**QProperty Information Management System**

Software Requirement Elicitation, Modelling and Design Document.

Client: QProperty real estate agency

Vendor: Tarique Turnbull, StudentID: 12177936

Due Date: Week 10 Friday, 17th May 2024 11.45pm

**TABLE OF CONTENTS**

[1 SYSTEM REQUIREMENTS 2](#_Toc166851202)

[1.1 Functional requirements 2](#_Toc166851203)

[1.2 Non-Functional requirements 2](#_Toc166851204)

[2 SYSTEM ARCHITECTURE 2](#_Toc166851205)

[2.1 Architectural Design 2](#_Toc166851206)

[2.2 Use Case Diagram 2](#_Toc166851207)

[2.3 Class Diagram 2](#_Toc166851208)

[Sequence Diagrams 2](#_Toc166851209)

[3 DATA DESIGN 2](#_Toc166851210)

[4 INTERFACE DESIGN 2](#_Toc166851211)

[Homepage View 2](#_Toc166851212)

[About View 2](#_Toc166851213)

[Account View 2](#_Toc166851214)

[Customer View 2](#_Toc166851215)

[Property View 2](#_Toc166851216)

[Repair job View 2](#_Toc166851217)

[Report View 2](#_Toc166851218)

[MVC 2](#_Toc166851219)

[Singleton 2](#_Toc166851220)

[7. REQUIREMENTS MATRIX 2](#_Toc166851221)

[8 Test Plan 2](#_Toc38188434)

# SYSTEM REQUIREMENTS

## Functional requirements

### Login/Register account

**As a user, I want to register an account.**

Acceptance criteria:

* Adds new user to the DB based on input fields.
* Must include and validate fields for:
  + First name.
  + Last name.
  + Email.
  + Address.
  + Phone number.
  + Username.
  + Password.
* Password must be encrypted.

**As a user, I want to login so that I can get full access to the program.**

Acceptance criteria:

* Authenticates using username and password.
* Refuses login attempt for incorrect credentials.

**As a user, I want to logout.**

Acceptance criteria:

* Logs user out of system.
* User should be unable to access the rest of the program after logout.

### Customer View

**As a user, I want to add new customers**

Acceptance criteria:

* Adds new customer record to DB based on input fields.
* Must include and validate fields for:
  + First name
  + Last name
  + Address
  + Phone number

**As a user, I want to delete customers.**

Acceptance criteria:

* Must delete the customer record from the database.

**As a user, I want to update existing customer records.**

Acceptance criteria:

* Updates existing customer record to DB based on input fields.
* Must include and validate fields for:
  + First name
  + Last name
  + Address
  + Phone number

**As a user, I want to view all customers.**

Acceptance criteria:

* User can view all customer records.

**As a user, I want to search for specific customer records.**

Acceptance criteria:

* Search is done using either phone number or last name.
* Displays any results that match search criteria.
* Search results should be selectable with the mouse.
* Selecting a result displays all related info.

### Property View

**As a user, I want to create new property entries**

Acceptance criteria:

* Adds new property record to DB based on input fields.
* Must include and validate fields for:
  + address
  + num bedrooms
  + num bathrooms
  + has car park
  + year built
  + property type
  + managing agents name

**As a user, I want to delete properties.**

Acceptance criteria:

* Must delete the property record from the database.

**As a user, I want to update existing properties**

Acceptance criteria:

* Updates property record to DB based on input fields.
* Must include and validate fields for:
  + address
  + num bedrooms
  + num bathrooms
  + has car park
  + year built
  + property type
  + managing agents name

**As a user, I want to view all properties.**

Acceptance criteria:

* User can view all property records.

**As a user, I want to search for properties by address so that I can view their details or update their properties.**

Acceptance criteria:

* Search is done using address
* Displays any results that match search criteria.
* Search results should be selectable with the mouse.
* Selecting a result displays all related info.

**As a user, I want to associate a property with a customer**

Acceptance criteria:

* Must be able to select a property from a list of options
* Must be able to select a customer from a list of options
* Must be able to make an association between selected customer with the selected property.
* Must be able to search for customers by ID or last name.
* Must be able to search for a specific property by ID or address.

### Repair jobs

**As a user, I want to create new repair job booking entries.**

Acceptance criteria:

* Adds new repair job record to DB based on input fields.
* Must include and validate fields for:
  + Description
  + Booking date
  + Expected Completion date
  + Charge
  + service staff name
  + job type
    - electrical
    - plumbing
    - structural
    - cleaning
    - gardening
    - pest control
  + Job status
    - Pending
    - In progress
    - Completed
    - Cancelled

**As a user, I want to update details of existing repair jobs.**

Acceptance criteria:

* Updates repair job record to DB based on input fields.
* Must include and validate fields for:
  + Description
  + Booking date
  + Expected Completion date
  + Charge
  + service staff name
  + job type
    - electrical
    - plumbing
    - structural
    - cleaning
    - gardening
    - pest control
  + Job status
    - Pending
    - In progress
    - Completed
    - Cancelled

**As a user, I want to delete a repair job record.**

Acceptance criteria:

  - Deletes it from the system.

**As a user, I want to associate a repair job with a property**.

Acceptance criteria:

* Must be able to select a repair job from a list of options
* Must be able to select a property from a list of options
* Must be able to make an association between selected repair job with the selected property.
* Must be able to search for repair jobs by ID or by job type.
* Must be able to search for a specific property by ID or address.

**As a user, I want to view all existing repair jobs.**

Acceptance criteria:

* User can view all repair job records.

**As a user, I want to search for a specific repair job so that I can view all of its information.**

Acceptance criteria:

* Search is done using Job ID or job type.
* Displays any results that match search criteria.
* Search results should be selectable with the mouse.
* Selecting a result displays all related info.

### Manager Report

**As a manager, I want to view a bar chart showing a distribution of all jobs across all properties.**

Acceptance criteria:

* shows all jobs marked as complete, categorised into job type.

**As a manager, I want to view the minimum, maximum and average charge costs for all repair jobs.**

Acceptance criteria:

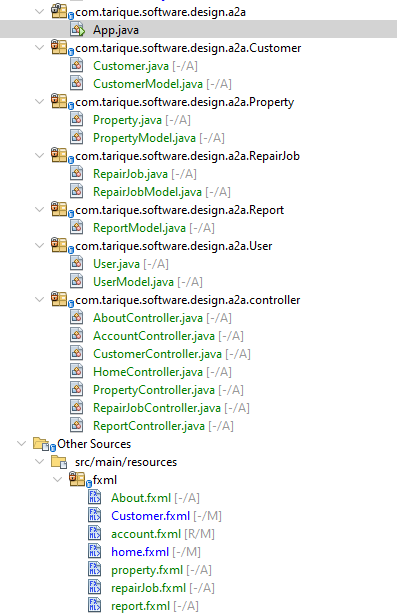
* displays statistics for minimum, maximum and average charge costs across all jobs.

## Non-Functional requirements

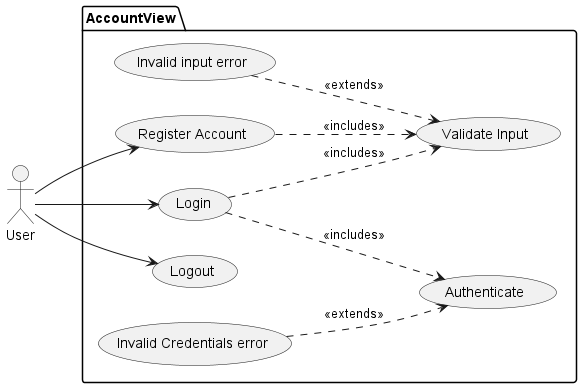
* Data saved must persist between application starts.
* Credentials must be encrypted.
* Password should be hidden from user when it is being typed into the field for privacy (example: show text as asterisk \*\*\*\*\*\*).
* User interface should be fast and responsive.
* User interface should be intuitive and easy to use.
* Must provide a brief description of how to use the program in an about page.
* Must cleanly handle errors without crashing the program.
* Any errors caught should be shown to the user with a brief description.

