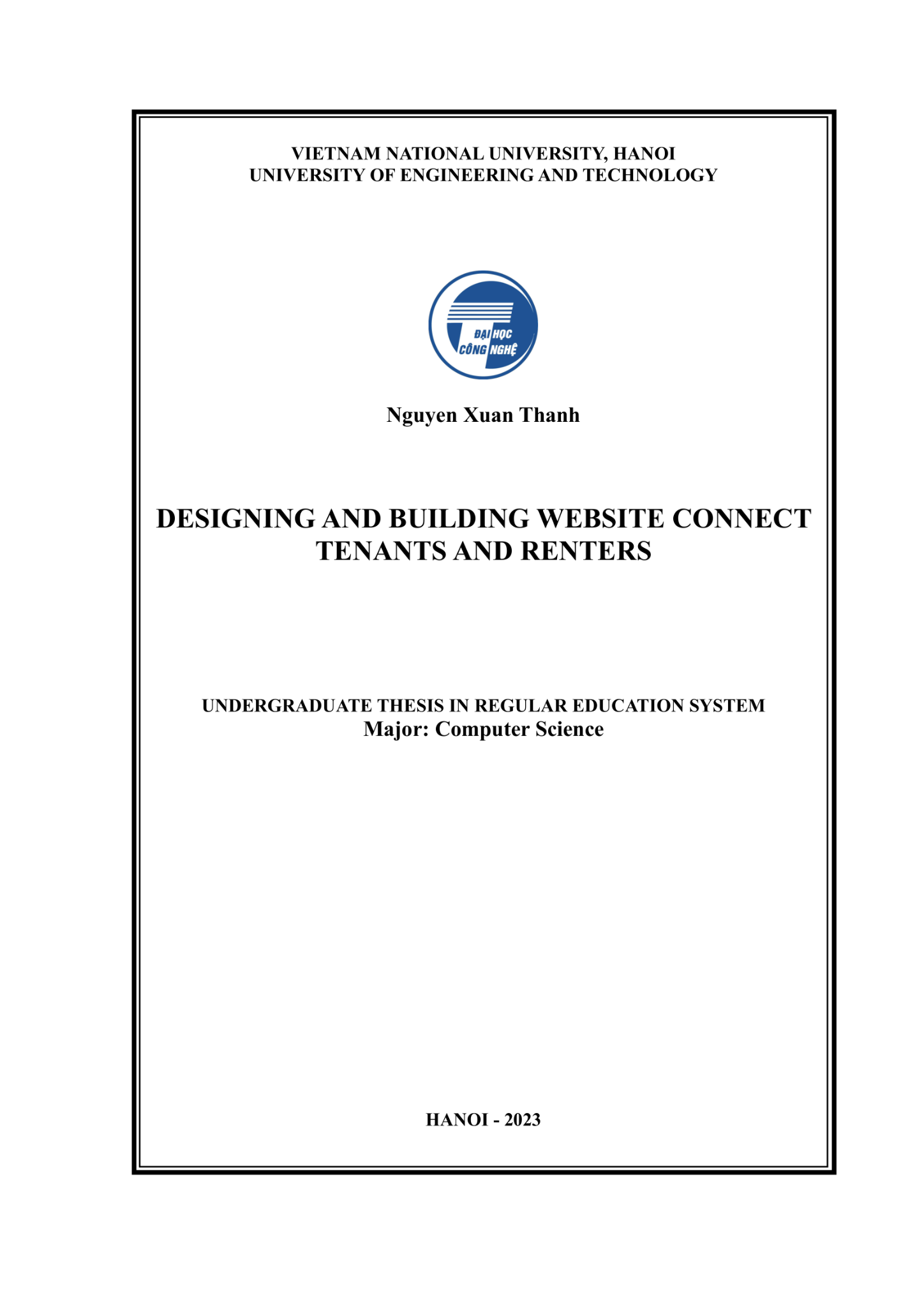
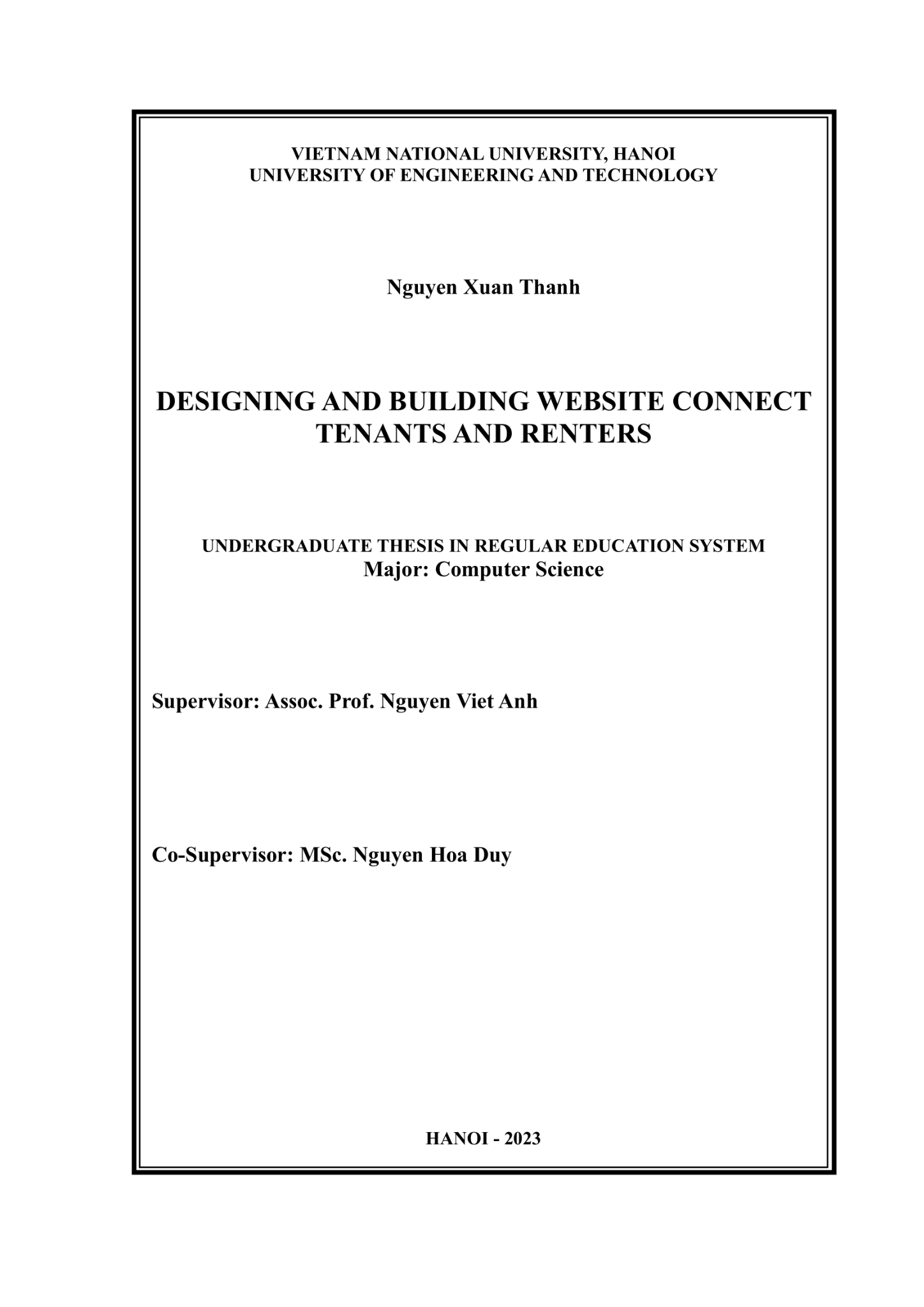
****

****

# Summary

**Abstract**: The website connecting renters and landlords is an increasingly popular online platform. With the growth of the tourism industry and the need for accommodation, this website helps improve the process of finding a place to stay and reduces time and effort for users.

With its search function, renters can easily find options for rooms, apartments, or houses for rent on this platform. Users can search by location, price, amenities, and size of the apartment to find options that meet their needs. The website also provides detailed information about available accommodations, including amenities that fit the user's budget.

To attract customers, landlords can post detailed information about their accommodation. Criteria such as price, location, size, and amenities are updated to attract the attention of potential customers. In addition, previous customer reviews and feedback are also displayed, helping users evaluate the quality of the accommodation and make smart choices.

The messaging feature between renters and landlords is also a highlight of the website. This feature allows both parties to communicate directly and exchange detailed information about the accommodation, prices, and requirements of the renters before making a transaction. This helps enhance trust between both parties and ensures the accuracy of the information.

In summary, the website connecting renters and landlords has helped improve the process of finding accommodation for customers while providing a diverse choice for landlords. This saves time, effort, and enhances trust between both parties. The website's security features and customer support also help create the best experience for users on the website.

**Keyword:** Booking website

# Acknowledgment

Firstly, I would like to express our sincere appreciation and deepest gratitude to Nguyen Viet Anh - lecturer of the Faculty of Information Technology at the University of Technology, our mentor who has provided enthusiastic guidance and instruction.

I would also like to extend our heartfelt thanks to all the teachers currently teaching at the Faculty of Information Technology at the University of Technology who have warmly supported, provided materials, and offered valuable contributions!

# Authorship

I hereby confirm that all the graduation theses were conducted and developed under the guidance and support of my supervisor, Assoc. Prof. Dr. Nguyen Viet Anh. I did not copy any materials or any other research works. All information and data used in the graduation theses have clear sources and are cited in the reference section.

I take all responsibility if I violate these commitments according to the regulations of the school!

Student

Nguyen Xuan Thanh

**Table of contents**

[**Summary 2**](#_plsk569jqrwj)

[**Acknowledgment 3**](#_voyopfw8q90)

[**Authorship 4**](#_ipu41jbgrpmg)

[**Abbreviations 9**](#_prr6x6u2h4g6)

[**Introduction 10**](#_aowhg2kvqn7u)

[1. Motivation 10](#_gfx85u252ifp)

[2 - Research objective 11](#_bzmojg53qto)

[3 - Area of research 12](#_wkluxpa99ok4)

[4 -Research content 12](#_aleckv9j9h0f)

[**Chapter 1 - Problem 14**](#_il5je1vss1fe)

[1.1 Description of problem 14](#_ivwdnfbs9qab)

[1.2 Similar systems 15](#_njf5w9mgeq21)

[1.3 The difference between Rentropolis and similar websites in Vietnam 16](#_9z3dk0hdpo8e)

[1.4 Problem details: 17](#_pfrxbab52d37)

[1.4.1: Property management: 17](#_d1lppwq0f6d6)

[1.4.2: Booking : 17](#_xg16kpp881ax)

[1.4.3: Payment: 17](#_fnpohsdcksyx)

[1.4.4: Notification: 17](#_x5sbr5x29ykb)

[1.4.5: The feature of messaging between the room owner and the customer. 18](#_nhk84au5w6dr)

[1.4.6: Accommodation management and occupancy status management: 18](#_rreg38hk5j3i)

[1.4.7: Revenue management: 18](#_ab1wcscfyob8)

[1.4.8: Search place: 18](#_47xdk7ndlyx6)

[1.4.9: Booking management: 18](#_udmm8l1cxpxq)

[1.4.9: Review 19](#_g6qakz42tguq)

[**Chapter 2 - System analysis and design 20**](#_9larxmz8jk0c)

[2.1 Use case diagram 20](#_gn4co0ibkj5t)

[2.2 Use case analysis 24](#_7hbzzt8s4usb)

[2.2.1 : Common 24](#_k3dycyfavz1d)

[2.2.1.1: "Sign in/Sign up with Google" 24](#_y1a9bi1b1vbg)

[2.2.1.1.1: Use case 24](#_vczqrbp5hd4z)

[2.2.1.1.2: Activity diagram 24](#_8kev9cr9ad72)

[2.2.1.1.3. Sequence diagram 25](#_j8a6le53h5da)

[2.2.1.5: Messaging 26](#_y0rkkgy1imfr)

[2.2.1.5.1: Use case 26](#_b6hnxnyz9rww)

[2.2.1.5.2: Activity diagram 26](#_6e01udaxracy)

[2.2.1.5.3: Sequence diagram 27](#_otz3vubvuvhu)

[2.2.1.6: Notification 28](#_se60dvl6nd8)

[2.2.1.6.1: Use case 28](#_hpsdc44lfxjt)

[2.2.1.6.2: Activity diagram 28](#_26h9y4ixf2gb)

[2.2.1.6.3: Sequence diagram 29](#_o68mhwk5la9o)

[2.2.2: Client 30](#_r2btakwa4yat)

[2.2.2.1: Search place 30](#_nzglpo7vum8p)

[2.2.2.1.1: Use case 30](#_tkeuotjm96ws)

[2.2.2.1.2: Activity diagram 31](#_wl0l7szfhu2r)

[2.2.2.1.3: Sequence diagram 31](#_j0n0414n0x7r)

[2.2.2.2: Booking accommodation 32](#_bluvr2i71bef)

[2.2.2.2.1: Use case 32](#_lzakv6gaxcbq)

[2.2.2.2.2: Activity diagram 34](#_723w2clhaufj)

[2.2.2.2.3: Sequence diagram 34](#_ggvkq3wh7eg7)

[2.2.2.3: Booking management 36](#_hayln7gjmj1t)

[2.2.2.3.1 Cancel booking 36](#_nnuhyu7sx4s9)

[2.2.2.3.1.1: Use case 36](#_7o67knvxrohj)

[2.2.2.3.1.2: Activity diagram 36](#_bwoa8bbipa4v)

[2.2.2.3.1.3: Sequence diagram 37](#_rtnl8f9u2rrt)

[2.2.2.4: Review 38](#_agznsyl5dgrk)

[2.2.2.4.1: Use case 38](#_7eq9zctaum9u)

[2.2.2.4.2:Activity diagram 39](#_eihm0lea0e0o)

[2.2.2.4.3: Sequence diagram 39](#_3fku05tmsuhy)

[2.2.3: Host 40](#_x929tg2x37q)

[2.2.3.1: Listing management 40](#_irgy7e8hrgv1)

[2.2.3.1.2.1: Use case 40](#_kdc8p7luk4av)

[2.2.3.1.2.2: Activity diagram 41](#_4o9bof6uvejg)

[2.2.3.1.2.3: Sequence diagram 42](#_9zlvxz2jb1zj)

[2.2.3.2: Listing request management 43](#_w1yh48tiazfu)

[2.2.3.2.1.2 : Approve 43](#_53mollgtfzf6)

[2.2.3.2.1.2.1: Use case 43](#_5kg8i2kqvsoa)

[2.2.3.2.1.2.2: Activity diagram 44](#_skdzvqtu18bc)

[2.2.3.2.1.2.3: Sequence diagram 44](#_vs0n46bc52yl)

[2.2.3.2.4: Already canceled 45](#_opv89ukn56y3)

[2.2.3.2.4.1: Use case 45](#_bt1l8nb94sgy)

[2.2.3.2.4.2: Activity diagram 45](#_ltp714srmv3c)

[2.2.3.2.4.3: Sequence diagram 46](#_2yjovakyd66p)

[2.2.3.3: Statistics 47](#_7wqc1yki269f)

[2.2.3.3.1: Use case 47](#_ph0t9v89gm03)

[2.2.3.3.2: Activity diagram 48](#_dsu5u6fdhh1n)

[2.2.3.3.3: Sequence diagram 48](#_16zhhbjg6v90)

[2.2.4: Admin 49](#_mc296s2mri9n)

[2.2.4.1: View user list (and report) 49](#_n9b1txihv6tb)

[2.2.4.1.1: Use case 49](#_e1l3r3g3ldta)

[2.2.4.1.2: Activity diagram 50](#_q7s451ps1ytu)

[2.2.4.1.3: Sequence diagram 51](#_nn6xsdw5q7us)

[2.2.4.2: Approve accommodation addition/modification requests 52](#_dkx9vo6amf1s)

[2.2.4.2.1: use case 52](#_wpw9ljrid6cv)

[2.2.4.2.2: Activity diagram 53](#_2em7hldphyua)

[2.2.4.2.3: Sequence diagram 53](#_1oiey5egk8va)

[2.2.4.3: Banned user 54](#_i98t79pgo1om)

[2.2.4.3.1: Use case: 54](#_qqyiosdjptwt)

[2.2.4.3.2: Activity diagram 54](#_tsaiclr9bd22)

[2.2.4.3.3: Sequence diagram 55](#_2evxen4bl17g)

[2.3 Database design 55](#_w39p469pnrgi)

[2.3.1: The Entity Relationship Diagram (ERD) 56](#_w6glxhq5yfbw)

[2.3.2: Detailed description 57](#_mf4zc4wiurb8)

[2.4 Architectural design 61](#_az7dt6feevog)

[2.5 Module design 62](#_h5r8vb223m88)

[2.6 Designing user interface 62](#_96qbitbledgv)

[**Chapter 3 - Test and review settings 67**](#_cqgc6jtyrnsy)

[3.1 Technology 67](#_m8e5y3f06fht)

[1 - Next.js 67](#_irrk11q92h47)

[2- Mantine 67](#_plb0fcmdfcwt)

[3-Typescript 67](#_d0vn41pt4gif)

[4 - tRPC 68](#_k8uswc1gj8lq)

[5 - Prisma 68](#_6olymfh8tli7)

[6 - NextAuth.js 69](#_ish7uxcsy4mb)

[7 - React 69](#_3et0wty49cth)

[8 - Firebase 69](#_a9jnc58eqgjs)

[9 - MySQL 70](#_i5om1j4glat)

[10 - Stripe 70](#_kqvhtnr3xuji)

[3.2. Installing system 71](#_hwcvyfvjd2h3)

[3.2.1. General setting 71](#_xpbawws9pau)

[3.2.2. Setting up client-side 72](#_iib5mrw9dupj)

[3.2.3. Setting up server-side 72](#_q6s35jzffjf4)

[3.3. Testing and evaluation. 73](#_np9bpzwxmk0z)

[3.3.1. Test environment 73](#_22iwxljxme0w)

[3.3.2. Test data and results 73](#_mjytmn3aht3t)

[3.3.2.1. Test data 73](#_bh8y8jygpkko)

[3.3.2.2. Results 73](#_rx2c32fk91te)

[**Chapter 4: Conclusion and Future Development Direction: 77**](#_mbvhvovk8d7i)

[4.1. Achievements 77](#_sfwj77t7ujl9)

[4.2. Limit 77](#_3jb5ns73nll1)

[**Chapter 5: References 78**](#_nyuwp7p848ft)

[**Appendix 79**](#_oew2a5ye6bzc)

[5.1: Update profile 79](#_70b8k3voggwz)

[5.1.1: Use case 79](#_1bh8wdropv4t)

[5.1.2: Activity diagram 80](#_bfx8t4egjjnv)

[5.1.3: Sequence diagram 80](#_4snm3vm3c43a)

[5.2: Add payment method 81](#_wny6xd9x57z)

[5.2.1: Use case 81](#_jjiklxs117q3)

[5.2.2: Activity diagram 82](#_tgawz8avnyxe)

[5.2.3: Sequence diagram 82](#_9ykibcgru7jg)

[5.3: Report user 83](#_crdv2xn9ust2)

[5.3.1: Use case 83](#_eadeq3y5ra64)

[5.3.2: Activity diagram 84](#_n0se54hndy3d)

[5.3.3: Sequence diagram 84](#_89hg1s7xkmw8)

[5.4: View accommodation details 85](#_zf0mcvk5jvs4)

[5.4.3.1. Use case 85](#_5ay4cpue9dva)

[2.2.2.2.2: Activity diagram 85](#_hwr92kmmyb24)

[2.2.2.2.3: Sequence diagram 86](#_czgd9p783ske)

[**Software testing 87**](#_oew2a5ye6bzc)

[6.1: Registration and Login Functionality 88](#_70b8k3voggwz)

[6.2: Search and Filtering Property Listings 89](#_70b8k3voggwz)

[6.3: Property Details and Booking Functionality 90](#_70b8k3voggwz)

[6.4: Scalability Testing 91](#_70b8k3voggwz)

# 

# Abbreviations

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Full name** | **Meaning** |
| Frameworks | Frameworks | Is a pre-built code framework that makes it easier for us in the software development process |
| API | Application Programming Interface | Application programming interface |
| REST | REpresentational State Transfer | An architecture that transforms data structures |
| JSON | JavaScript Object Notation | JavaScript Object Notation |
| User | User | The user of the application can be an organization or an individual |
| Admin | Administrators | The user with the highest authority in the application |

# 

# 

# Introduction

## 1. Motivation

- In the era of explosive development of information technology, which has become a dominant factor in the daily lives of people, the emergence of strong technological applications, economic and shared services has followed. Uber and Grab are the clearest examples of this trend. Similar to Uber and Grab, in the restaurant industry, Rentropolis (Let's book a room together) is a website designed for renters and landlords looking for rooms.

