# SUMMARY

Topic title: Real estate business

Student name: Nguyen Duc Van

Student ID: 102210195. Class: 21TCLC\_DT2

|  |  |
| --- | --- |
| DA NANG UNIVERSITY  **UNIVERSITY OF SICIENCE AND TECHNOLOGY**  FALCUTY………………………………………… | **THE SOCIALIST REPUBLIC OF VIETNAM**  Independence - Freedom - Happiness |

# GRADUATION PROJECT REQUIREMENTS

Student Name: …..…………….………….…….. Student ID :………………...

Class:…………… Faculty:..................................... Major:…………...........................

1. *Topic title:*

………………………………………………..…………………………………………

…………………………………………………………………………………………..

1. *Project topic :* ☐*has signed intellectual property agreement for final result*
2. *Initial figure and data:*

……………………………………..……………………………………………..……......……………………………………………………………………………………………..………………………………….…..………………………..………………………*Content of the explanations and calculations:*

…...………………………………………………………………………………………

…...………………………………………………………………………………………

…...………………………………………………………………………………………

…...………………………………………………………………………………………

…...………………………………………………………………………………………

1. *Drawings, charts (specify the types and sizes of drawings):*

…...………………………………………………………………………………………

…...………………………………………………………………………………………

…...………………………………………………………………………………………

…...………………………………………………………………………………………

|  |  |
| --- | --- |
| 1. *Name of instructor:* | *Content parts:* |
|  |  |
|  |  |
|  |  |

1. *Date of assignment: ……../……./201…..*
2. *Date of completion: ……../……./201…..*

|  |  |
| --- | --- |
|  | *Đà Nẵng, date month year 201* |
| **Head of Division**…………………. | **Instructor** |

# PREFACE

# ASSURRANCE

I hereby declare that:

1. The graduation project report, Topic name: Real Estate Business System, is my own research work under the direct guidance of the lecturer MSc. Do Thi Tuyet Hoa
2. I have read, researched, translated documents, and synthesized the knowledge that has made this report, and ensured that it has not been copied anywhere.
3. The theories in the thesis are all used from the documents as I have referred to in the reference section in the report.

If there is any violation, I will take full responsibility.

Student Performed

Nguyen Duc Van

# TABLE OF CONTENT

[SUMMARY 3](#_Toc198135736)

[GRADUATION PROJECT REQUIREMENTS 4](#_Toc198135737)

[PREFACE i](#_Toc198135738)

[ASSURRANCE ii](#_Toc198135739)

[TABLE OF CONTENT iii](#_Toc198135740)

[LIST OF PICTURES v](#_Toc198135741)

[LIST OF TABLES vi](#_Toc198135742)

[LIST OF SYMBOLS, ACRONYMS vii](#_Toc198135743)

[INTRODUCTION 1](#_Toc198135744)

[1. Purpose of implementation 1](#_Toc198135745)

[2. Objective of the topic 1](#_Toc198135746)

[2.1. System Target 1](#_Toc198135747)

[2.2. Features 1](#_Toc198135748)

[3. Scope 2](#_Toc198135749)

[4. Target customers 2](#_Toc198135750)

[5. Development technology 2](#_Toc198135751)

[CHAPTER 1: THEORETICAL FOUNDATION 3](#_Toc198135752)

[1.1. Overview of Python and Jupyter Notebook 3](#_Toc198135753)

[1.1.1. Python 3](#_Toc198135754)

[1.1.2. Jupyter Notebook 3](#_Toc198135755)

[1.2. Overview of Java and MySQL 4](#_Toc198135756)

[1.2.1. Java 4](#_Toc198135757)

[1.2.2. Spring Boot 5](#_Toc198135758)

[1.2.3. MVC model 6](#_Toc198135759)

[1.2.4. MySQL 8](#_Toc198135760)

[1.3. Overview of ReactJS and ChakraUI 9](#_Toc198135761)

[1.3.1. Reactjs 9](#_Toc198135762)

[1.3.2. ChakraUI 9](#_Toc198135763)

[CHAPTER 2: SYSTEM DESIGN AND ANALYSIS 10](#_Toc198135764)

[3.1. Business analysis 10](#_Toc198135769)

[3.1.1. Buyer's business 10](#_Toc198135770)

[3.1.2. Broker's business 11](#_Toc198135771)

[3.1.3. Admin's business 11](#_Toc198135772)

[3.2. System design 11](#_Toc198135773)

[3.2.1. Use-case diagram 11](#_Toc198135774)

[3.2.2. Activity diagram 15](#_Toc198135775)

[3.2.3. Sequence diagram 16](#_Toc198135776)

[3.3. Database design 21](#_Toc198135777)

[3.3.1. ER (entity-relationship) 21](#_Toc198135778)

[3.3.2. Converting an implementation model into a system model 21](#_Toc198135779)

[3.3.3. Database schema 27](#_Toc198135780)

[3.4. Real estate prediction system development 29](#_Toc198135781)

[3.4.1. Real estate valuation model 33](#_Toc198135782)

[3.4.2. Real estate price forecast model 34](#_Toc198135783)

[3.5. Apply NSFW image and toxic content classification models 34](#_Toc198135784)

[3.5.1. NSFW image classification models 34](#_Toc198135785)

[3.6. Chapter Conclusion 35](#_Toc198135786)

[CHAPTER 3: IMPLEMENTATION AND RESULTS EVALUATION 35](#_Toc198135787)

[CHAPTER 4: CONCLUSION AND DEVELOPMENT DIRECTION 36](#_Toc198135788)

[REFERENCES 37](#_Toc198135789)

[APPENDIX 1 1](#_Toc198135790)

[APPENDIX 2 2](#_Toc198135791)

# LIST OF PICTURES

[Figure 2.1 General use case diagram 5](#_Toc197611720)

[Figure 2.2 User management use-case diagram 5](#_Toc197611721)

[Figure 2.3 Post moderation use case diagram 6](#_Toc197611722)

[Figure 2.4 Profile management use case diagram 6](#_Toc197611723)

[Figure 2.5 Account upgrade management use case diagram 7](#_Toc197611724)

[Figure 2.6 Post management use case diagram 7](#_Toc197611725)

[Figure 2.7 Favourite post management use case diagram 8](#_Toc197611726)

[Figure 2.8 Client Operation Principle 9](#_Toc197611727)

[Figure 2.9 Admin Operation Principle 9](#_Toc197611728)

[Figure 2.10 Registration/Login Flow 10](#_Toc197611729)

[Figure 2.11 Real estate article management flow as a user 12](#_Toc197611730)

[Figure 2.12 Account upgrade flow 13](#_Toc197611731)

[Figure 2.13 The ER entity relationship model 14](#_Toc197611732)

[Figure 2.14 Relational Model 20](file:///D:\VisualStudioCode\Project\RealEstate\GraduationProjectReport\MyReport\PL04_B_RuotThuyetMinh.docx#_Toc197611733)

[Figure 2.15 Real estate valuation model features 21](#_Toc197611734)

[Figure 2.16 Real estate price forecast features 22](#_Toc197611735)

[Figure 2.17 Q-Q Plots After Optimal Normalization per Feature 23](#_Toc197611736)

[Figure 2.18 Comparison of Q-Q Plots of Price per m² across different transformation methods 23](#_Toc197611737)

[Figure 2.19 Scatter plot of features vs. Price 24](#_Toc197611738)

[Figure 2.20 Selected features in order from best to worst 24](#_Toc197611739)

[Figure 2.21 Optimal number of features 24](#_Toc197611740)

[Figure 2.22 Recursive Feature Elimination with correlated features 25](#_Toc197611741)

# LIST OF TABLES

[Table 2.1 Describe role table attributes 14](#_Toc197611443)

[Table 2.2 Describe user table attributes 15](#_Toc197611444)

[Table 2.3 Describe token table attributes 15](#_Toc197611445)

[Table 2.4 Describe transaction histories table attributes 16](#_Toc197611446)

[Table 2.5 Describe favourite properties table attributes 16](#_Toc197611447)

[Table 2.6 Describe category table attributes 16](#_Toc197611448)

[Table 2.7 Describe property table attributes 17](#_Toc197611449)

[Table 2.8 Describe house table attributes 17](#_Toc197611450)

[Table 2.9 Describe furnished status table attributes 17](#_Toc197611451)

[Table 2.10 Describe type of house table attributes 18](#_Toc197611452)

[Table 2.11 Describe house characteristics table attributes 18](#_Toc197611453)

[Table 2.12 Describe house characteristic mapping table attributes 18](#_Toc197611454)

[Table 2.13 Describe land table attributes 18](#_Toc197611455)

[Table 2.14 Describe type of land table attributes 18](#_Toc197611456)

[Table 2.15 Describe land characteristics table attributes 19](#_Toc197611457)

[Table 2.16 Describe land characteristic mapping table attributes 19](#_Toc197611458)

# LIST OF SYMBOLS, ACRONYMS

|  |  |
| --- | --- |
| Symbol, Acronym | Detailed explanation |
| PMML | Predictive Model Markup Language |
| HTTPS | HyperText Transfer Protocol Secure |
| PKL | Extension for files created with Pickle |
| API | Application Programming Interface |
| ONNX | Open format for storing machine learning models |
| NSFW | Not safe for work |

# INTRODUCTION

## Purpose of implementation

* Searching and buying real estate today is difficult, especially in determining the real value of the property, comparing suitable options and accessing transparent information. Buyers often spend a lot of time searching for properties that meet their needs in terms of location, area, amenities and finance, while sellers have difficulty reaching potential customers.
* Therefore, an online real estate business system can help display information visually, support filtering and comparing options, and integrate price prediction technology to help buyers and sellers make more accurate decisions

## Objective of the topic

### System Target

* Help users quickly post and search for suitable real estate based on filters with many characteristics.
* The real estate valuation feature provides a reference price close to the market, helping users minimize the risk of incorrect valuation, supporting the brokerage team or real estate sellers and buyers to save time in valuation and negotiation.
* The system will also support predicting real estate fluctuations to support customers' buying or selling decisions.
* Integrating VNPay, safe and convenient online payment when customers want to upgrade their account to use more features of the application.

### Features

* For guest
  + Search and filter by needs: price, area, location, property type, ...
  + View property details with images, prices and legal information, ...
  + Displays detailed information of the property along with the price determined by the property valuation system.
* For member
  + Post real estate with full information: images, descriptions, prices, legal status, ...however, the number of posts will be limited to 3.
  + You can save posts that interest you as well as upgrade your account to be able to use more features of the application.
* For brokers
  + Customers with upgraded accounts will be supported by the system to forcast real estate price in the future.
  + The number of posts posted by the brokerage account will also be up to 30.
* For admins:
  + Update account status.
  + View article details and moderate it.

## Scope

* Property type: house, land.
* Geographical scope: Da Nang
* Language: Vietnamese

## Target customers

* Real Estate Buyers/Sellers
* Real Estate Companies or Brokers

## Development technology

* Project management: Github
* Language: Java, Python, Reactjs
* Framework: Spring boot
* Database: Mysql
* Encryption & security: Security, HTTPS Token

# THEORETICAL FOUNDATION

## Overview of Python and Jupyter Notebook

### Python

Python is a widely used high-level programming language applied in web development, software engineering, data science, and machine learning (ML). Developers favor it due to its simplicity, efficiency, and cross-platform compatibility.

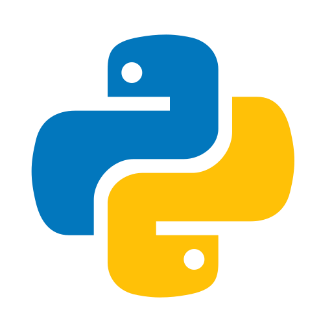


Figure . Python

**Benefits of Python include:**

* Developers can easily read and understand Python programs because the language has a simple syntax that resembles English.
* Python improves developer productivity as fewer lines of code are typically needed to write a program compared to other programming languages.
* Python has a large standard library that contains reusable code for almost any task, so developers don't need to write everything from scratch.
* Developers can easily integrate Python with other popular programming languages such as Java, C, and C++.
* Python is cross-platform and can be used on various operating systems, including Windows, macOS, Linux, and Unix.

### Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing code, equations, visualizations, and narrative text. It is widely used in data science, machine learning, academic research, and educational settings.



Figure . Jupyter

Benefits of Jupyter Notebook:

* Jupyter allows users to run code in small chunks, making it ideal for testing and debugging.
* The integration with libraries like Matplotlib, Seaborn, and Plotly makes visualizing data easy within the notebook itself.
* Users can mix narrative text (using Markdown) with code and outputs, creating comprehensive reports or tutorials.
* Notebooks make it easy to share code, data, and results, ensuring that research is reproducible.