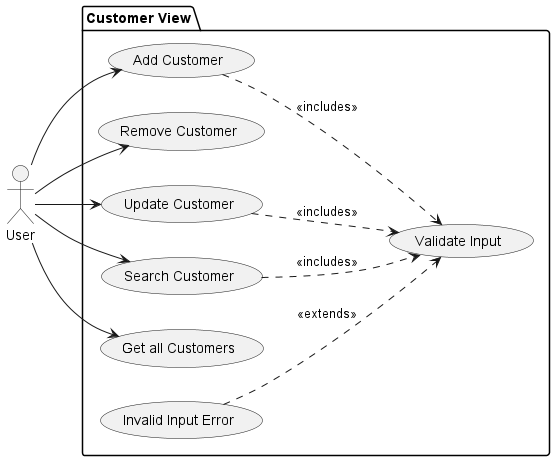
# SYSTEM ARCHITECTURE

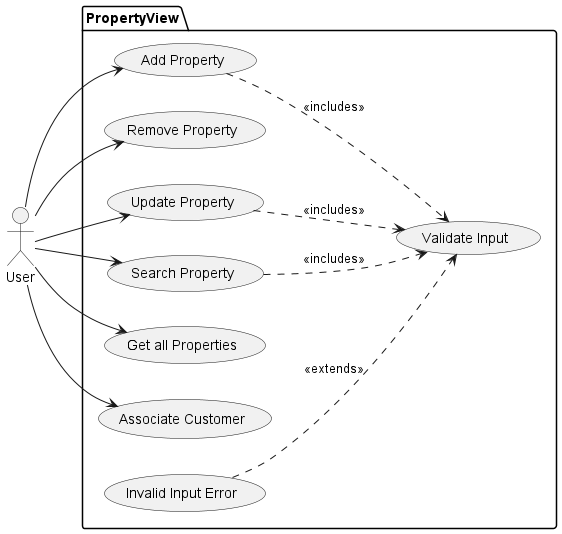
## Architectural Design

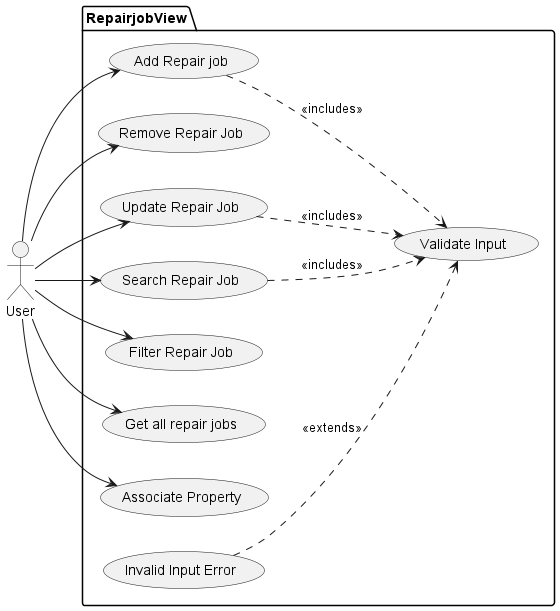


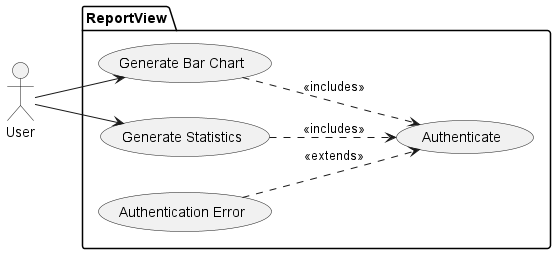
## Use Case Diagram





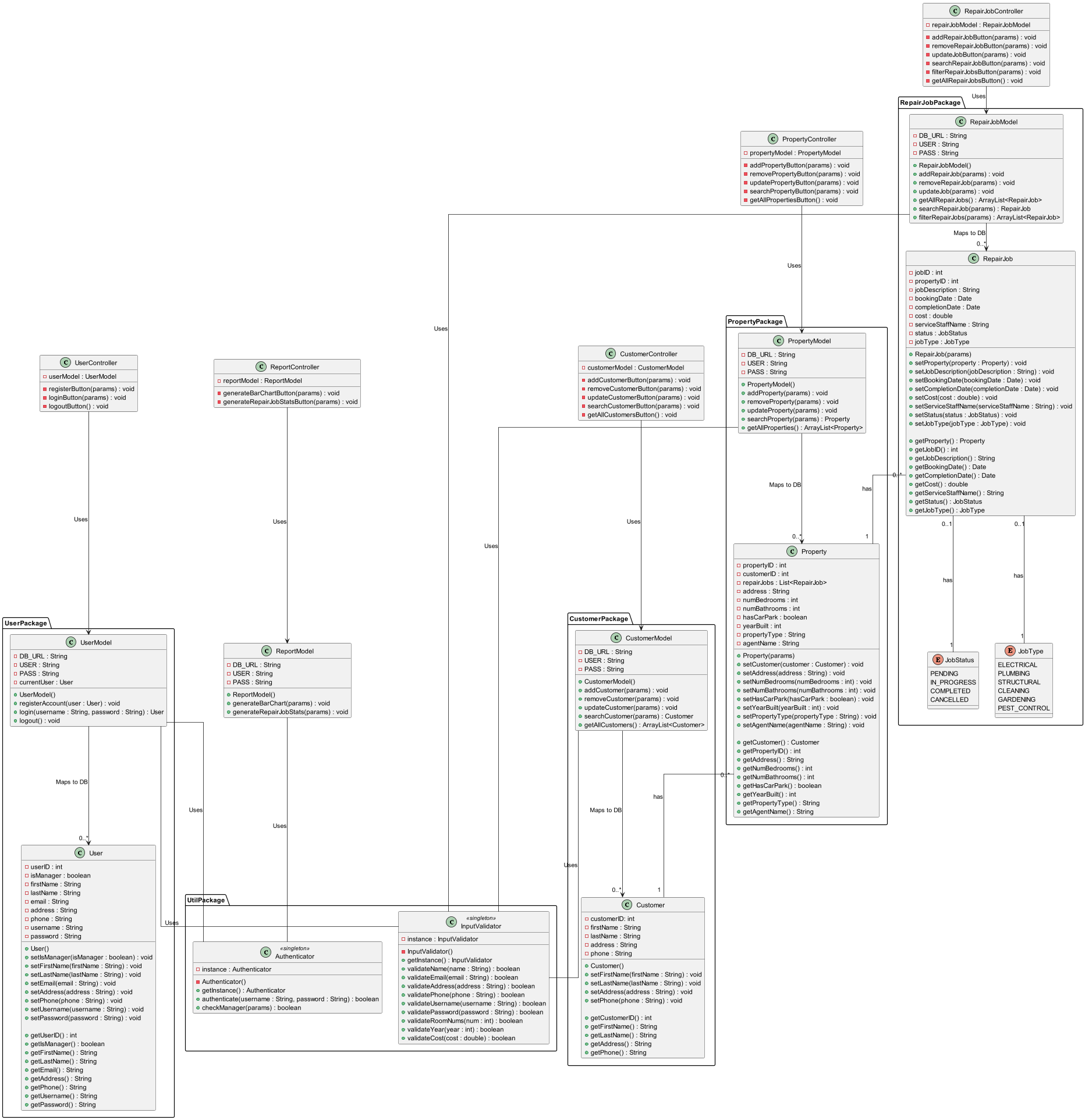






## Class Diagram

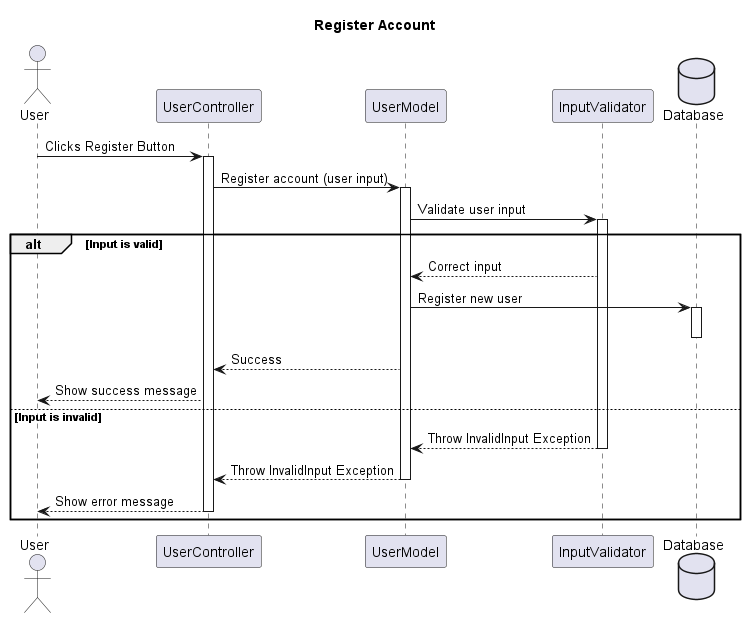
Diagram too large for Microsoft Word. Please set zoom to 230%

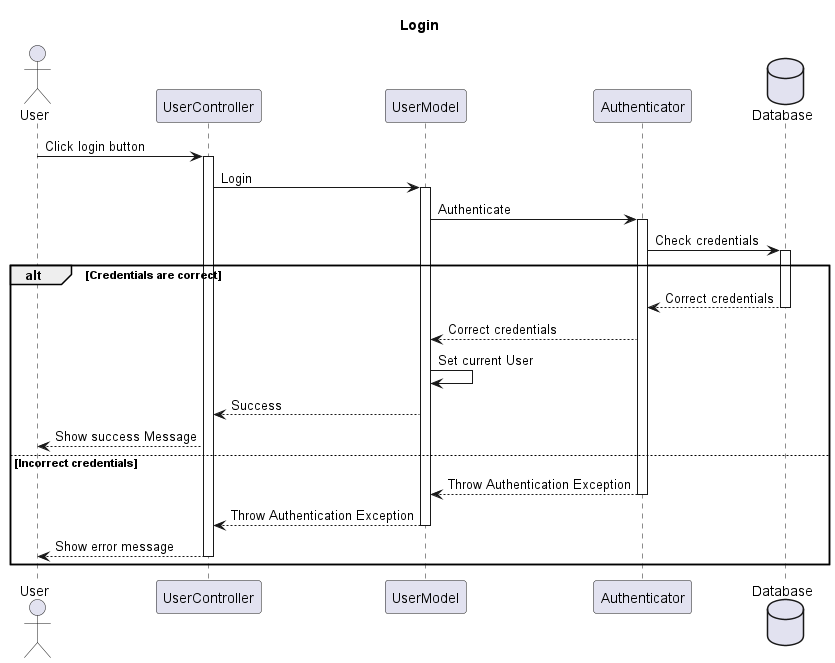


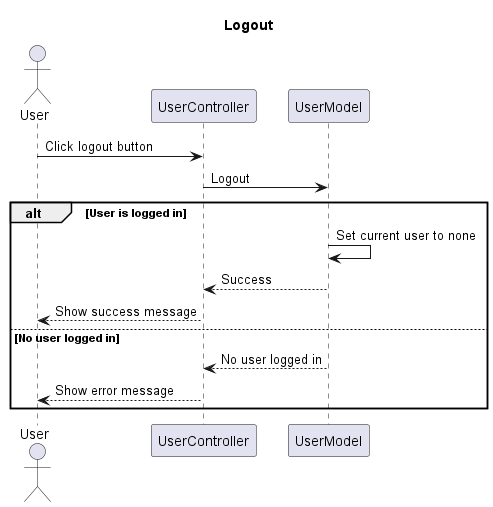
## Sequence Diagrams

All sequence diagrams are separated based on the use cases defined in the use case diagrams.

