Design Document

for

Homestay Booking Website Project

Version 1.0

Prepared by

Group 6

Vu Nguyet Hang	**22028079**	**22028079@vnu.edu.vn**
Duong Quoc Khanh	**22028090**	**22028090@vnu.edu.vn**
Nguyen Thi Thu Trang	**22028254**	**22028254@vnu.edu.vn**
Nguyen Van Son	**22028020**	**22028020@vnu.edu.vn**
Nguyen Truong Giang	**19021259**	**19021259@vnu.edu.vn**

Instructor: Assoc. Prof. Dr. Dang Duc Hanh

Course: INT2208E_23

Teaching Assistant: Kieu Van Tuyen

Date: 04/05/2024

CONTENTS

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
#1	Group 6	Initial Design Document	04/05/24

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed architecture design of the Homestay Booking System by focusing on four key quality attributes: usability, availability, maintainability, and testability. This document will help the development team to determine how the system will be structured at the highest level. It is also intended for the project manager to sign off on the high-level structure before the team shifts into detailed design, and to validate that the development team is meeting the agreed-upon requirements during evaluation process.

1.2 Scope

This document will address the architecturally significant functional requirements as well as a prototype of the user interface design. Additionally, a class diagram and sequence diagram are provided for each use case. By addressing these aspects comprehensively, this document aims to provide stakeholders, including developers, designers, and project managers, with a clear understanding of the homestay booking website's design and functionality. It serves as a reference point throughout the development lifecycle, guiding decision-making and facilitating effective communication among project team members.

1.3 Definitions, Acronyms and Abbreviations

User	refers to any individual who interacts with the homestay booking website.
Guest	a user of the homestay booking website who seeks temporary accommodation in various locations.
Host	a user who offers accommodation on the homestay booking website.
UI	User Interface

1.4 References

Application Architecture Guide 2.0 patterns and practices. © 2008 Microsoft Corporation.
Retrieved from: http://fizyka.umk.pl/~jacek/docs/net/Application_Architecture_Guide_v2.pdf

Textbook: Software Engineering PEARSON Tenth Edition – Ian Sommerville.

Architecture model, use case diagram templates provided by teaching assistant.

2. Architectural Design

2.1 Overview

This figure shows a high-level overview of the system's architecture. Further details on the system components and their interactions will be explained in detail in the following sections.

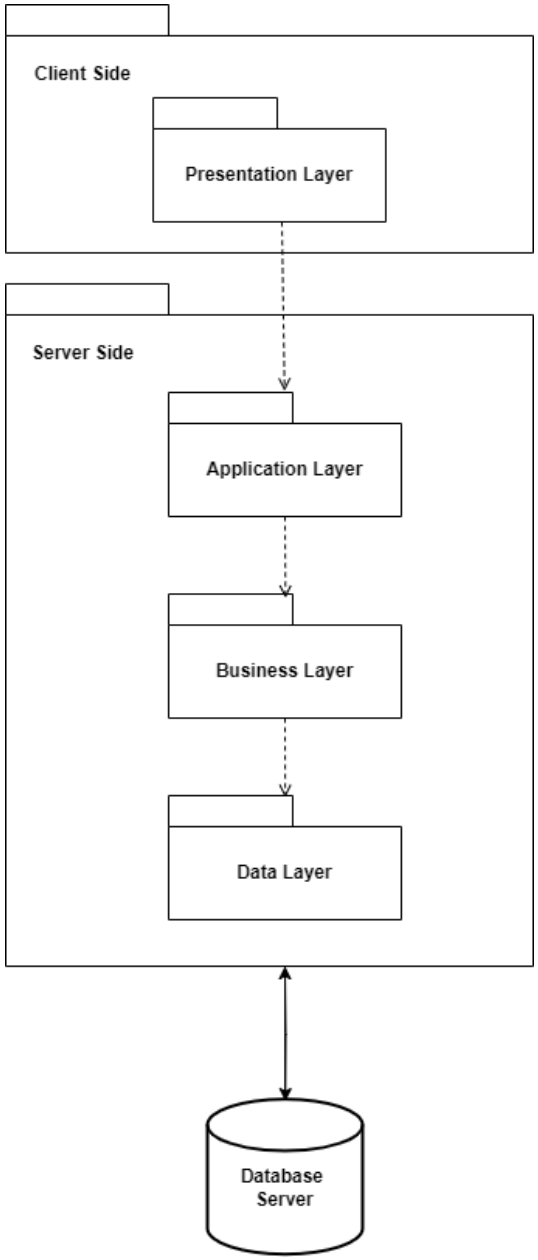


Figure 1. Architecture overview

Overall, the sections of the application design can be thought of as four basic sets of services:

- **Presentation services:** Theses are the user-oriented services responsible for managing user interaction with the system, and generally consist of componets located within the presentation layer. They provide a common bridge into the core business logic encapsulated in the business services.
- **Application services:** These services handle the presentation layer requests, transformation of disparate data for presentation and serve as the controller component.
- **Business services:** These services implement the core functionality of the system and encapsulate the relevant business logic. They generally consist of components located within the business layer, which may expose service interfaces that other callers can use.
- **Data services:** These services provide access to data that is hosted within the boundaries of the system, and data exposed by other back-end systems; perhaps accessed through services. The data layer exposes data to the business layer through generic interfaces designed to be convenient for use by business services. This layer implements the communication with the data source (in this case, a database).

2.2 Architectural Details

2.2.1 Presentation Layer Components

Presentation layer components implement the functionality required to allow users to interact with the application. In our Homestay Booking System, these components are separated into two types:

- **User inteface (UI) components.** These components provide the mechanism for users to interact with the application. They format data and render it for display, acquire and validate data entered by user. Specifically, they are React components such as header, search bar, pop-up messages, etc.