## Overview of Java and MySQL

### Java

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is intended to let application developers Write Once and Run Anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java was developed by James Gosling at Sun Microsystems Inc. in May 1995 and later acquired by Oracle Corporation, and is widely used for developing applications for desktop, web, and mobile devices.



Figure . Java

Java is known for its simplicity, robustness, and security features, making it a popular choice for enterprise-level applications. Java applications are compiled to bytecode that can run on any Java Virtual Machine. The syntax of Java is similar to C/C++.

Since Java runs on the JVM, it may not be as fast as languages that are compiled directly, such as C/C++. This can be a concern for applications that require high performance.

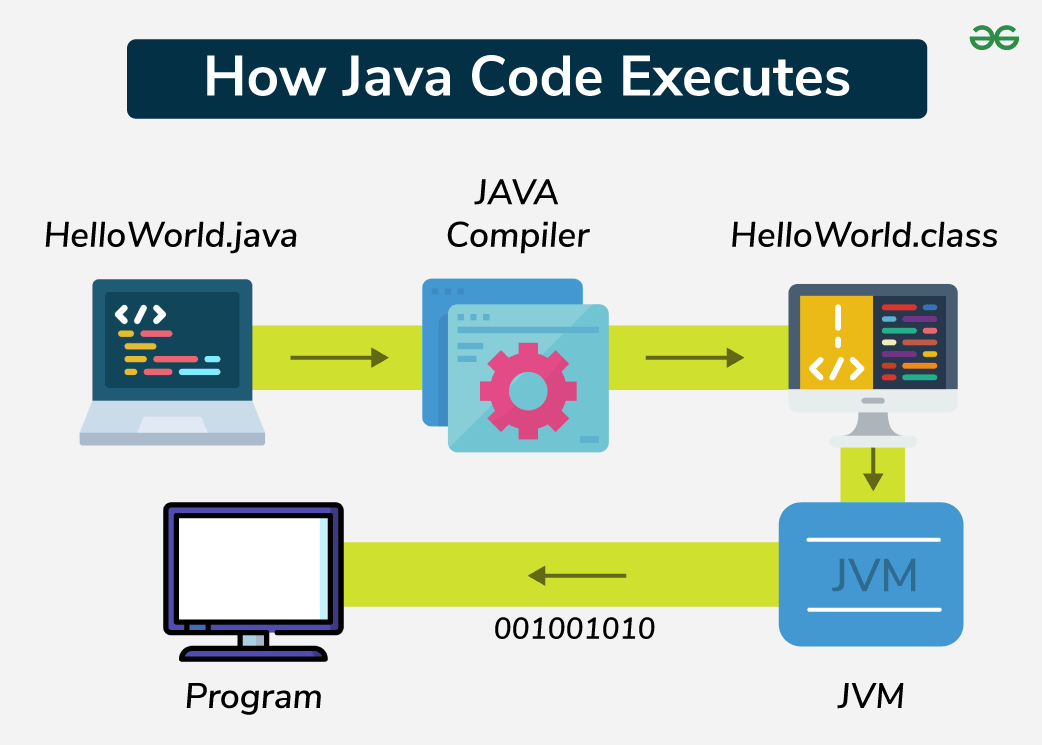


Figure . The compilation and interpretation process of Java

### Spring Boot

Spring Boot is a combination of the Spring Framework and an embedded server, allowing you to build web applications, microservices, and standalone JAR applications. It supports features like Microservices, REST API, and JPA databases, making app development simpler and more efficient.



Figure . Spring Boot

Advantages of Spring Boot:

* It creates standalone Spring applications.
* It makes testing web apps easier with embedded HTTP servers like Tomcat, Jetty, and others. No need to distribute WAR files.
* No XML configuration required.

### MVC model

* **MVC** stands for **Model-View-Controller**.
* It is a design pattern used in software engineering.
* MVC is a software architecture pattern used to create user interfaces on computers.
* MVC is divided into three connected parts, and each part has its own responsibility and works independently from the others.
  + **Model**
    - Responsible for working with the **database**.
    - It contains all the functions and methods that directly query or update the data.
    - The **Controller** uses these functions to get data and sends it to the **View**.
  + **View**
    - The **user interface**.
    - Contains all the elements that the user interacts with, such as menus, buttons, images, text, etc.
    - Receives data from the **Controller** and displays it.
  + **Controller**
    - Acts as the **middle layer** between Model and View.
    - Handles user requests, uses the **Model** to get data, and then passes that data to the **View** to show to the user.

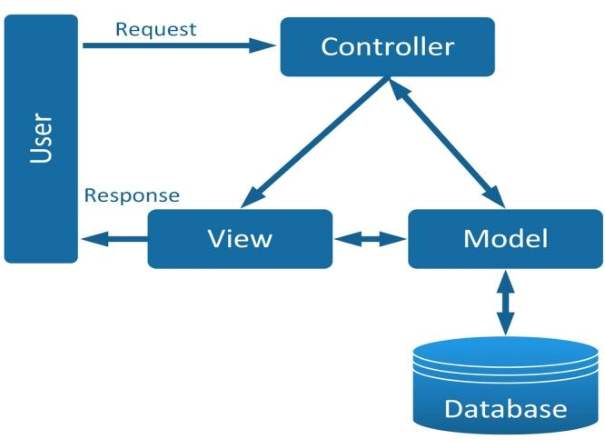


Figure . Flow of operation in MVC model

* MVC workflow
  + First, a request from the user is sent from the client to the server.
  + Then, the Controller, based on the user’s request, communicates with the Model to get the required data from the database.
  + Finally, the Controller sends the retrieved data to the View, which displays it to the user in the browser.
* Three-tier model: this is a common source code structure used in Spring Boot applications. Specifically, the application is divided into three tiers (or layers) as follows:
  + Presentation layer: this layer interacts directly with the user. It can include Views and Controllers (in traditional MVC) or APIs (in RESTful applications).
  + Business logic layer: this layer contains the core logic of the application. Most of the application’s code resides here. It is responsible for processing data and enforcing business rules.
  + Data access layer: this layer interacts directly with the database, executing queries and returning results to the business logic layer.

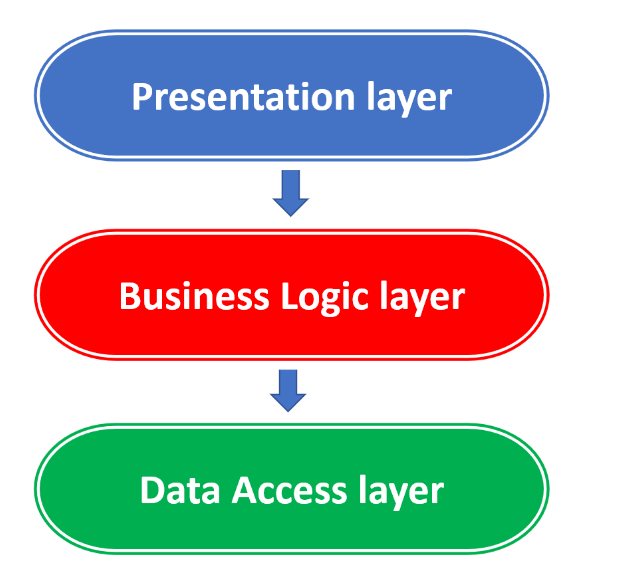


Figure . Three-tier architecture

* By combining the two models, we get a complete Spring Boot application consisting of the following components:
  + Controller: the controller returns a View (which contains the data ready for display in HTML format) or a Model represented through an API for the View (the View is separately written using React, Vue, or Angular).
  + Service: The Service contains the logic and processing code. When the Controller requests it, the Service will handle the request and return the data (in the form of a Model) back to the Controller. The Controller will then pass this data to the View as described earlier.
  + Repository: The Repository is used by the Service to interact with the database. The Repository directly interacts with the database, reading and writing data, and then returning the results to the Service.

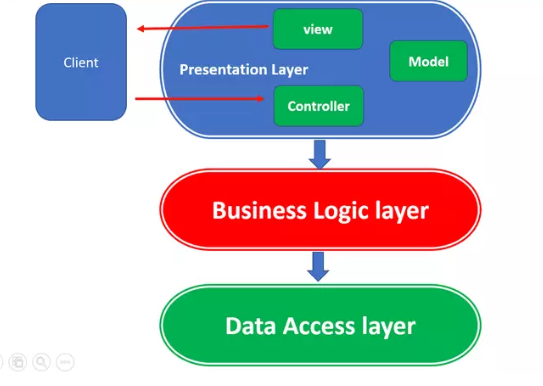


Figure . Three-Tier architecture vs MVC pattern

### MySQL

* MySQL is an open-source, relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data. It is one of the most popular database systems used in web applications, known for its speed, reliability, and ease of use.



Figure . MySQL

* Key Features of MySQL:
  + **Developed by Oracle Corporation.**
  + **Supports multiple platforms**: MySQL runs on operating systems such as Windows, Linux, and macOS.
  + **Widely used by developers** due to its scalability, data security features, and large community support.
* Advances of MySQL:
  + **Open-source**: MySQL is free and open-source, allowing modification and redistribution.
  + **High performance**: MySQL offers fast data retrieval and processing for large datasets.
  + **ACID compliance**: ensures data integrity and reliability, especially with the InnoDB storage engine.
  + **Scalability**: it supports large databases and high traffic with features like partitioning and clustering.
  + **Variety of storage engines**: MySQL offers multiple storage engine options (e.g., InnoDB, MyISAM) for flexible usage.
  + **Replication**: it supports master-slave replication for data redundancy and high availability.
  + **Security features**: provides user authentication, SSL encryption, and safe data storage options.

## Overview of ReactJS and ChakraUI

### Reactjs

### ChakraUI

# SYSTEM DESIGN AND ANALYSIS



## Business analysis

### Buyer's business

* **Visiting customers**
  + View property list: can view all properties posted on the system.
  + View property details: view description, images, price, area, number of rooms,… At the same time, the real estate price will be displayed as determined by the system.
  + Register account: if you want to save posts or make transactions, visitors must register an account.
  + Search and filter properties: search by keywords, filter by characteristics of property.
* **Members** are those who have registered an account and can use all the features:
  + All rights of visitors.
  + Update profile and account information: Change contact information and fullname.
  + Upgrade account: choose an account upgrade package to use more features and pay directly through the payment gateway.
  + View transaction history: track completed or pending transactions.
  + Buyer accounts can also post properties for sale, but are limited to 3
  + Manage personal post: edit, update, or delete posted real estate post.
  + Save to favorites: add properties to your favorites list for easy access later.

### Broker's business

* Brokerage account is a registered and upgraded account
* Having all rights of member
* This account will be used for additional features of the system: predict real estate trends.
* The article limits the posts that can be posted by this account to 30

### Admin's business

* Account status control: lock or unlock client accounts.
* Post moderation: review, approve or reject real estate posts.