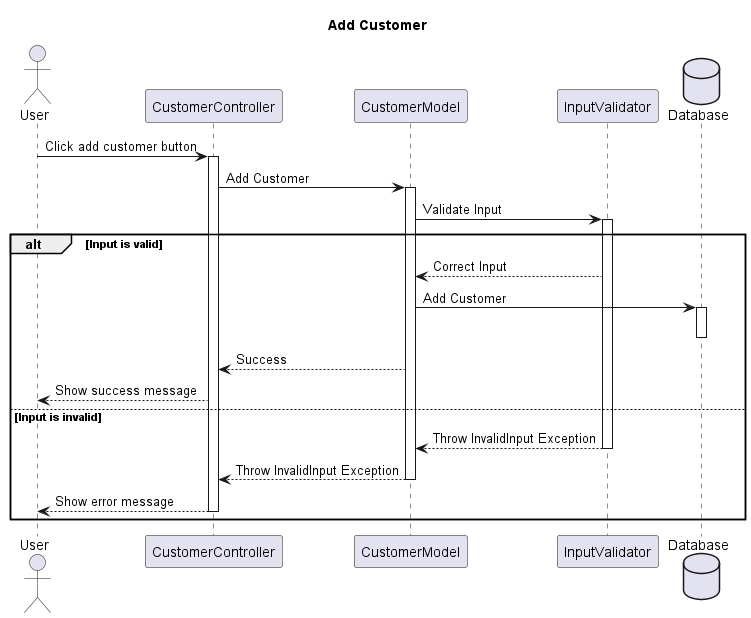
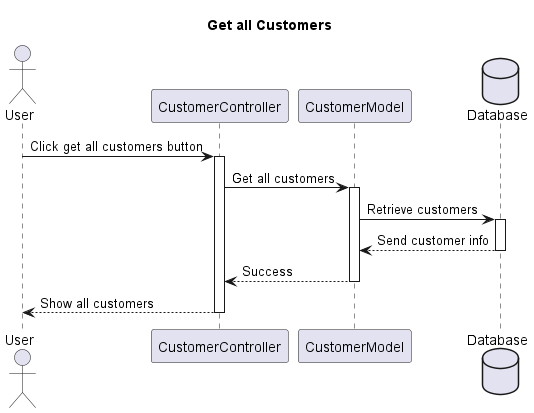
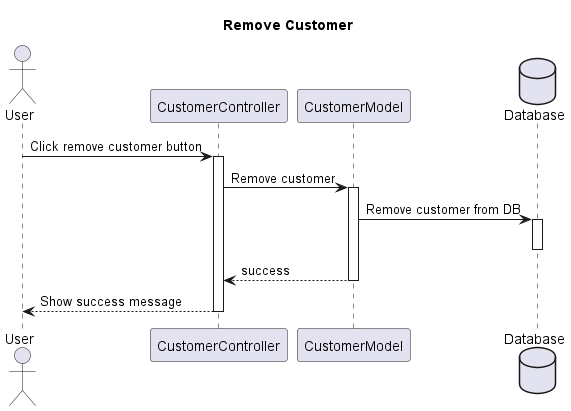
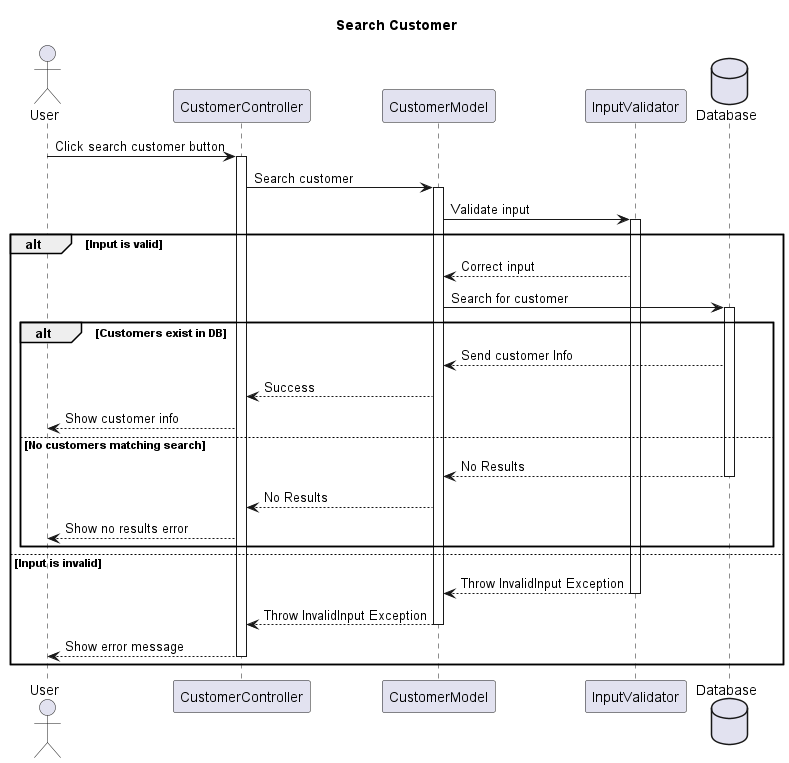
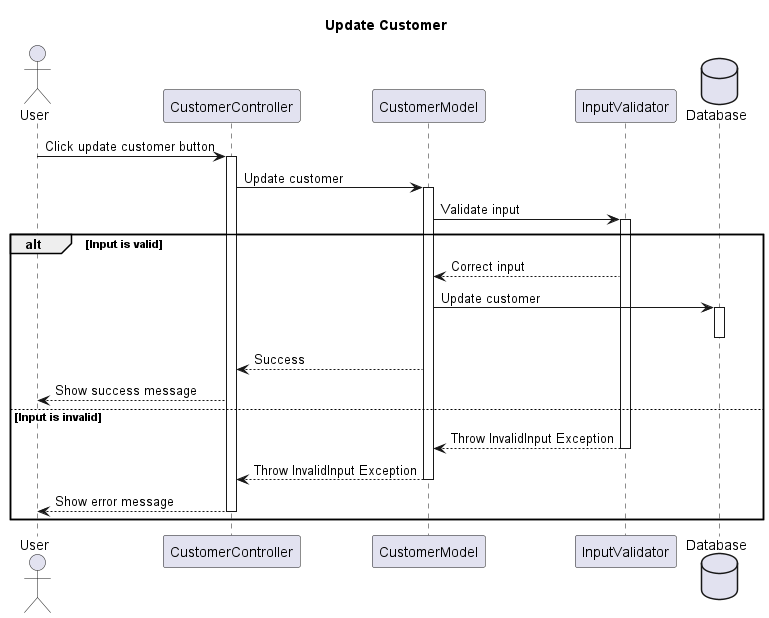
### Account View