This is the Problem Statement for the Rentropolis system, which connects renters and landlords through a booking website. The name Rentropolis is a combination of "rent" and "metropolis," symbolizing the connection of renters and landlords across major cities. :

Finding a place to live or a tenant for your rental property can be time-consuming and stressful. With so many options available, organizing listings, scheduling viewings, and negotiating deals can be overwhelming. That's where Rentropolis comes in - as a comprehensive platform that connects renters with landlords and property managers, Rentropolis streamlines the entire process and makes your search easier than ever before..

One of the key benefits of using Rentropolis is the ability to search and filter through a range of properties that match your specific needs and preferences. With advanced search options such as location, price range, number of bedrooms, and more, finding the perfect rental property has never been easier. Additionally, Rentropolis provides a range of features to help landlords and property managers find suitable tenants, including detailed property listings, applicant screening tools, and communication options.

Another important advantage of using Rentropolis is the peace of mind that comes with knowing that you're working with a reliable and verified platform. All properties and landlords on Rentropolis are thoroughly checked and verified to ensure a safe and trustworthy rental experience for all parties involved. Whether you're a renter searching for your dream home or a landlord looking for the perfect tenant, Rentropolis provides a comprehensive solution for all your rental needs.

In summary, Rentropolis is an essential tool for anyone looking for a home or wanting to rent out their property. It saves time and effort by providing a convenient platform to search and connect tenants with landlords and property managers. With advanced search features, thoroughly verified properties, and supportive features for searching and negotiating, Rentropolis is an excellent choice for anyone who wants to rent or lease their property.

In addition, Rentropolis also provides an easy and convenient experience for managing transactions related to property rentals. With the feature of rent contract management, payments, and direct message exchanges on the platform, Rentropolis helps to simplify and make the property rental process more efficient than ever before.

Therefore, if you are seeking a platform to connect tenants and landlords or looking to rent out your own property, Rentropolis is an excellent choice. Join us now and experience a more convenient and trustworthy property rental experience than ever before.

## 2 - Research objective

**-** The overall goal of the project is to build an integrated platform between the host and customers, in order to provide convenience and connectivity for all parties involved. To achieve this goal, specific objectives have been set as follows:

* Build a system that connects renters and landlords, allowing users to easily plan and experience their trips, or conveniently manage their living spaces.
* Developing a website that is useful, user-friendly, and secure for its users.
* Enhancing interaction and communication between hosts and customers, creating a civilized travel community and sharing experiences with each other.
* Developing reliable partnerships to enable the platform to quickly and effectively meet the needs and demands of customers.

## 3 - Area of research

Developing and deploying a web platform that connects renters and landlords using Next.js and TypeScript technology. Specifically, researching and applying various techniques, technologies, and design aspects to develop an efficient and functional website, including:

* Designing an attractive and user-friendly user interface, optimizing user experience, and increasing user engagement on the website.
* Create search and filtering features for renters and landlords to easily find suitable properties that meet their needs.
* Integrating payment and secure authentication features to ensure that transactions between renters and landlords are carried out safely and securely.
* Using TypeScript to ensure that your code is type-safe and easier to maintain.
* Use Next.js to build your website, as it provides better performance and scalability compared to other frameworks.

## 4 -Research content

-Developing a housing quality rating system: Researching the criteria for assessing housing quality and designing a rating system to ensure the quality of housing meets the needs and expectations of customers.

-Optimizing the booking process: Analyzing and improving the booking process to ensure that it is simple and fast, thereby enhancing the user experience and attracting new customers.

-Develop new features: Research and propose new features to enhance the customer experience, including search, result filtering, room selection, payment, and feedback on service quality.

-Managing user information and accounts: Building a system for managing user information and accounts to ensure the security and accuracy of information, thereby increasing user trust.

# Chapter 1 - Problem

## 1.1 Description of problem

*Requests from non-logged-in users.*

* Login / Register account
* Search for houses/rooms for rent.
  + Users can search for houses/rooms based on the destination, check-in date, check-out date, number of guests, price per night, type of accommodation, and type of services provided at the accommodation.
* View details of the accommodation.
* View detailed information of the host.

*Requests of logged-in users.*

* Account settings.
  + Users are allowed to change their personal information as well as their account passwords, except for their email address (because the email address is used as equivalent to the account name for login).
* Notification
  + Users will receive notifications related to their booking requests.

*Requests from renters:*

* Search for houses/rooms.
  + A tenant can quickly and accurately search for the desired house/room.
* Managing favorite accommodation list
  + The renter can view their favorite list, as well as add, edit, and delete their favorite places to stay.
* Booking
  + A renter fills in the necessary information including check-in date, check-out date, number of guests, and the system will calculate and provide the total cost for the user immediately afterwards.
* Manage trips
  + A renter can keep track of past trips, upcoming trips, and current trips.
* Guests can leave comments and reviews for the accommodations they stayed in after each trip.

*Request from hosts/property owners*

* Manage booking requests
  + A host has the right to view the list of booking requests sent to them.
  + The host can approve or reject any booking request in the list.
* View revenue statistics.
* Managing transaction history.
  + The host has the right to view a list of past, current, and upcoming transactions.
* Setting up a host profile.

## 1.2 Similar systems

**Agoda[1]:**

\*Advantages:

Provides a wide range of hotel room options globally, from rental apartments to luxury villas, catering to various needs and budgets of travelers.

Has a user review feature that allows users to evaluate the hotel's quality before booking.

Offers promotional programs and discounts to help users save costs when booking hotel rooms.

Provides 24/7 customer support through the customer service center.

\*Disadvantages:

Service fees for customers may be higher than some competing rivals.

The quality of hotel rooms and services may not be fully guaranteed in some cases.

**Traveloka[2]:**

\*Advantages:

Ability to book hotels, flights, transportation, and other services on a single platform.

Offer many promotions and discounts for users.

Have a smart search function that helps users easily find what they need.

Organize special events and discount programs on holidays and other special occasions.

\*Disadvantages:

May have difficulty finding special hotels or resorts in English.

Information about hotel rooms and services may not be 100% accurate.

System errors or delays may occur during payment or booking confirmation.

## 1.3 The difference between Rentropolis and similar websites in Vietnam

- Rentropolis is a website that connects renters and landlords with a primary focus on home-sharing. Unlike popular hotel or resort booking websites in Vietnam, Rentropolis focuses on apartments, private homes, homestays, vacation rentals, and other similar accommodation types.

- With Rentropolis, renters can find a convenient and cost-effective place to stay, as well as experience local living and culture. The website also offers renters a variety of accommodation options with different prices and amenities, from cozy apartments to luxurious villas.

- For landlords, Rentropolis provides a simple and convenient way to promote and rent out their properties. The website helps landlords reach a large pool of potential customers, especially home-sharing enthusiasts.

- In summary, Rentropolis brings benefits to both renters and landlords when searching for accommodation options. It is a novel choice that meets the needs of customers and the community in effectively utilizing and utilizing lodging resources.

## 1.4 Problem details:

### 1.4.1: Property management:

* Allow creating, editing, and deleting rooms in the hotel
* Allow searching for rooms based on criteria such as price, area, floor level, number of beds, facilities
* Display detailed information of each room such as images, prices, facilities, descriptions.

### 1.4.2: Booking :

* Allow customers to book rooms based on criteria such as room type, check-in date, and check-out date.
* Display booking information for each customer such as room number, check-in date, check-out date, price, and booking status.

### 1.4.3: Payment:

* Allow customers to pay using various payment methods such as cash, credit cards, and e-wallets.
* Automatically calculate the amount due based on the number of days stayed, room type, and price.

### 1.4.4: Notification:

* Send notifications to customers about hotel activities such as successful booking, successful payment, special events at the hotel.

### 1.4.5: The feature of messaging between the room owner and the customer.

* Allow customers and room owners to send messages to each other to exchange information related to booking, payment, and other requests.

### 1.4.6: Accommodation management and occupancy status management:

* The system will manage information about accommodations, including the number of rooms, room types, room rates, room status (available/fully booked/under maintenance), and the location of the rooms within the hotel.
* When a customer books a room, the system will check and update the room's status. If the room is available and the customer has paid a deposit, the system will mark the room as booked and change its status to "booked". If the room has already been booked or is under maintenance, the system will inform the customer that the room is not available.
* The system will also allow accommodation management to perform operations to update room information, including changing room rates, updating room status, adding new rooms, and deleting rooms from the system.

### 1.4.7: Revenue management:

* The system will display the monthly revenue earned by the owner throughout the year.

### 1.4.8: Search place:

* The search feature: The system allows customers to search for accommodations based on criteria such as location, price, type of accommodation, amenities, number of guests, and booking dates.

### 1.4.9: Booking management:

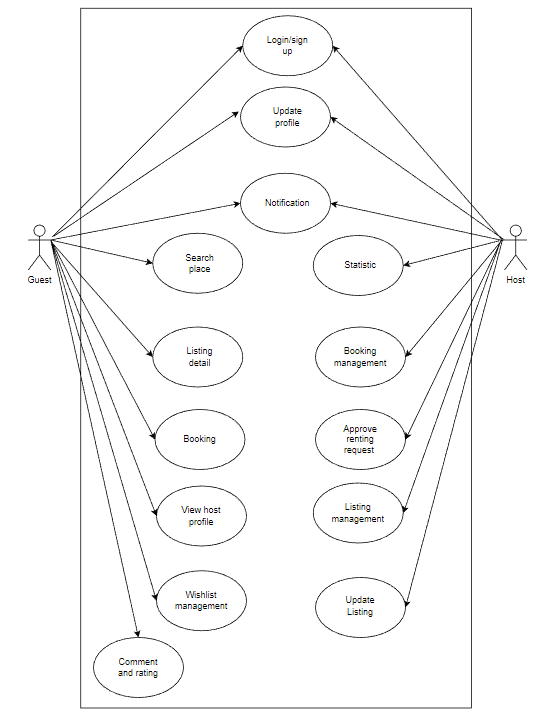
* The system will retain the information of confirmed accommodation bookings and pending bookings.
* When a new accommodation booking is received, the system will automatically send a notification to the manager for confirmation and recording.
* The manager needs to check the feasibility of the booking (availability, suitable booking time...) and confirm the booking. If the booking is not feasible, the manager will inform the customer of the issue and propose alternative solutions.
* After the manager confirms the booking, the system will send a confirmation notification to the customer along with the details of the booking and payment instructions.

### 1.4.9: Review

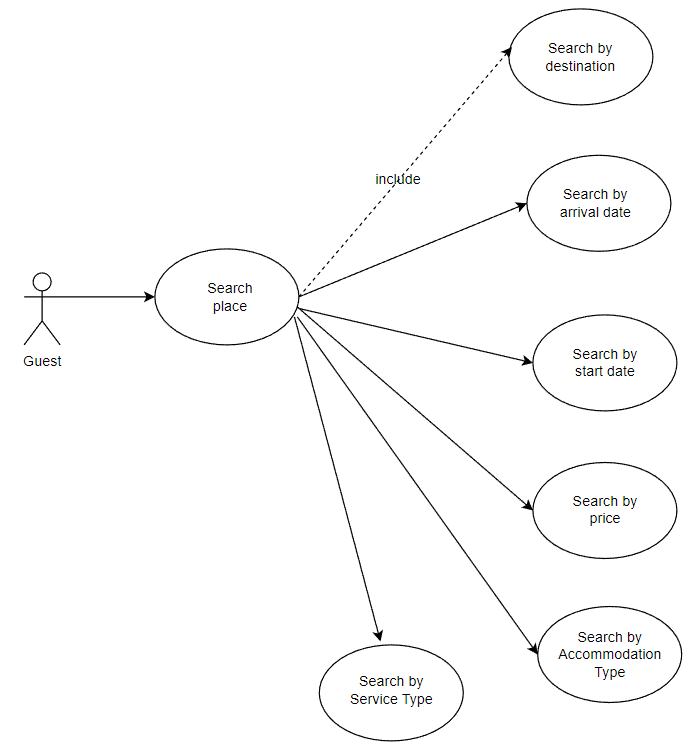
* The review feature allows users to rate, comment, and share their experience using the services of the accommodation.

