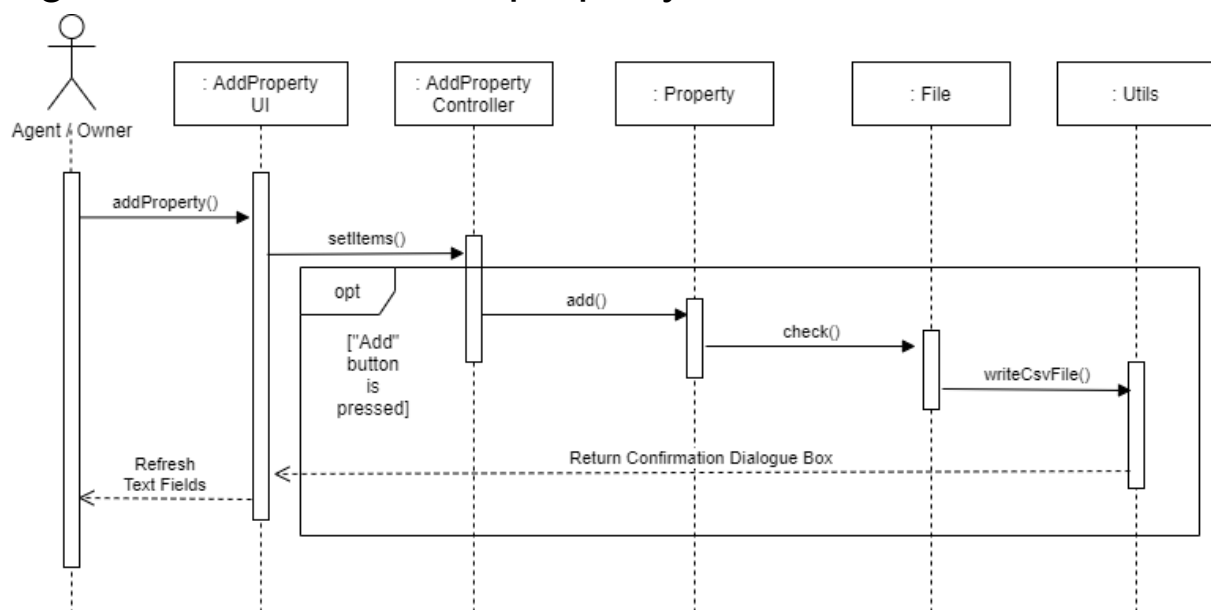
6.16. Agent/Owner Approve Tenant

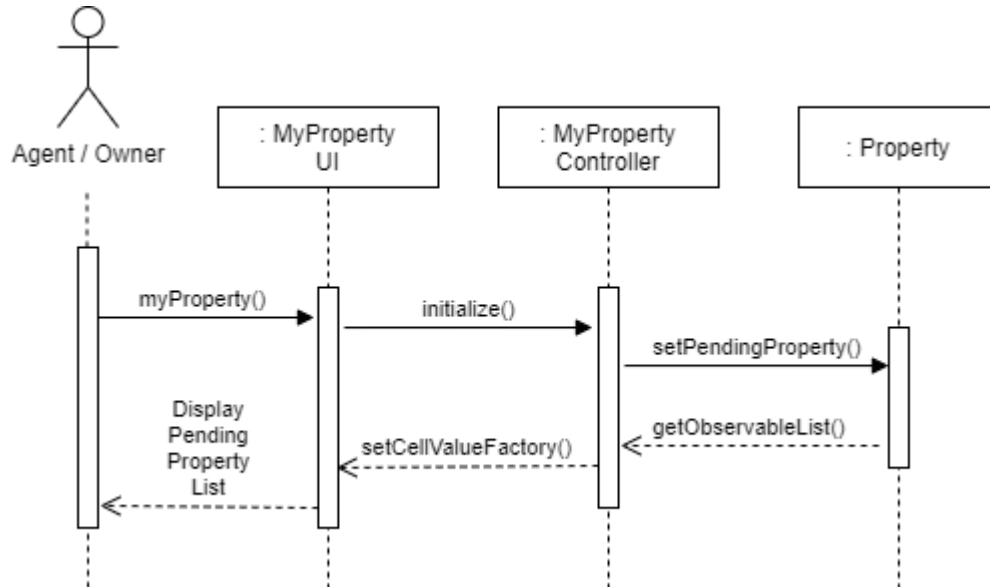Agent/Owner