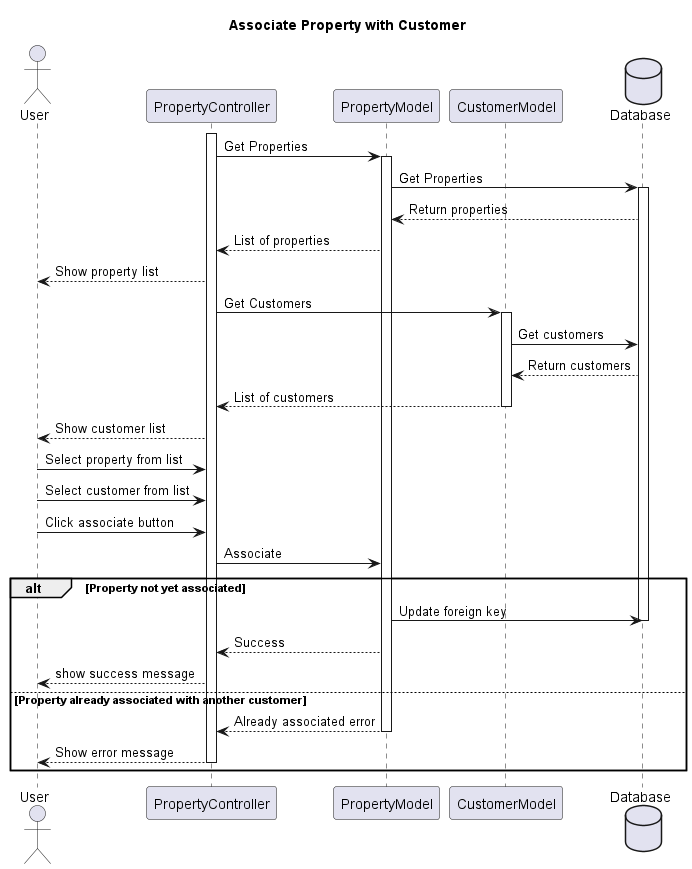


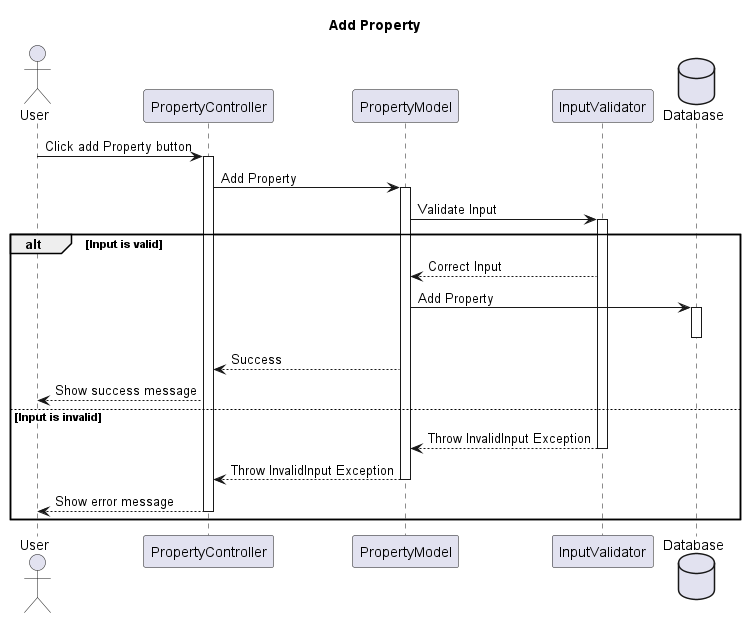
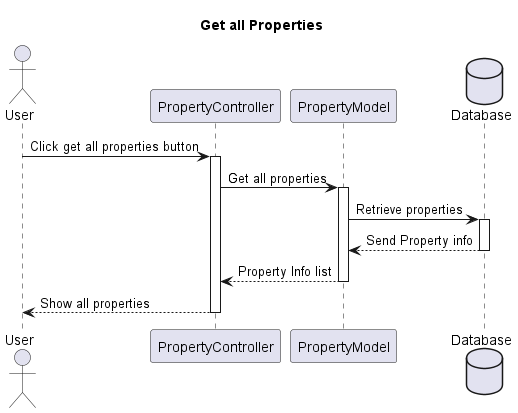
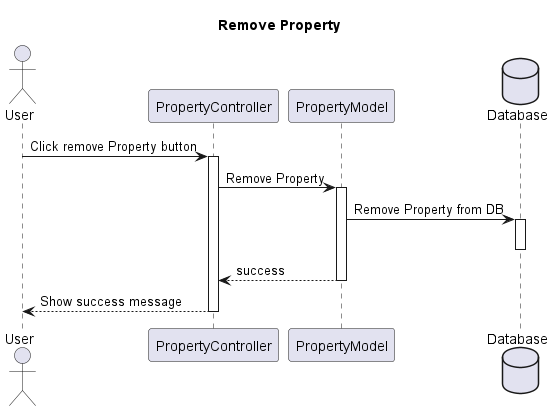
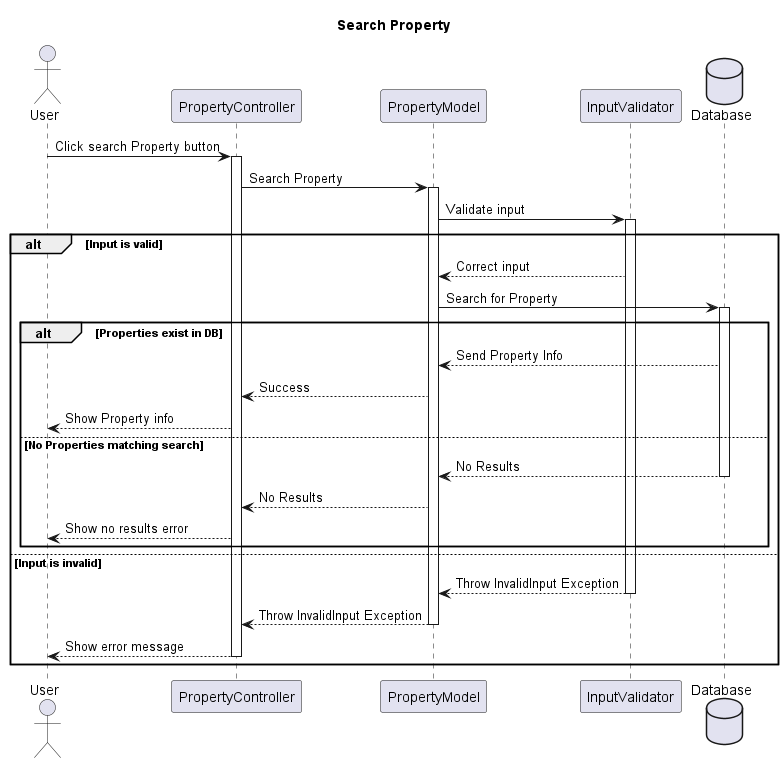
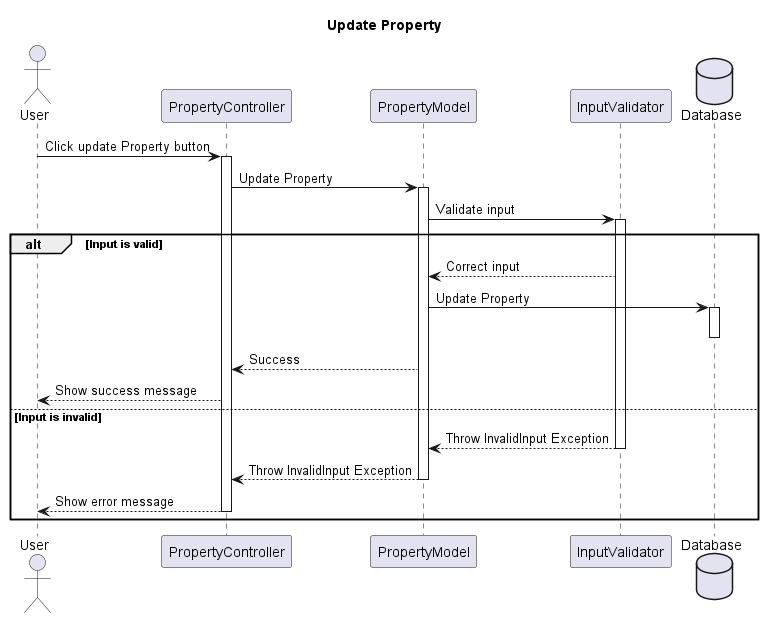