## System design

### Use-case diagram

#### General use-case diagram

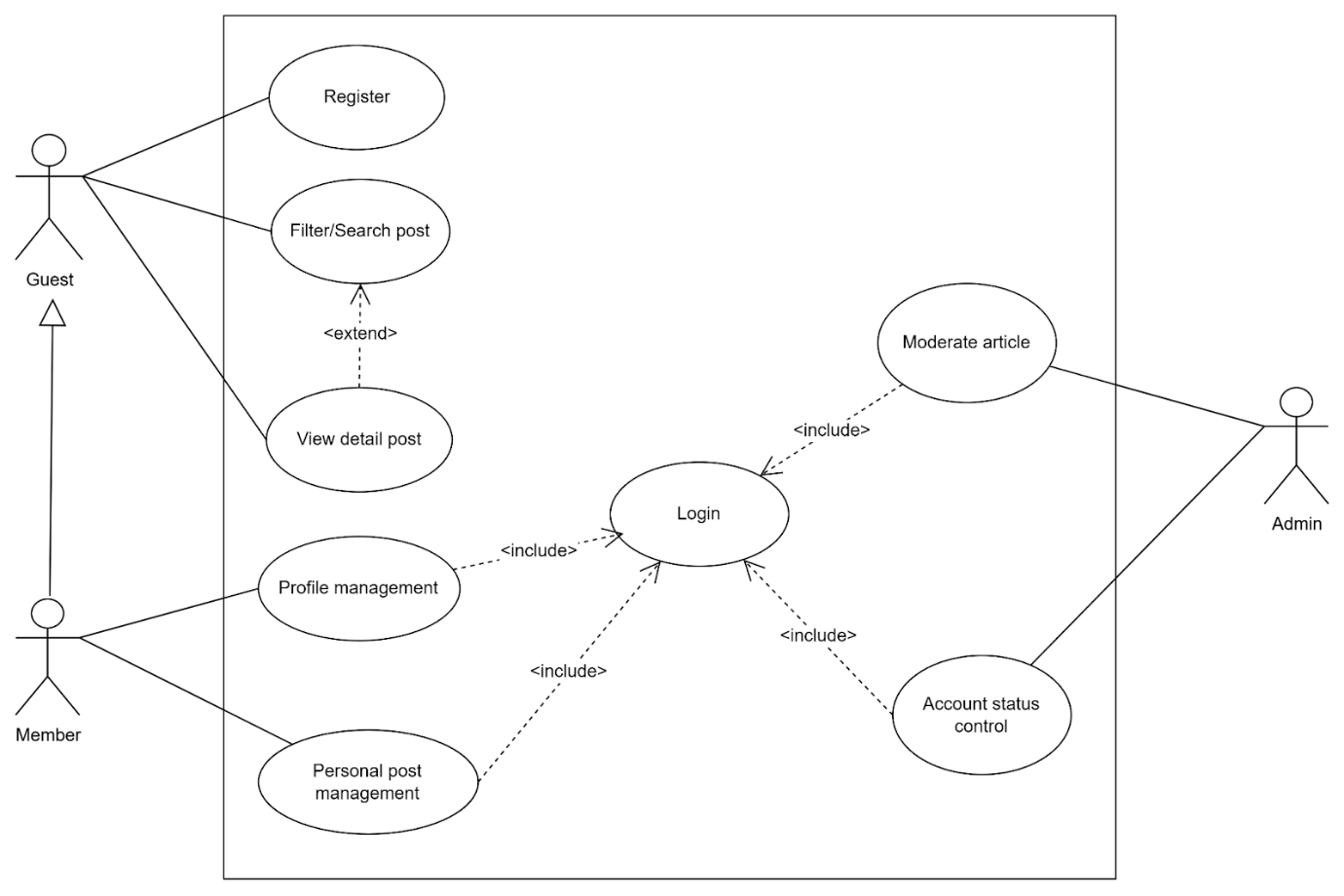


Figure . General use case diagram

#### Admin use-case diagram

* Account status control

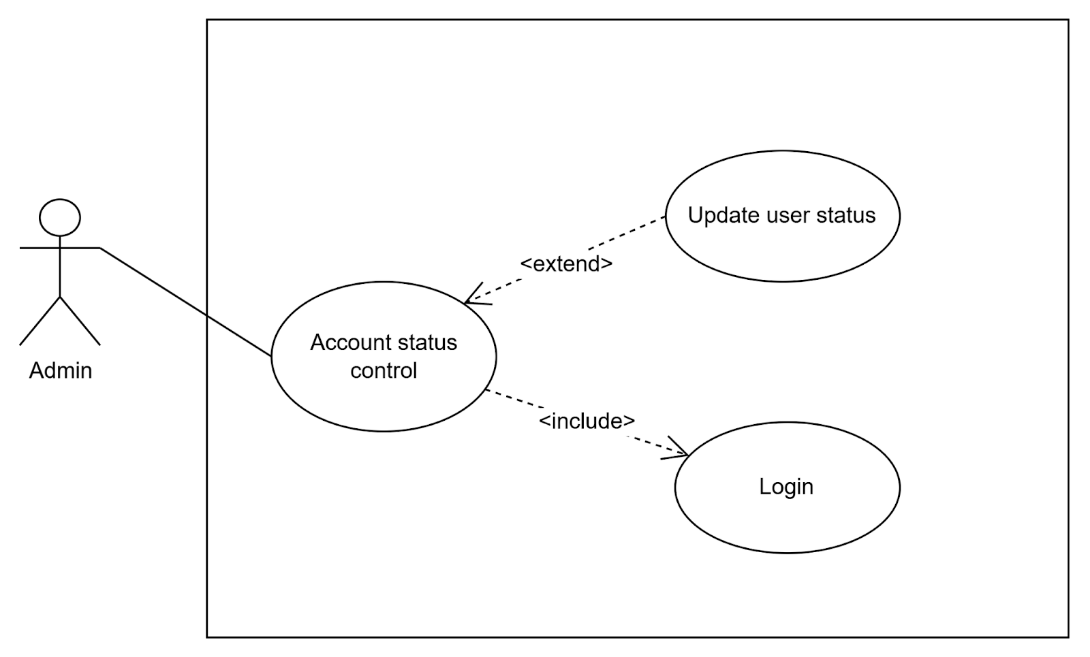


Figure . User management use-case diagram

* Post moderation

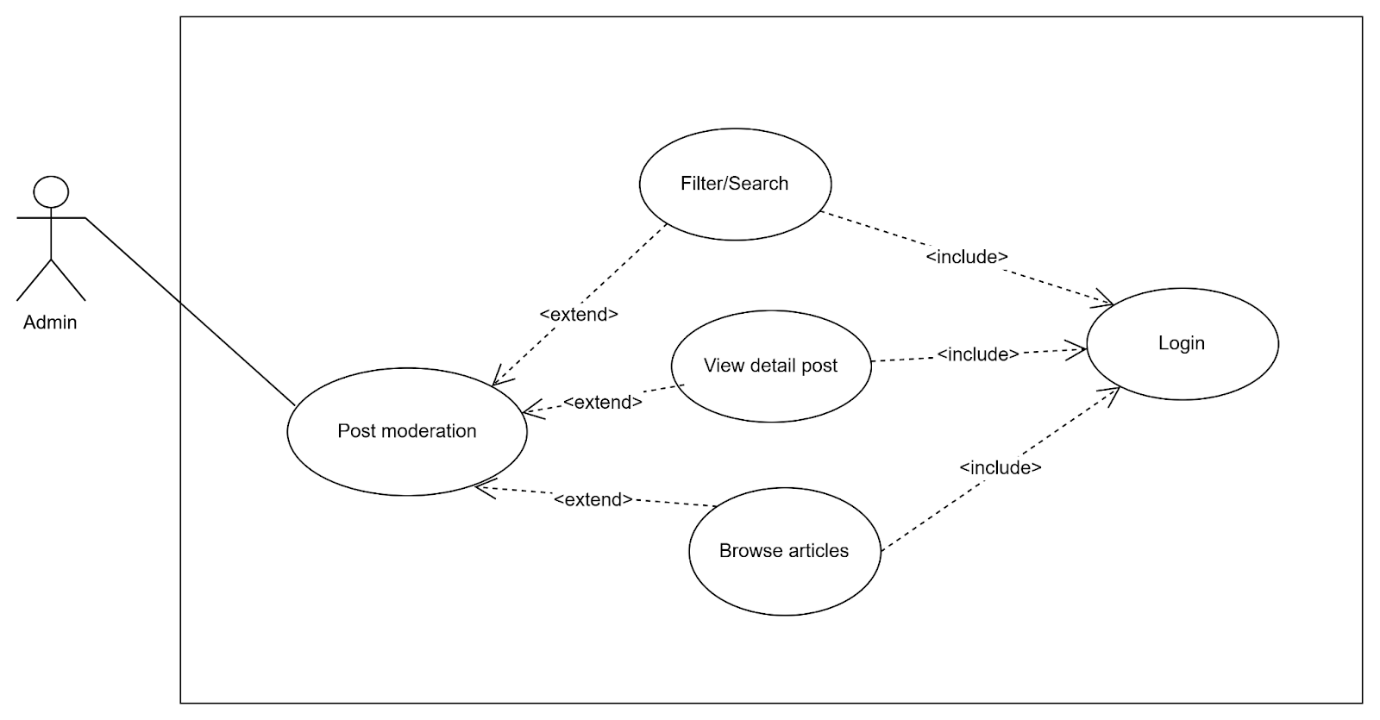


Figure . Post moderation use case diagram

#### User use-case diagram

* Profile management