approves the tenant's rental application.
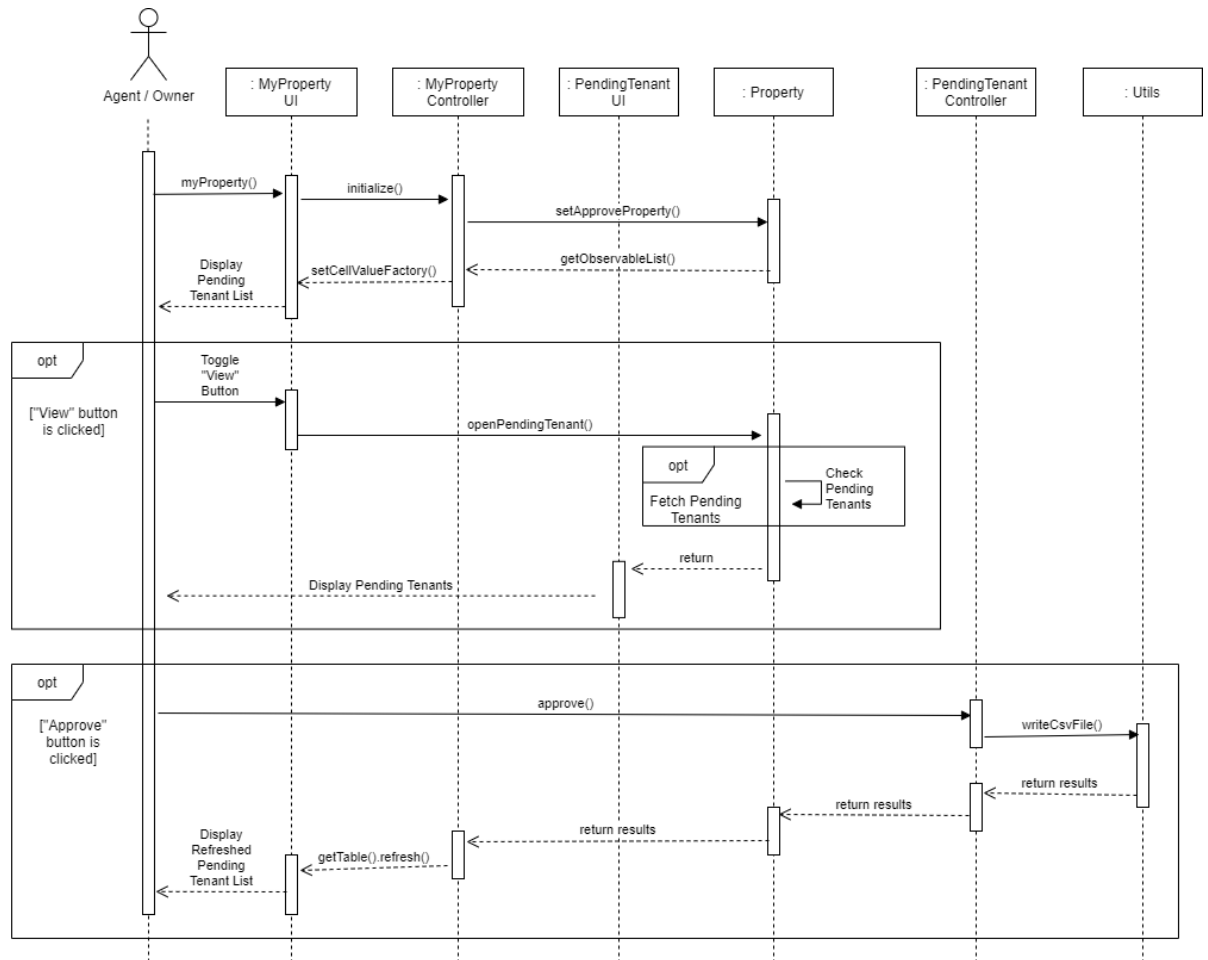
6.17. Tenant Rent Property