### Customer View

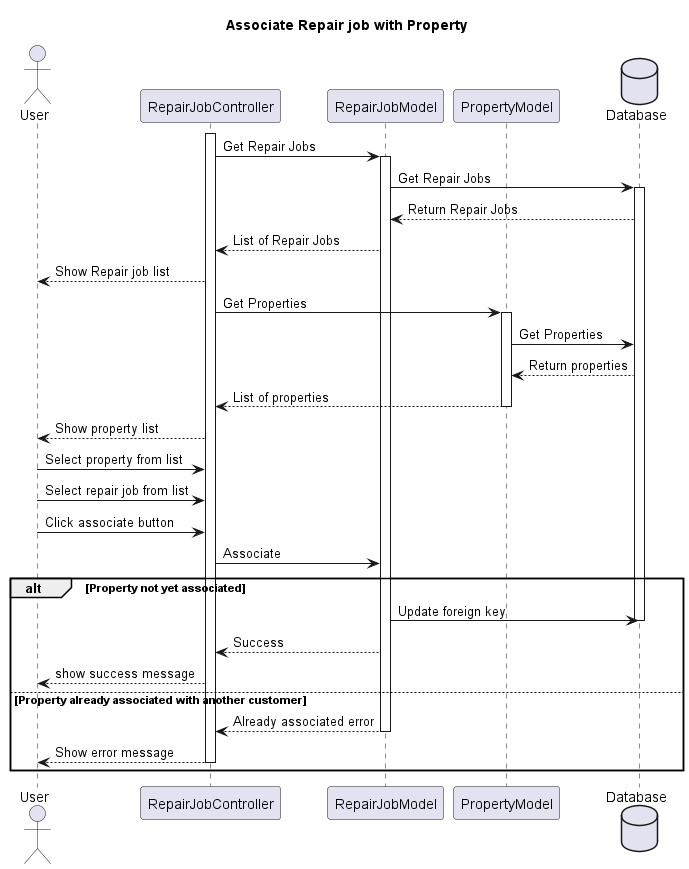
=    

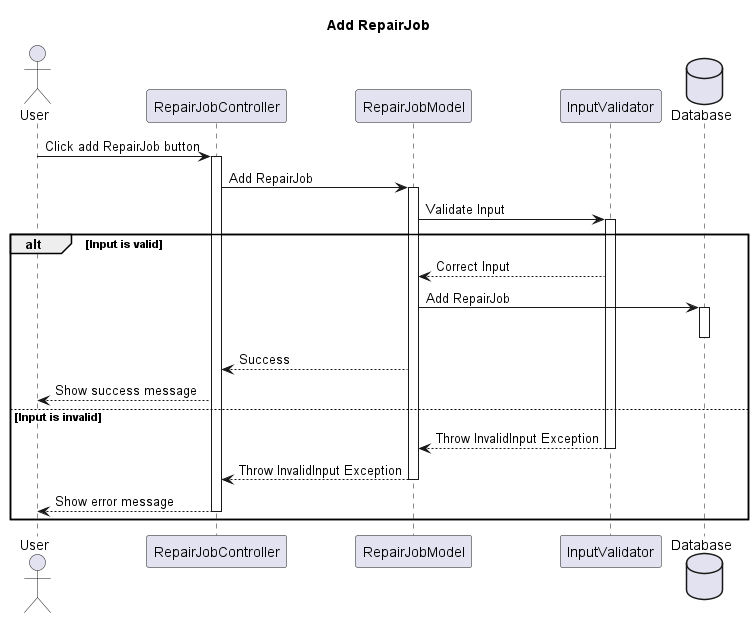
### Property View

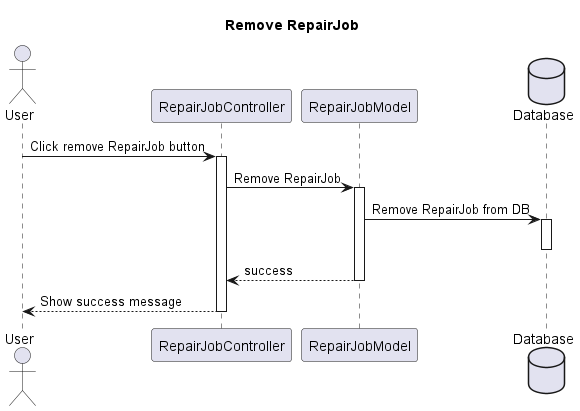
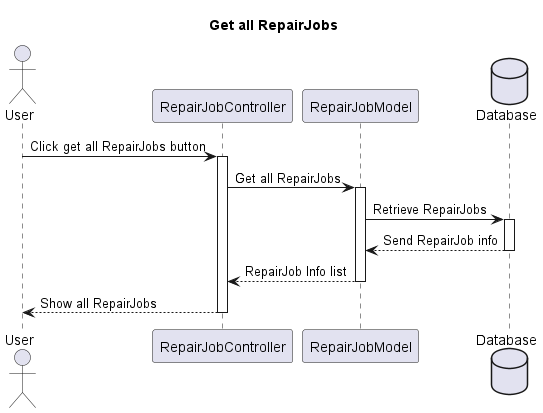