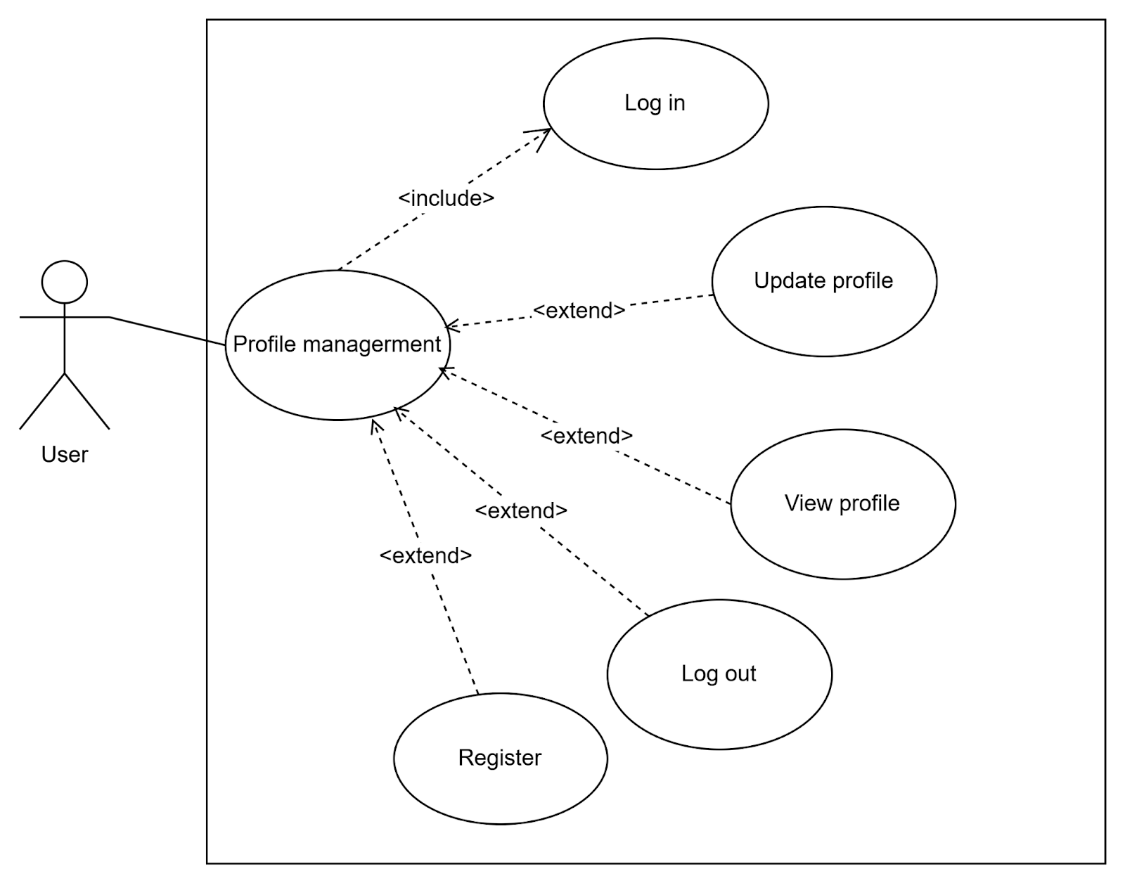


Figure . Profile management use case diagram

* Account upgrade management

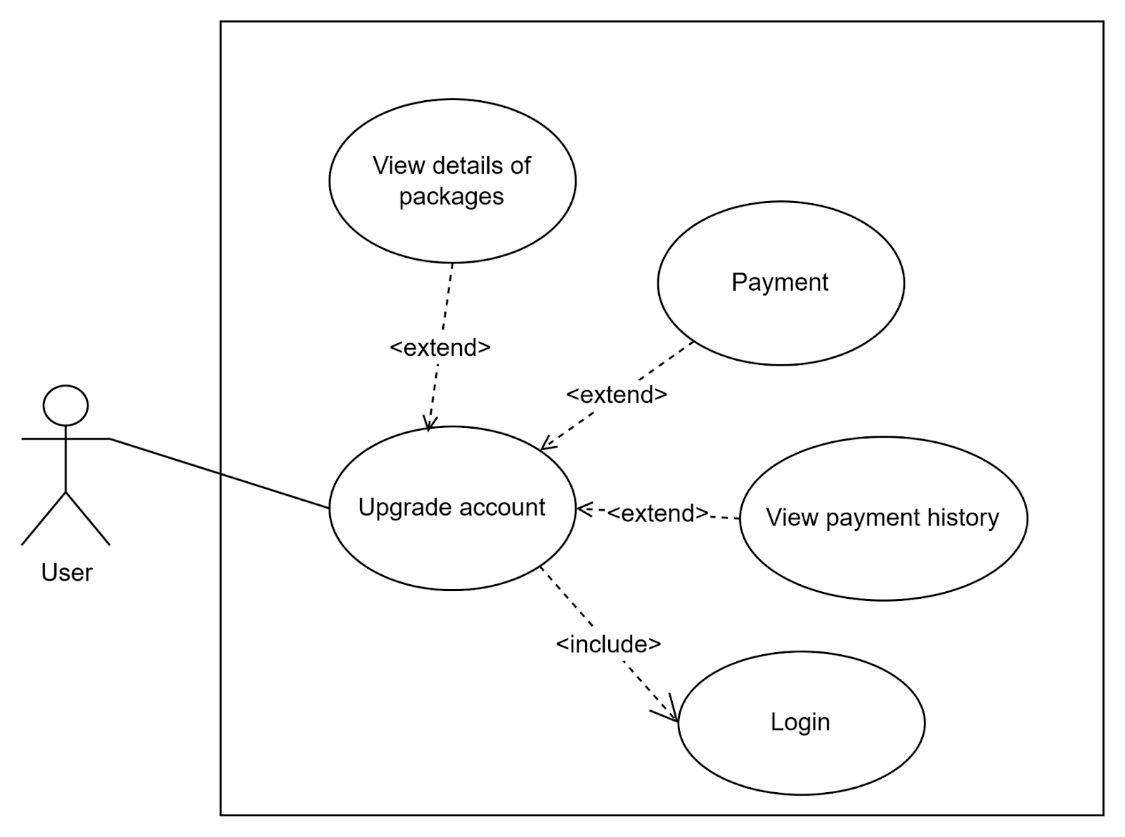


Figure . Account upgrade management use case diagram

* Post management

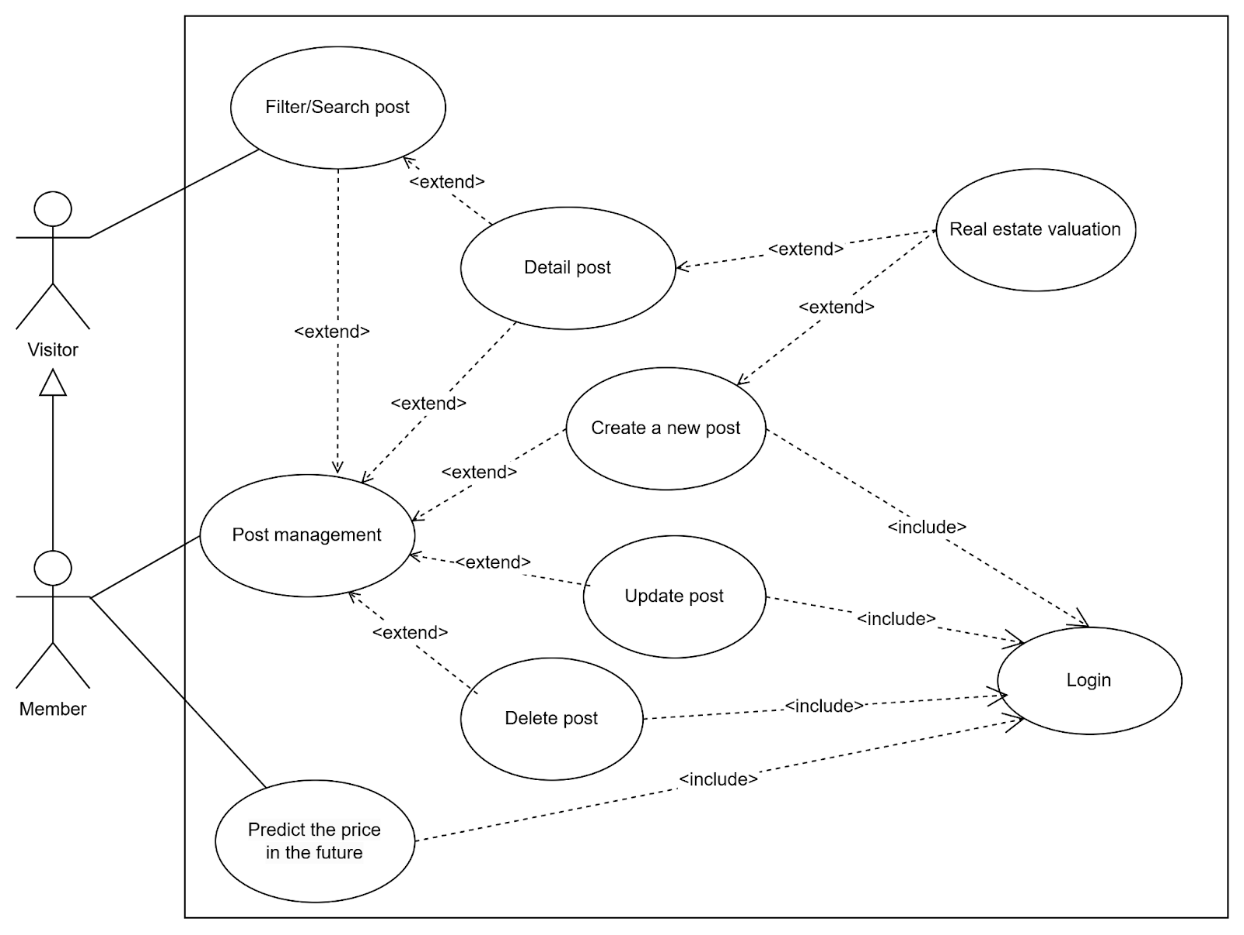


Figure . Post management use case diagram

* Favourite post management

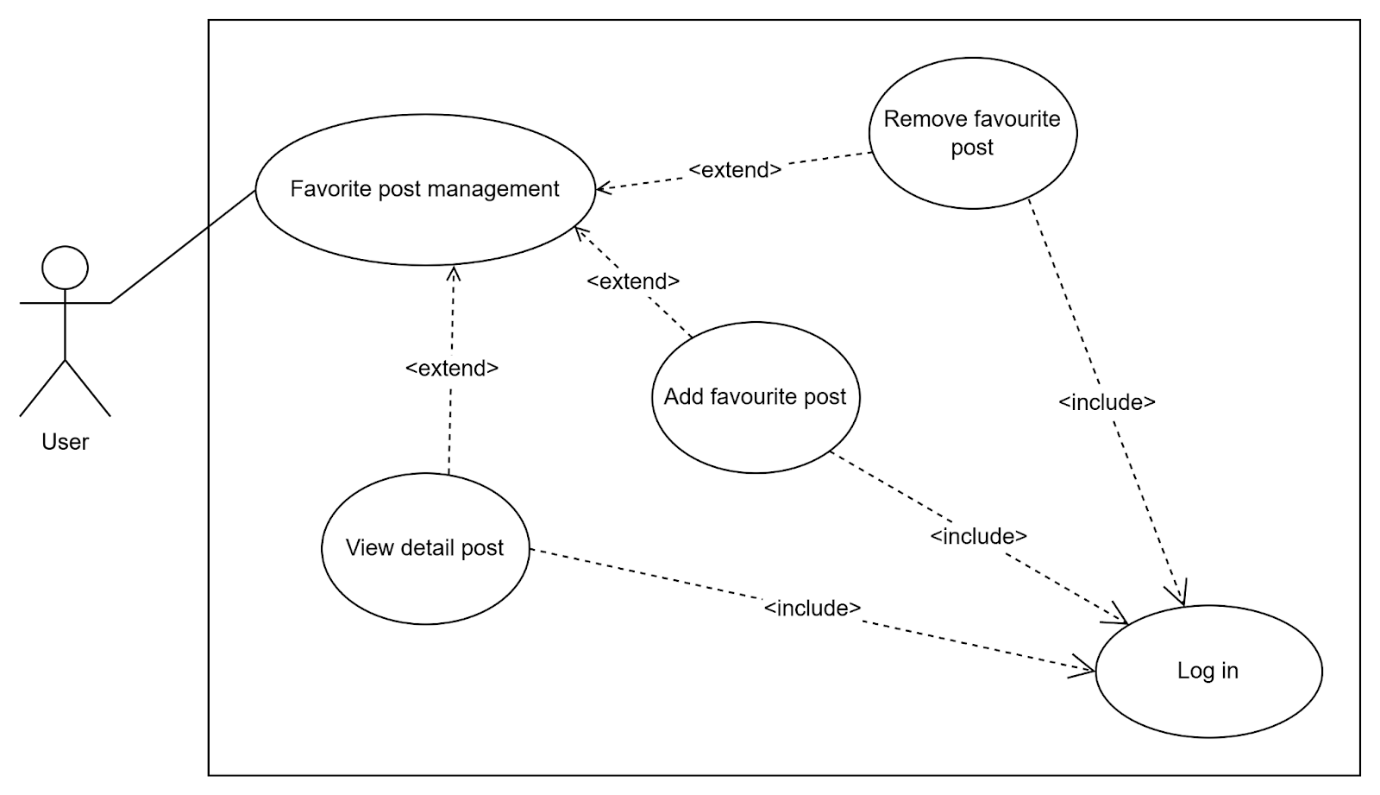


Figure . Favourite post management use case diagram