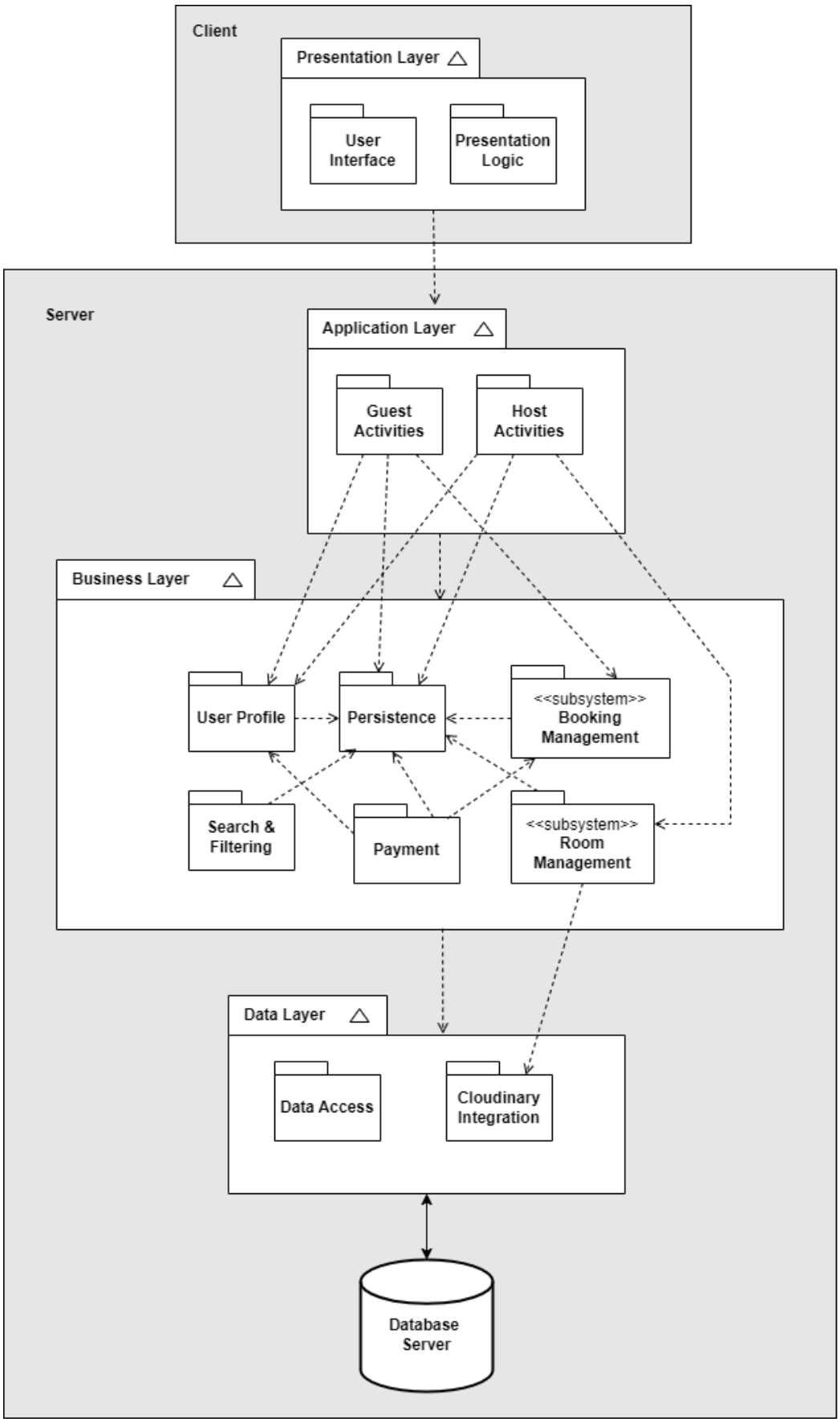


Figure 2. Architectural model

- **Presentation Logic components.** They are responsible for managing the user interface and controlling how data is presented to the user. These components handle tasks such as managing the state of UI elements and responding to user interactions. Within the homestay booking system, we also leverage caching mechanisms to optimize room lookups and avoid network round trips.

2.2.2 Application Layer Components

By placing guest and host activities in this layer, we separate the concerns of managing user interactions and business rules from other layers as they represent different sets of functionalities available to users acting as guests and hosts within the Homestay Booking System.

- Both guest and host activities rely on **User Profile** in the Business Layer for managing user information, preferences, and authentication. They have common user managament functionalities, such as registration, login, profile editing.
- Both guest and host activities depend on **Persistence** component in the Business Layer for storing and retrieving data from the database.
- Guest activities rely on the **Booking Management** subsystem for functionalities such as searching for available rooms, making reservations, and managing bookings.
- Host activities depend on the **Room Management** subsystem for functionalities such as listing properties, managing room availability, editing room, viewing guests' bookings.

2.2.3 Business Layer Components

- The **Payment** component handles payment processing for homestay booking (in our project, this is done through Stripe). It interacts with both **User Profile** component to retrieve user payment information and the **Booking Management** component to associate payments with specific bookings.
- The **Search & Filtering** component enables users to search for rooms based on criteria such as location, dates, number of guests.
- All components in this layer interacts with **Persistence** component to ensure that each component can access and manipulate data stored in the database efficiently.
- **Room Management** subsystem is responsible for managing room inventory, including descriptions, amenities, and images. **Integrating with Cloudinary** in the Data Layer allows system to store room images and media assets in the cloud, providing a centralized and scalable solution for managing and serving media content.

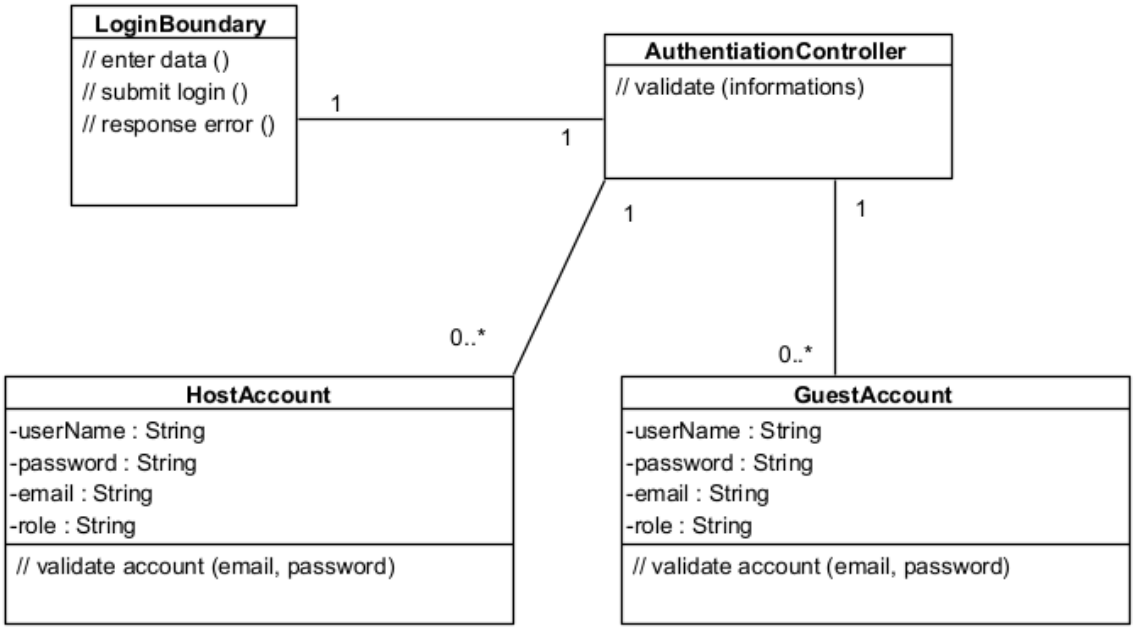
2.2.4 Data Layer Components

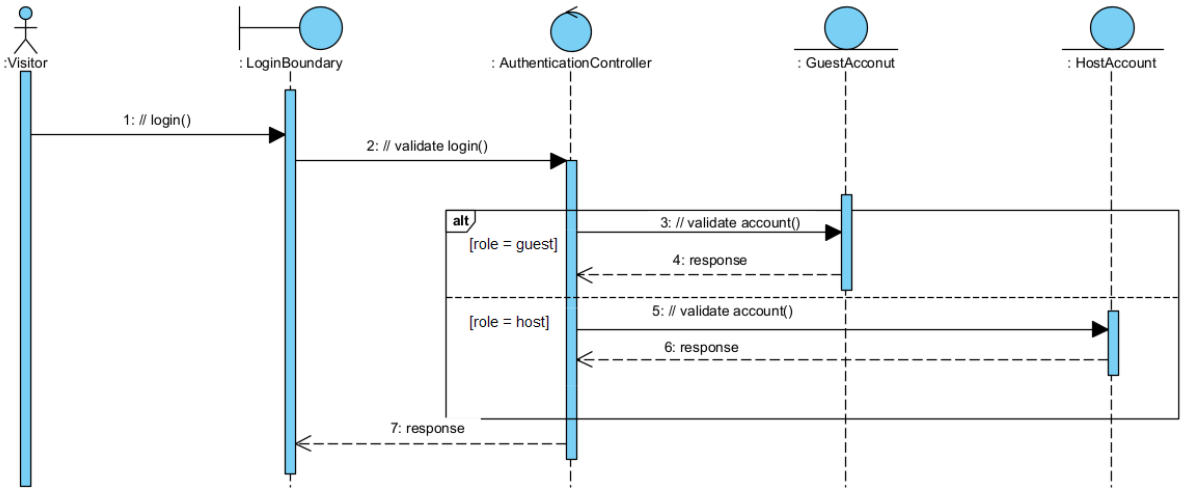
- The **Data Access** component is responsible for handling interactions with the database, providing a set of interfaces, methods, or classes that abstract the details of data storage and retrieval operations, allowing other components in higher layers to interact with the database without needing to know the underlying implementation details. It includes functionalities such as: connecting to the database, executing queries, inserting – updating – deleting data.
- **Cloudinary Integration**, as mentioned before, is a cloud-based media management platform that manages host's images, directly involves in the process of adding room which belongs to **Room Management** subsytem.

Figure 2 illustrates components of each layer and the dependencies between them.