# Chapter 2 - System analysis and design

## 2.1 Use case diagram

**

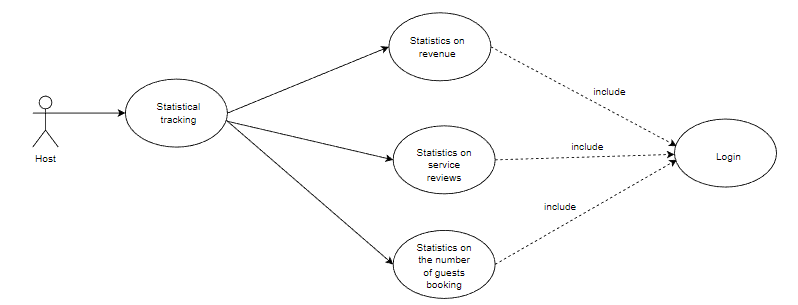
*Figure 2.1 System overview*

**

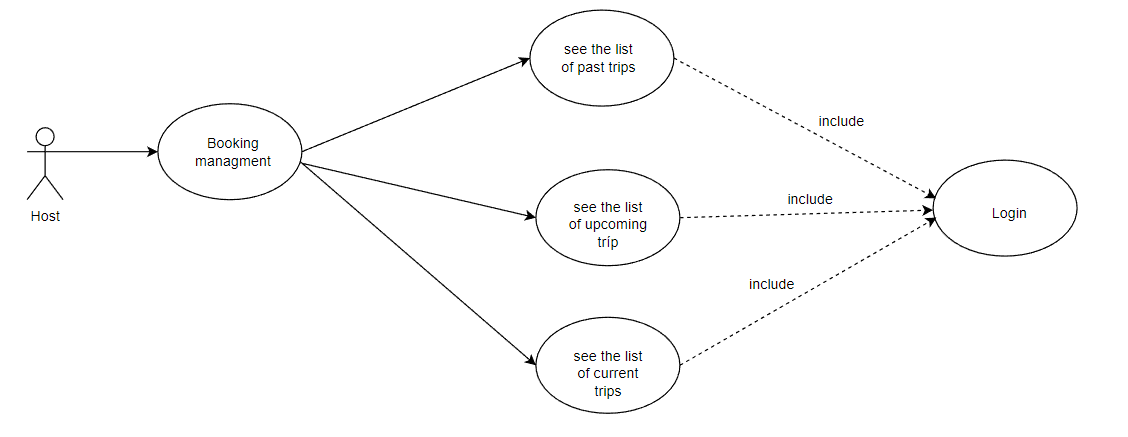
*Figure 2.2Search for a place*

**

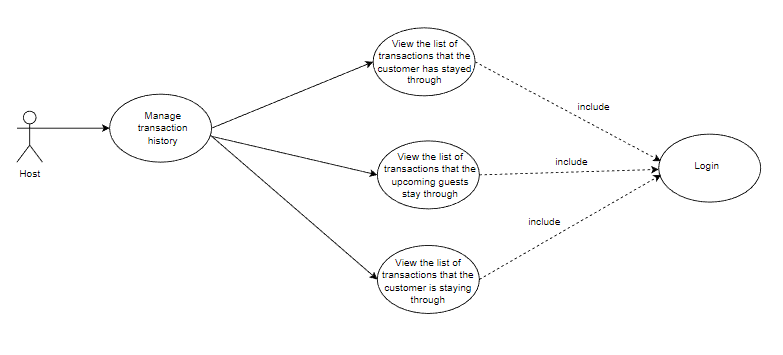
*Figure 2.3 Book a house/room to rent*

**

*Figure 2.4 Statistical tracking*

**

*Figure 2.5 Booking Management*

**

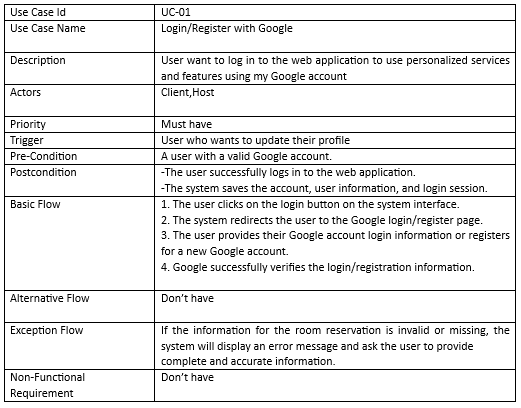
*Figure 2.6 Manage transaction history*

## 2.2 Use case analysis

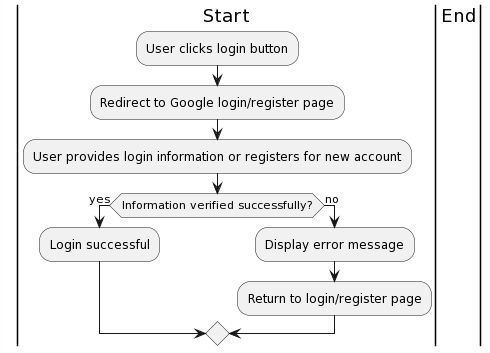
### 2.2.1 : Common

### 2.2.1.1: "Sign in/Sign up with Google"

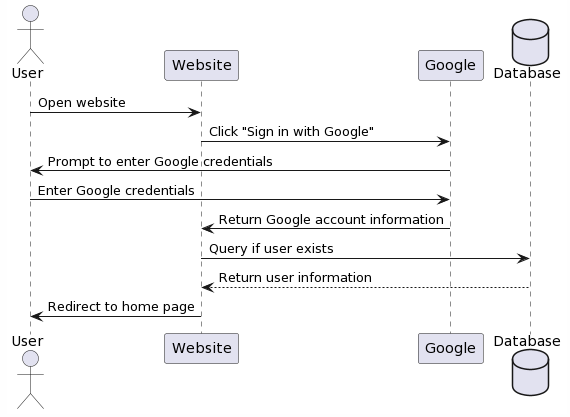
##### 2.2.1.1.1: Use case



##### 2.2.1.1.2: Activity diagram

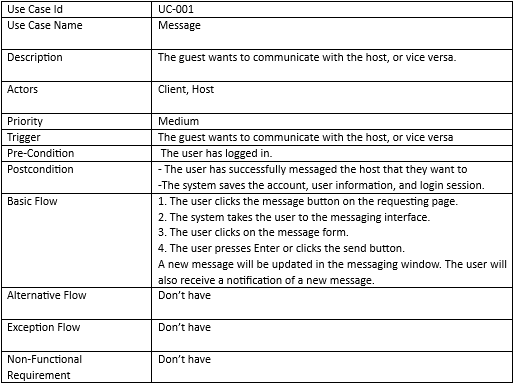


##### 2.2.1.1.3. Sequence diagram

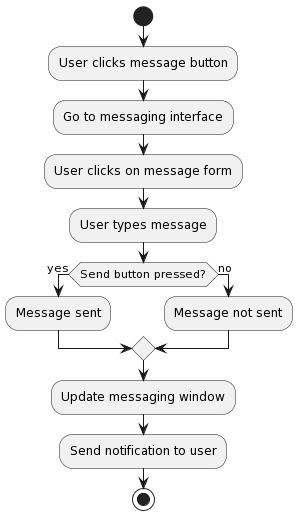


#### 2.2.1.5: Messaging

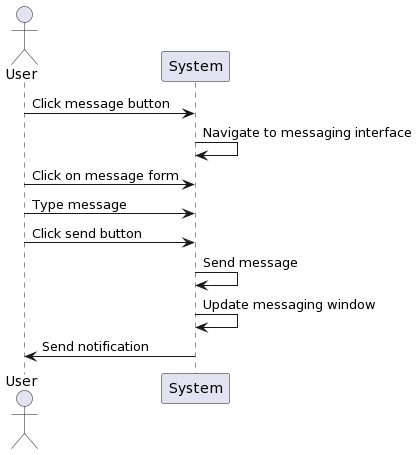
##### 2.2.1.5.1: Use case



##### 2.2.1.5.2: Activity diagram

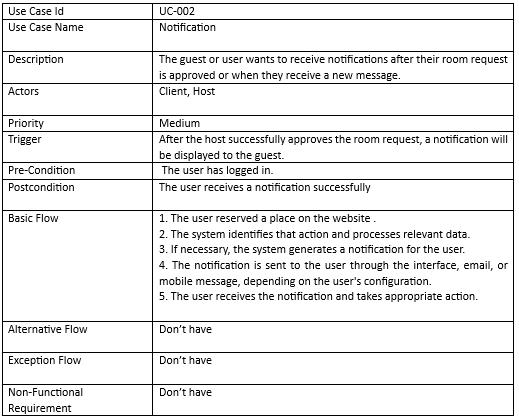


##### 2.2.1.5.3: Sequence diagram

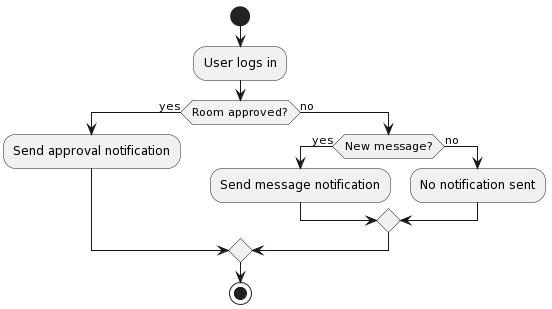


#### 2.2.1.6: Notification

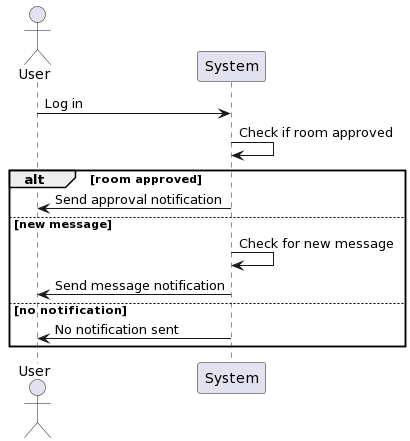
##### 2.2.1.6.1: Use case



##### 2.2.1.6.2: Activity diagram



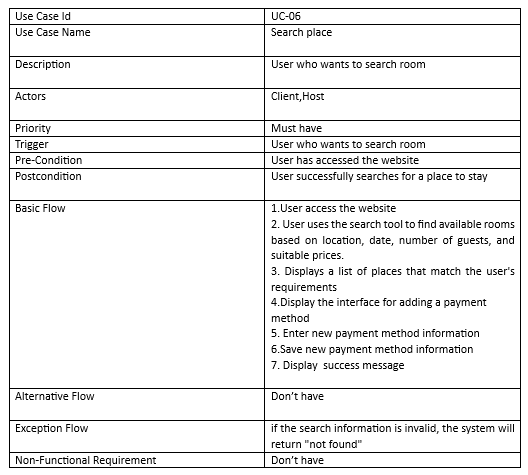
##### 2.2.1.6.3: Sequence diagram



### 2.2.2: Client

#### 2.2.2.1: Search place

##### 2.2.2.1.1: Use case

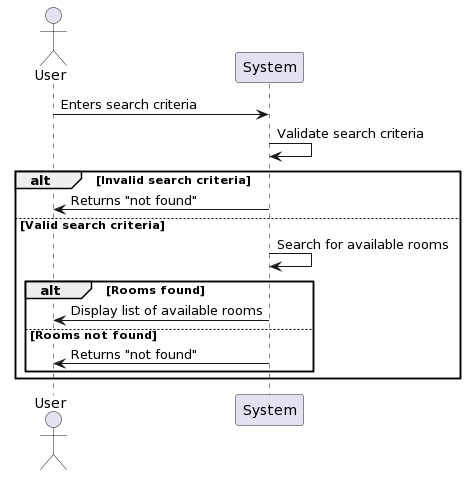