### Activity diagram

* The stream of activities access to the buyer/seller 's website

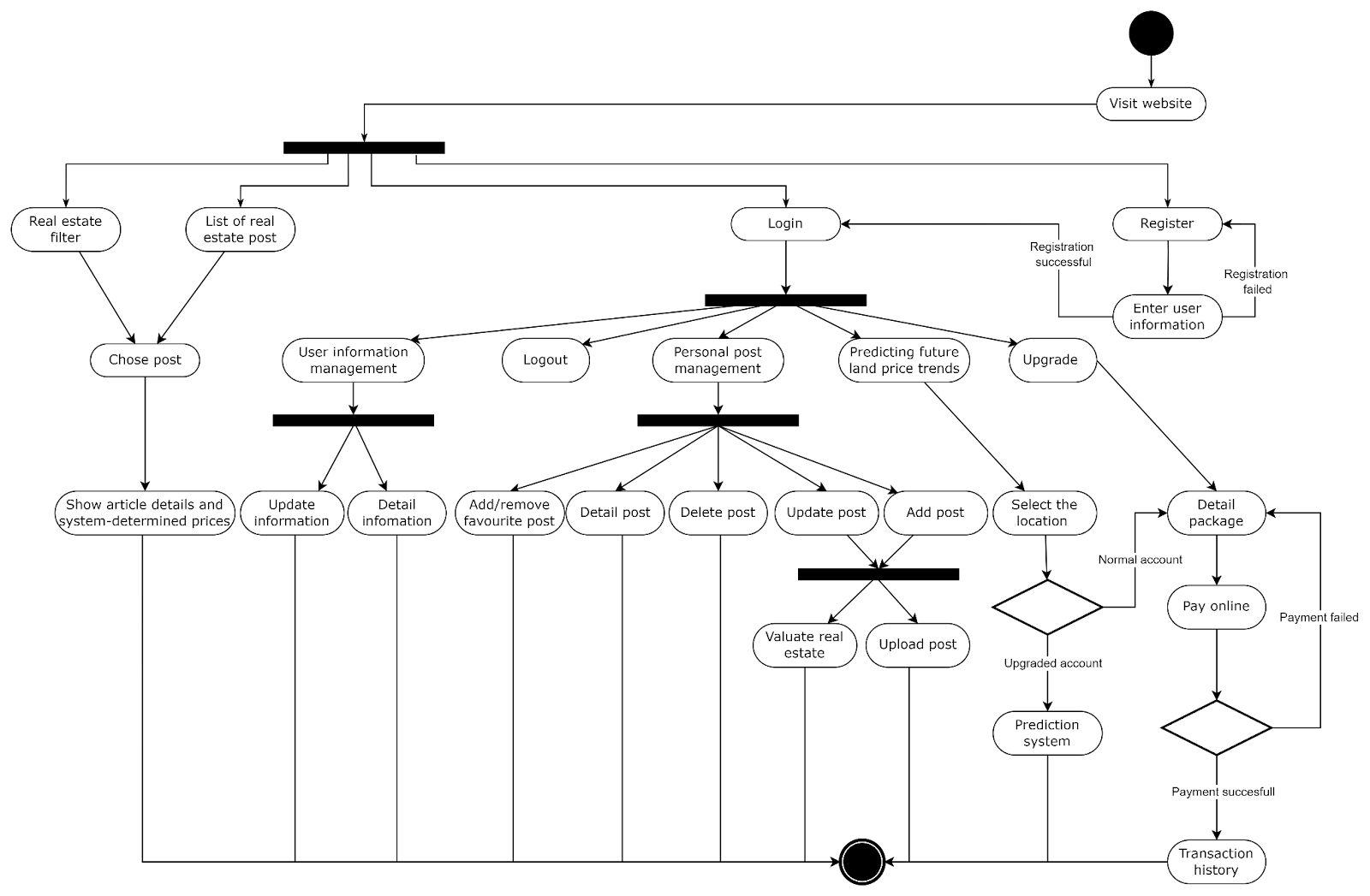


Figure . Client Operation Principle

* The stream of activities access to the admin website

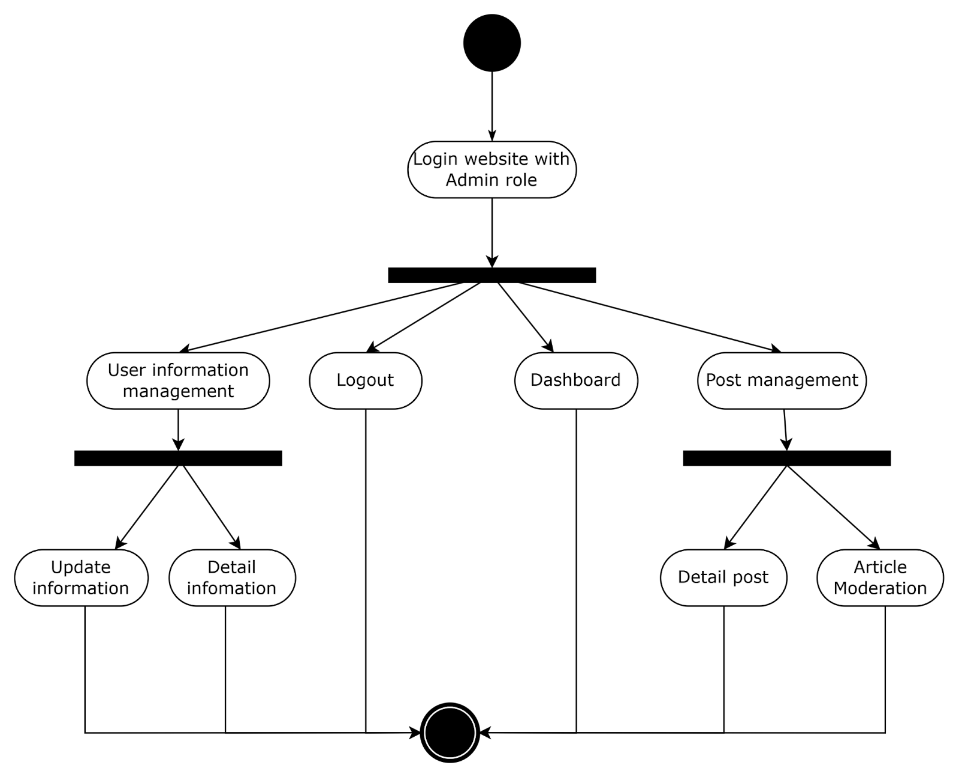


Figure . Admin Operation Principle

### Sequence diagram

* Login/Signup flow design

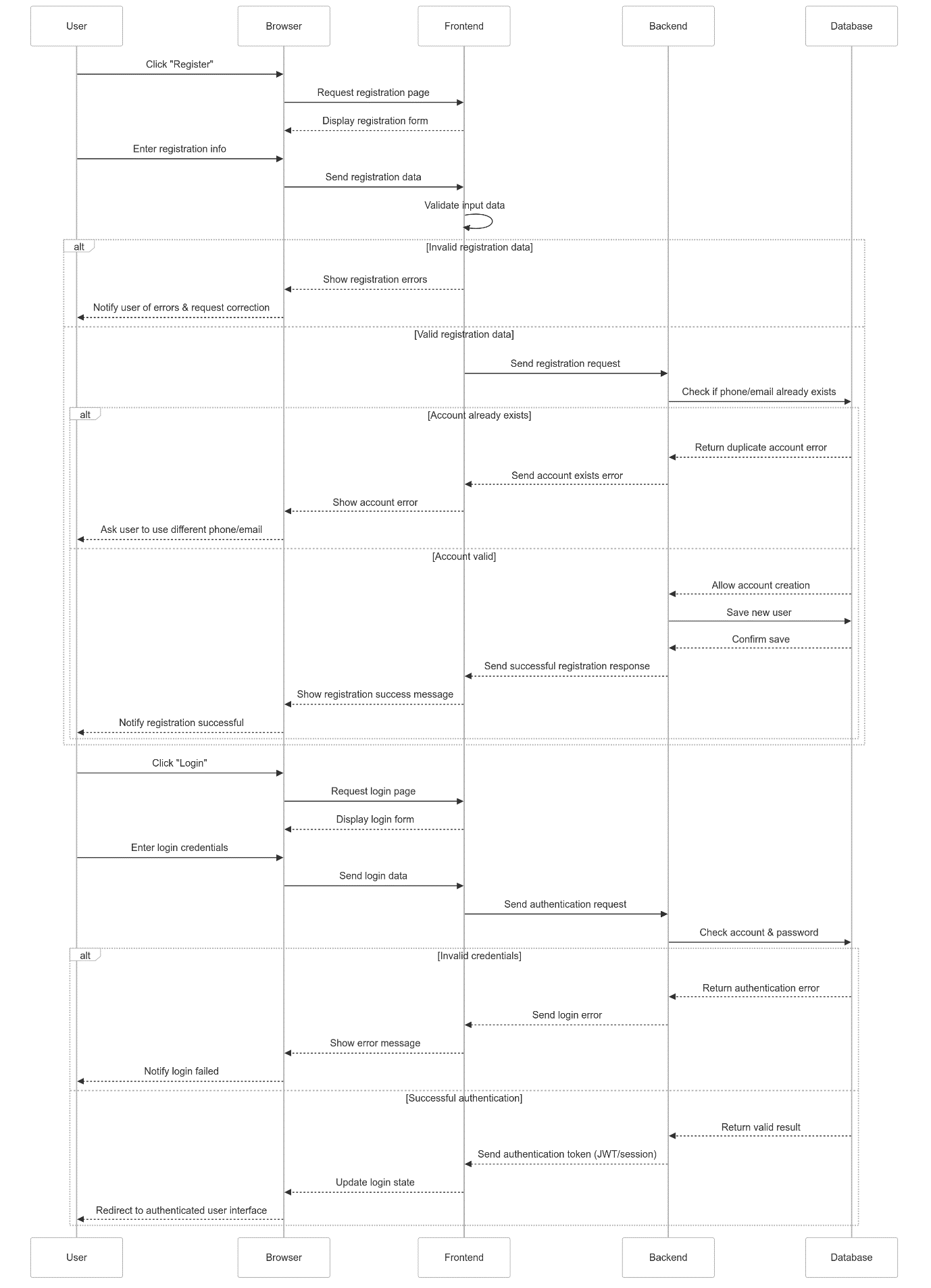
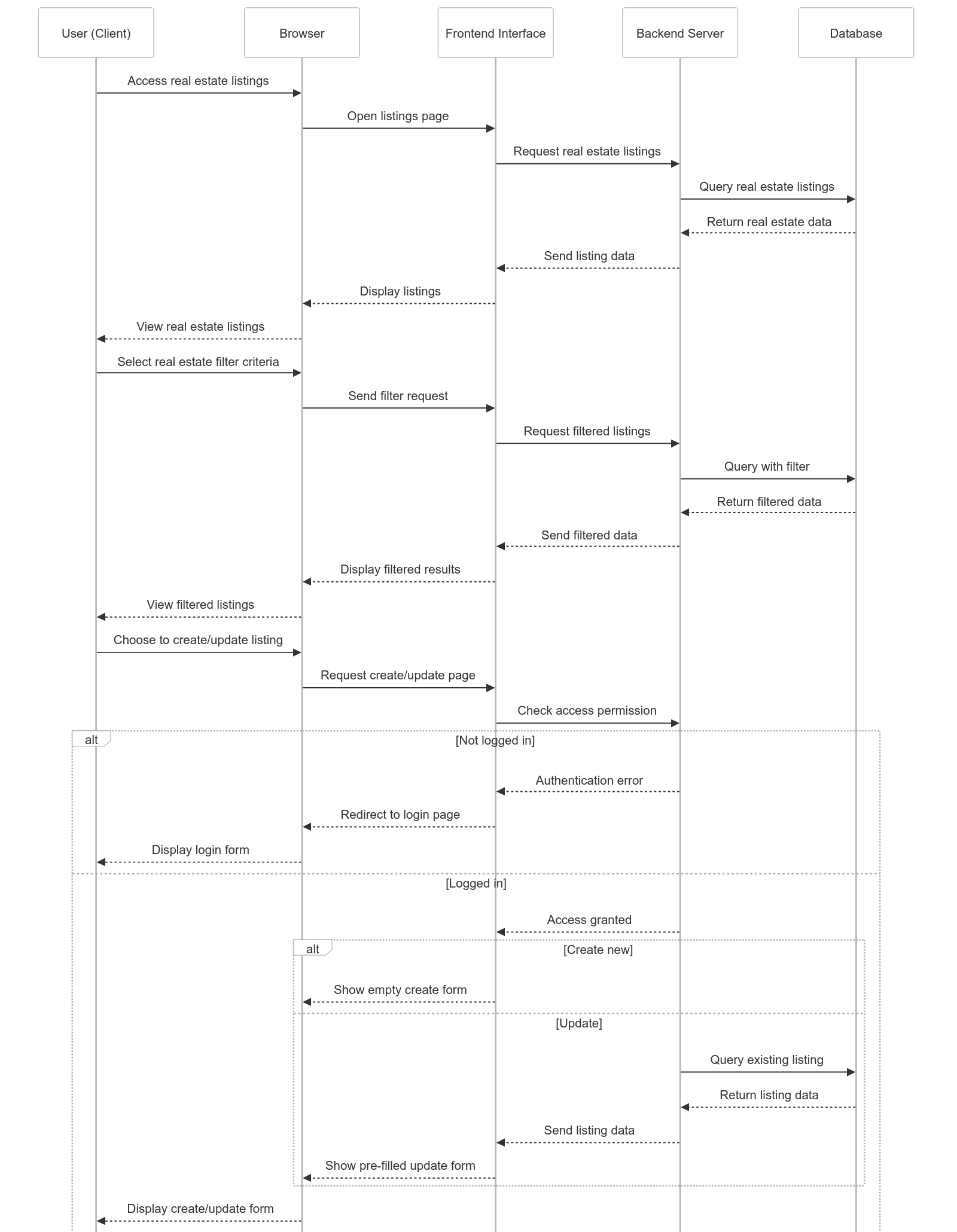