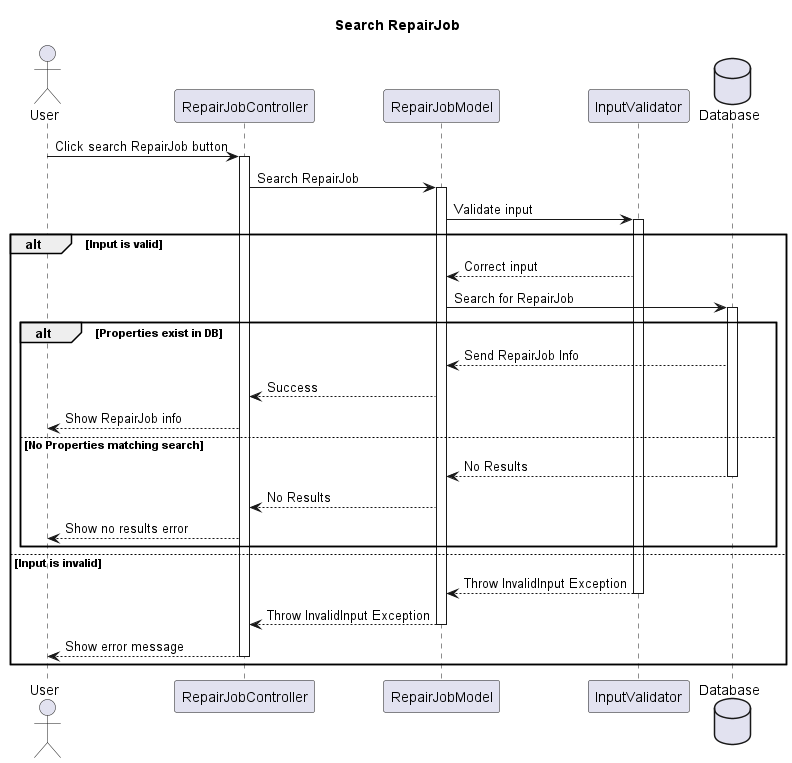
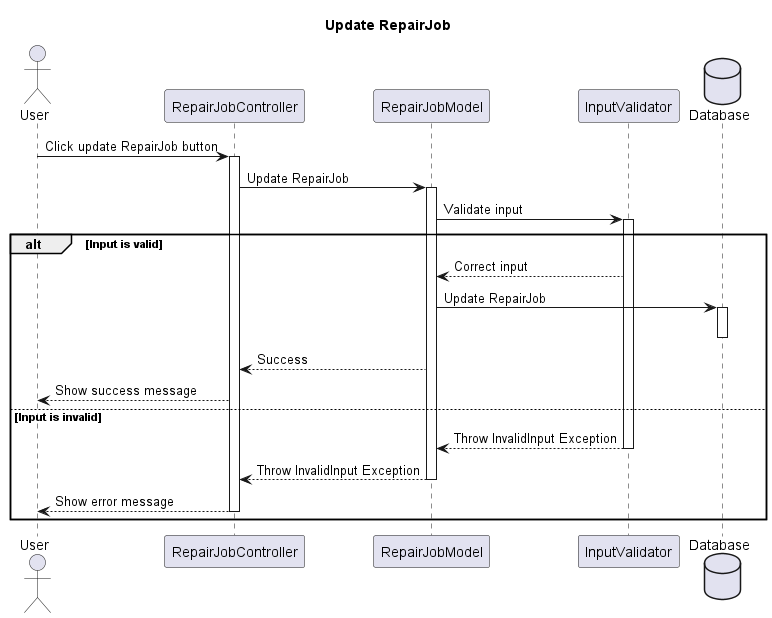


### Repair job View





### Report View

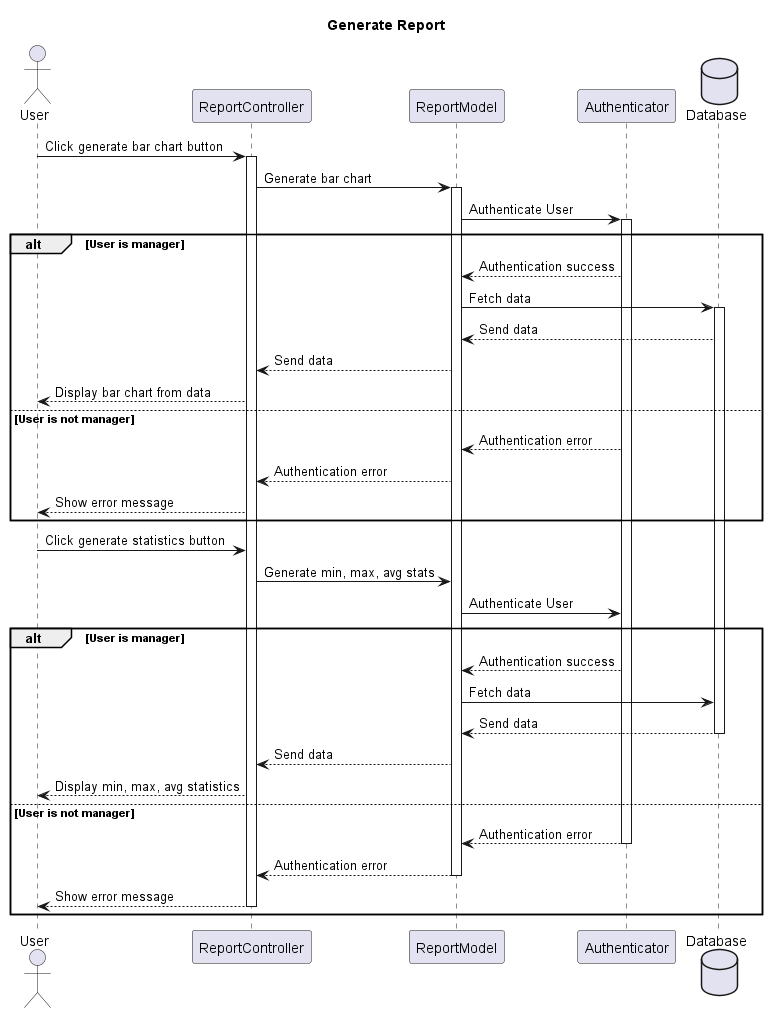
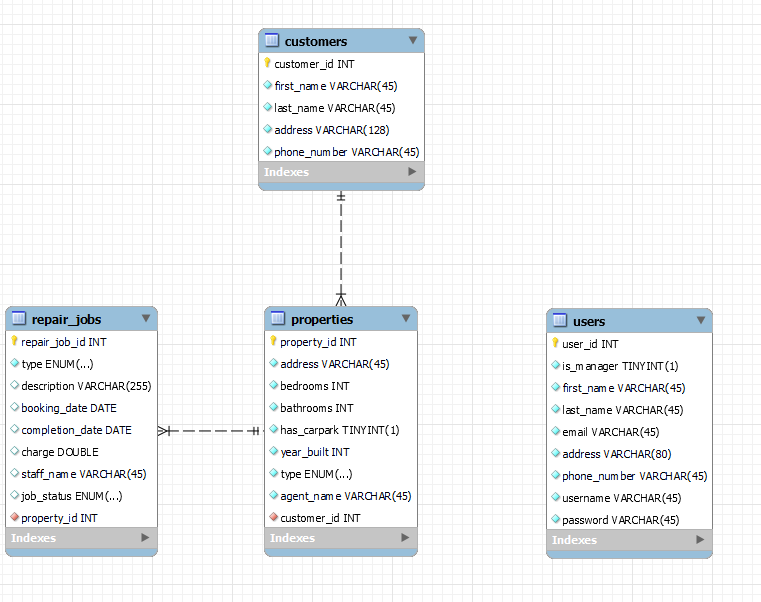