Tenants can view the details of the properties approved by the Admin. Rental applications can also be submitted on the same page.
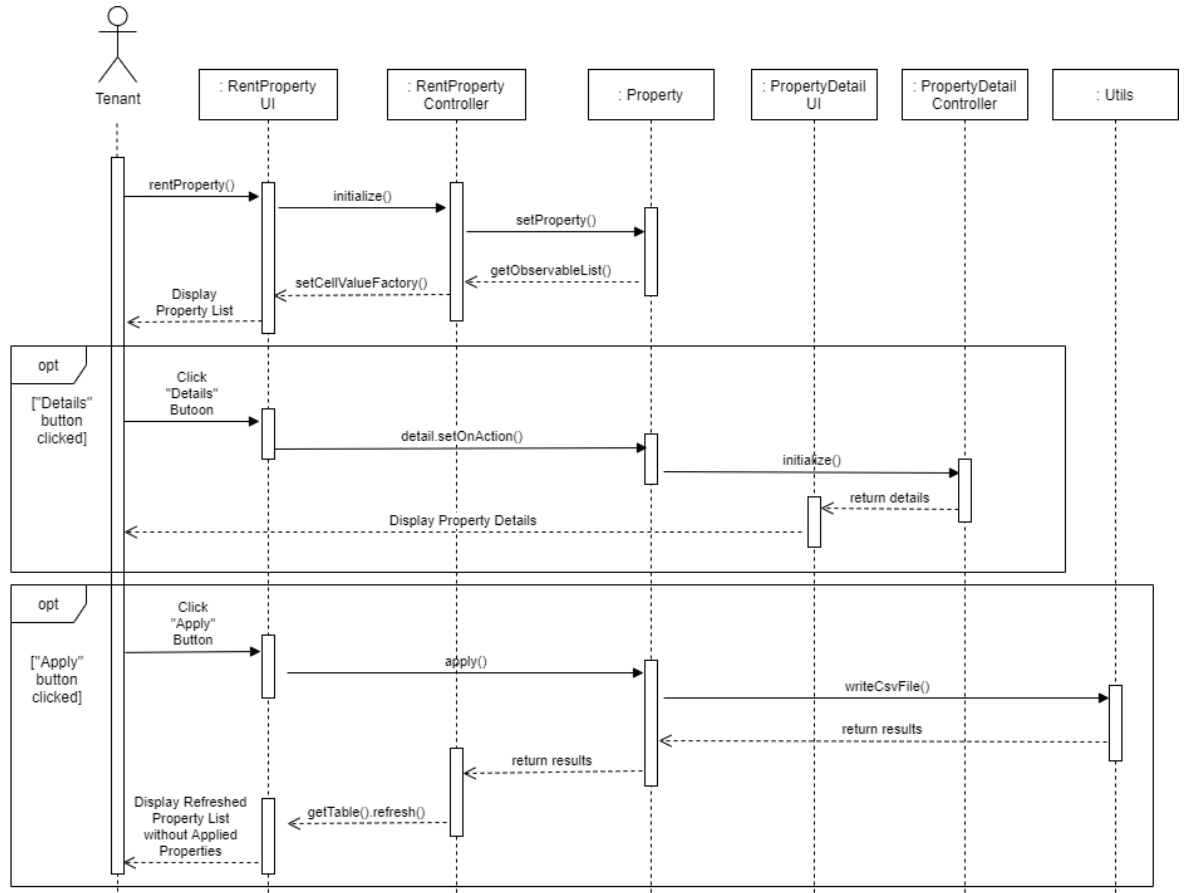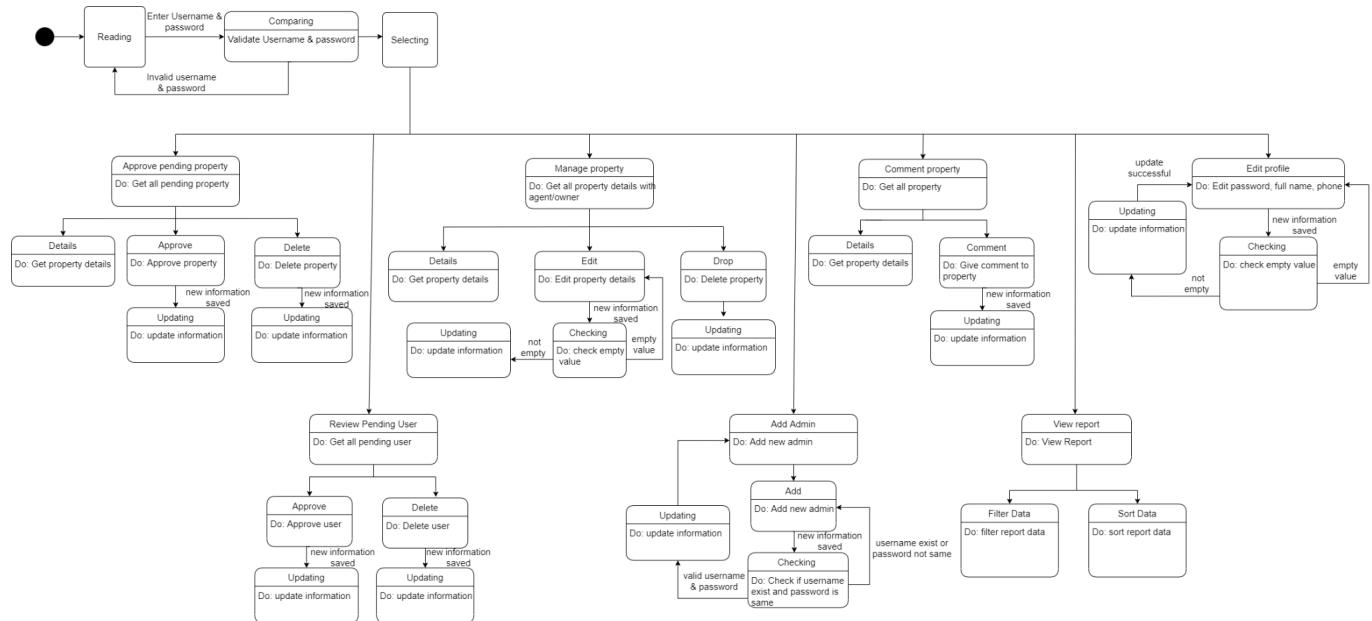
6.18. Agent/Owner View Pending Property