Figure . Registration/Login Flow

* Real estate article management flow design



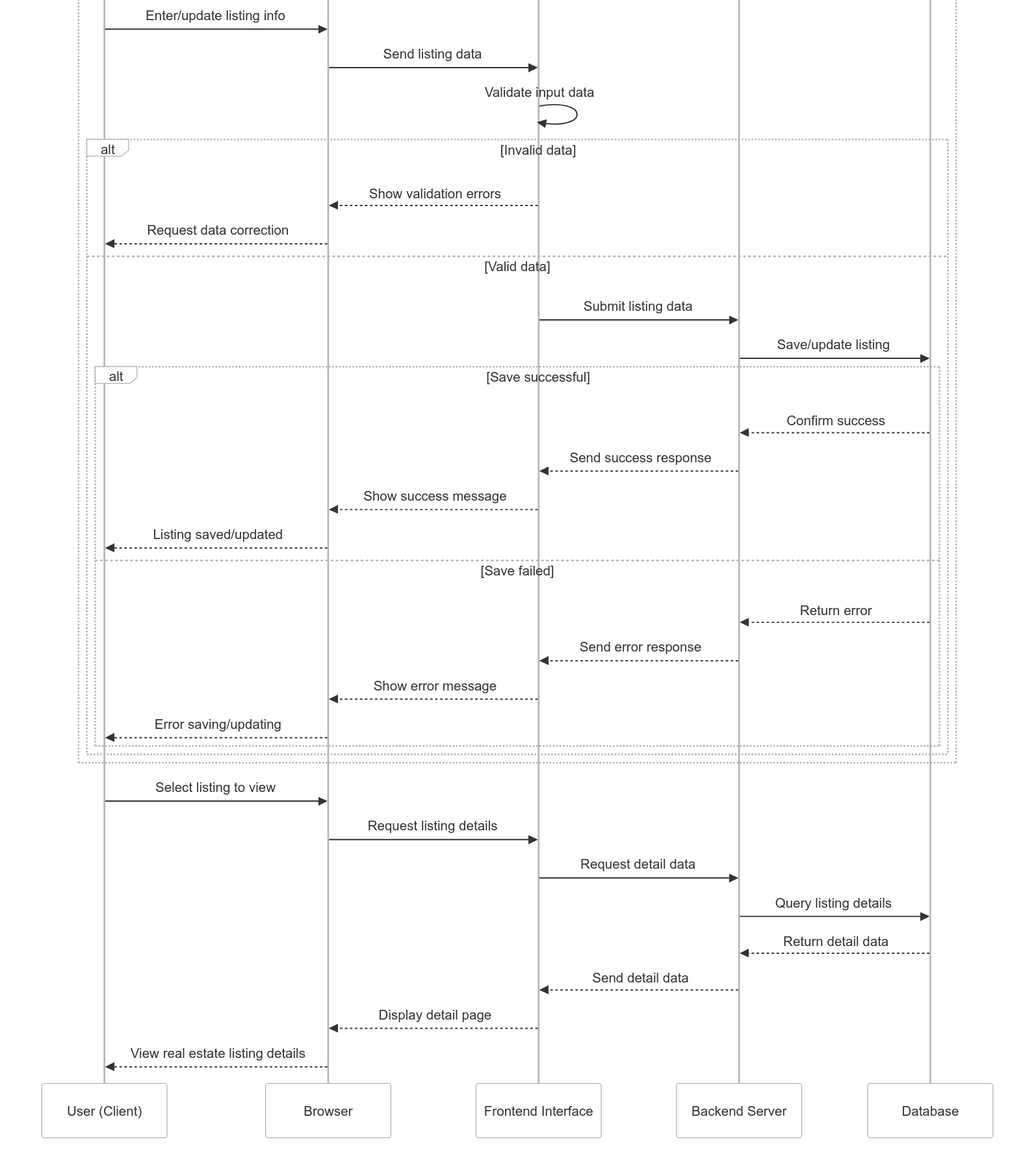


Figure . Real estate article management flow as a user

* Design of account upgrade flow integrating online payment

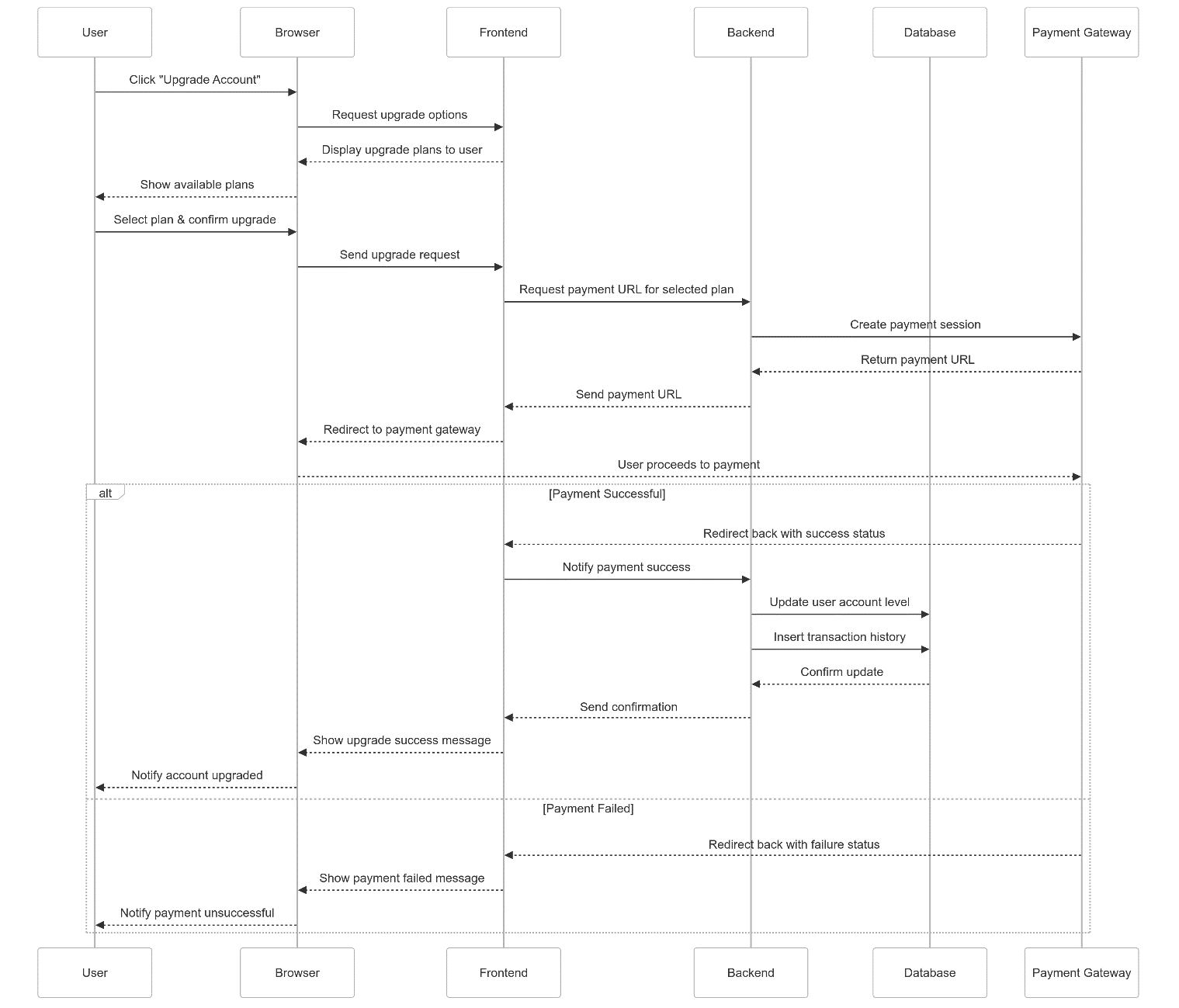


Figure . Account upgrade flow

## Database design

### ER (entity-relationship)

The ER entity relationship model is represented as

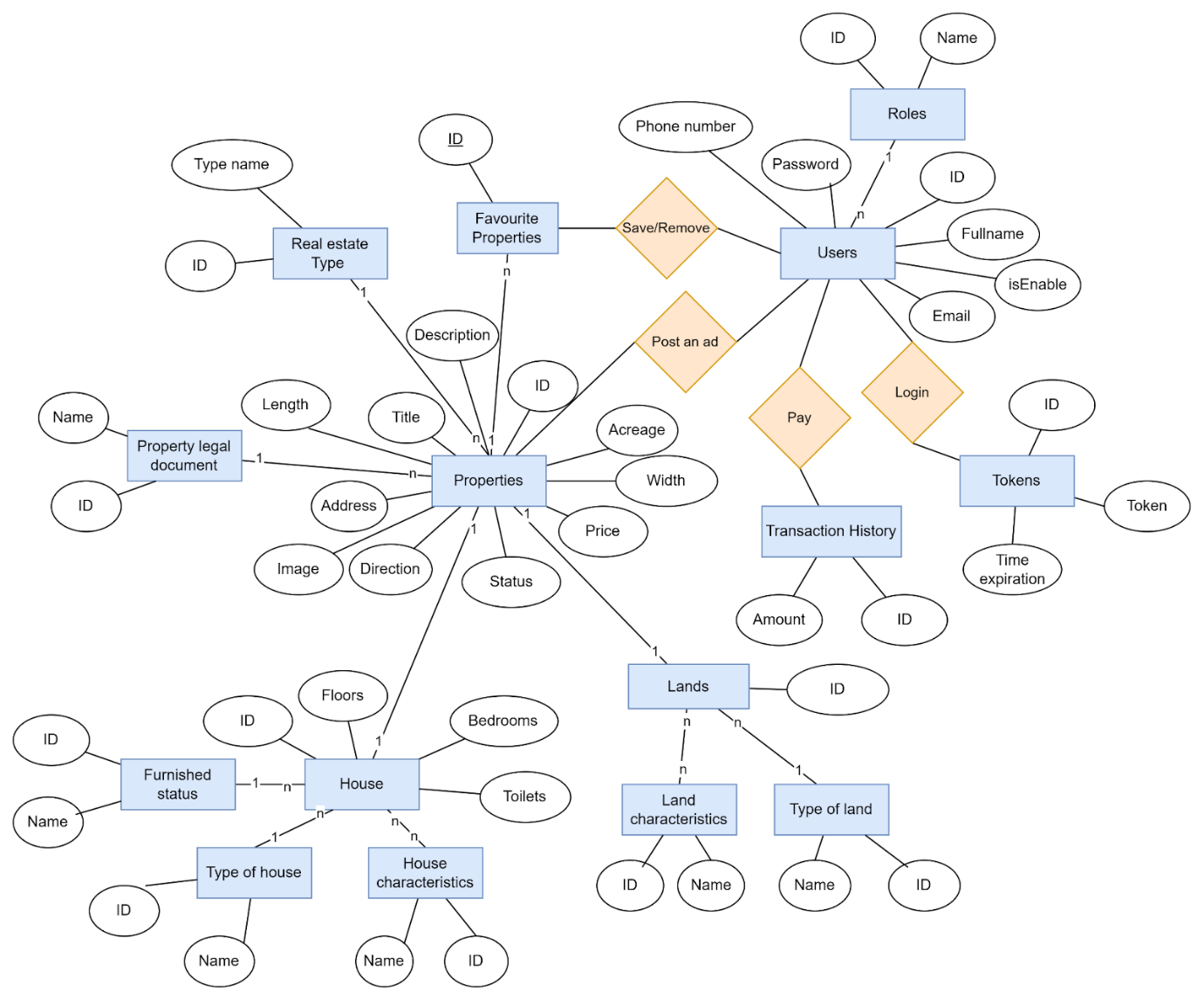


Figure . The ER entity relationship model

### Converting an implementation model into a system model

* Role table (roles)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| role\_id | Integer | Not null | Primary Key | Code of permission |
| name | Enum | Not null |  | Name of permission |

Table . Describe role table attributes

* User table (users)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| user\_id | Integer | Not null | Primary Key | Code of user |
| role\_id | Integer | Not null | Foreign Key | Permission |
| isEnable | Boolean | Not null |  | User account is active or inactive |
| fullName | String | Not null |  | Full name of user |
| email | String | Not null |  | Email of user |
| password | String | Not null |  | Password of user(will be encrypted before entering the database) |
| phone | String | Not null |  | Phone number of user |
| created\_at | LocalDataTime |  |  |  |
| updated\_ad | LocalDataTime |  |  |  |

Table . Describe user table attributes

* Token table (tokens)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| token\_id | Integer | Not null | Primary Key | Code of token |
| user\_id | Integer | Not null | Foreign Key | User |
| token | String | Not null |  | A temporary or persistent string used to authenticate and authorize a user or system |
| created\_at | LocalDataTime |  |  |  |
| updated\_ad | LocalDataTime |  |  |  |

Table . Describe token table attributes

* Transaction history table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| transaction\_id | Integer | Not null | Primary Key | Code of token |
| user\_id | Integer | Not null | Foreign Key | User |
| amount | Number | Not null |  | Amount paid |
| created\_at | LocalDataTime |  |  |  |
| updated\_ad | LocalDataTime |  |  |  |

Table . Describe transaction histories table attributes

* Favourite property table (favourite\_properties)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| favourite\_ property\_id | Integer | Not null | Primary Key | Code of favourite property |
| user\_id | Integer | Not null | Foreign Key | User |
| property\_id | Integer | Not null | Foreign Key | Property |
| created\_at | LocalDataTime |  |  |  |

Table . Describe favourite properties table attributes

* Category table (categories)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| category\_id | Integer | Not null | Primary Key | Code of category |
| Name | String | Not null |  | Name of category |