##### 2.2.2.1.2: Activity diagram

##### 

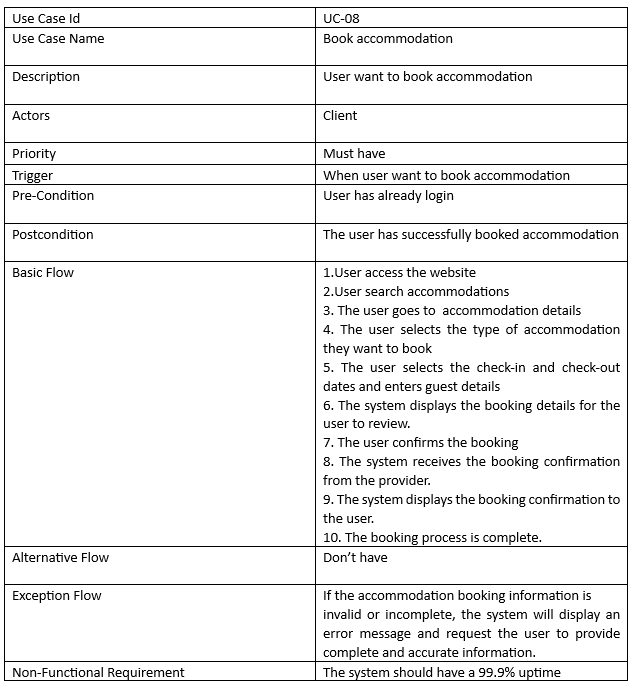
Biểu đồ hoạt động ca tìm kiếm

##### 2.2.2.1.3: Sequence diagram



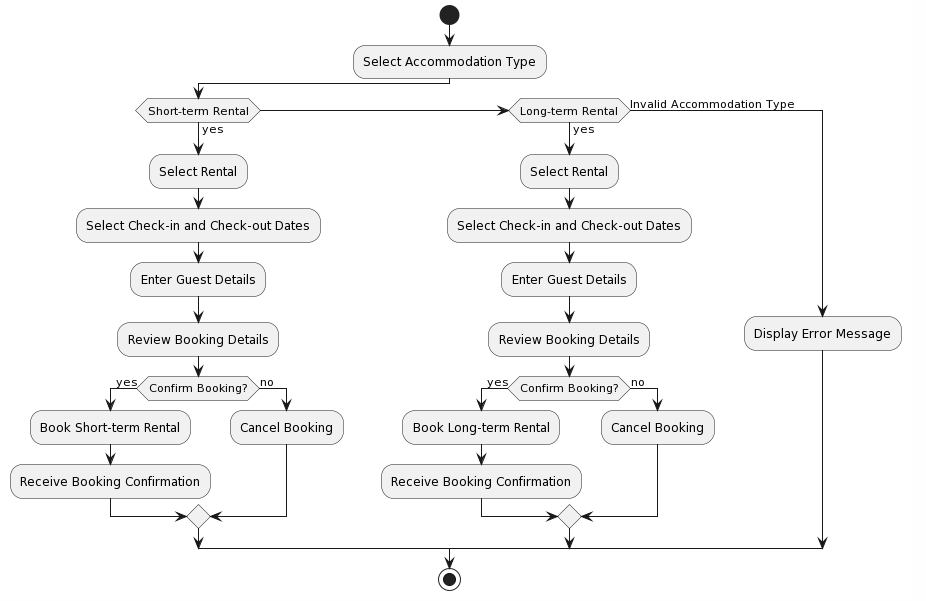
#### 2.2.2.2: Booking accommodation

##### 2.2.2.2.1: Use case



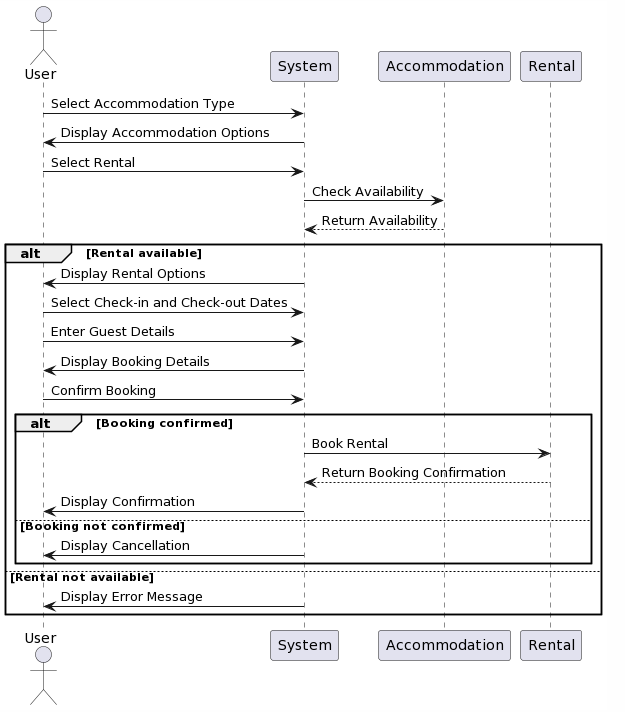
##### 2.2.2.2.2: Activity diagram

##### 



Biểu đồ hoạt động ca đặt phòng

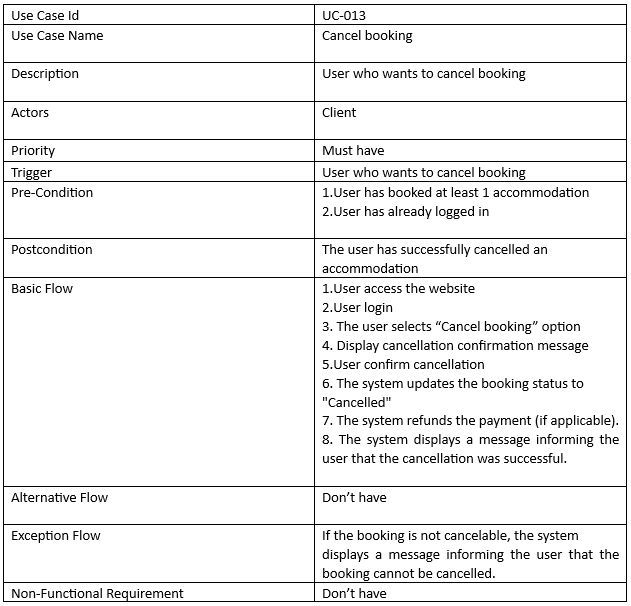
##### 2.2.2.2.3: Sequence diagram



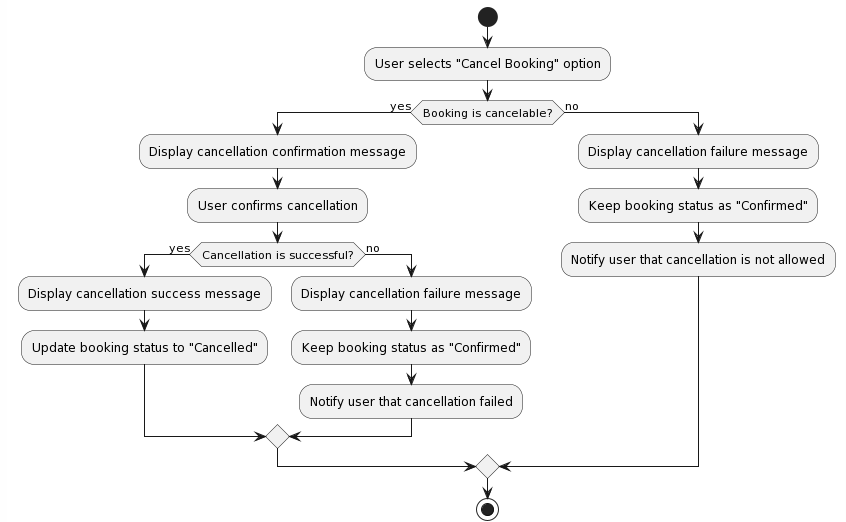
#### 2.2.2.3: Booking management

##### 2.2.2.3.1 Cancel booking

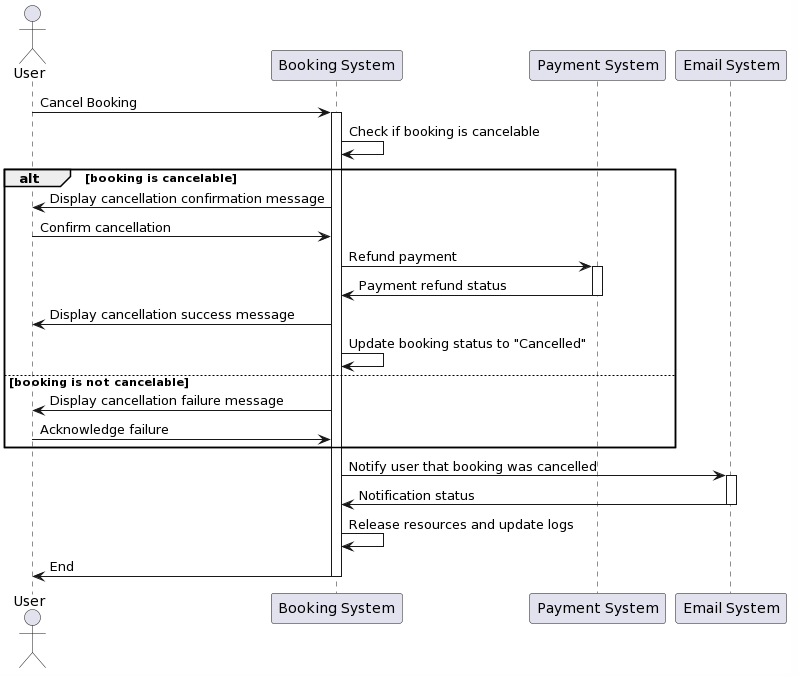
###### 2.2.2.3.1.1: Use case



###### 2.2.2.3.1.2: Activity diagram

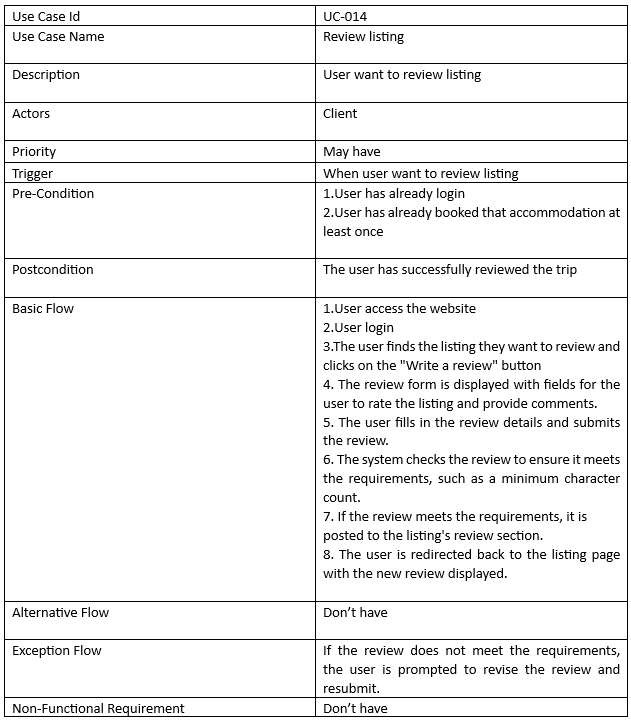


###### 2.2.2.3.1.3: Sequence diagram

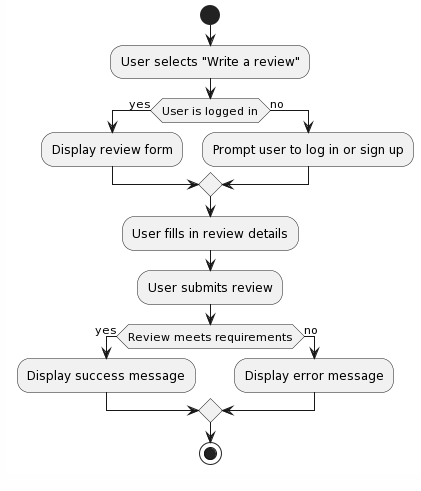


#### 2.2.2.4: Review

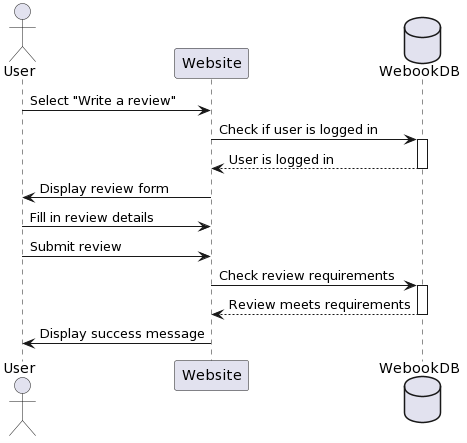
##### 2.2.2.4.1: Use case

****

###### 2.2.2.4.2:Activity diagram

****

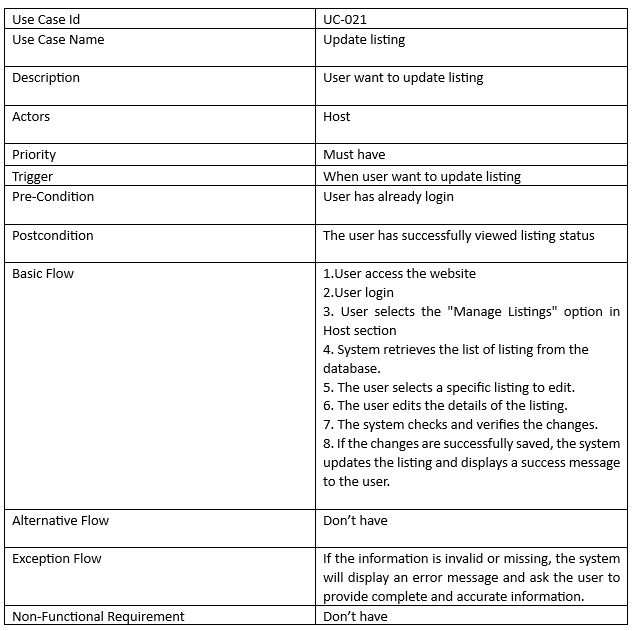
###### 2.2.2.4.3: Sequence diagram

****

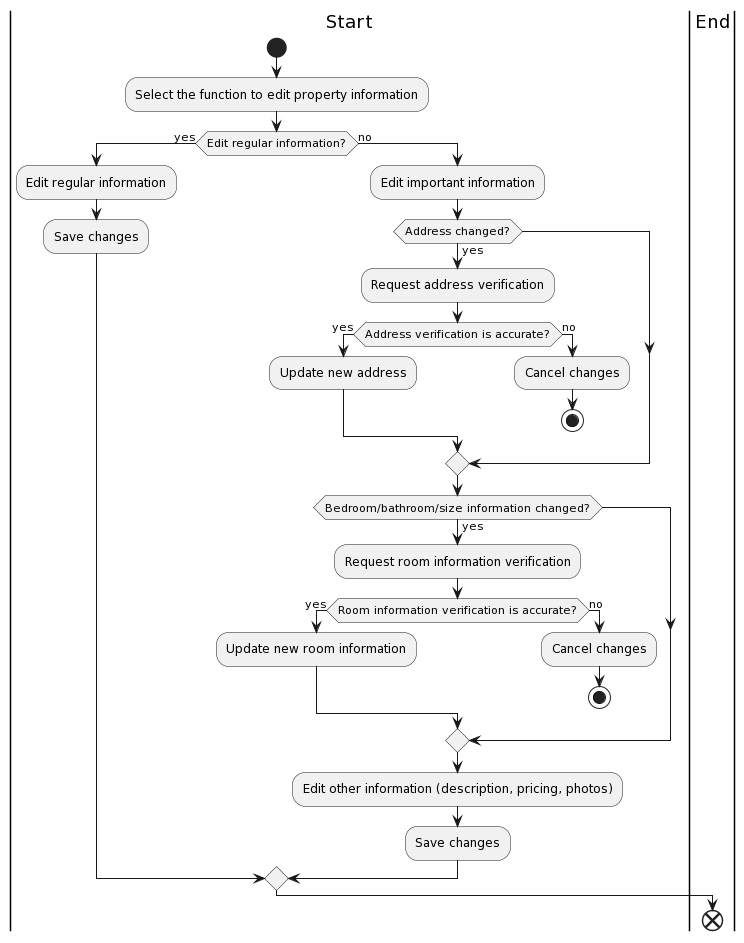
### 2.2.3: Host

#### 2.2.3.1: Listing management

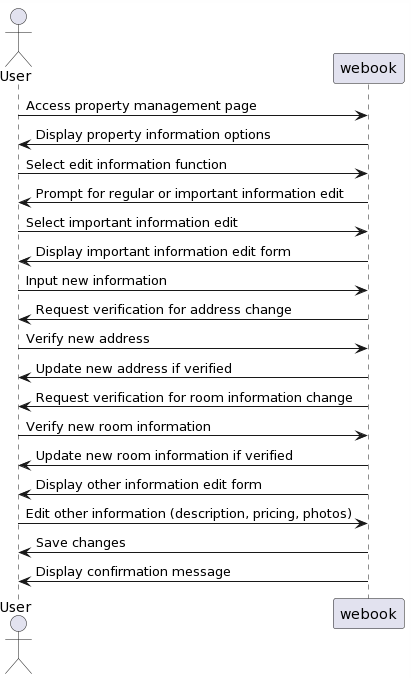
##### 2.2.3.1.2.1: Use case



##### 2.2.3.1.2.2: Activity diagram



##### 2.2.3.1.2.3: Sequence diagram

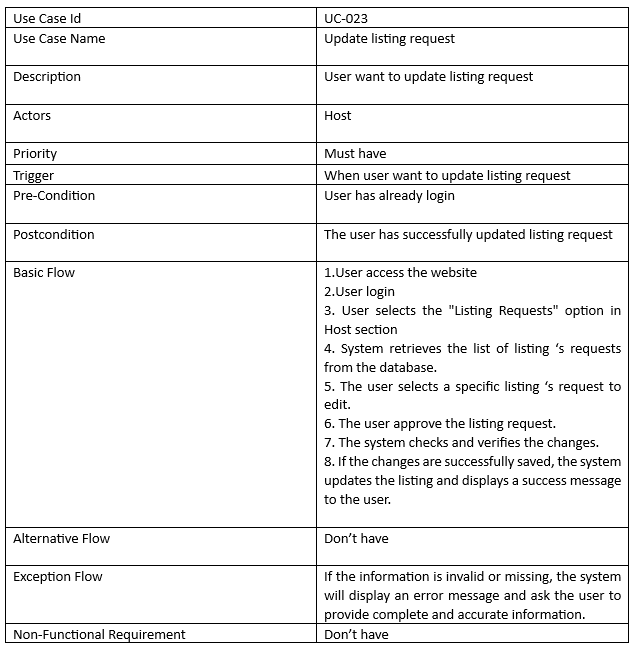


###### 

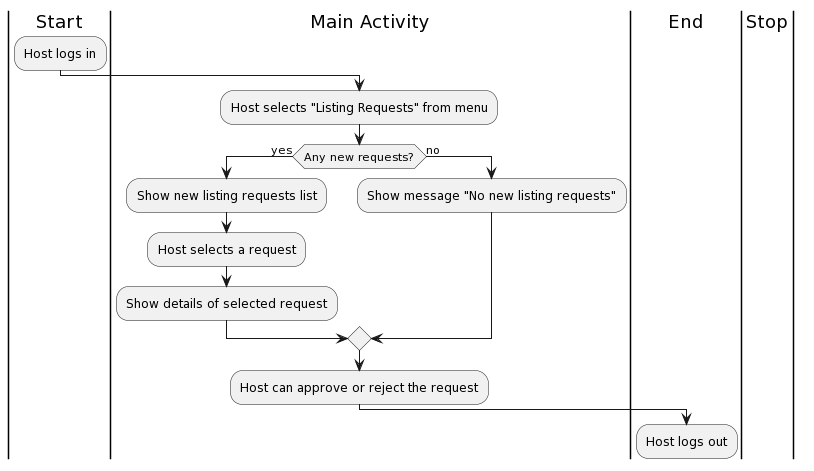
#### 2.2.3.2: Listing request management