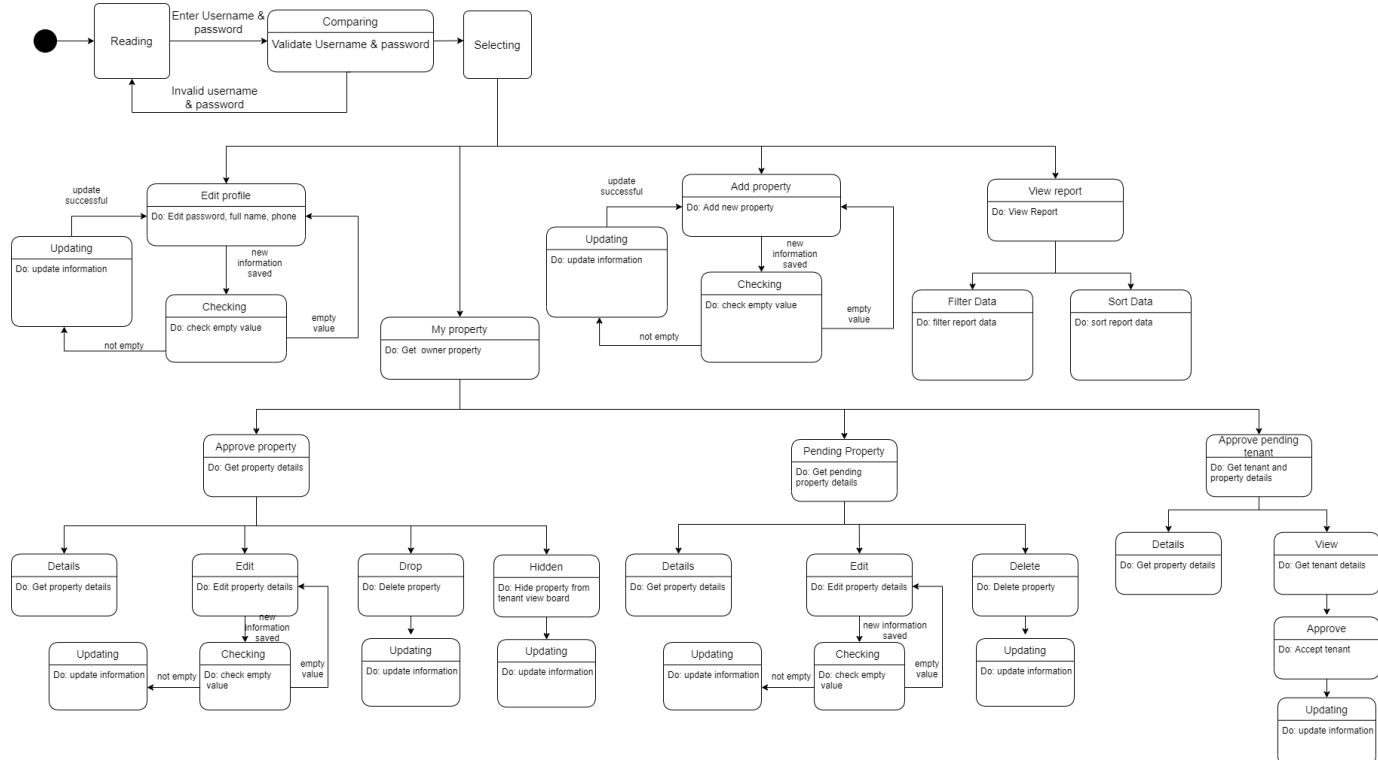Agent/Owner loads up the list of properties which are waiting for approval by the Admin