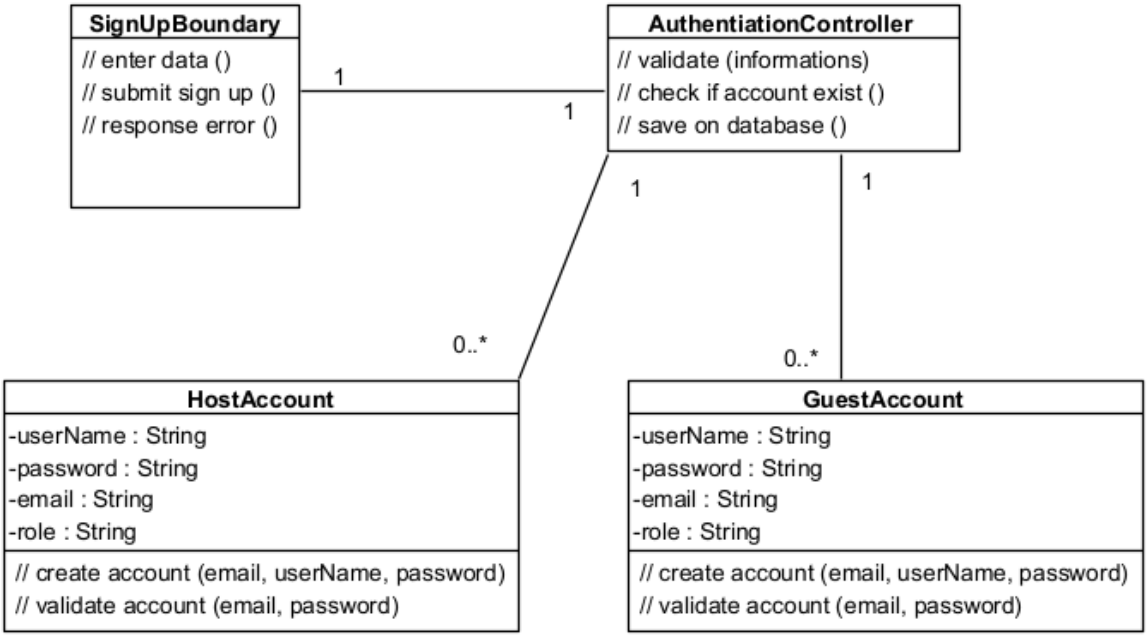
2.3 Use Case Diagrams

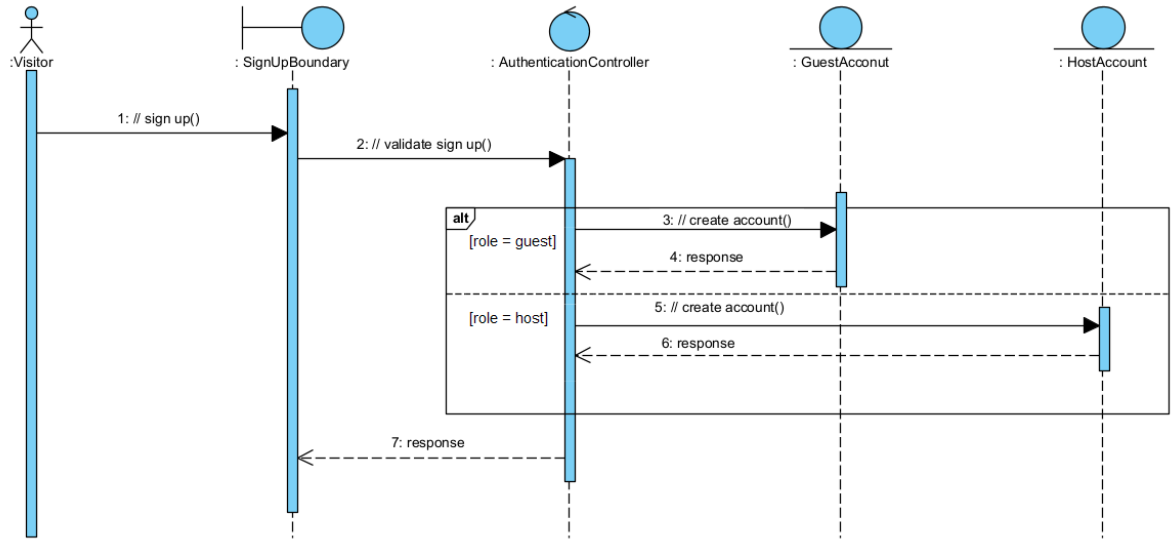
2.3.1 Login


LoginBoundary
// enter data ()
// submit login ()
// response error ()
AuthentiationController
// validate (informations)
1
1
1
1
0..*
0..*
HostAccount
-userName : String
-password : String
-email : String
-role : String
// validate account (email, password)
GuestAccount
-userName : String
-password : String
-email : String
-role : String
// validate account (email, password)

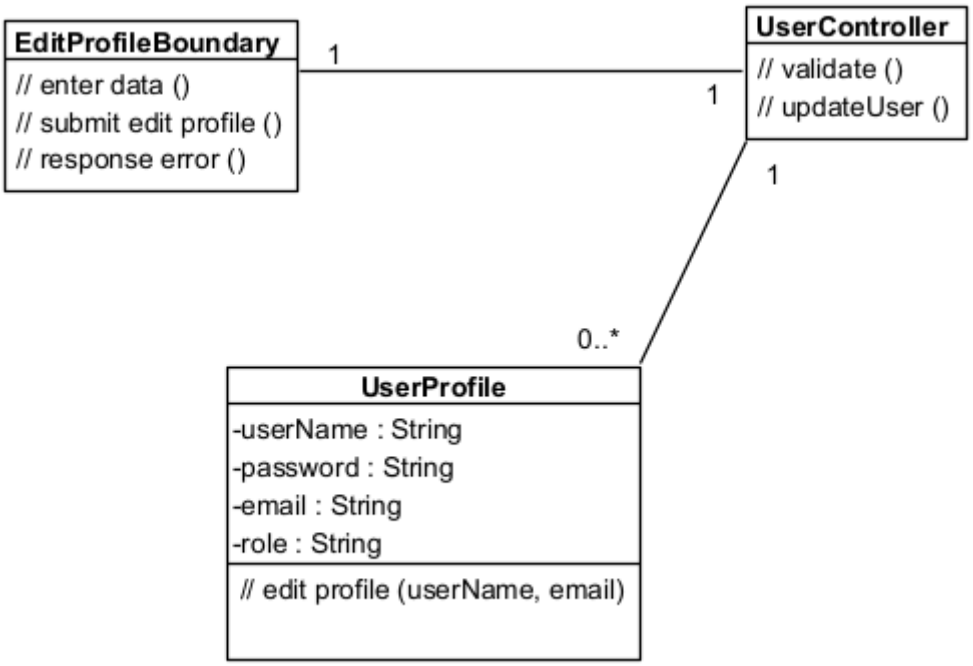

:Visitor
: LoginBoundary
: AuthenticationController
: GuestAcconut
: HostAccount
1: // login()
2: // validate login()
alt
[role = guest]
3: // validate account()
4: response
[role = host]
5: // validate account()
6: response
7: response