##### 2.2.3.2.1.2 : Approve

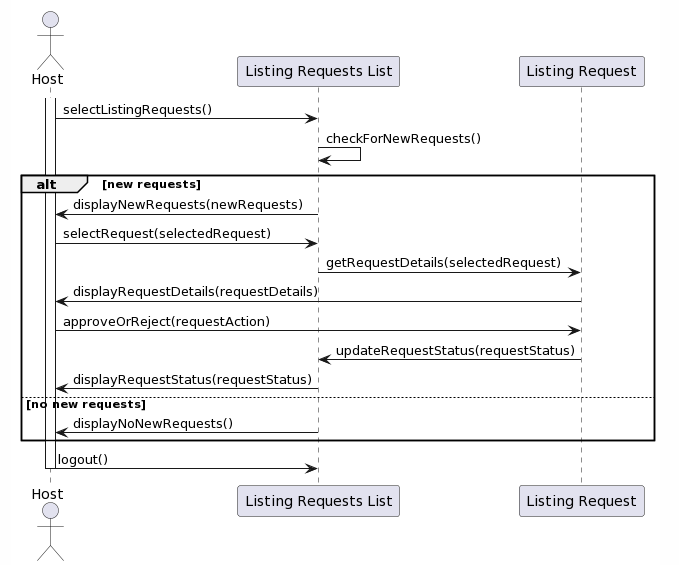
###### 2.2.3.2.1.2.1: Use case



###### 2.2.3.2.1.2.2: Activity diagram

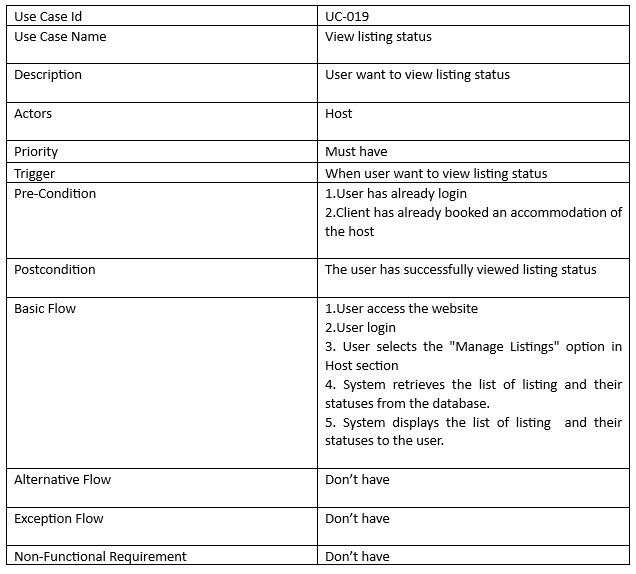


###### 2.2.3.2.1.2.3: Sequence diagram

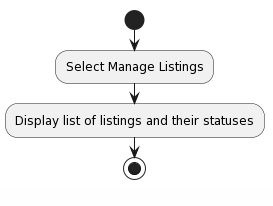


##### 2.2.3.2.4: Already canceled

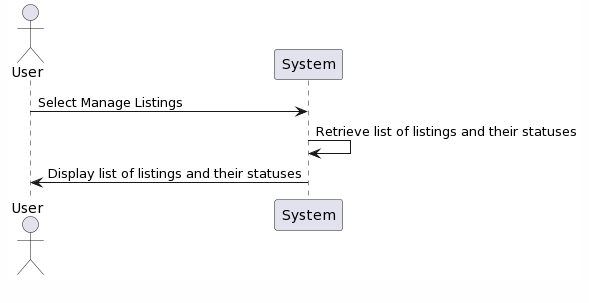
###### 2.2.3.2.4.1: Use case

****

###### 2.2.3.2.4.2: Activity diagram

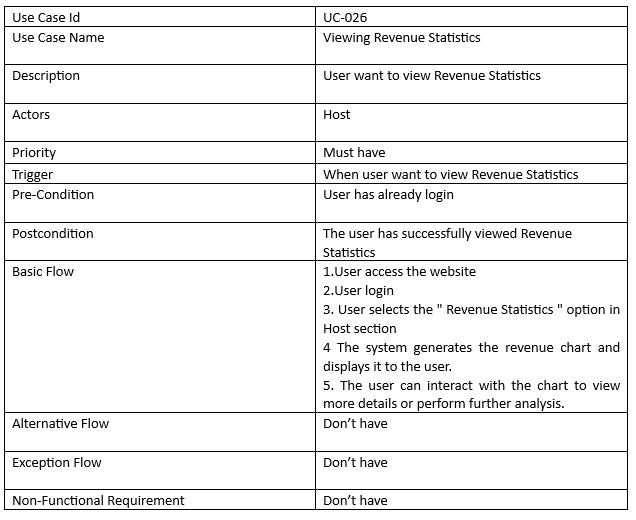


###### 2.2.3.2.4.3: Sequence diagram

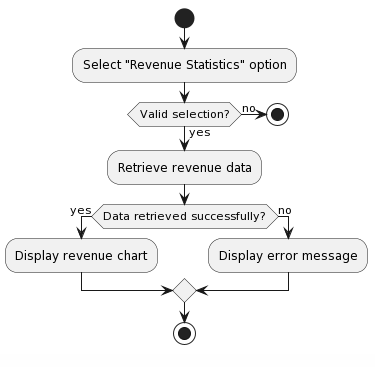


#### 2.2.3.3: Statistics

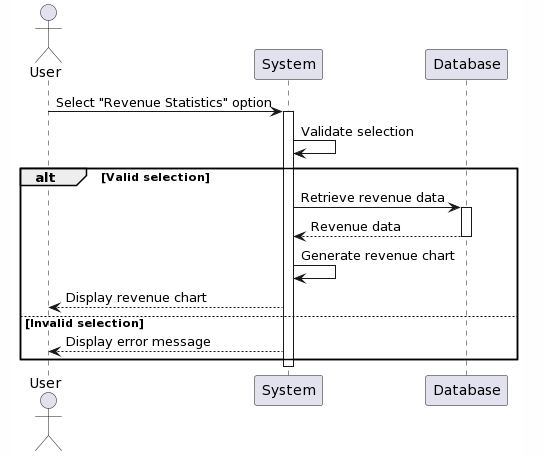
##### 2.2.3.3.1: Use case



##### 2.2.3.3.2: Activity diagram



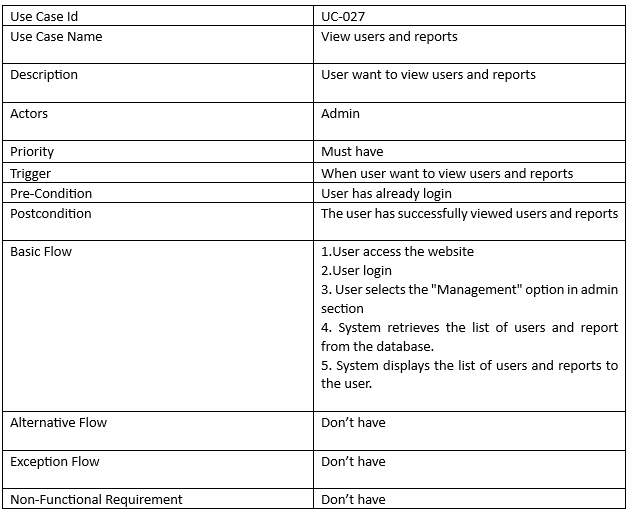
##### 2.2.3.3.3: Sequence diagram



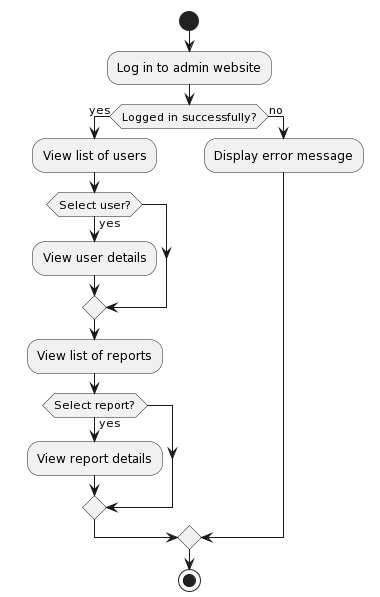
### 2.2.4: Admin

#### 2.2.4.1: View user list (and report)

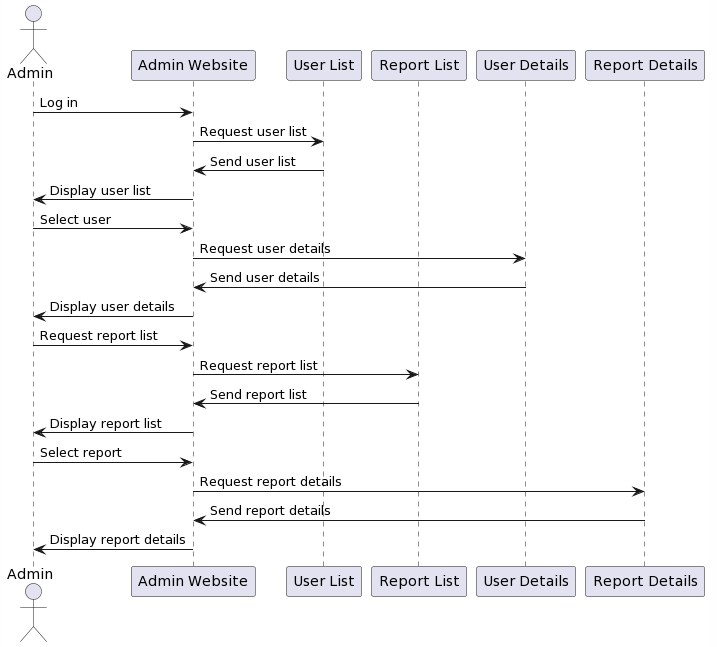
##### 2.2.4.1.1: Use case



##### 2.2.4.1.2: Activity diagram



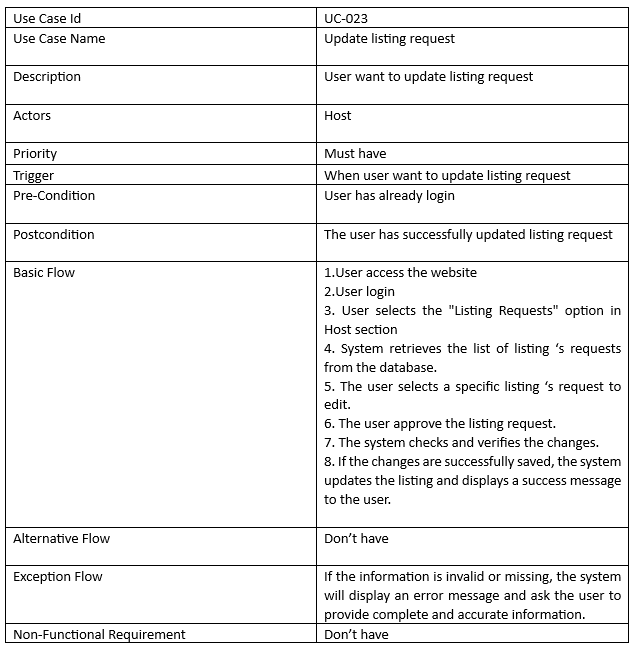
##### 2.2.4.1.3: Sequence diagram



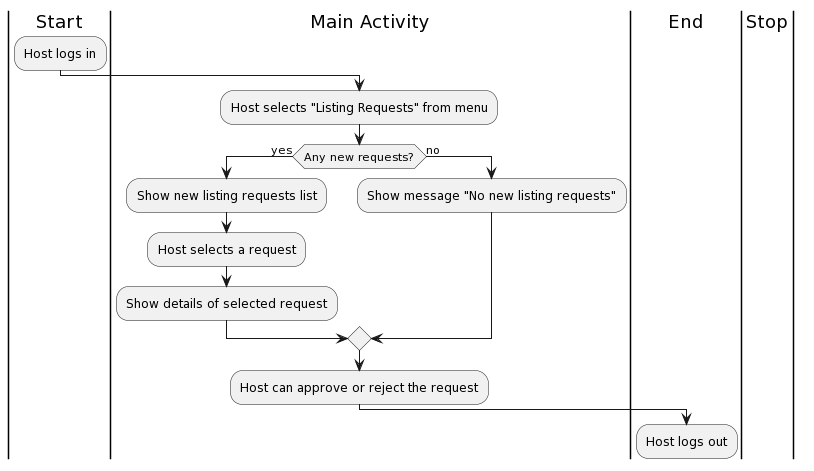
#### 2.2.4.2: Approve accommodation addition/modification requests

##### 2.2.4.2.1: use case

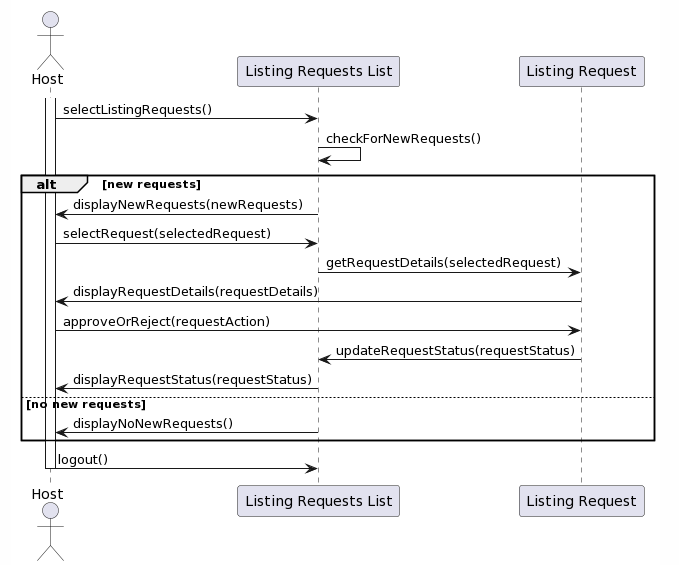
###### 



##### 2.2.4.2.2: Activity diagram



##### 2.2.4.2.3: Sequence diagram



#### 2.2.4.3: Banned user

##### 2.2.4.3.1: Use case:

### 

##### 2.2.4.3.2: Activity diagram

##### 

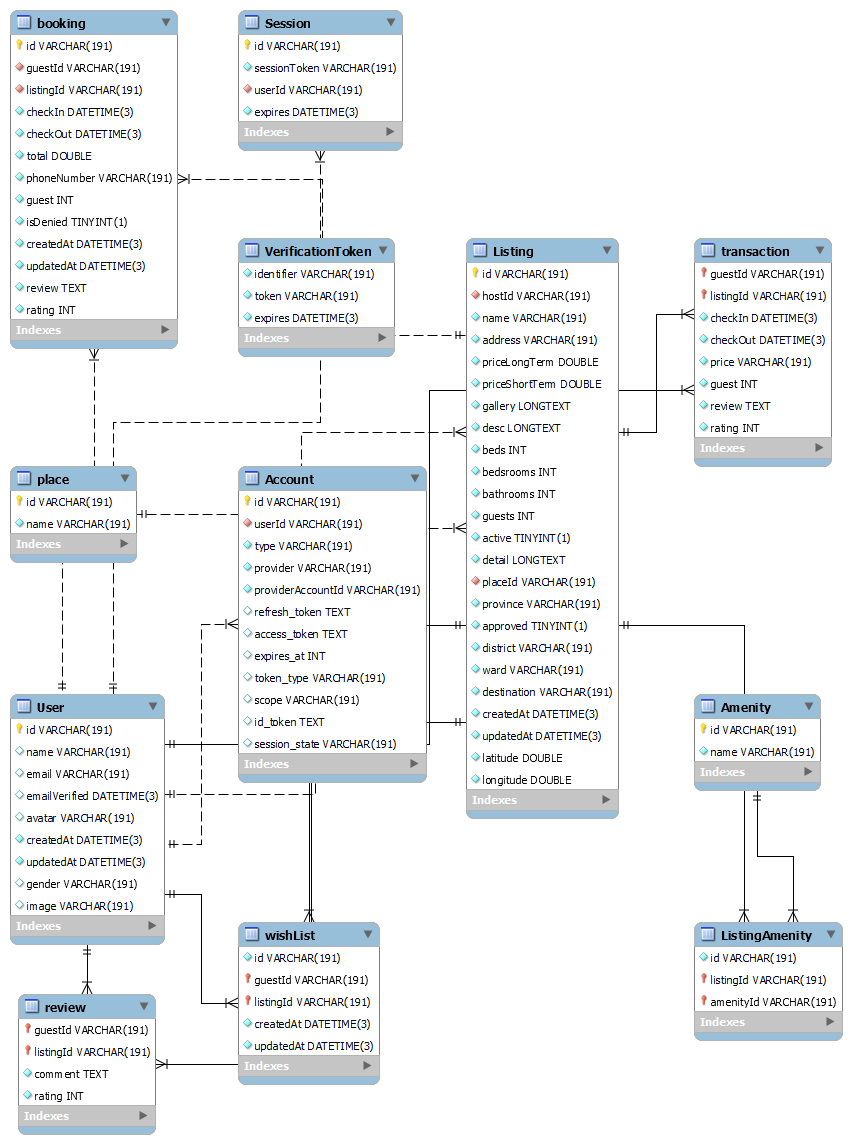
##### 

##### 2.2.4.3.3: Sequence diagram

##### 

## 2.3 Database design

### 2.3.1: The Entity Relationship Diagram (ERD)



*Figure 2.3.1 ERD diagram*

### 2.3.2: Detailed description

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| id | VARCHAR(191) | Account identifier code |
| userId | VARCHAR(191) | User ID (User Identification) |
| type | VARCHAR(191) | Type of authentication (OAuth) |
| provider | VARCHAR(191) | Authentication provider (Google) |
| providerAccountId | VARCHAR(191) | Provider Identifier |
| refresh\_token | TEXT | The token used to obtain a new access\_token |
| access\_token | TEXT | Token is used for authentication. |
| expires\_at | INT | Expiration time of access\_token. |
| token\_type | VARCHAR(191) | Bearer token authentication. |
| scope | VARCHAR(191) | The scope of a token |
| id\_token | TEXT | Token identifier/code. |
| session\_state | VARCHAR(191) | Active session status |

*Table Account*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| id | VARCHAR(191) | Session ID |
| sessionToken | VARCHAR(191) | Token for active session. |
| userId | VARCHAR(191) | User ID (User Identification) |
| expires | DATETIME(3) | Session end time |

*Bảng Session*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| identifier | VARCHAR(191) | Identifier |
| token | VARCHAR(191) | Token |
| expires | DATETIME(3) | Expire date |

*Table VerificationToken*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| id | VARCHAR(191) | User ID (User Identification) |
| name | VARCHAR(191) | Name |
| email | VARCHAR(191) | Email |
| emailVerified | DATETIME(3) | Confirmed email. |
| image | VARCHAR(191) | Profile picture |
| gender | VARCHAR(191) | Gender |
| phone | VARCHAR(191) | Phone number |
| address | VARCHAR(191) | Address |
| paymentMethod | VARCHAR(191) | Payment method |
| cardNumber | VARCHAR(191) | Card number |
| dueDateCard | VARCHAR(191) | Due date of card |
| secretNumber | VARCHAR(191) | 3 secret number of card |
| createdAt | DATETIME(3) | Create date |
| updatedAt | DATETIME(3) | Update date |

*Table User*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| id | VARCHAR(191) | Booking id |
| guestId | VARCHAR(191) | User ID (User Identification) |
| listingId | VARCHAR(191) | Listing id |
| checkIn | DATETIME(3) | Check-in time |
| checkOut | DATETIME(3) | Check-out time |
| total | FLOAT | Total cost |
| guest | INT | Total guest |
| isDenied | BOOLEAN | Is approved or not |
| createdAt | DATETIME(3) | Create date |
| updatedAt | DATETIME(3) | Update date |

*Table Booking*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| guestId | VARCHAR(191) | User ID (User Identification) |
| listingId | VARCHAR(191) | Listing id |
| checkIn | DATETIME(3) | Check-in time |
| checkOut | DATETIME(3) | Check-out time |
| price | FLOAT | Total cost |
| guest | INT | Total guest |

*Table Transaction*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| guestId | VARCHAR(191) | User ID (User Identification) |
| listingId | VARCHAR(191) | Listing id |
| comment | VARCHAR(191) | comment |
| rating | INT | Rating |

*Table Review*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| **id** | VARCHAR(191) | Wishlist id |
| guestId | VARCHAR(191) | User ID (User Identification) |
| listingId | VARCHAR(191) | Listing id |
| createdAt | DATETIME(3) | Create date |
| updatedAt | DATETIME(3) | Update date |

*Table WishList*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| **id** | VARCHAR(191) | Listing amenity id |
| listingId | VARCHAR(191) | Listing id |
| amenityId | VARCHAR(191) | Amenity id |

*Table ListingAmenity*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| **id** | VARCHAR(191) | Amenity id |
| name | VARCHAR(191) | Amenity name |

*Table Amenity*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| id | VARCHAR(191) | Listing id |
| hostId | VARCHAR(191) | Host id |
| name | VARCHAR(191) | Listing name |
| checkIn | DATETIME(3) | Check-in time |
| checkOut | DATETIME(3) | Check-out time |
| total | FLOAT | Tổng tiền |
| guest | INT | Số lượng khách |
| isDenied | BOOLEAN | Được duyệt hay chưa |
| createdAt | DATETIME(3) | Create date |
| updatedAt | DATETIME(3) | Update date |

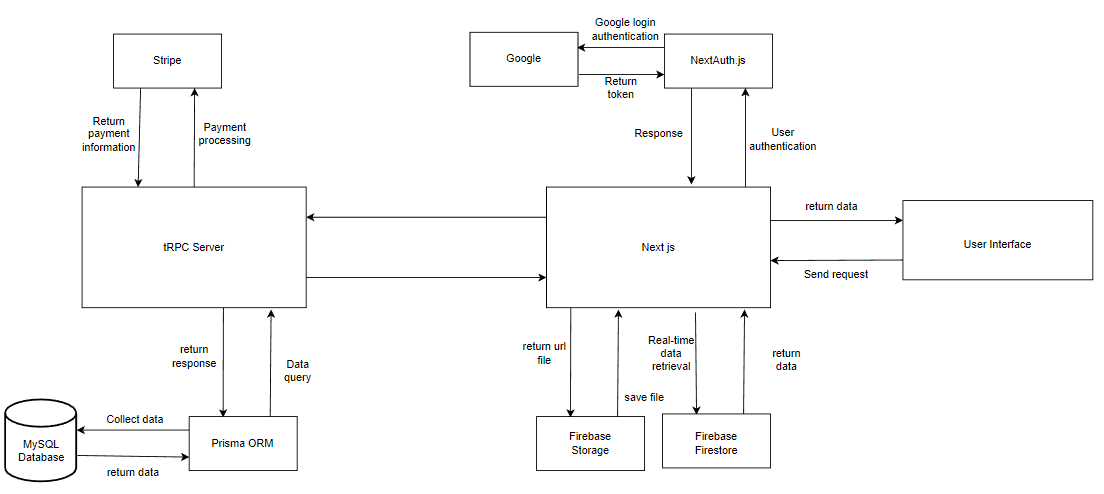
*Table Listing*

|  |  |  |
| --- | --- | --- |
| **Field name** | **type** | **Description** |
| **id** | VARCHAR(191) | Place id |
| name | VARCHAR(191) | Place name |