to be listed for rental.
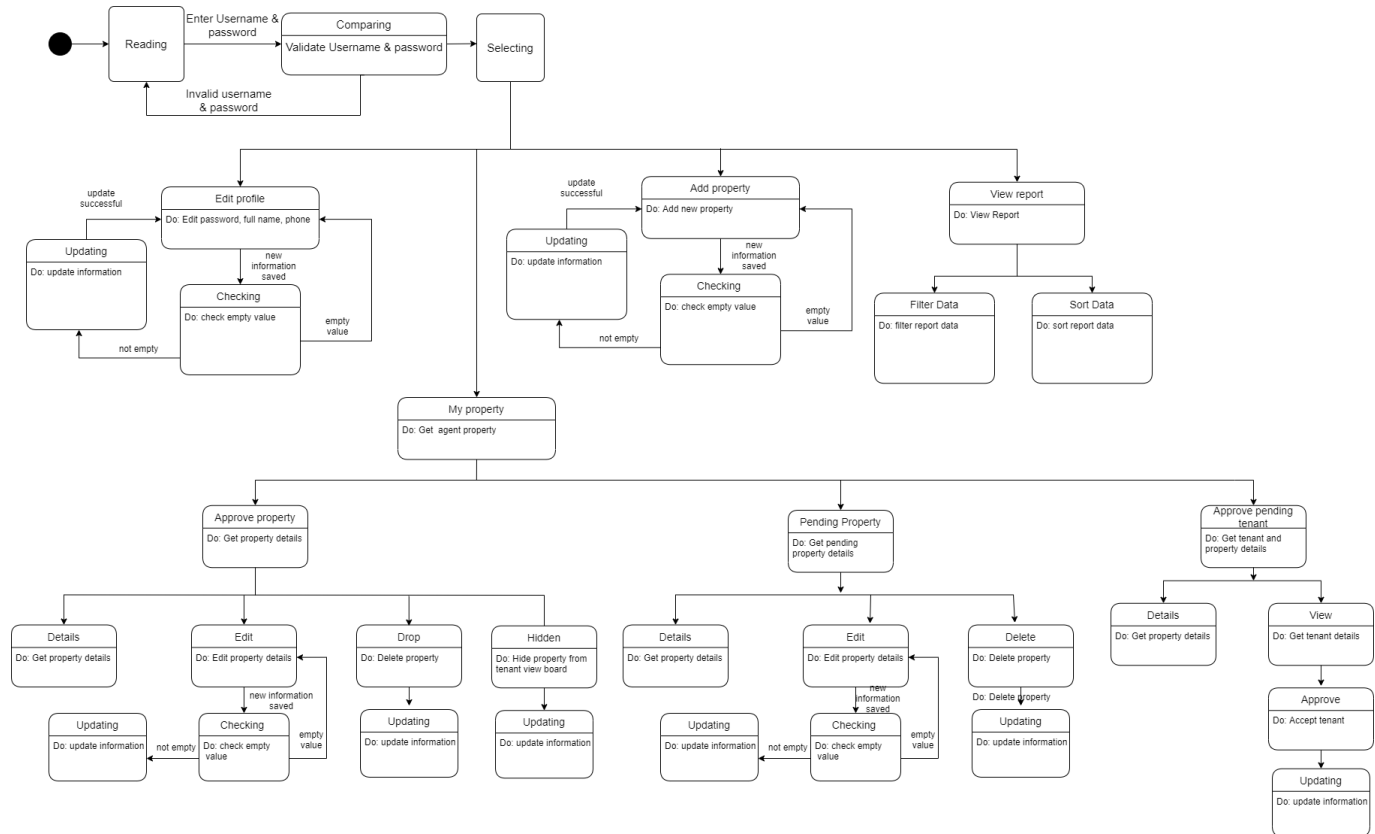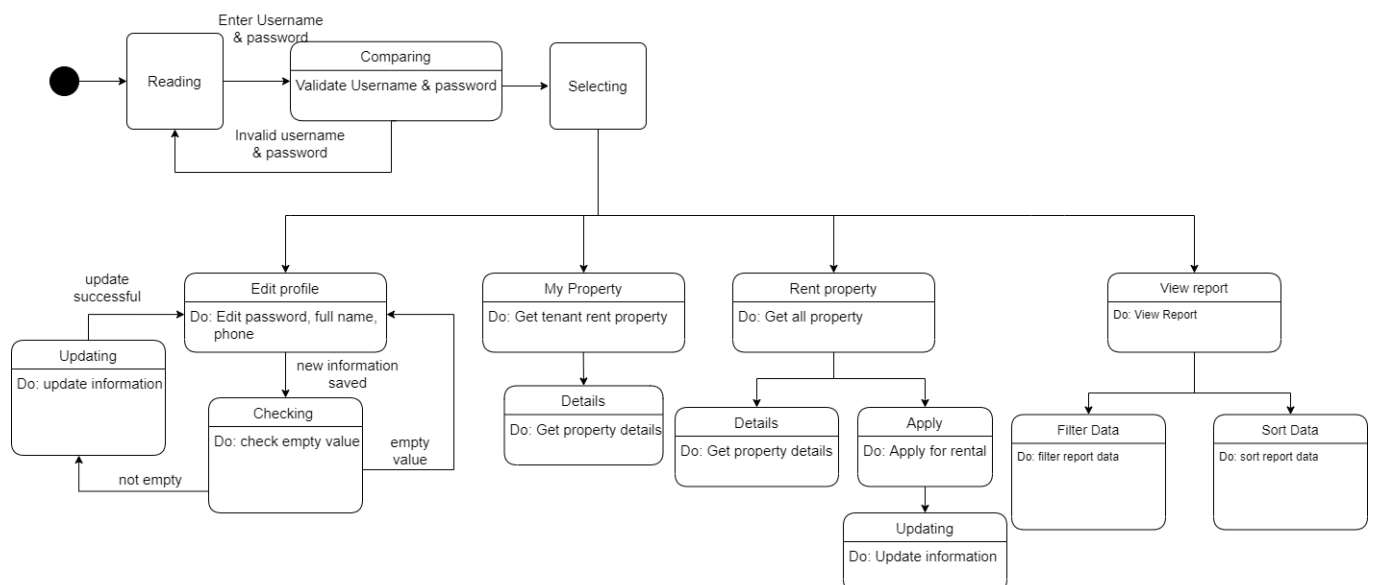
7.State Diagram