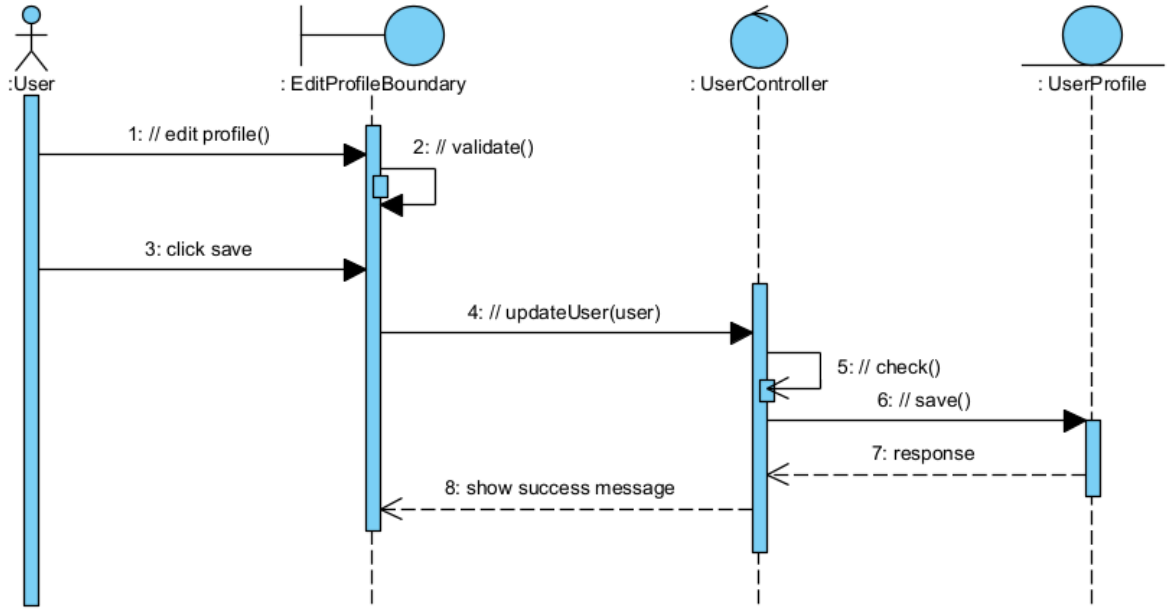
2.3.2 Register



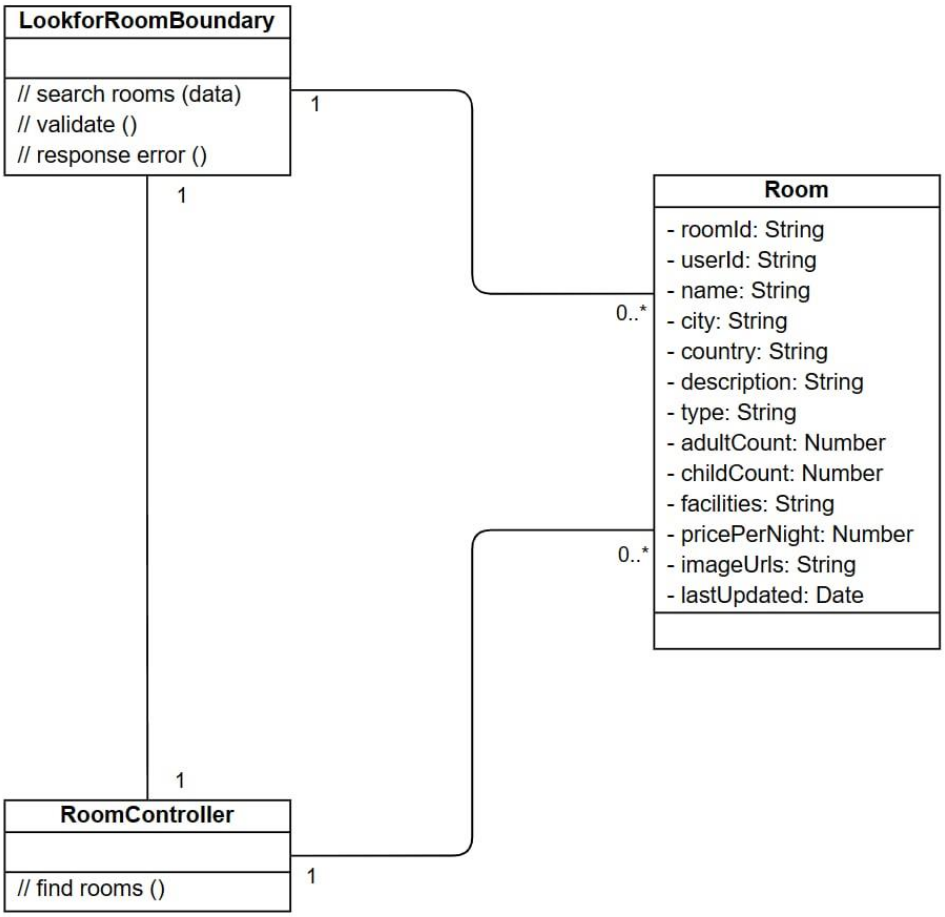


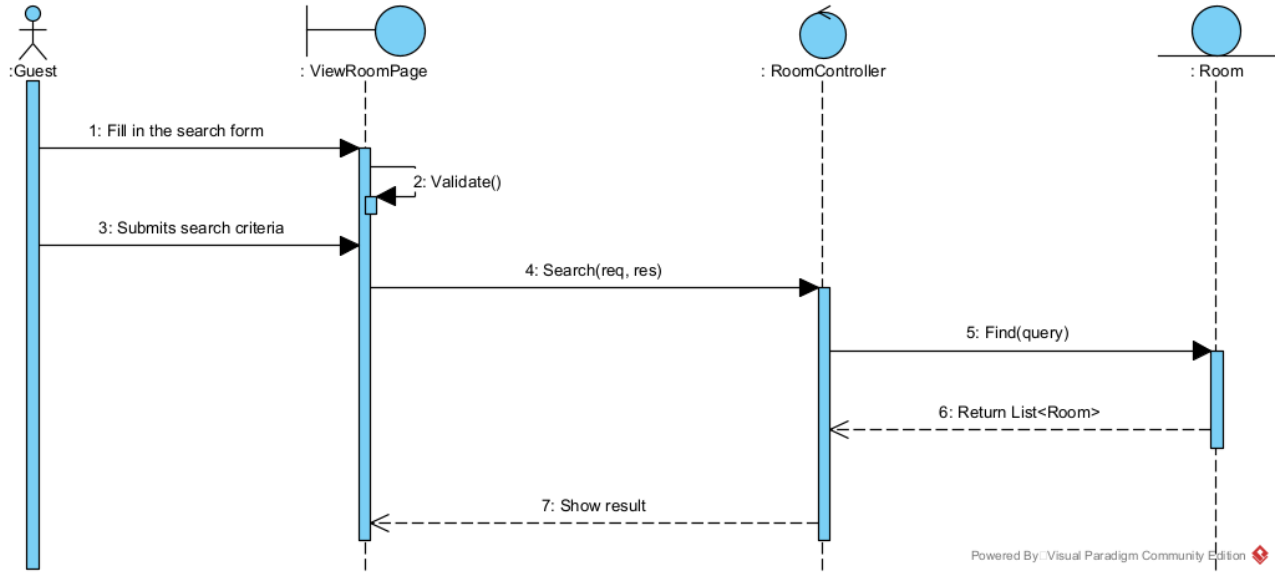
2.3.3 Edit Profile



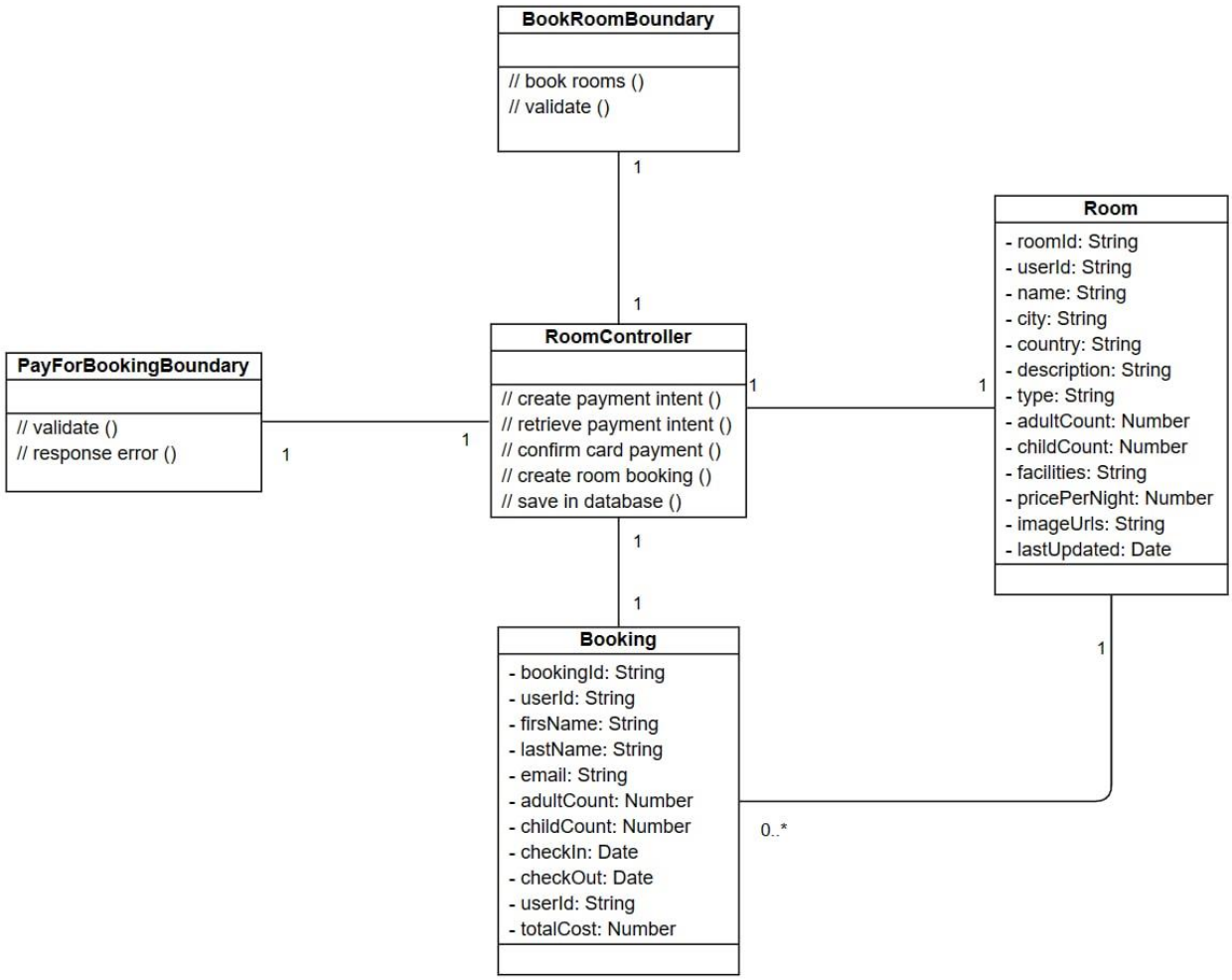


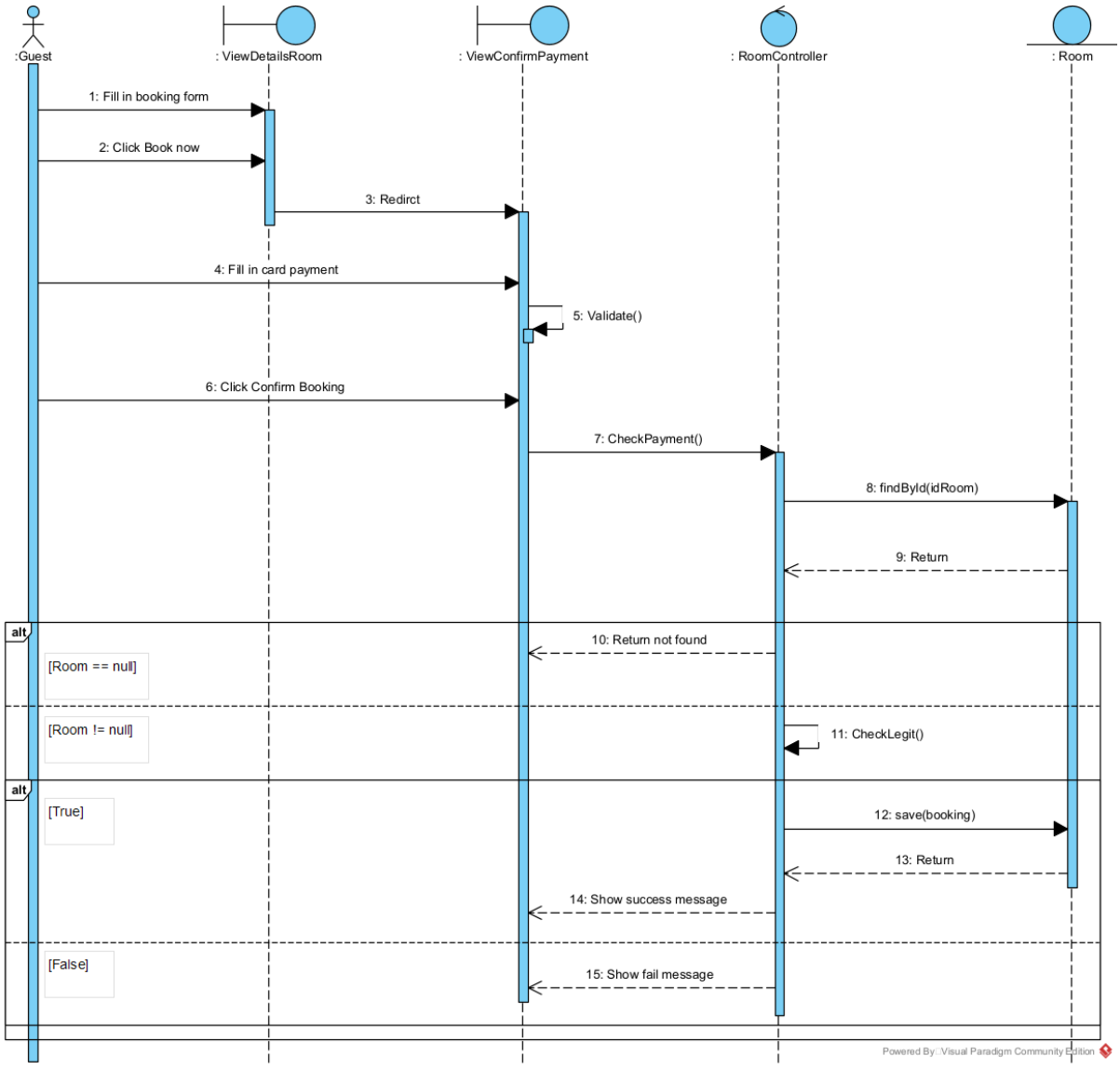
2.3.4 Search Room