*Table Place*

## 2.4 Architectural design

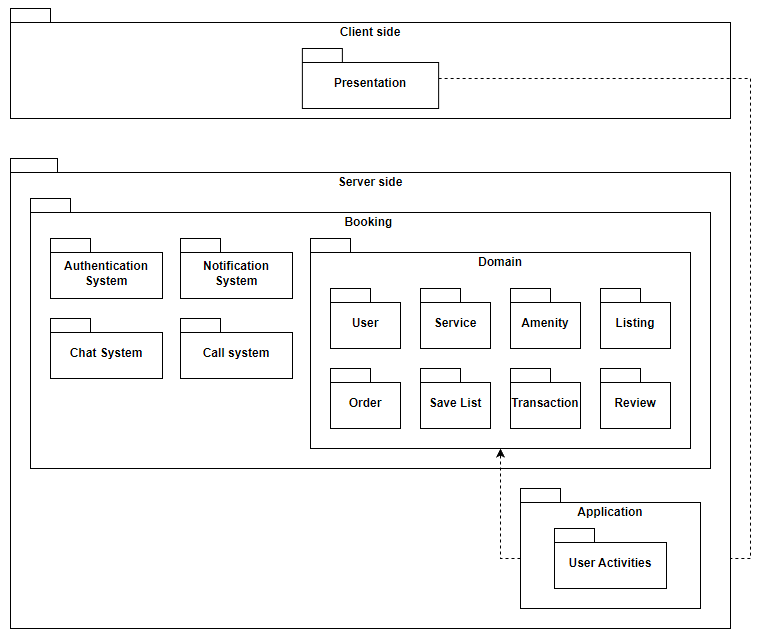
#### 



*Figure 2.4 Architectural design*

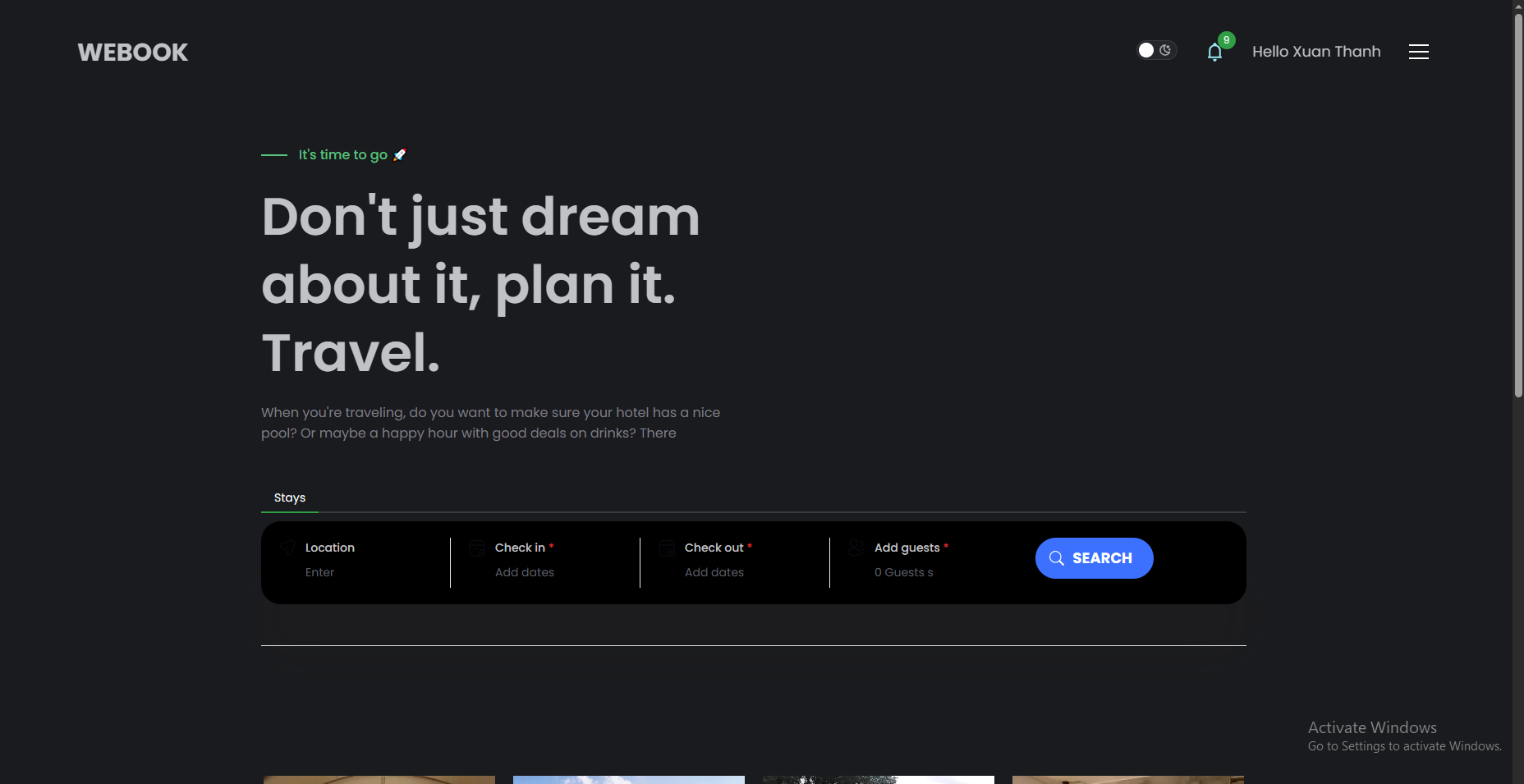
* **User interface :** Representing the client-side interface of the web application
* **Next.js:** Using server-side rendering to build web applications.
* **tRPC Server:** Handling communication between clients and servers
* **Prisma ORM:** Act as an Object-Relational Mapper (ORM) for the MySQL database.
* **MySQL Database:** Store system data
* **NextAuth.js:** The authentication library for Next.js applications
* **Google:** Google Login Provider.
* **Stripe:** Payment processing platform.
* **Firebase Storage:** file storage platforms
* **Firebase Firestore:** Real-time data storage.

## 2.5 Module design

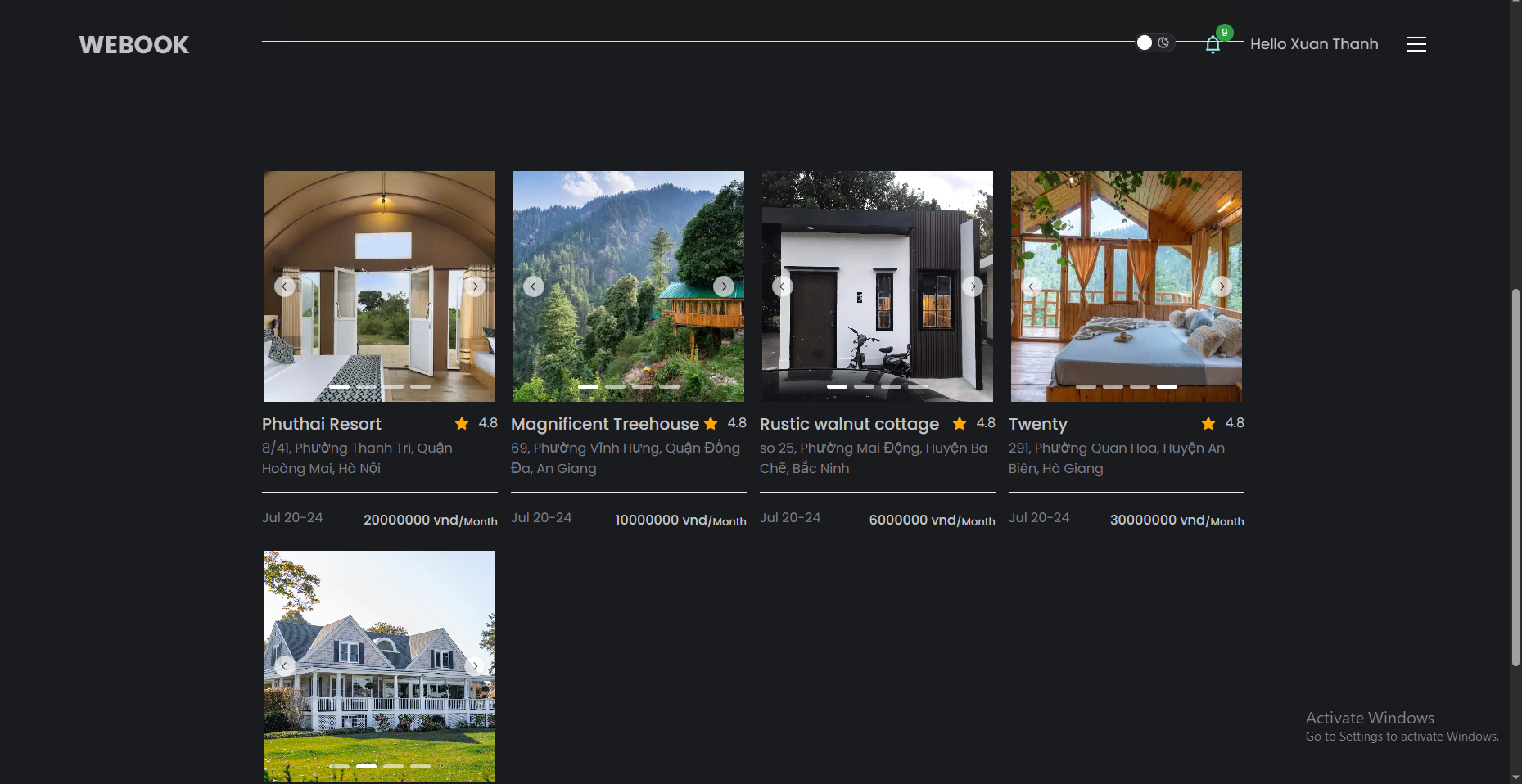


*Figure 2.5 Module design*

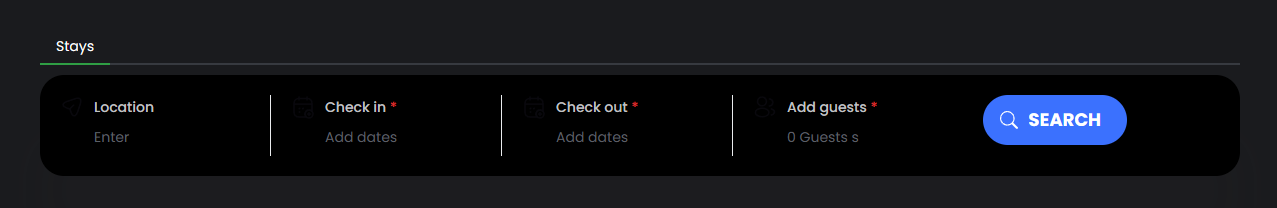
## 2.6 Designing user interface



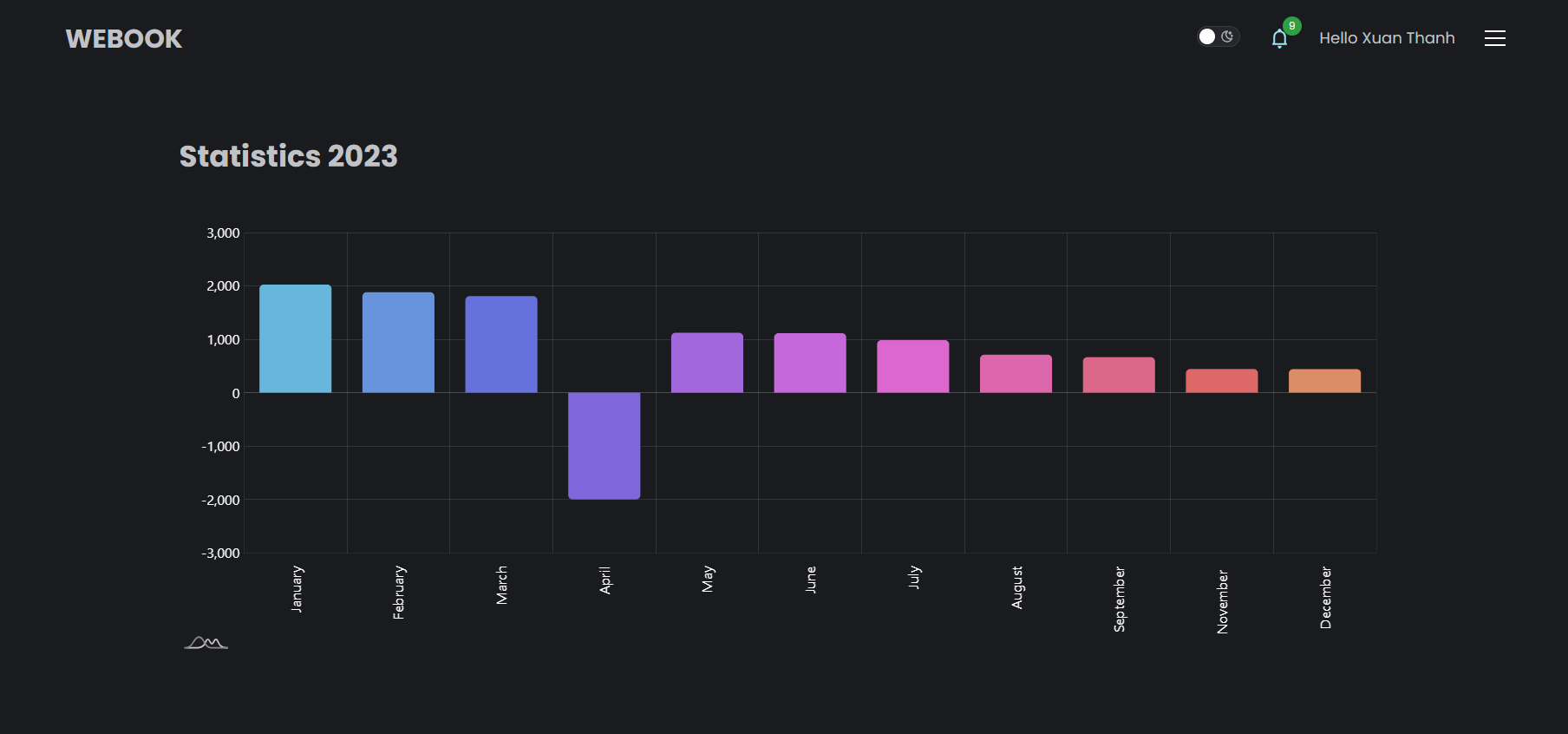
*Figure 2.6 Homepage interface*

**

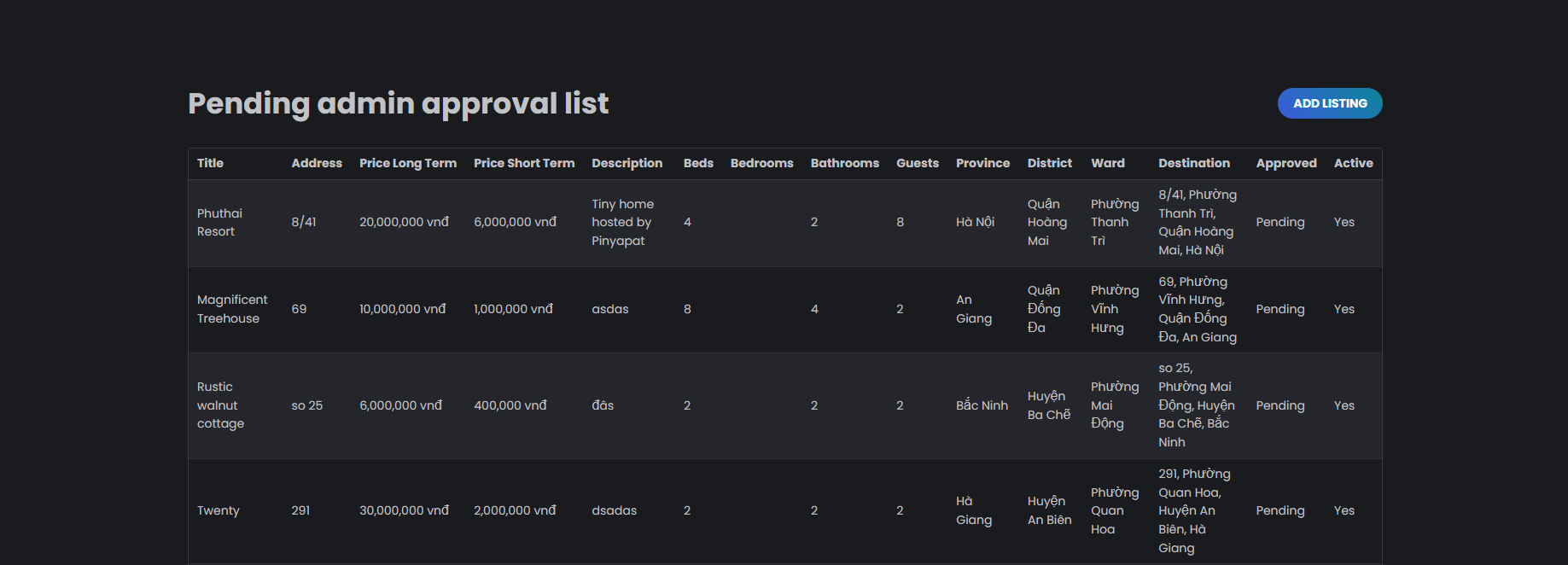
*Figure 2.7 Interface of the accommodations list*

**

*Figure 2.8 Search bar*

**

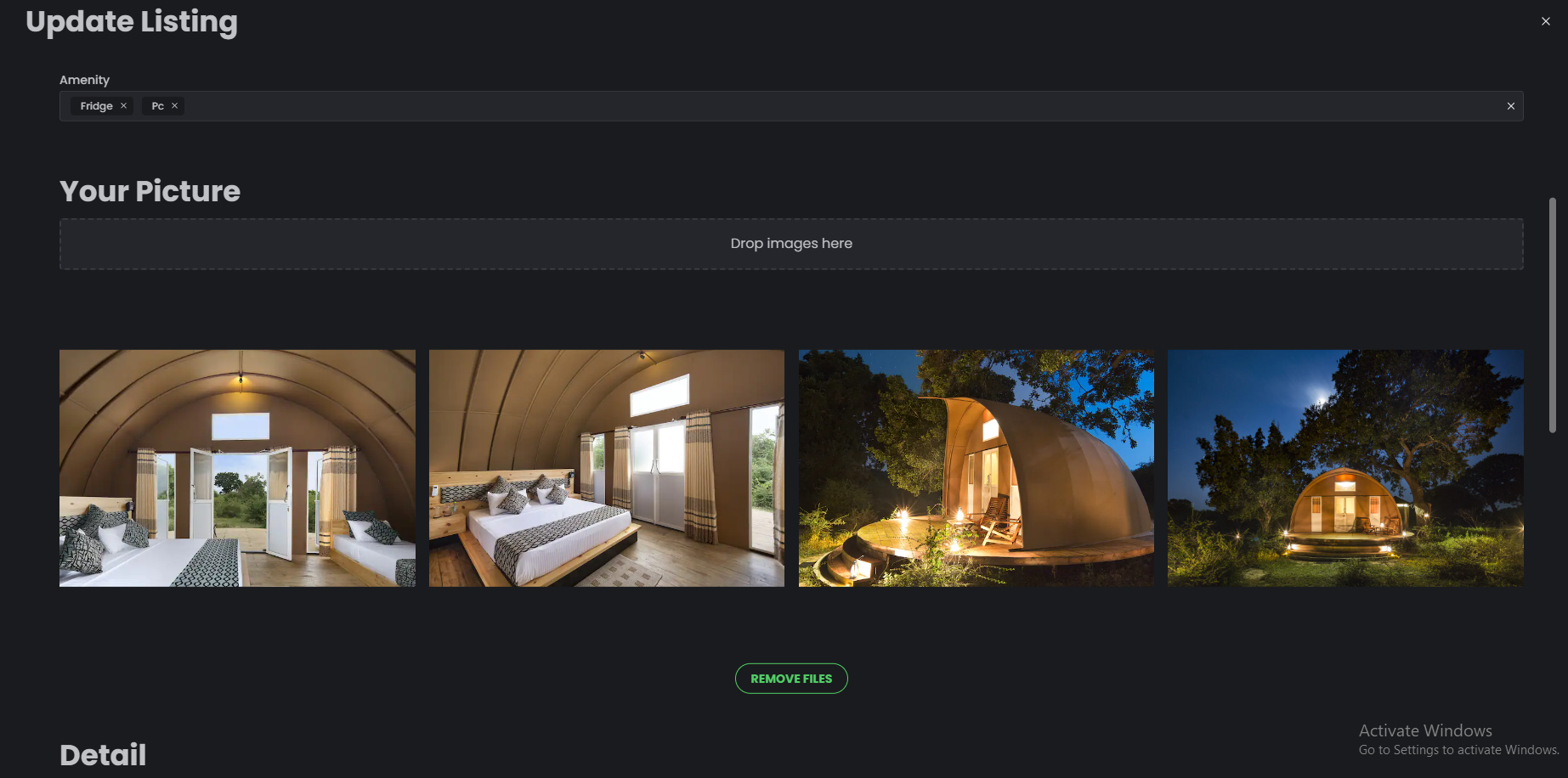
*Figure 2.9 Revenue statistics for hosts*