Figure 1. Account view sequence diagram

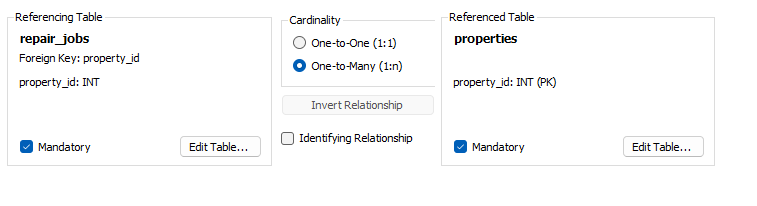
# DATA DESIGN



Notes: ENUM values were too long to display in the diagram. The values are as follows

* Type enum for repair jobs: electrical, plumbing, structural, cleaning, gardening, pest\_control
* Type enum for properties: house, apartment.

The diagram was created using mysql workbench, which did not appear to support zero to many relationships as shown in the figure below:



The relationship between property and repair job can be more accurately represented as:

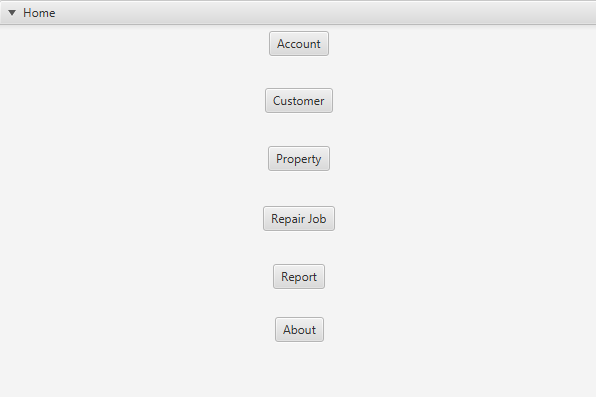
* Property to repair job: Zero or more.
* Repair job to Property: 1 and only one.

The same can be said for the relationship between customer and property.

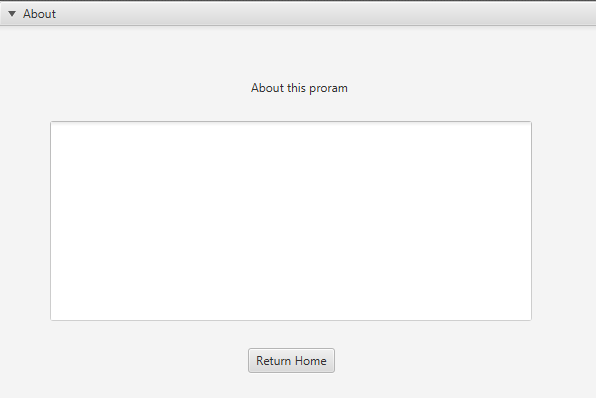
Additionally, a separate table depicting table design is not included in this document as it would duplicate what is shown in the ER diagram.