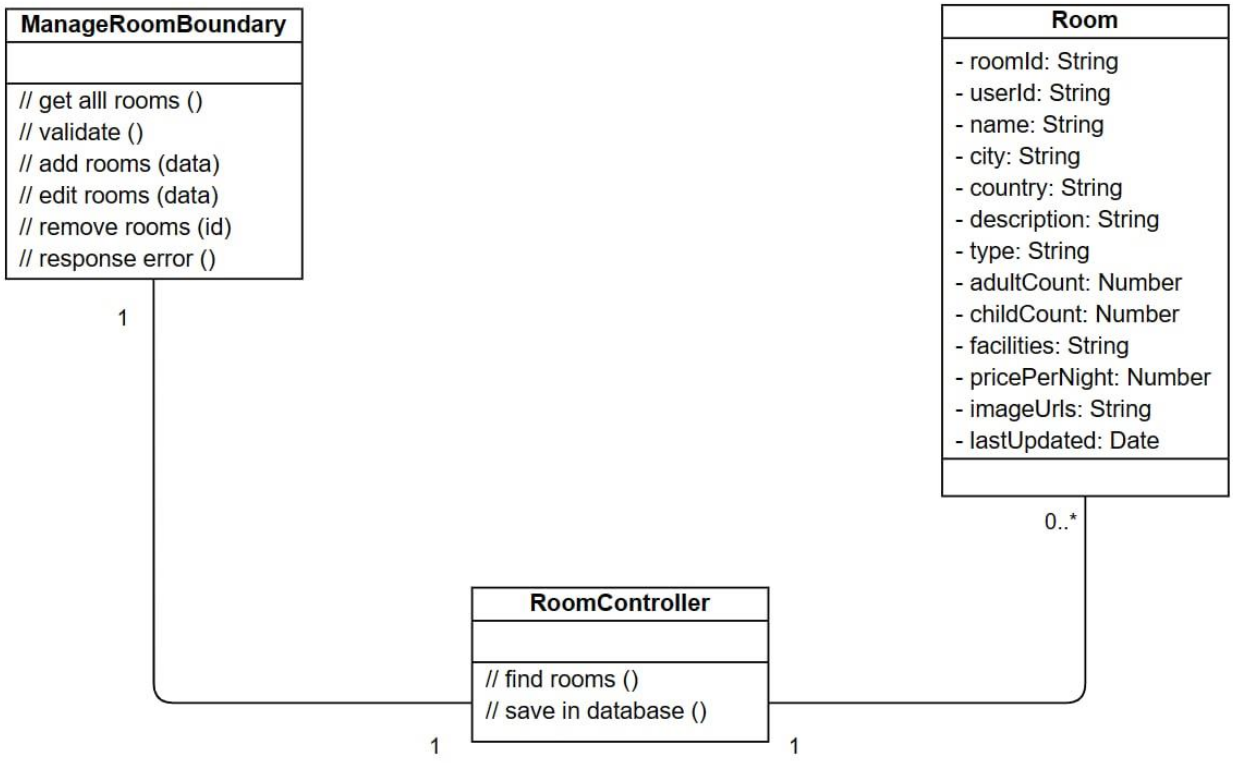


2.3.5 Book Room

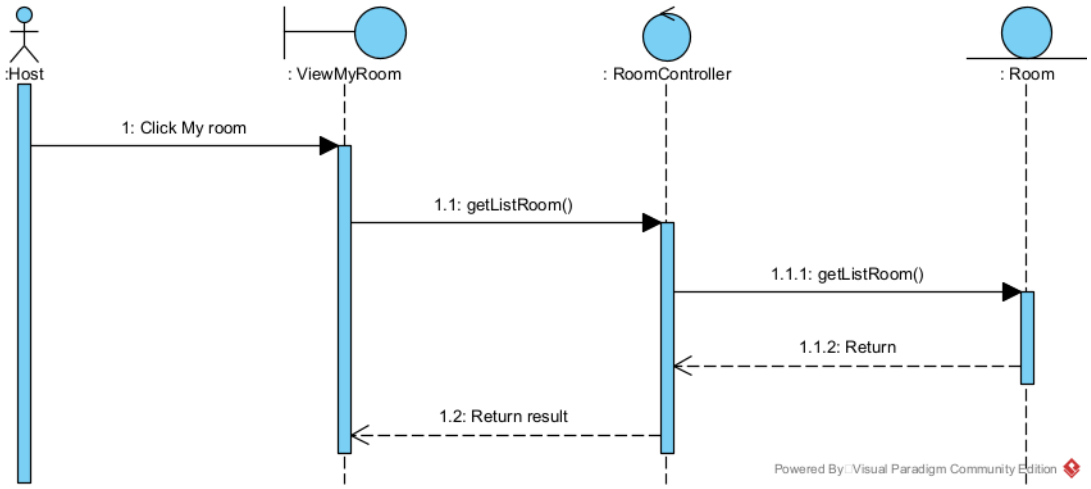


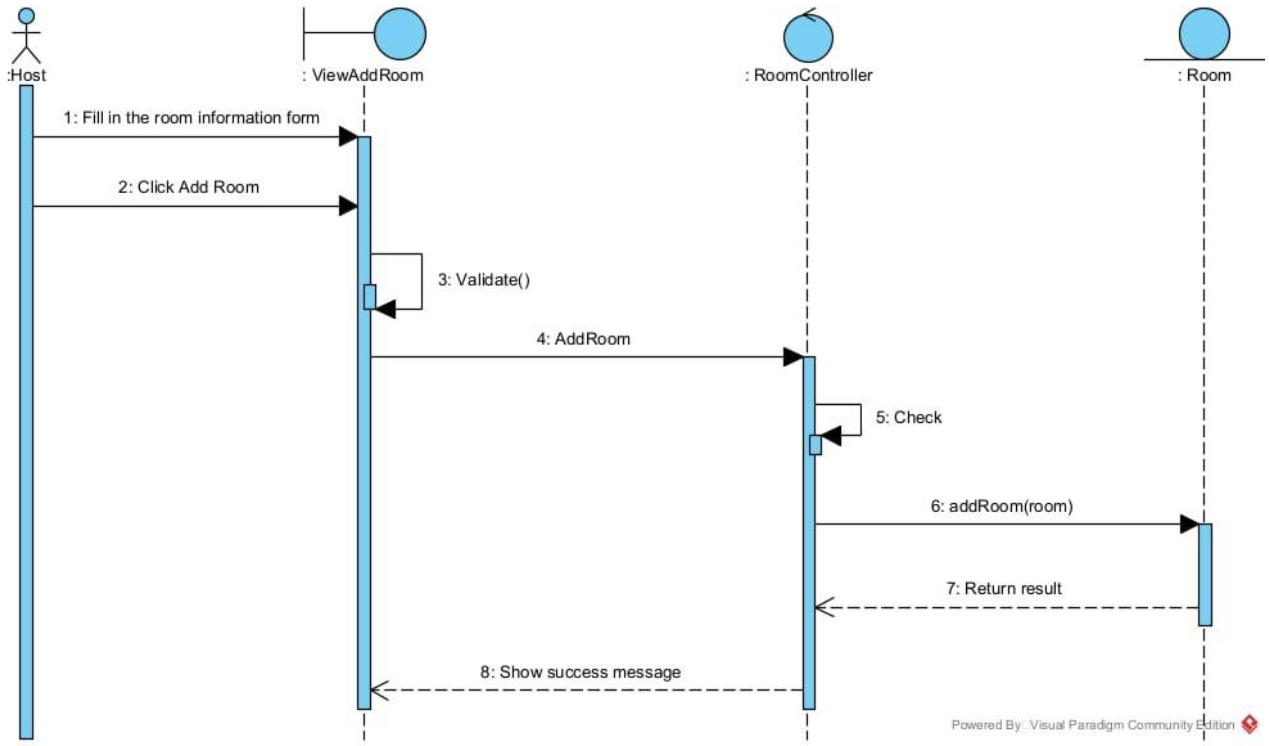

:Guest
: ViewDetailsRoom
: ViewConfirmPayment
: RoomController
: Room
1: Fill in booking form
2: Click Book now
3: Redirct
4: Fill in card payment
5: Validate()
6: Click Confirm Booking
7: CheckPayment()
8: findById(idRoom)
9: Return
alt
[Room == null]
10: Return not found
[Room != null]
11: CheckLegit()
alt
[True]
12: save(booking)
13: Return
14: Show success message
[False]
15: Show fail message
Powered By: Visual Paradigm Community Edition