**

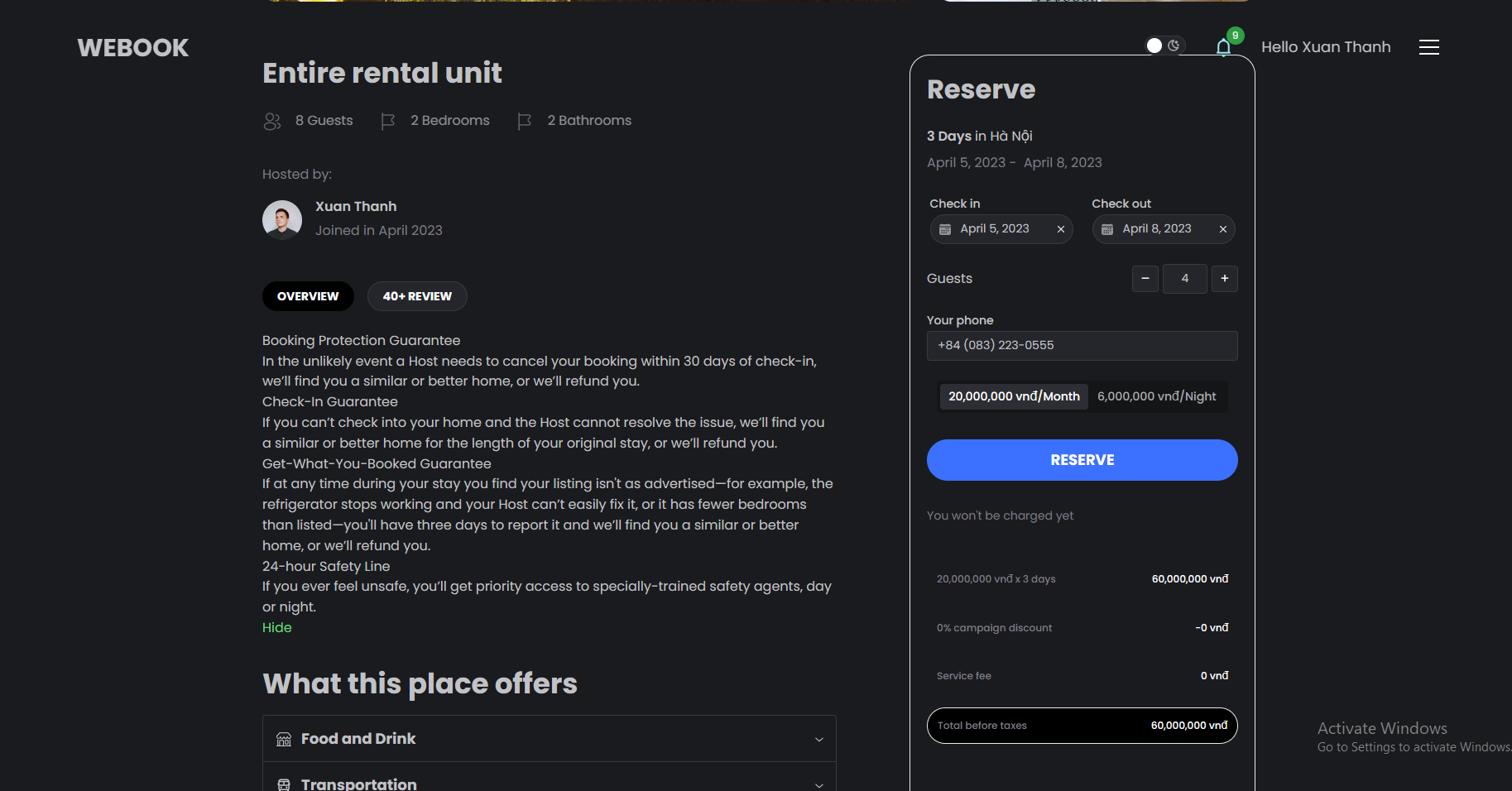
*Figure 2.10 The list of rooms awaiting approval.*

**

*Figure 2.11 List of managed accommodations*

**

*Figure 2.12 Update Listing*

**

*Figure 2.13 Reservation interface*

# 

# Chapter 3 - Test and review settings

## 3.1 Technology

#### 1 - Next.js

* Translation: Next.js[3] is a server-side web framework for React, built on the Node.js platform. It enables efficient development of web applications by providing features such as server-side rendering (SSR), static site generation (SSG), code splitting, and runtime data updates.
* Next.js is used to build dynamic web applications, static websites, or static websites with dynamic features. I use it for features such as routing, API routes, serverless functions, and many other features to help develop web applications quickly and easily.

#### 2- Mantine

* Mantine is an open-source UI (User Interface) library for React, designed to help develop web applications with beautiful, user-friendly and highly customizable user interfaces. It provides a large number of UI components such as buttons, forms, tables, sliders, and many others to help develop complex and professional-looking user interfaces.
* Mantine also features support for TypeScript, standard CSS-in-JS, accessibility features, and performance optimizations. The library is designed to be easy to use and customize, with simple and understandable APIs to help me customize my interface.

#### 3-Typescript

* TypeScript[5] is an open-source programming language that is an extension of JavaScript. TypeScript provides powerful features for developing more complex web applications by offering static type checking, static error checking, and other features such as classes, interfaces, scope, declarations, lambda functions, and many more.
* TypeScript allows me to write more precise code, avoiding errors due to incorrect data types or other errors that the JavaScript language cannot detect during development. TypeScript also provides a separate compiler, allowing TypeScript code to be compiled into JavaScript code to run on any web browser.

#### 4 - tRPC

* tRPC is an open-source library that allows the development of APIs using RPC (Remote Procedure Call) in both client-side and server-side web applications. tRPC helps to reduce programming work for processing requests and HTTP responses through a method-based approach instead of REST API.
* tRPC supports features such as authentication and authorization, latency control, application programming interfaces (API) interface created with TypeScript, and supports all popular user interface libraries such as React, Vue, Angular, and Svelte.

#### 5 - Prisma

* Prisma[7] is an open-source Object-Relational Mapping (ORM) designed to make developing web applications with databases easier. Prisma supports multiple databases including PostgreSQL, MySQL, and SQLite.
* Prisma provides me with a simple and optimized approach to accessing the database by using the GraphQL language to create queries and mutations. Prisma allows developers to use TypeScript to define data types for the application and provides features like authentication and authorization to protect user data.

#### 6 - NextAuth.js

* ReactJS is an open-source library for building user interfaces for web applications. It was developed by Facebook and widely used in the web development community. ReactJS uses a Component-Based model, allowing for independent UI components to be easily reused and managed.
* ReactJS uses JSX to define UI components and supports writing code with JavaScript. It also provides a flexible approach to managing the state of the application with Redux or the Context API. In addition, ReactJS also supports features such as Virtual DOM (Document Object Model) to optimize performance and improve user experience.

#### 7 - React

* ReactJS is an open-source library for building user interfaces for web applications. It was developed by Facebook and widely used in the web development community. ReactJS uses the Component-Based model, which allows building independent UI components that are easy to reuse and manage.
* ReactJS uses JSX to define UI components and supports writing code in JavaScript. It also provides a flexible approach for managing the state of an application with Redux or Context API. Additionally, ReactJS supports features such as Virtual DOM (Document Object Model) to optimize performance and improve user experience.

#### 8 - Firebase

* Firebase is a Backend-as-a-Service (BaaS) platform provided by Google, offering tools for developing web and mobile applications quickly and easily. Firebase includes a number of popular services such as:
  + Realtime Database: a real-time NoSQL database that allows data to be synchronized between devices quickly.
  + Authentication: provides user authentication mechanisms with authentication providers such as Google, Facebook, Twitter, and GitHub.
  + Storage: stores and manages multimedia files and data.
  + Cloud Functions: provides serverless functions for processing events and server-side logic.
  + Hosting: allows domain registration and management, providing high availability and static content distribution.
* Firebase is also used to store some special files such as images.

#### 9 - MySQL

* MySQL[11] is an open-source relational database management system widely used worldwide. It was developed by MySQL AB (now part of Oracle Corporation) and provides the ability to store and manage relational data.
* MySQL uses the SQL query language to query and process data, providing features such as creating, modifying, and deleting data tables, managing constraints and relationships between tables, and supporting functions such as indexing, grouping, sorting, and filtering data.
* MySQL also provides some advanced features such as the ability to backup and restore data, schedule tasks, partition data, and performance optimization tools.

#### 10 - Stripe

* Stripe[12] is a company that provides online payment solutions for businesses. Stripe offers tools for businesses to accept payments from customers around the world. Stripe provides features such as:
  + Payment Processing: Stripe provides tools to process payments from different payment formats such as credit cards, debit cards, e-wallets, and bank transfers.
  + Subscriptions: Stripe provides tools to manage recurring payment subscriptions, including managing payment cycles and automatic renewal features.
  + Fraud Detection: Stripe provides tools to detect fraudulent transactions, including security features and identity verification.
  + Payouts: Stripe provides tools to pay partners and service providers.
  + Billing: Stripe provides tools to manage customer invoices and services.
* I use Stripe as a payment method.

## 3.2. Installing system

#### 3.2.1. General setting

* Since NextJS is a full-stack framework, all of the client-side code and a portion of the server-side code are located within the "src" (source) directory. Within the "src" directory, there are several commonly used subdirectories, including:
  + **constants**: This directory contains variables and data that are constants.
  + **env**: This directory contains environment variable settings, checks whether the required environment variables are declared in the ".env" file (in the development environment) or in the environment variable settings on the hosting (in the production environment), and checks whether those environment variables have the correct data format.
  + **libs**: This directory contains commonly used library settings.
  + **utils**: This directory contains files defining synchronous common utility functions.
  + **helpers**: This directory contains files defining asynchronous common utility functions.

#### 3.2.2. Setting up client-side

* Client-side is developed on ReactJS. This part focuses on the following directories:
  + **assets**: This directory contains images, videos, vectors, fonts used on the website.
  + **pages**: Following the routing structure of NextJS, web pages will correspond to files in the "pages" directory with the path name being the file name.
  + **features**: Each web page has features inside, and all of them are wrapped in the "features" directory. The "features" directory will include subdirectories named similarly to the web page file name in the "pages" directory.
  + **components**: This directory contains reusable web elements that are reused multiple times on different web pages.
  + **hooks**: ReactJS has a hook feature, and this folder contains more complex hooks that are created with the purpose of being shared across multiple web pages.
  + **layouts**: This directory contains layout files of the interface, such as the Header and Footer of the website.

#### 3.2.3. Setting up server-side

* **prisma:** The folder containing the Prisma schema files that define the tables in the MySQL database.
* **server**: The folder containing the files that define the routes of the APIs used in the web application.

## 3.3. Testing and evaluation.

#### 3.3.1. Test environment

The testing environment of the Rentropolis system is built on the implementation of NextJS, a full-stack framework. This means that both the client-side and server-side are deployed on the same hosting.

The hosting used for Rentropolis is Vercel. Vercel is a cloud hosting platform that provides deployment and development services for web applications with high scalability. This platform allows for fast and easy web application development while providing features such as multi-region locations, code sharing, performance optimization, and automatic deployment.

The database used for Rentropolis is MySQL and is deployed on Railway. Railway is a hosting platform for web applications and databases. Railway provides features such as automatic deployment, easy management, and tools to help develop and deploy applications quickly and efficiently.

Overall, the testing environment of Rentropolis uses the Vercel and Railway hosting platforms for the client-side, server-side, and database. This allows me to focus on developing Rentropolis quickly and efficiently, while ensuring that the system operates stably and meets the needs of users.

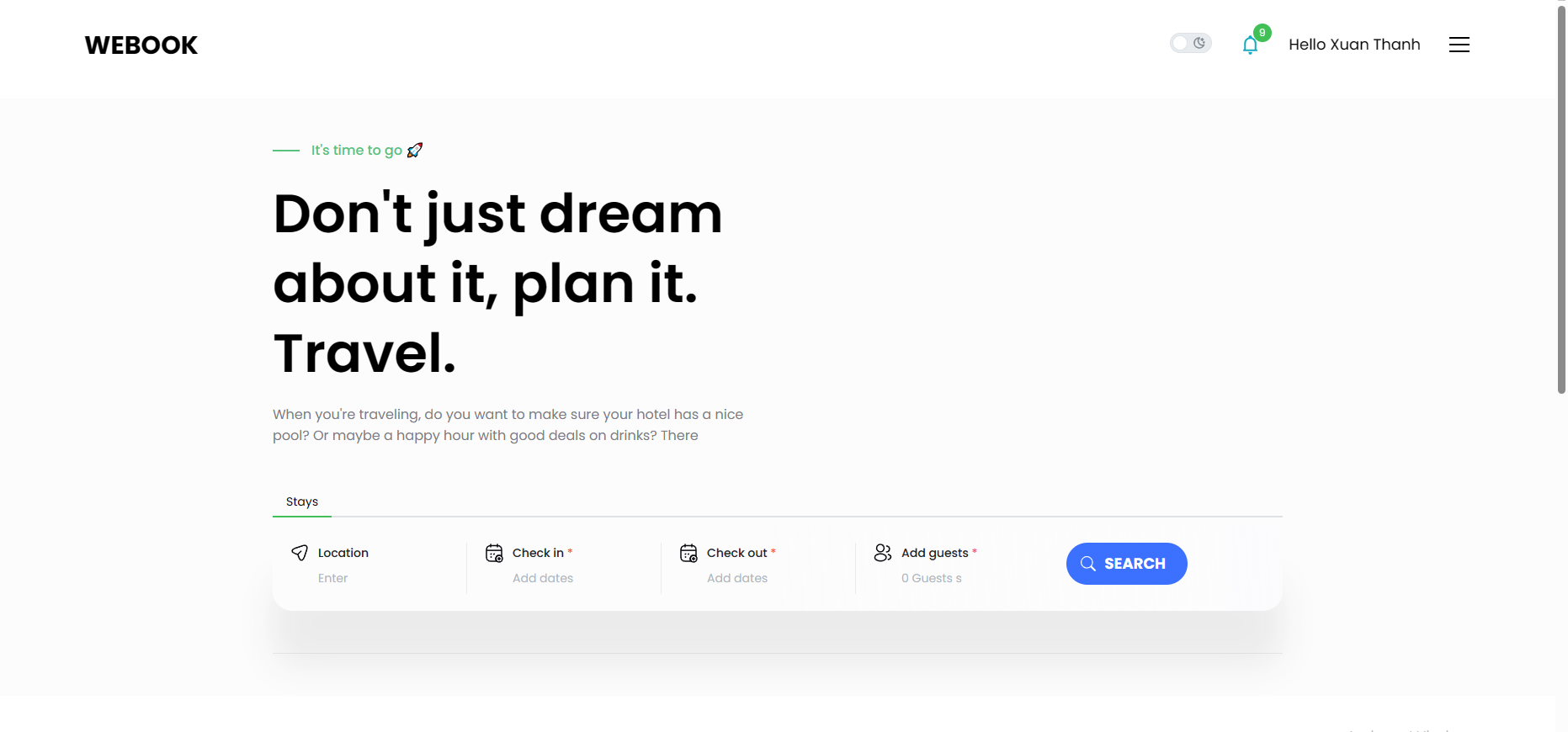
#### 3.3.2. Test data and results

##### 3.3.2.1. Test data

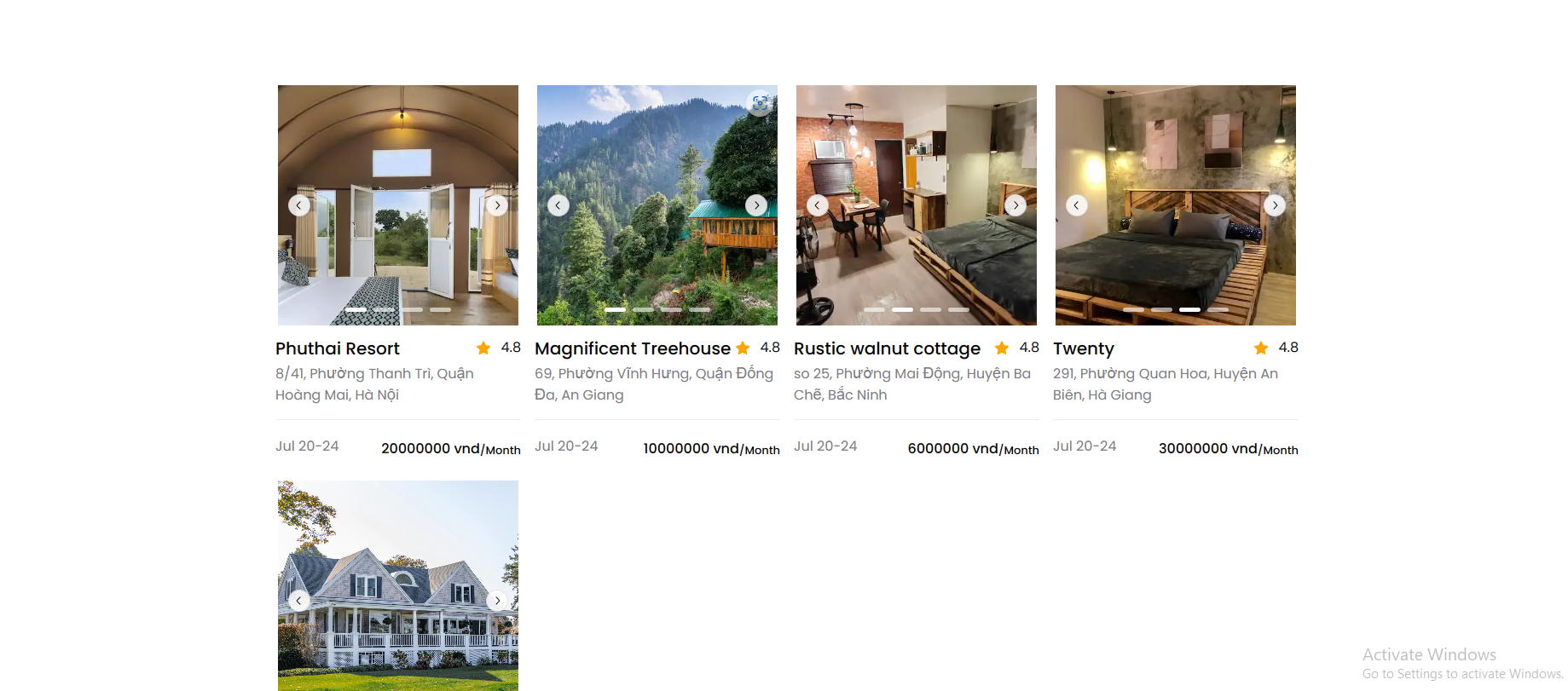
The test data was referenced from Traveloka and Agoda, which are two well-known booking websites in Vietnam.