Table . Describe category table attributes

* Property table (properties)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| property\_id | Integer | Not null | Primary Key | Code of property |
| user\_id | Integer | Not null | Foreign Key | User |
| category\_id | Integer | Not null | Foreign Key | Type of property |
| property\_legal \_document\_id | Integer | Not null | Foreign Key | Type of property legal document |
| status | Enum | Not null |  | Approved or canceled |
| title | String | Not null |  | Title appears at the top of the article |
| description | String | Not null |  | Detailed description of the article |
| region | String | Not null |  | Province name |
| district\_name | String | Not null |  | District name |
| ward\_name | String | Not null |  | Ward name |
| street\_name | String | Not null |  | Street name |
| longitude | Number | Not null |  | Coordinates |
| latitude | Number | Not null |  |
| direction | Number | Not null |  | Main direction of real estate |
| area | Number | Not null |  |  |
| length | Number | Not null |  |  |
| width | Number | Not null |  |  |
| images | String | Not null |  | Contains image links |
| price | Number | Not null |  |  |
| created\_at | LocalDataTime |  |  |  |
| updated\_ad | LocalDataTime |  |  |  |

Table . Describe property table attributes

* House table (houses)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| house\_id | Integer | Not null | Primary Key | Code of house |
| property\_id | Integer | Not null | Foreign Key | Code of property |
| furnished\_ status\_id | Integer | Not null | Foreign Key | Interior condition |
| house\_ type\_id | Integer | Not null | Foreign Key | Type of house |
| floors | Integer | Not null |  |  |
| bedrooms | Integer | Not null |  |  |
| toilets | Integer | Not null |  |  |

Table . Describe house table attributes

* Furnished status table (furnished\_status)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| furnished\_ status\_id | Integer | Not null | Primary Key | Code of furnished status |
| name | String | Not null |  | Describe the condition of the interior |

Table . Describe furnished status table attributes

* Type of house table (house\_type)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| house\_ type\_id | Integer | Not null | Primary Key | Code of house type |
| name | String | Not null |  | House type description |

Table . Describe type of house table attributes

* House characteristics table (house\_characteristics)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| house\_ characteristic\_id | Integer | Not null | Primary Key | Code of house  characteristics |
| name | String | Not null |  | House characteristics description |

Table . Describe house characteristics table attributes

* + Here create an additional table (house\_characteristic\_mappings) to hold the characteristic keys of the house object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| house\_characteristic\_ mapping\_id | Integer | Not null | Primary Key | Code of house  characteristic mapping |
| house\_ characteristics\_id | Integer | Not null | Foreign Key | House  characteristics |
| house\_id | Integer | Not null | Foreign Key | House |

Table . Describe house characteristic mapping table attributes

* Land table (lands)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| land\_id | Integer | Not null | Primary Key | Code of house |
| property\_id | Integer | Not null | Foreign Key | Code of property |
| land\_type\_id | Integer | Not null | Foreign Key | Type of land |

Table . Describe land table attributes

* Type of land table (land\_types)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| land\_type\_id | Integer | Not null | Primary Key | Code of land type |
| name | String | Not null |  | Land type description |

Table . Describe type of land table attributes

* Land characteristics table (land\_characteristics)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| land \_ characteristic\_id | Integer | Not null | Primary Key | Code of land  characteristics |
| name | String | Not null |  | Lanđ characteristics description |

Table . Describe land characteristics table attributes

* + Here create an additional table (land\_characteristic\_mappings) to hold the characteristic keys of the land object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Required | Constraint | Description |
| land \_characteristic\_ mapping\_id | Integer | Not null | Primary Key | Code of land  characteristic mapping |
| land \_ characteristics\_id | Integer | Not null | Foreign Key | Land  characteristics |
| house\_id | Integer | Not null | Foreign Key | Land |

Table . Describe land characteristic mapping table attributes

### Database schema

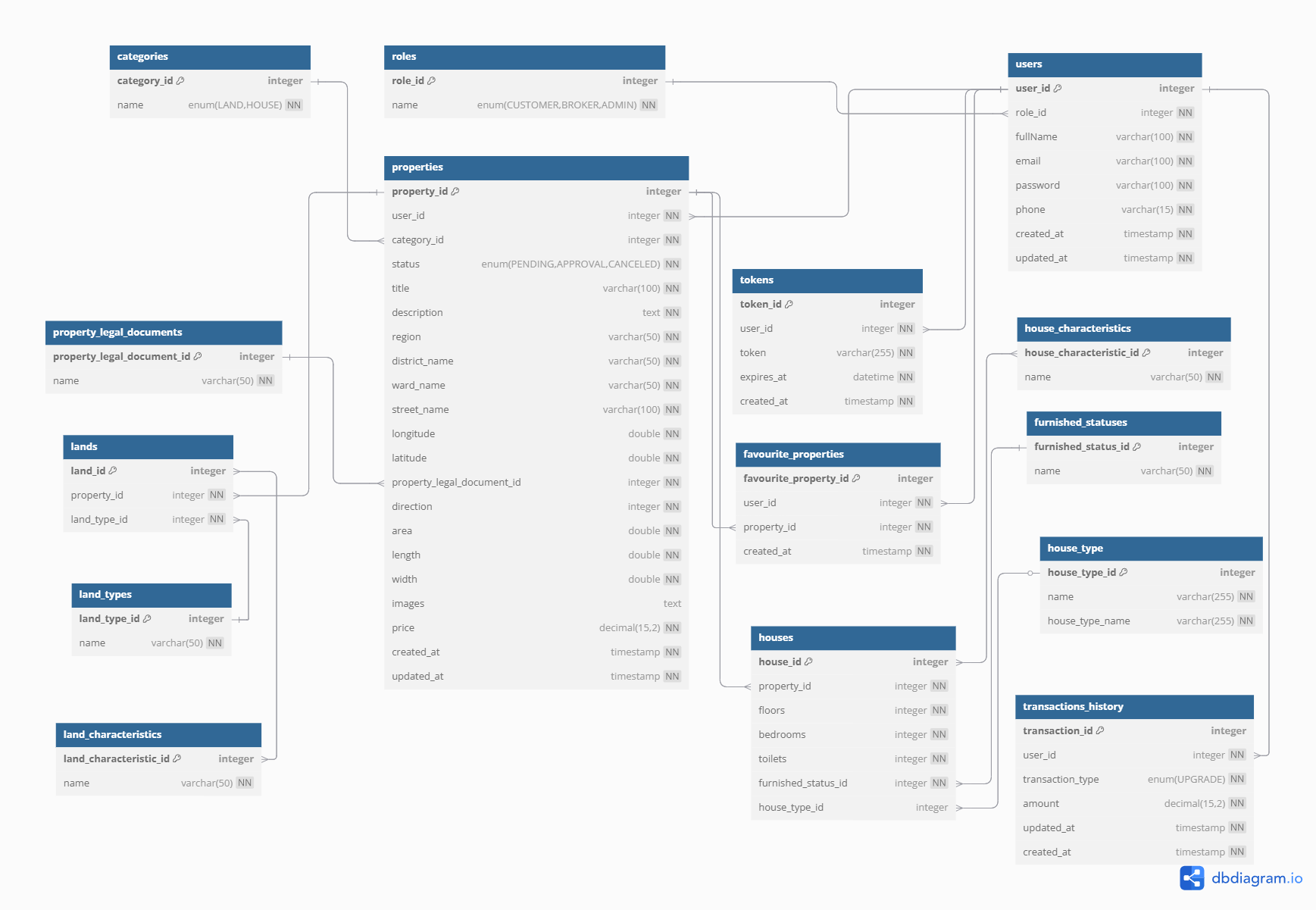


Figure . Relational Model

## Real estate prediction system development

* The problem of predicting real estate prices will be divided into two types including:
  + Real estate valuation: users will provide information related to land or the house they want to sell or buy for the system to evaluate.
  + Real estate price forecast: the system will predict the price in the near future of a certain area given by users at the same level.
* Common processing steps of the two models
  + Data Crawling: collect data from the “Nhà tốt” website and have “RawData.csv” file with … samples.
  + Data Cleaning: after handling missing and outlier values, the process returns a cleaned dataset containing the following fields and is exported as “Cleandata.csv”:
    - Real estate valuation:

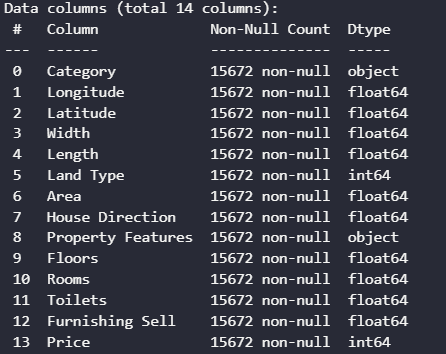


Figure . Real estate valuation model features

* + - Real estate price forecast:

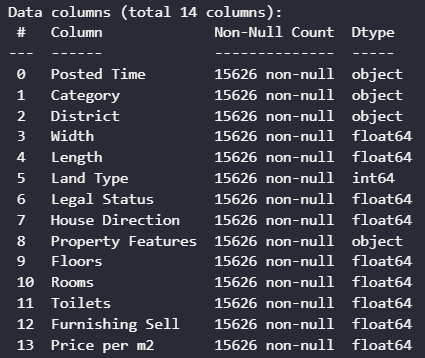


Figure . Real estate price forecast features

* + - The two problems use slightly different features. For the valuation model, exact coordinates are included since price can vary significantly even along the same street. For the forecasting model, time becomes a crucial factor, and detailed coordinate data is sparse. Therefore, we generalize location by district and use **price per m²** as the target variable.
  + Feature Engineering:
    - Feature Tranformation:
      * First, one-hot encoding is applied to categorical variables such as **property category, land type, direction, furnishing status** and **district.**
      * For numerical features, various normalization techniques (e.g., Z-score, log, sqrt, min-max, arsinh) are tested. The most effective method is selected based on distribution analysis.
      * The result is a transformed dataset exported as “TransformedData.csv”.