2.3.6 Manage Room

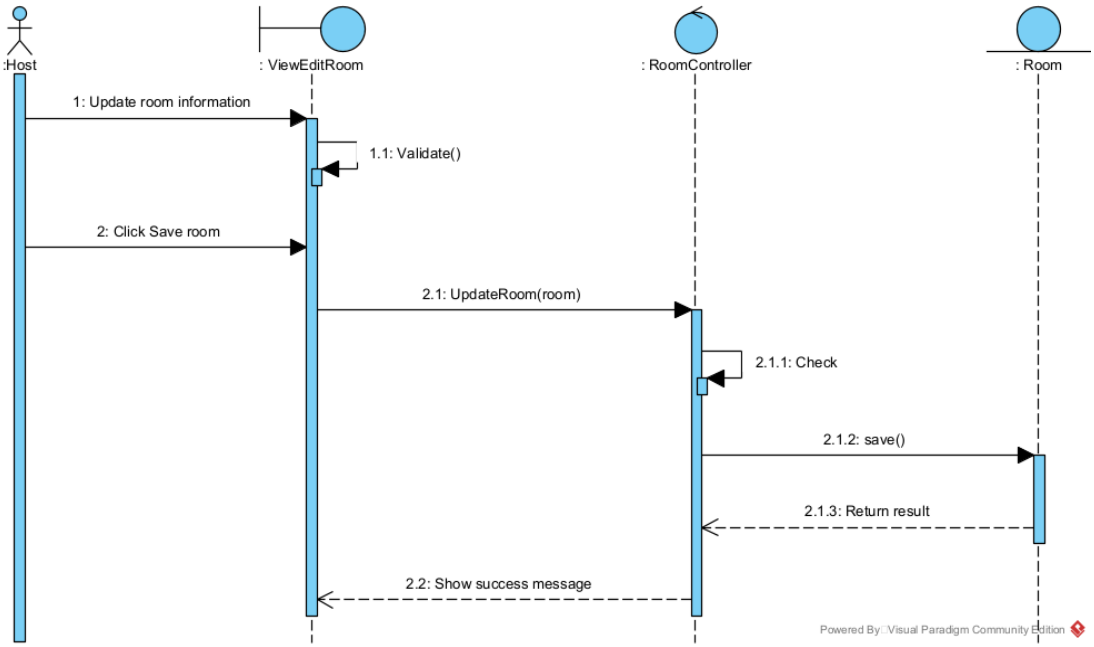


2.3.6.1 View my room

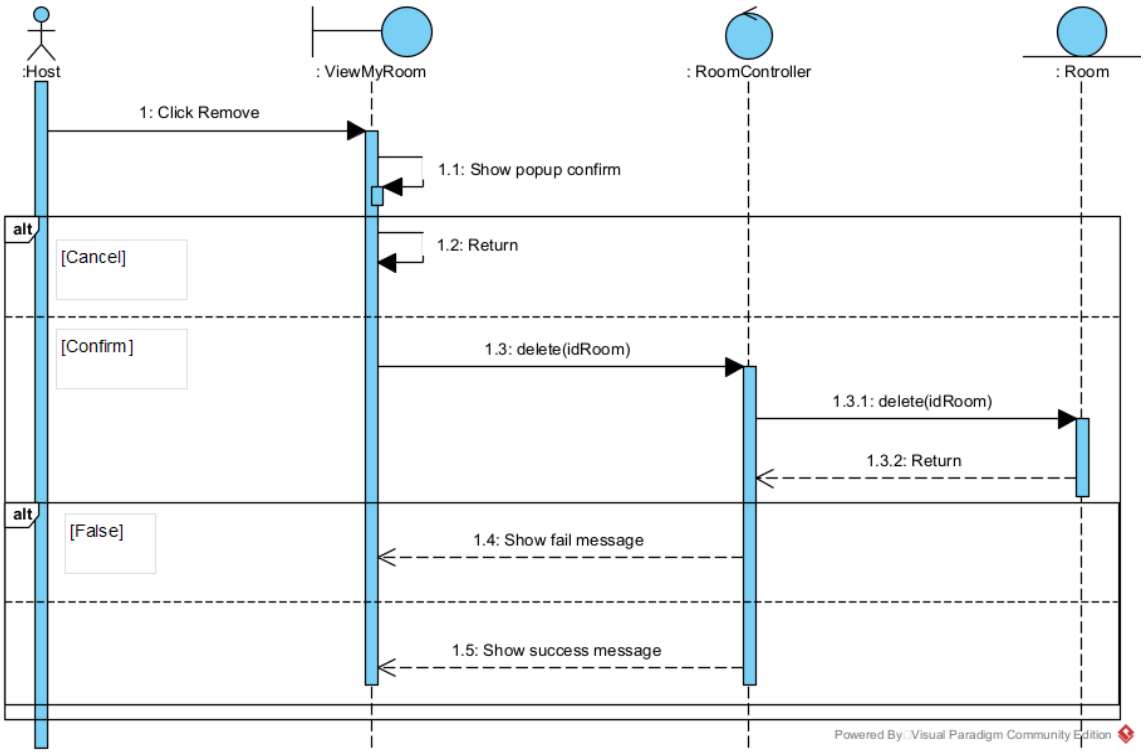




2.3.6.2 Add room



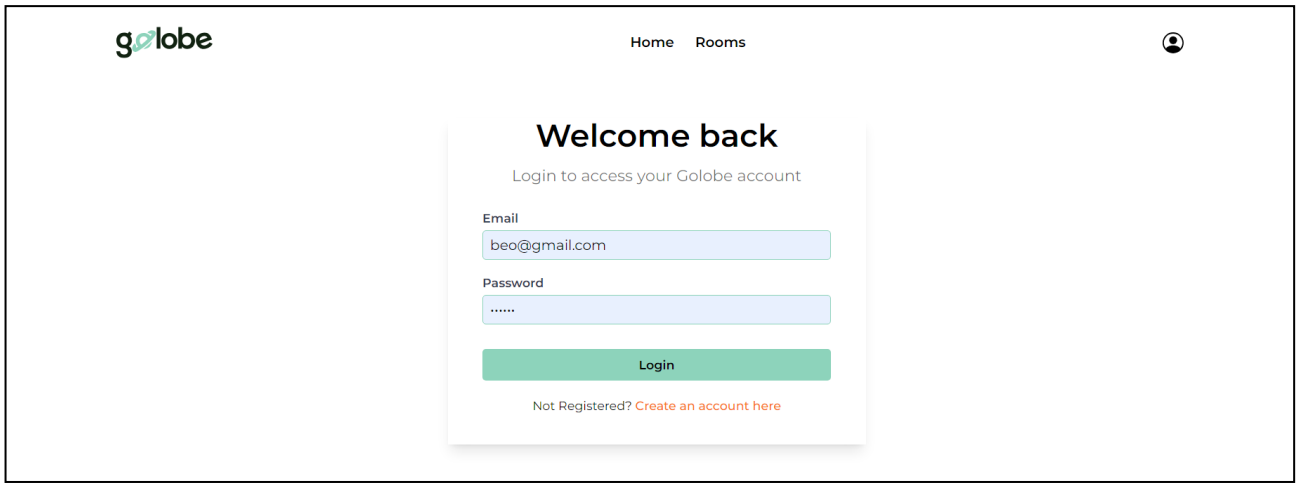
2.3.6.3 Edit room



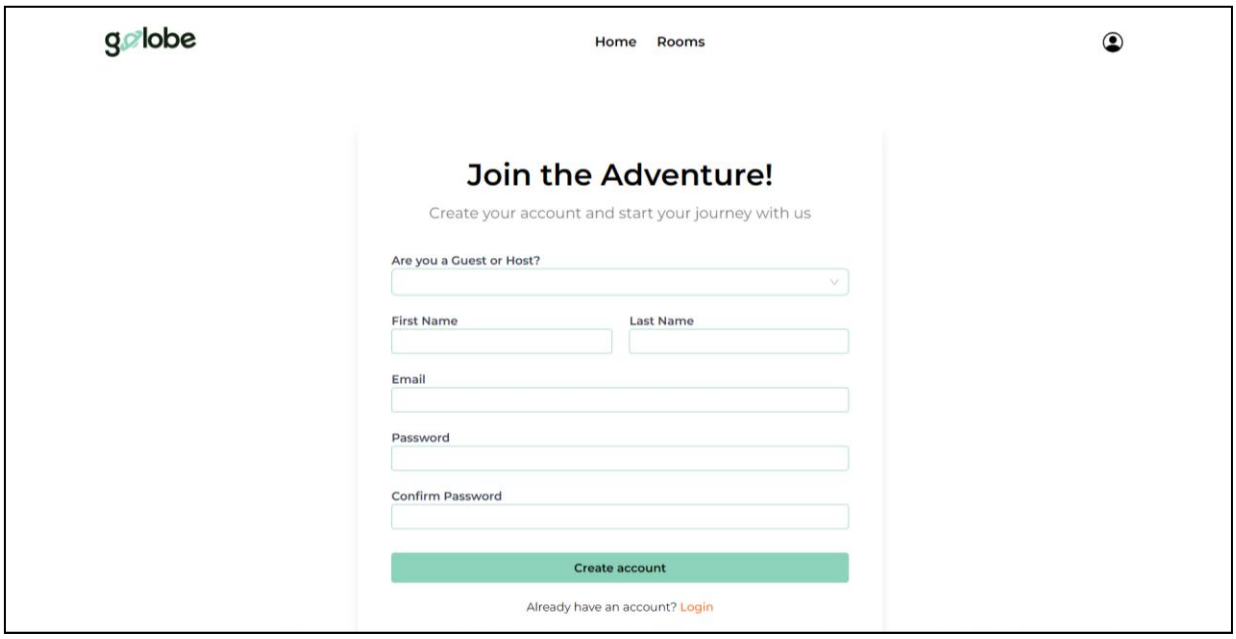