Admin

Reading — Enter Username & password → **Comparing** (Validate Username & password) → **Selecting**
Invalid username & password

Approve pending property — Do: Get all pending property
- **Details** — Do: Get property details
- **Approve** — Do: Approve property → new information saved → **Updating** — Do: update information
- **Delete** — Do: Delete property → new information saved → **Updating** — Do: update information

Manage property — Do: Get all property details with agent/owner
- **Details** — Do: Get property details
- **Edit** — Do: Edit property details → new information saved → **Checking** — Do: check empty value → not empty / empty value → **Updating** — Do: update information
- **Drop** — Do: Delete property → **Updating** — Do: update information

Comment property — Do: Get all property
- **Details** — Do: Get property details
- **Comment** — Do: Give comment to property → new information saved → **Updating** — Do: update information

Edit profile — Do: Edit password, full name, phone → update successful → **Updating** — Do: update information → new information saved → **Checking** — Do: check empty value → not empty / empty value

Review Pending User — Do: Get all pending user
- **Approve** — Do: Approve user → new information saved → **Updating** — Do: update information
- **Delete** — Do: Delete user → new information saved → **Updating** — Do: update information

Add Admin — Do: Add new admin
- **Add** — Do: Add new admin → new information saved → **Checking** — Do: Check if username exist and password is same → valid username & password → **Updating** — Do: update information / username exist or password not same

View report — Do: View Report
- **Filter Data** — Do: filter report data
- **Sort Data** — Do: sort report data

Property Owner

Reading — Enter Username & password → **Comparing** (Validate Username & password) → **Selecting**
Invalid username & password

Edit profile — Do: Edit password, full name, phone → update successful → **Updating** — Do: update information → new information saved → **Checking** — Do: check empty value → not empty / empty value

Add property — Do: Add new property → update successful → **Updating** — Do: update information → new information saved → **Checking** — Do: check empty value → not empty / empty value

View report — Do: View Report
- **Filter Data** — Do: filter report data
- **Sort Data** — Do: sort report data

My property — Do: Get owner property

Approve property — Do: Get property details
- **Details** — Do: Get property details
- **Edit** — Do: Edit property details → new information saved → **Checking** — Do: check empty value → not empty / empty value → **Updating** — Do: update information
- **Drop** — Do: Delete property → **Updating** — Do: update information
- **Hidden** — Do: Hide property from tenant view board → **Updating** — Do: update information

Pending Property — Do: Get pending property details
- **Details** — Do: Get property details
- **Edit** — Do: Edit property details → new information saved → **Checking** — Do: check empty value → not empty / empty value → **Updating** — Do: update information
- **Delete** — Do: Delete property → **Updating** — Do: update information

Approve pending tenant — Do: Get tenant and property details
- **Details** — Do: Get property details
- **View** — Do: Get tenant details → **Approve** — Do: Accept tenant → **Updating** — Do: update information