# INTERFACE DESIGN

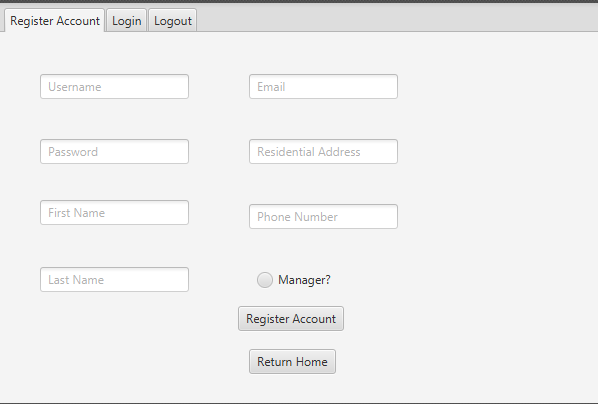
## Homepage View

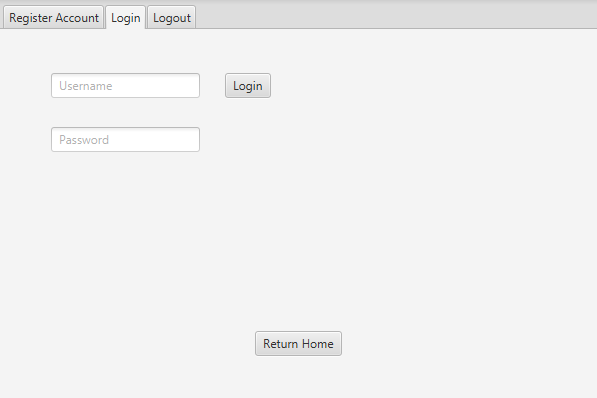


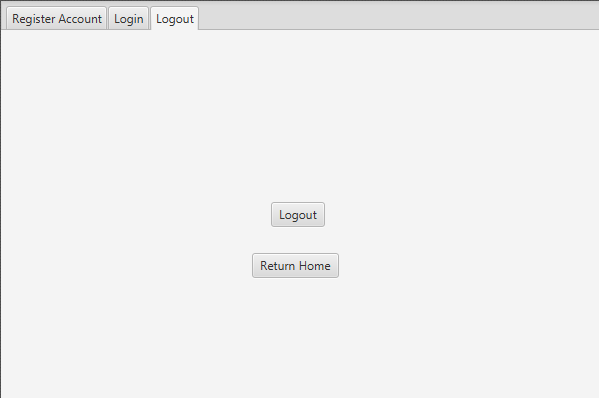
## About View



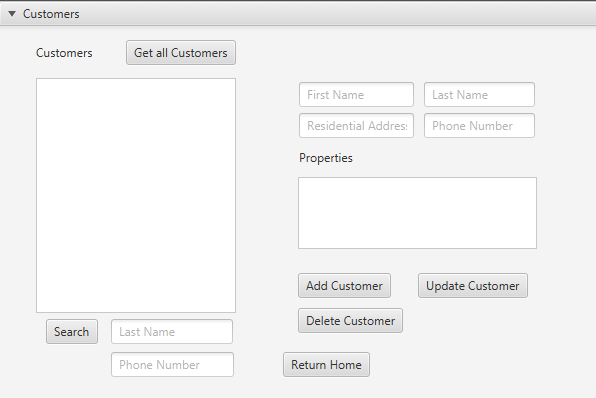
## Account View



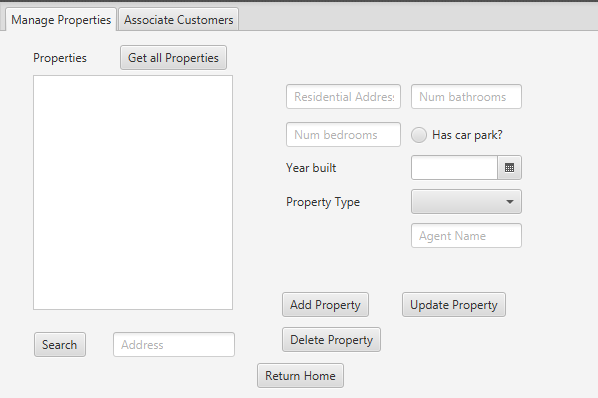


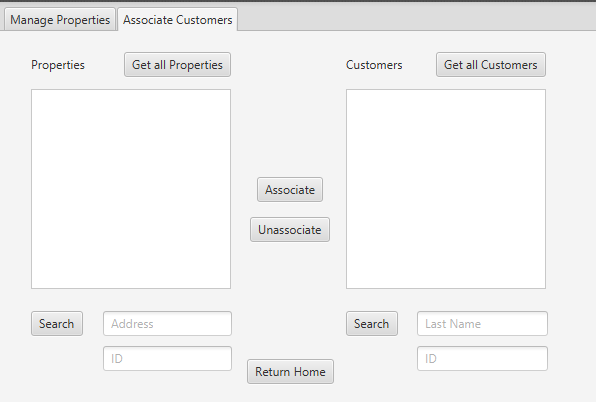


## Customer View

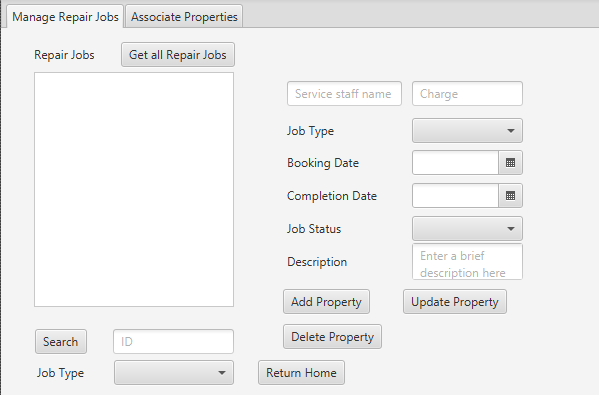


## Property View





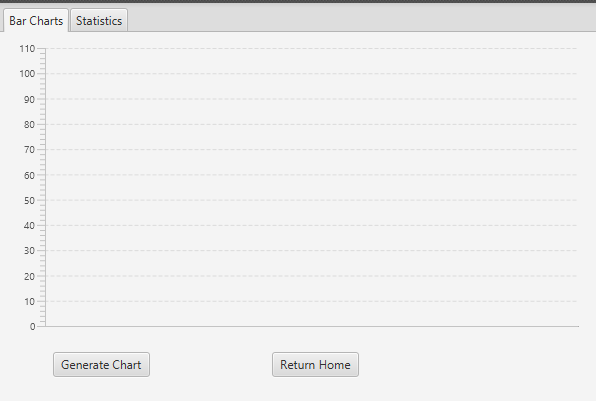
## Repair job View

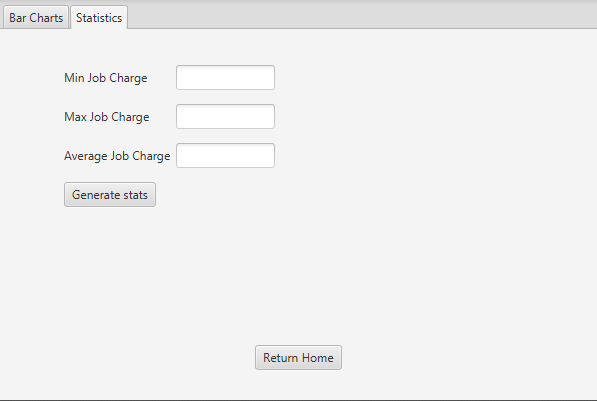


A screenshot of a computer

Description automatically generated

## Report View





**5. Design Pattern**

## MVC

The Model View Controller (MVC) design pattern is used for this program design. The reason for this is because the program will be developed using JavaFX, which already supports MVC quite well, leading to an easier development process.

## Singleton

A singleton design pattern is chosen for the InputValidator and Authenticator classes. The reason for this is to save on resources and to ensure that input validation and authentication is standardised for all parts of the program that will use it. By using a singleton, there will always only be one instance of the object for input validator and authenticator, saving memory. Additionally, having only one instance will ensure that functionality will always remain the same, no matter how many other classes use the instance.

A singleton may have also been suitable for creating a database connection class to ensure that all other classes use the same connection settings. This was not included in the design but may be considered during implementation.

**6. Data validation**

|  |  |
| --- | --- |
| **Field to validate** | **Validation requirements** |
| Name (first or last name) | * Alphabetical letters only * No spaces * Not longer than 45 characters |
| Email | * Follows the email format. Example: [abc123@mail.com](mailto:abc123@mail.com) * Not longer than 45 characters |
| Address | * Not longer than 128 characters |
| Phone | * Numbers only * No spaces * Not longer than 10 characters |
| Username | * No spaces * Not longer than 45 characters |
| Password | * No spaces * Not longer than 255 characters |
| Number of rooms (num bedrooms, bathrooms) | * Numbers only |
| Year | * Follows Date format YYYY-MM-DD |
| Cost | * Numbers only |

# 7. REQUIREMENTS MATRIX

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Button** | **Inputs** |
| Register account | Register Button | First name, last name, email, address, phone, username, password. |
| Login | Login button | Username, password |
| Logout | Logout button | None |
| Add new customer | Add customer button | First name, last name, address, phone number |
| Update customer | Update customer button | First name, last name, address, phone number |
| Delete customer | Delete customer button | Customer selection from list using mouse |
| Search for customer | Search customer button | Last name, phone number |
| View all customers | Get all customer button | None |
| Add new Property | Add property button | Address, num bedrooms, num bathrooms, car park (true/false), year built, property type, agent name. |
| Delete property | Delete property button | Property selection from list using mouse |
| Update property | Update property button | Address, num bedrooms, num bathrooms, car park (true/false), year built, property type, agent name. |
| Search property | Search property button | address |
| View all properties | Get all properties button | None |
| Associate property with customer | Associate property button | Property and Customer selection from lists using mouse. |
| Add repair job | Add repair job button | Description, booking date, completion date, charge, service staff name, job type, job status |
| Remove repair job | Remove repair job button | Repair job selection from list using mouse |
| Update repair job | Update repair job button | Description, booking date, completion date, charge, service staff name, job type, job status |
| Search repair job | search repair job button | Job id or job type |
| View all repair jobs | Get all repair jobs button | None |
| Associate repair job with property | Associate repair job button | Repair job and property selection from list using mouse |
| Show job distribution bar chart | Generate bar chart button | None |
| Show min, max, average stats for job charge | Generate statistics button | None |

**8.Test Plan**

**Unit testing**

Unit testing with Junit will be conducted to ensure that all functionality is correct. All model classes that interact with the database, as well as utility classes such as input validators and authenticators, will be tested. For example, model classes will be tested to ensure they can add, update, delete, etc. Additionally, whenever a new feature is complete, all tests will be run again to make sure no new bugs were introduced.

**Acceptance testing**

Once functionality has been developed, developers will begin acceptance testing by interacting with the user interface and emulating the user stories previously described. If all acceptance criteria for that story pass, then the feature is considered complete. If not, then revision must be done.