##### 3.3.2.2. Results

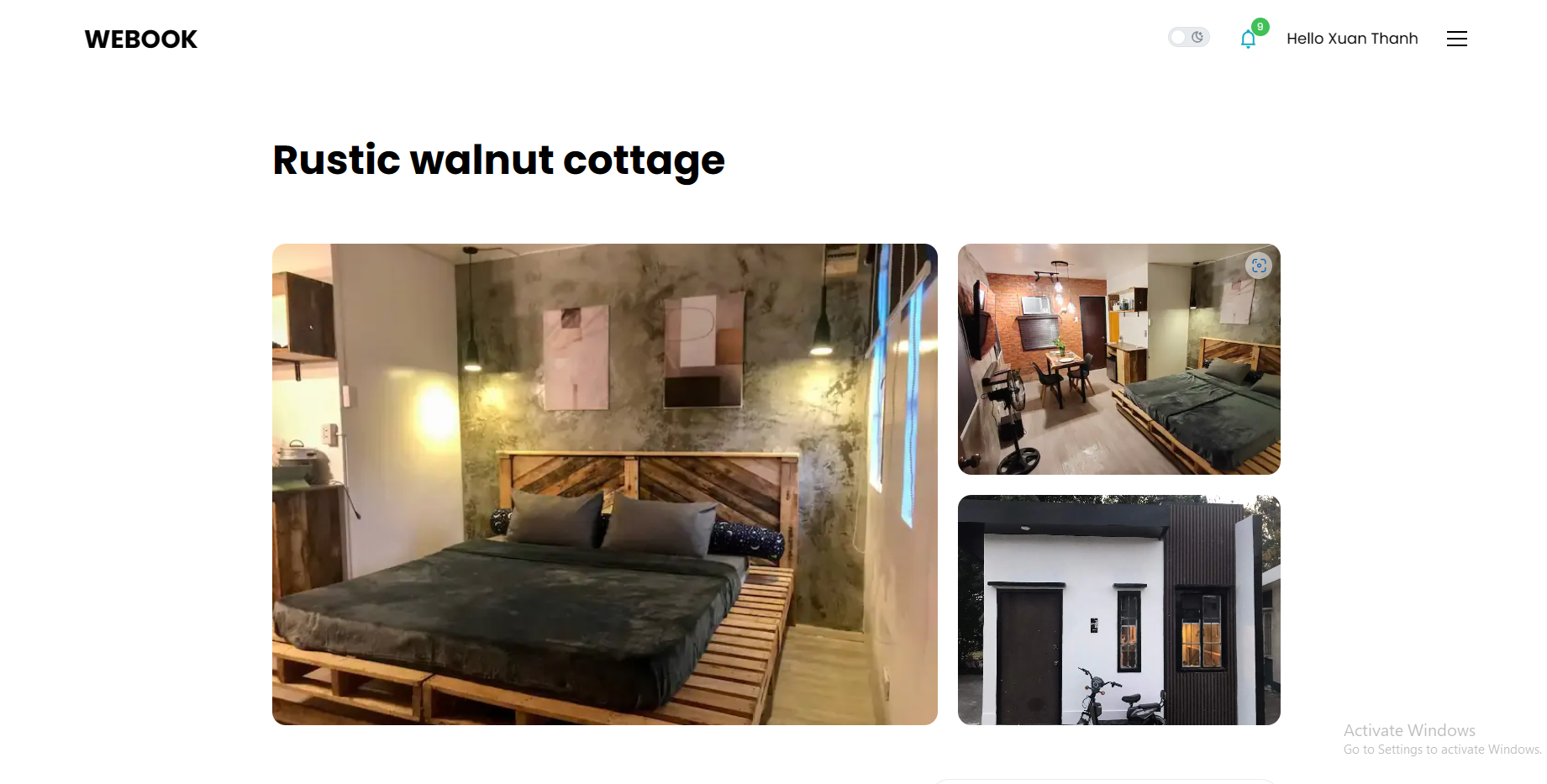
Below is the description and images of some of the main interfaces during testing:

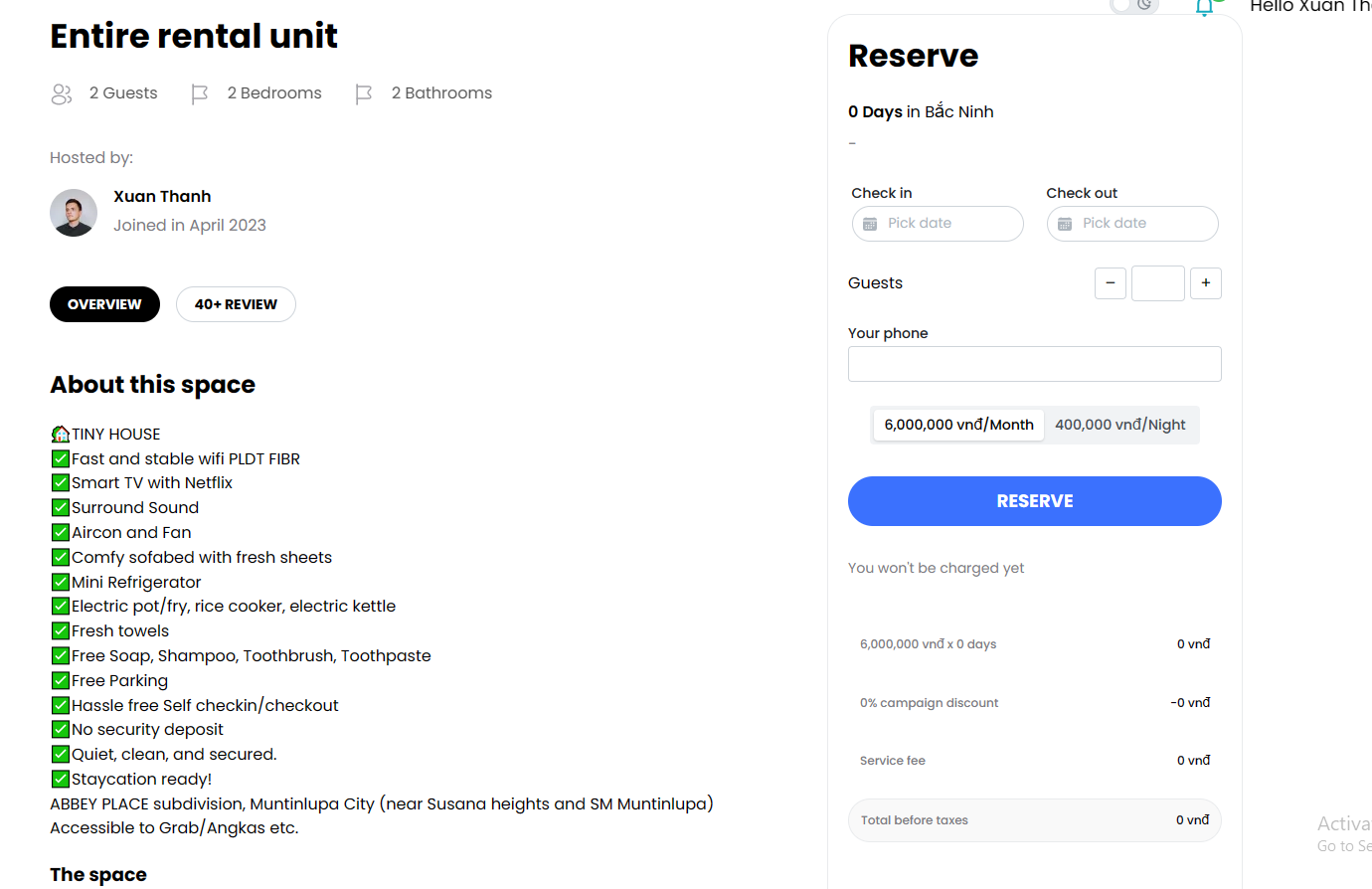


*Figure 3.3.2.2* Home page

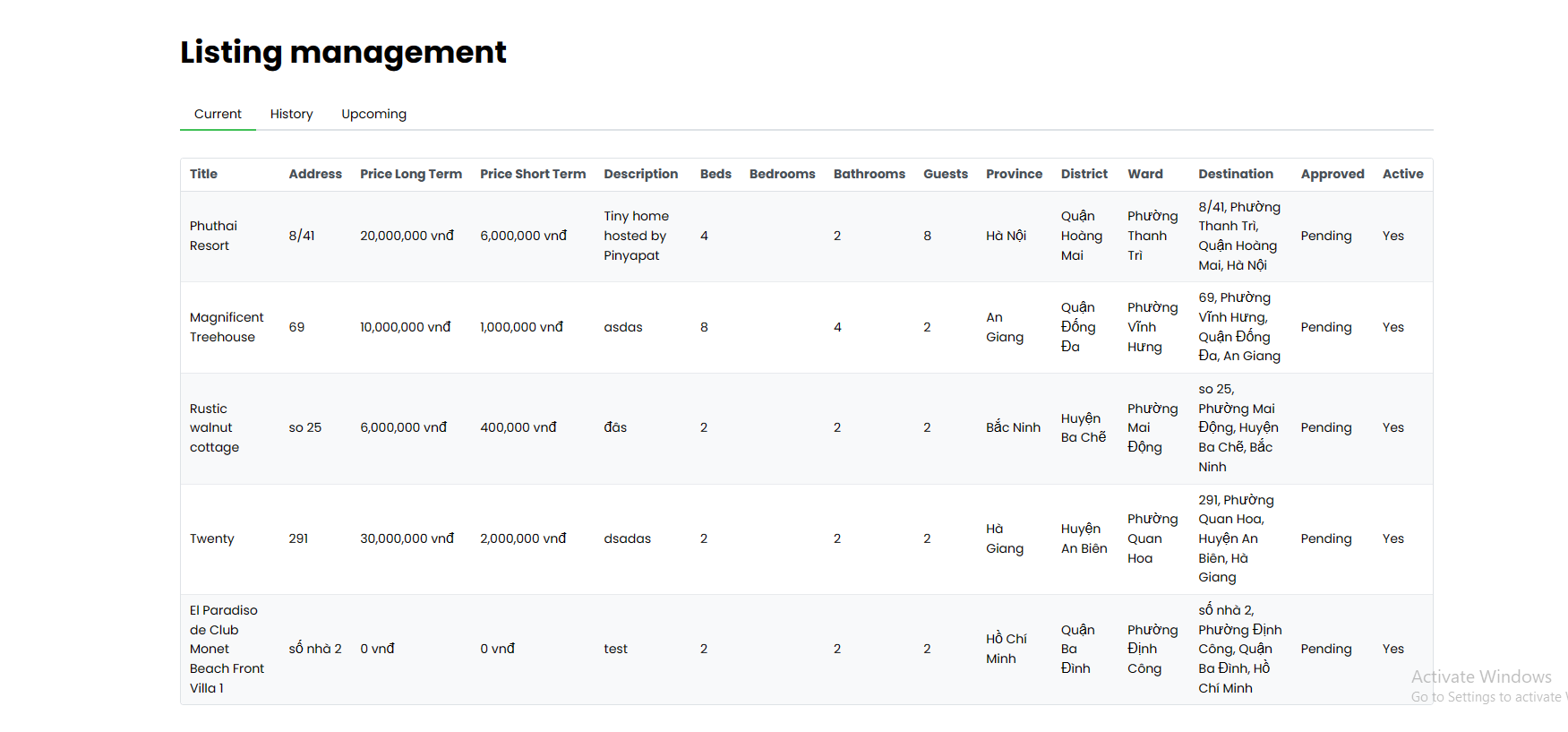


*Figure 3.3.2.3*  Room listing interface





*Figure 3.3.2.4* Detail booking and reservation



*Figure 3.3.2.5* Listing management

# Chapter 4: Conclusion and Future Development Direction:

## 4.1. Achievements

* Through this thesis, I have learned a lot of valuable knowledge and technologies that I have never used before, such as Typescript language, NextJS framework, and accompanying packages like Prisma, tRPC, etc. These knowledge will be a solid foundation for my future career.
* In terms of theory, through this thesis, I have gained a deeper understanding of system analysis and design. I have been able to grasp the process of creating a system, and have an overview of the software development process. I have been able to carry out all the steps to create a product from collecting documentation, analyzing requirements, designing, implementing, testing, to deploying the system.
* In terms of system development, the system has provided all the necessary functions for booking, searching, and managing rooms. Overall, the system has met the non-functional requirements set out above.

## 4.2. Limit

* Due to lack of experience, technology knowledge, business understanding, and time, the system has not yet fully implemented all the required features. Some of the features that I desired to have in my product have not been implemented.
* Some of the features need to be improved and optimized for a better user experience.

# Chapter 5: References

[1] Agoda. Available: https://www.agoda.com/vi-vn/. [Accessed 26 April 2023].

[2] Traveloka,. Available: https://www.traveloka.com/vi-vn/. [Accessed 26 April 2023].

[3] Next.js . Available: https://nextjs.org/. [Accessed 26 April 2023].

[4]Mantine. Available: https://mantine.dev/. [Accessed 26 April 2023].

[5] TypeScript, [Online]. Available: https://www.typescriptlang.org/. [Accessed 26 April 2023].

[6] trpc. Available: https://trpc.io/. [Accessed 26 April 2023].

[7] Prisma. Available: https://www.prisma.io/. [Accessed 26 April 2023].

[8] NextAuth.js. Available: https://next-auth.js.org/. [Accessed 26 April 2023].

[9] React . Available: https://react.dev/. [Accessed 26 April 2023].

[10] Firebase . Available: https://firebase.google.com/. [Accessed 26 April 2023].

[11] MySQL . Available: https://www.mysql.com/. [Accessed 26 April 2023].

[12] Stripe . Available: https://stripe.com/. [Accessed 26 April 2023].

|  |  |
| --- | --- |
| Appendix5.1: Update profile5.1.1: Use case  5.1.2: Activity diagram  5.1.3: Sequence diagram  5.2: Add payment method5.2.1: Use case  5.2.2: Activity diagram  5.2.3: Sequence diagram  5.3: Report user5.3.1: Use case  5.3.2: Activity diagram  5.3.3: Sequence diagram | |
|  | |
|  | |
|  |  |

#### 5.4: View accommodation details

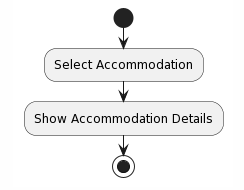
##### 5.4.3.1. Use case

##### 

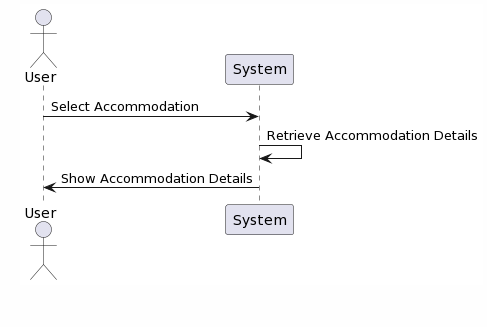


##### 5.4.3.2: Activity diagram

##### 



##### 5.4.3.3: Sequence diagram



|  |  |
| --- | --- |
| **Software testing**  **6.1 Registration and Login Functionality:**  **a**. Verify successful registration of new user accounts:   * Enter valid registration details and confirm successful account creation. * Attempt registration with invalid or missing information and ensure appropriate error messages are displayed.   b. Validate successful login with registered accounts:   * Enter correct login credentials and verify successful login. * Attempt login with incorrect username or password and ensure appropriate error messages are displayed.   c. Test authentication and restrict unauthorized access to protected pages:   * Access protected pages without logging in and verify redirection to the login page. * Log in with a valid account and ensure access is granted to protected pages.   **6.2 Search and Filtering Property Listings:**  a. Verify the search functionality and display a list of relevant properties based on search criteria:   * Enter specific search criteria (e.g., location, guest range) and validate that relevant properties are displayed. * Perform a search with no matching properties and verify an appropriate message is shown.   **b.** Test the filtering of search results based on specific requirements:   * Apply filters (e.g., price, area, number of rooms) and confirm that the search results are correctly refined. * Test scenarios where no properties meet the applied filters and ensure proper messaging is displayed.   c. Validate the sorting feature of search results based on criteria:   * Sort search results by price, area, or other criteria and verify that the sorting order is accurate. * Test scenarios where sorting results in ascending or descending order**.**   **6.3 Property Details and Booking Functionality:**   * Verify the display of detailed information for a specific property: * Click on a property listing and ensure that all relevant details (e.g., description, amenities, photos) are displayed correctly.   b. Test the booking process and confirm successful booking:   * Select a property and proceed with the booking process, ensuring that all necessary information is collected. * Confirm that the booking is successfully completed and relevant notifications are sent to both the renter and the landlord.   **6.4 Scalability Testing:**  a. Test the performance of the website with a significant number of concurrent users:   * Simulate a high user load on the website and monitor response times and server resources. * Validate that the website can handle the increased traffic without significant performance degradation.   b. Validate the system's ability to handle increased traffic and user load:   * Conduct load testing by simulating multiple concurrent users accessing different features of the website. * Monitor server response times, database performance, and overall system stability under various load conditions. * These detailed testing scenarios cover the major features of the website connecting renters and landlords. However, it is important to adapt and expand these scenarios based on the specific requirements and functionalities of your project. Additionally, consider including edge cases and boundary testing to ensure robustness and reliability of the website. | |
|  | |
|  | |
|  | |
|  |  |