Property Agent

Potential Tenant

8. Appendix

7.1 Sample Interface Screen

Image 1: Login Screen
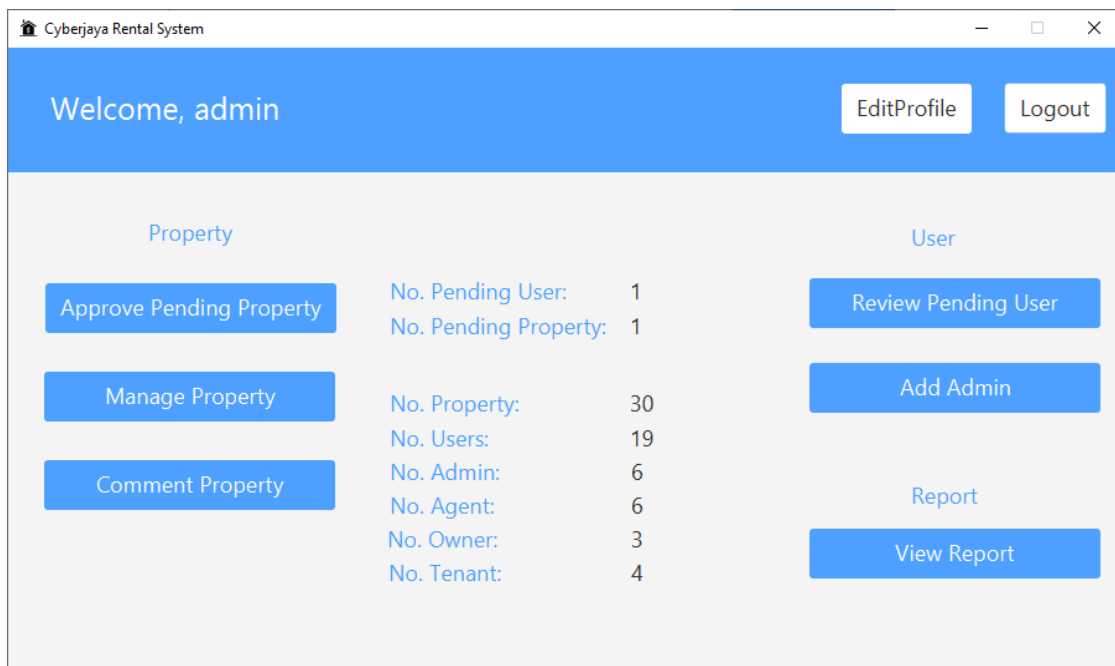

Image 2: Registration Screen

Image 3: User Home Screen
NOTE: Different users will have different home screen layouts and functions.
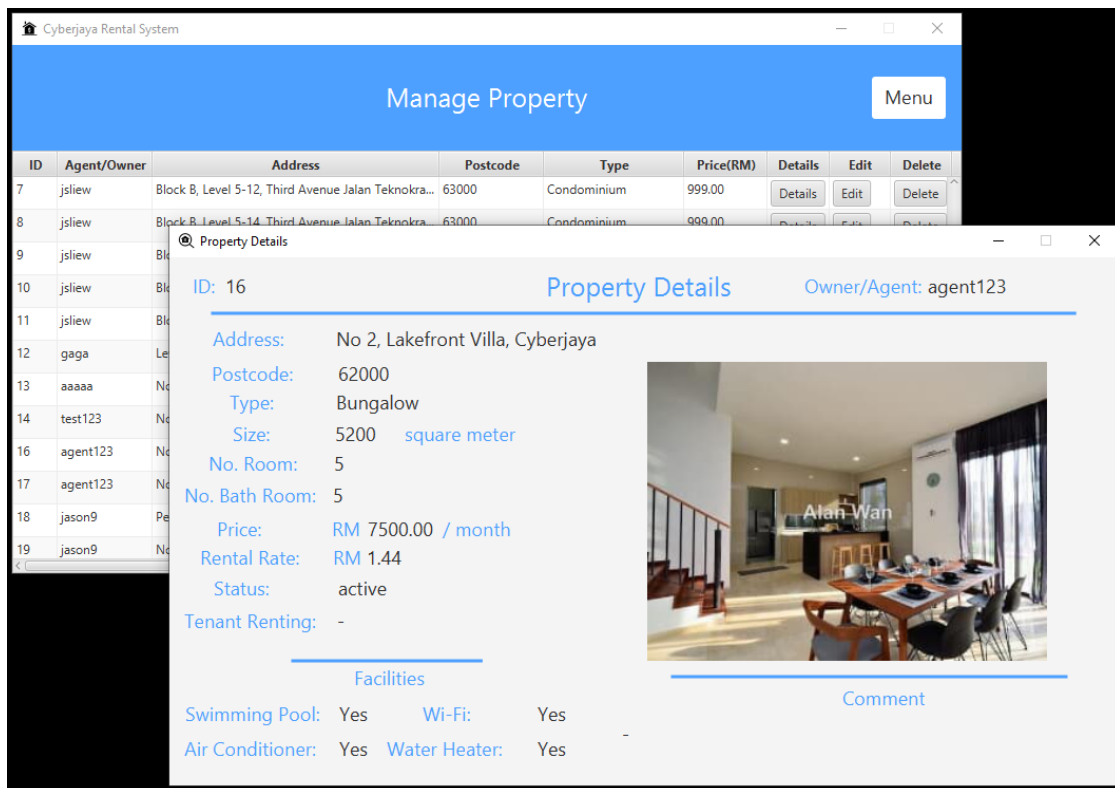

Image 4: Property List with Pop-up Property Detail

Image 5: Edit Profile Screen

Image 6: View Property Report

7.2 Execution Instructions

The development environment of this program is as follows:

Target Platform	Microsoft Windows
JDK Version	OpenJDK 11.0.12
JavaFX Version	JavaFX 11.0.2
IDE	Visual Studio Code (Source Code Editor) IntelliJ IDEA (JAR Compilations)

Users can run the program by either executing the included `CORMS.jar` or running the code within the IDE editor.

ATTENTION: Do NOT modify the location of all the files and directories in order to avoid any errors while executing the program.