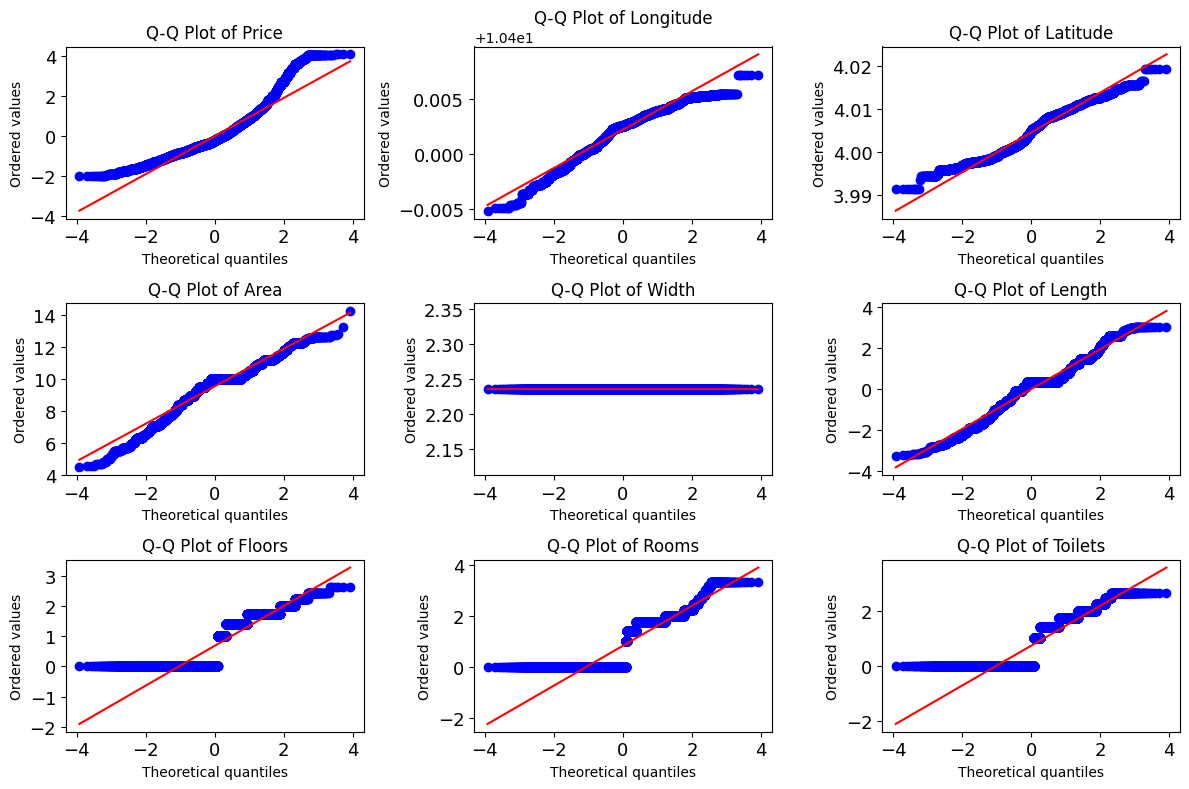


Figure . Q-Q Plots After Optimal Normalization per Feature

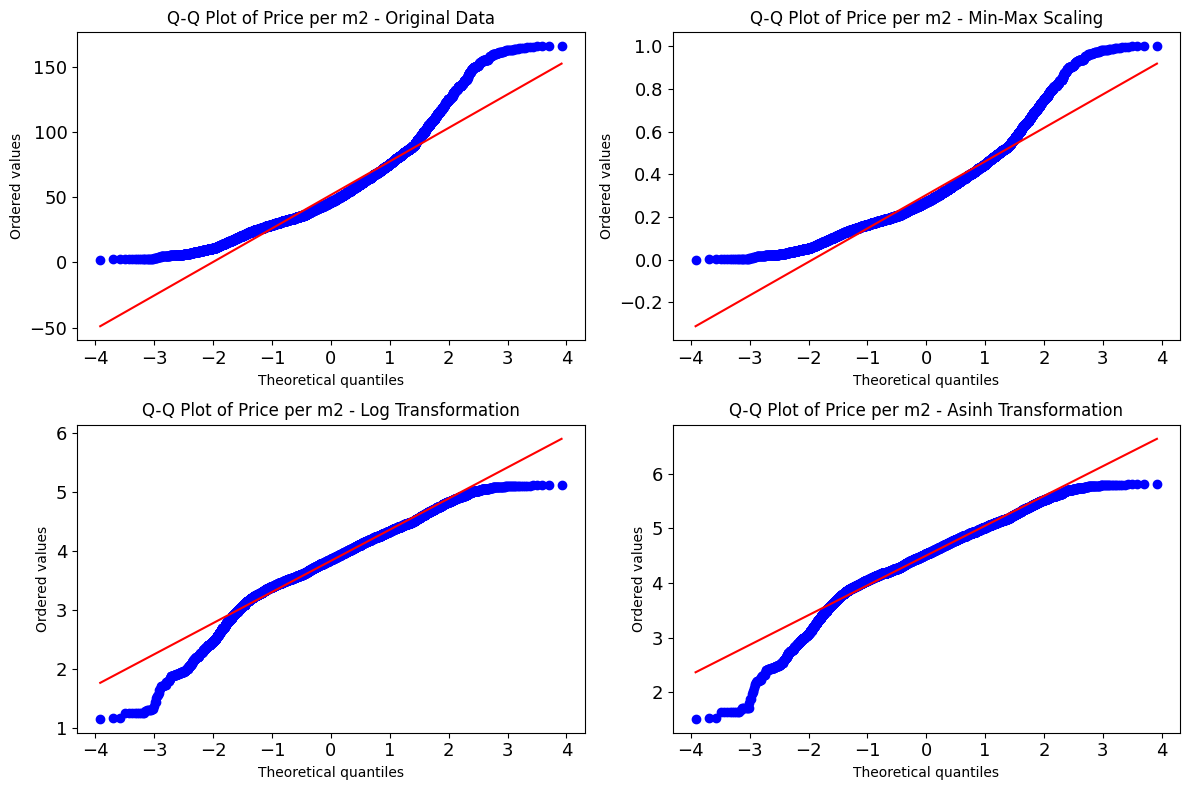


Figure . Comparison of Q-Q Plots of **Price per m²** across different transformation methods

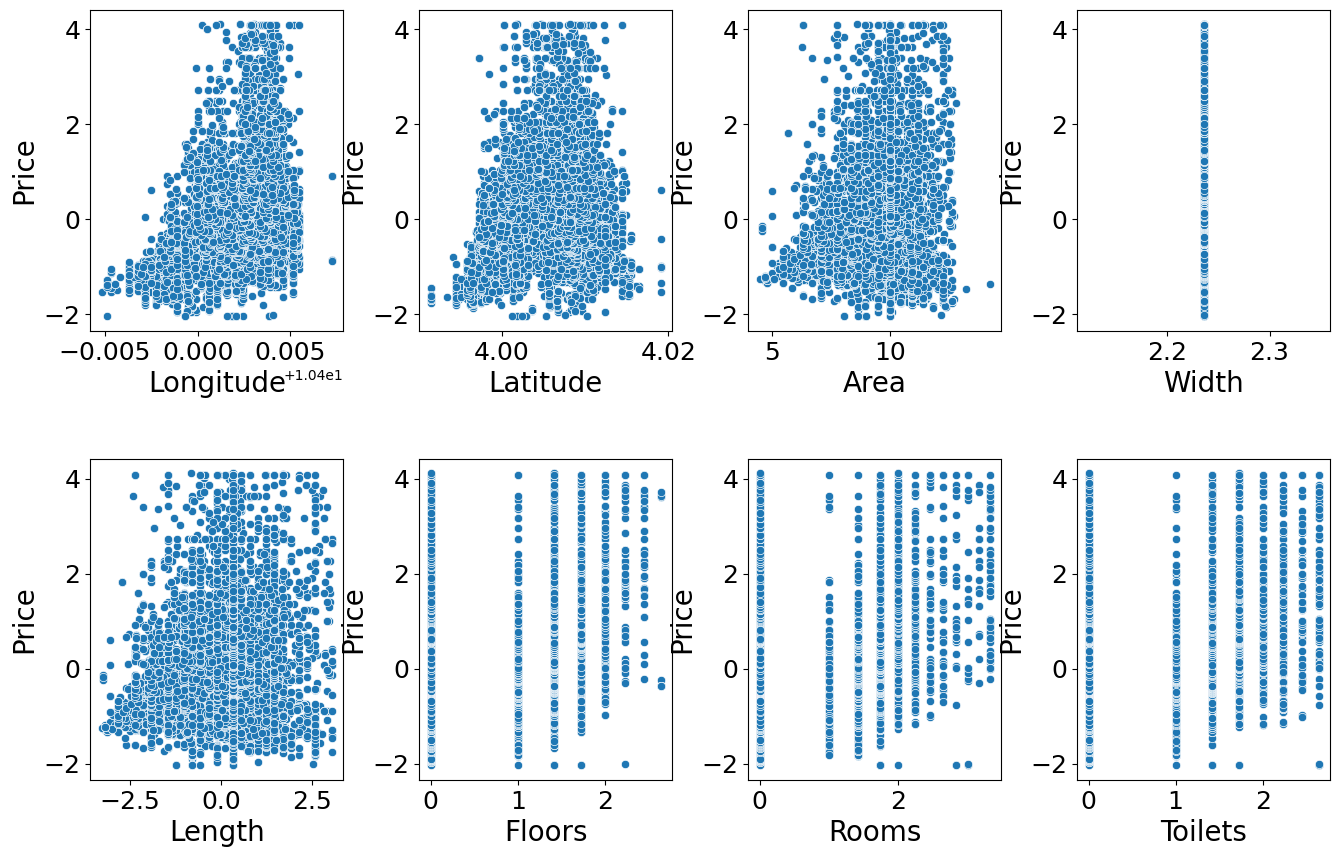
* + - Feature Selection:
      * Observing the relationships between the coordinates (Longitude, Latitude), area, and width with the target variable (Price), it was noted that these features have a slight correlation with the target. 

Figure . Scatter plot of features vs. **Price**

* + - * **SelectKBest** was used to rank the features based on their correlation with the target, with **Longitude** and **Latitude** being the most influential.

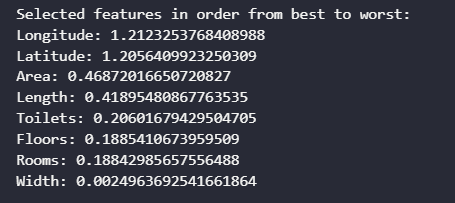


Figure . Selected features in order from best to worst

* + - * The **RFECV** method was then applied, selecting the optimal number of features, which is 8 in this case. The best set of features for predicting the target variable includes **Longitude, Latitude, Area, Width, Length, Floors, Rooms,** and **Toilets**.

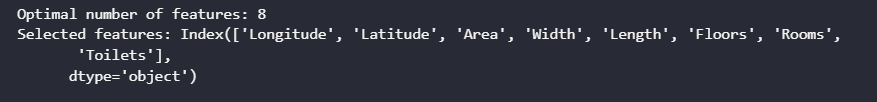


Figure . Optimal number of features

* + - * The plot of the **RFECV** shows the relationship between the number of selected features and the mean test accuracy.

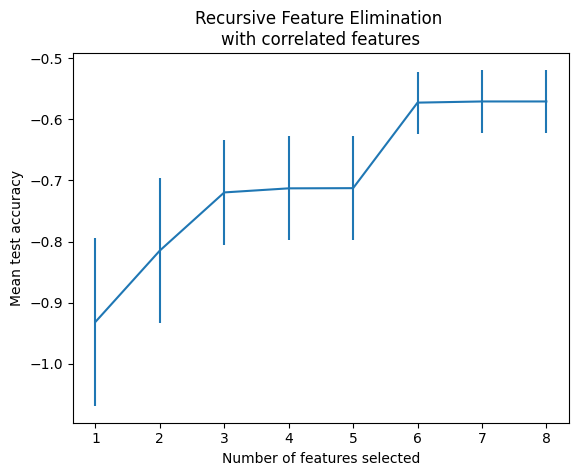


Figure . Recursive Feature Elimination with correlated features

### Real estate valuation model

* The dataset is divided into two parts:
  + 80% for the training set (…sample)
  + 20% for the test set (…sample)
* Three regression algorithms are applied to build the models: Linear Regression, Ridge Regression, and Random Forest Regression.
* The evaluation metrics for Random Forest Regression model achieved the best performance are presented below

|  |  |  |
| --- | --- | --- |
|  | Train set | Test set |
| MSE |  |  |
| RMSE |  |  |
| MAPE |  |  |
| R-Squared | 80.7% | 76.02% |

Table . Evaluation metrics table for the real estate valuation models

### Real estate price forecast model

* To handle time-based patterns, we extracted temporal features from the **Posted Time** field:
  + **year, month, day, day of week** and **quarter**: These help capture seasonality and time-related trends.
  + Cyclical Encoding: **month** and **dayofweek** are converted into sine and cosine values to preserve their cyclical nature (e.g., December is close to January).
* The dataset was split into two parts:
  + 80% for training (… samples)
  + 20% for testing (…samples)
* Three forecasting models were applied for comparison: XGBoost Regressor, LightGBM Regressor, Facebook Prophet.
* The evaluation metrics for XGBoost Regressor model achieved the best performance are presented below

|  |  |  |
| --- | --- | --- |
|  | Train set | Test set |
| MSE |  |  |
| RMSE |  |  |
| MAPE |  |  |
| R-Squared | 80.42% | 68.29% |

Table . Evaluation metrics table for the real estate price forecast models

## Apply NSFW image and toxic content classification models

### NSFW image classification models

In this project, I integrated the NudeNetv2 model to detect and classify sexually explicit content in images. I utilized the pre-trained “classifier\_model.onnx” for efficient inference.

The model categorizes content into two classes:

* safe – The image/video is not sexually explicit
* unsafe – The image/video contains nudity or explicit content

I cloned the repository and used the available ONNX version of the classifier to perform predictions on image data.

This classification component plays a crucial role in automatically filtering out inappropriate content, enhancing the safety and compliance of our platform.

## Chapter Conclusion

This chapter provides the reader with an overview of the real estate business operations, offering insights into the key activities and processes involved in the industry. It also includes an analysis and design of the system architecture, along with a detailed discussion of the database model. Additionally, the chapter focuses on the development of a price support system, which aims to assist buyers by offering pricing assistance and making real estate transactions more accessible.

# IMPLEMENTATION AND RESULTS EVALUATION

# CONCLUSION AND DEVELOPMENT DIRECTION

# REFERENCES

<https://aws.amazon.com/vi/what-is/python/>

<https://200lab.io/blog/jupyter-notebook-la-gi>

<https://docs.jupyter.org/en/latest/what_is_jupyter.html>

<https://www.geeksforgeeks.org/introduction-to-java/>

<https://www.geeksforgeeks.org/spring-boot/>

https://viblo.asia/p/tat-tan-tat-ve-mo-hinh-mvc-Rk74avjAJeO

https://viblo.asia/p/luong-di-trong-spring-boot-ORNZqdELK0n

https://www.geeksforgeeks.org/what-is-mysql/

# APPENDIX 1

# APPENDIX 2