7.2.1 Executing The JAR File

1. Double click the `CORMS.jar` to execute the program.


CORMS.jar

2. The program will pop up and users may start using it.

3. Alternatively, users may execute the `CORMS.jar` via the command line terminal. Simply navigate to the directory where the project folder is and execute the command: `java -jar CORMS.jar`. This method is preferred for debugging purposes.
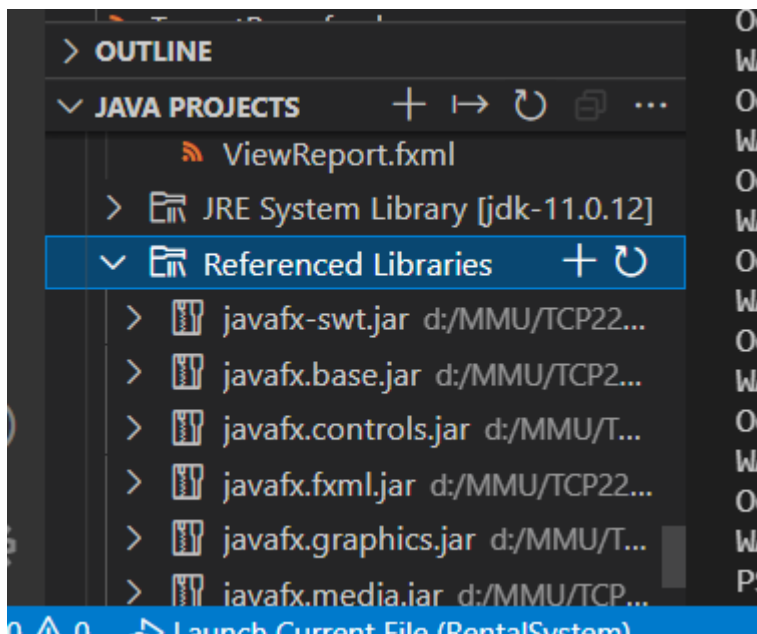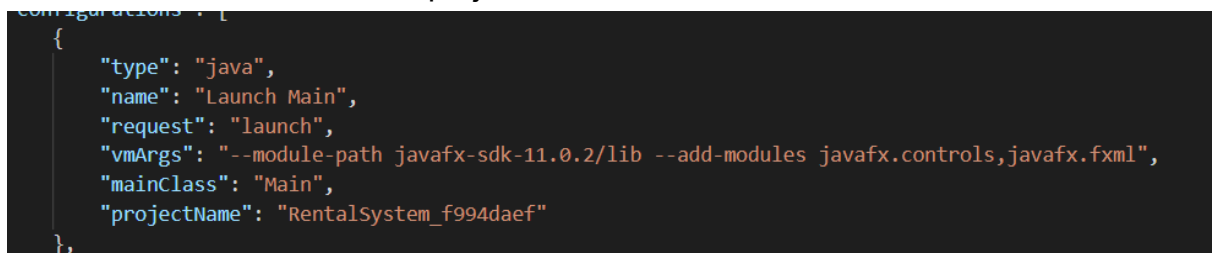
```
PS C:\Users\ycteoh> cd D:\CORMS
PS D:\CORMS> java -jar CORMS.jar
```

7.2.2 Execute via Source Code In Visual Studio Code

1. Create a java project in VS code .
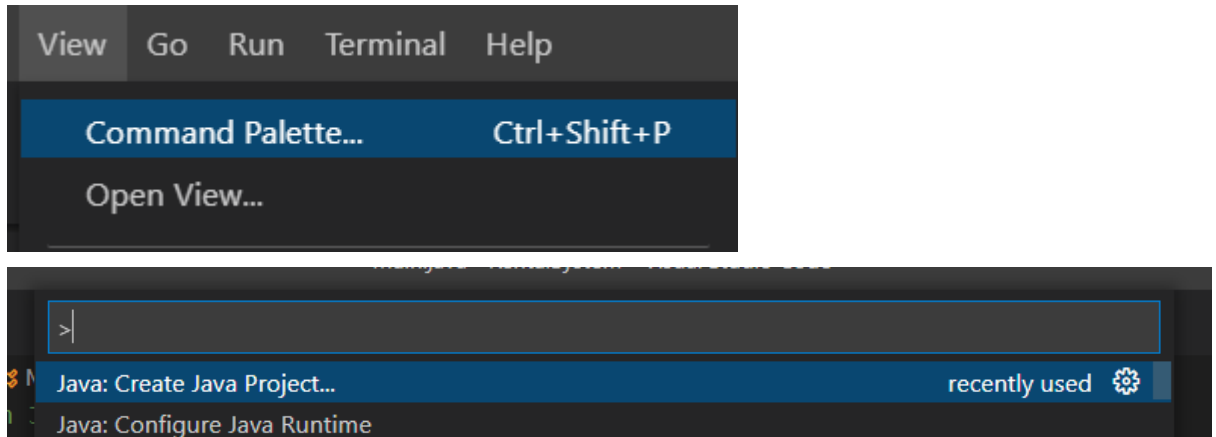
2. Set up project files, go to launch.json add the javafx library location and link the referenced libraries to the project folder.

```
configurations : [
    {
        "type": "java",
        "name": "Launch Main",
        "request": "launch",
        "vmArgs": "--module-path javafx-sdk-11.0.2/lib --add-modules javafx.controls,javafx.fxml",
        "mainClass": "Main",
        "projectName": "RentalSystem_f994daef"
    },
```


3. Paste the all source code into the src folder.

4. Paste the assets and csv folder outside the src folder but inside the project folder.

7.2.3 Execute via Source Code In CMD

1. Open the cmd inside the source code folder.

2. Compile the source code using the following command:
   ```
   javac --module-path javafx-sdk-11.0.2\lib --add-modules
   javafx.controls,javafx.fxml Main.java
   ```

   Please replace the `javafx-sdk-11.0.2\lib` with your own library location.
   Do keep in mind that some java files might not compile in one go, you might
   need to manually compile every java file and make sure every java file is
   compiled.

3. To run the program using the following command:
   ```
   java javafx-sdk-11.0.2\lib --add-modules
   javafx.controls,javafx.fxml Main
   ```

   Replace the `javafx-sdk-11.0.2\lib` with your own library location.