2.3.6.4 Delete room

3. User Interface Design

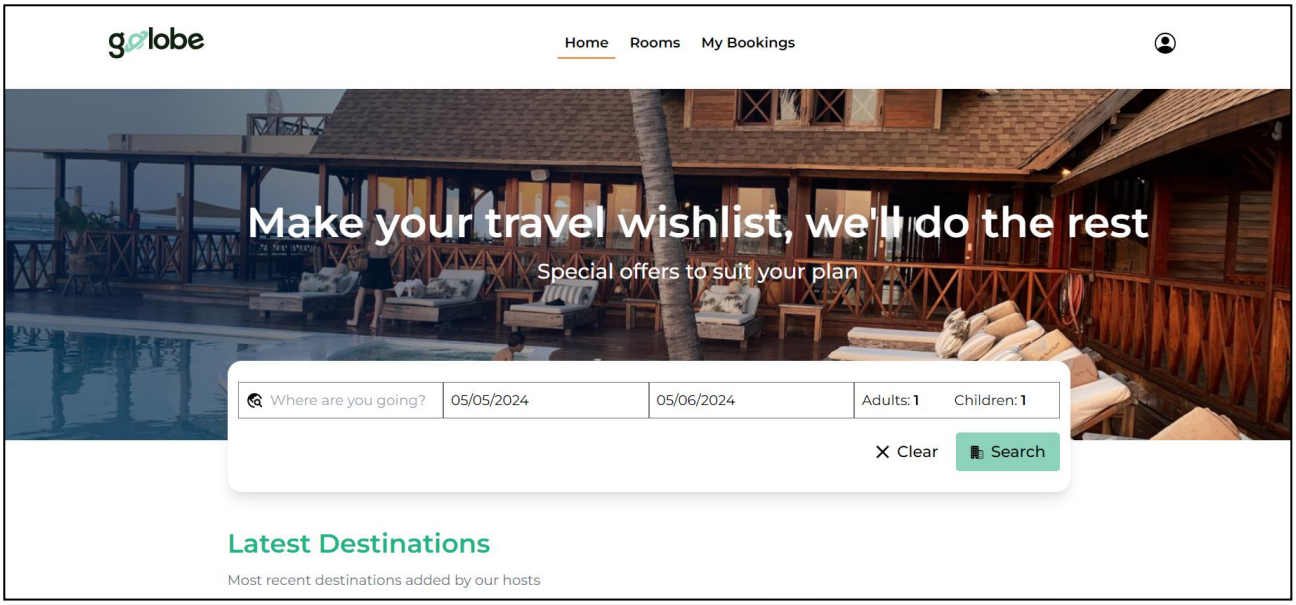
3.1 Login



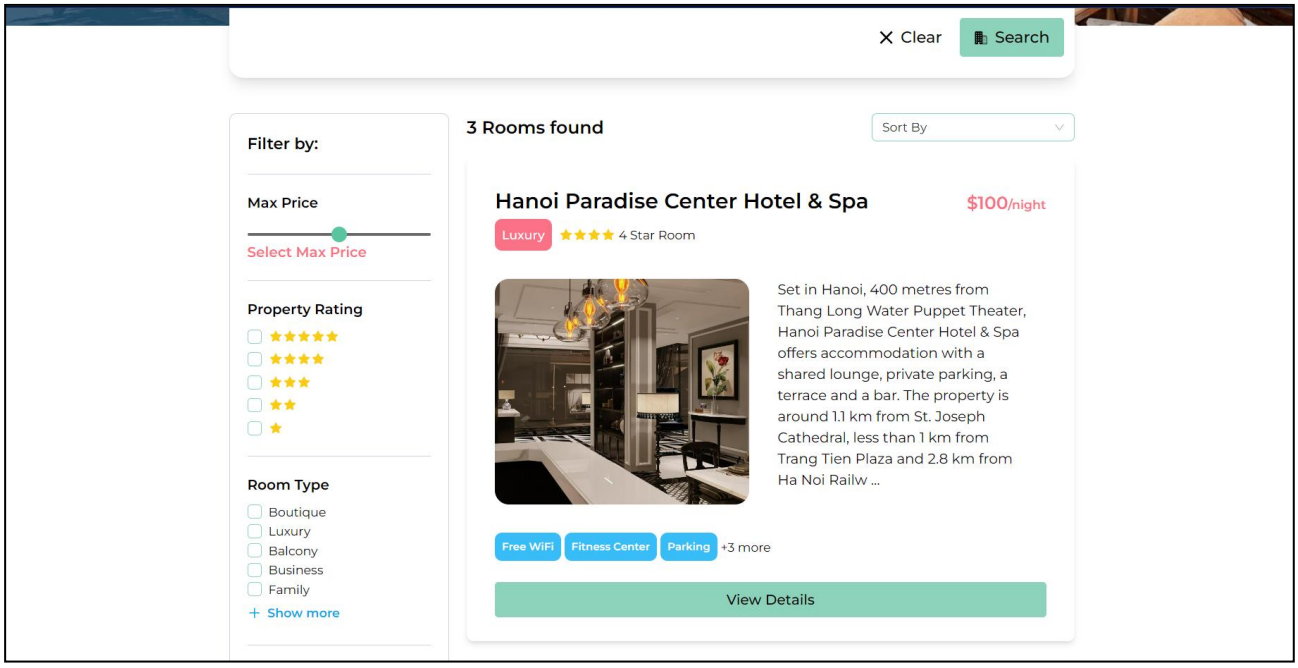
3.2 Register



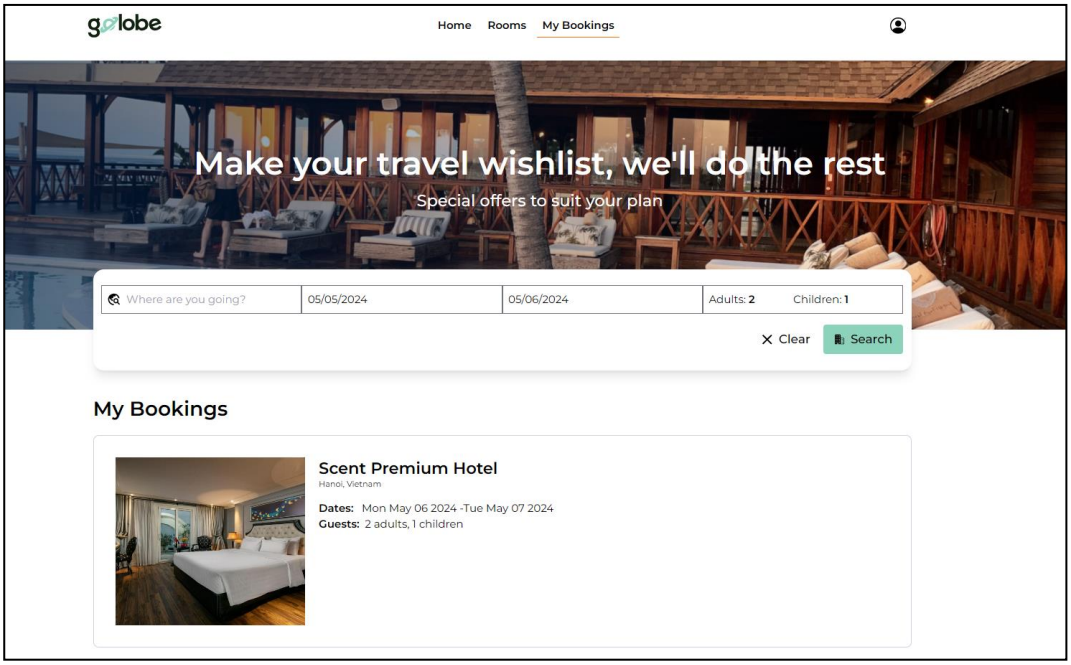
3.3 Landing Page



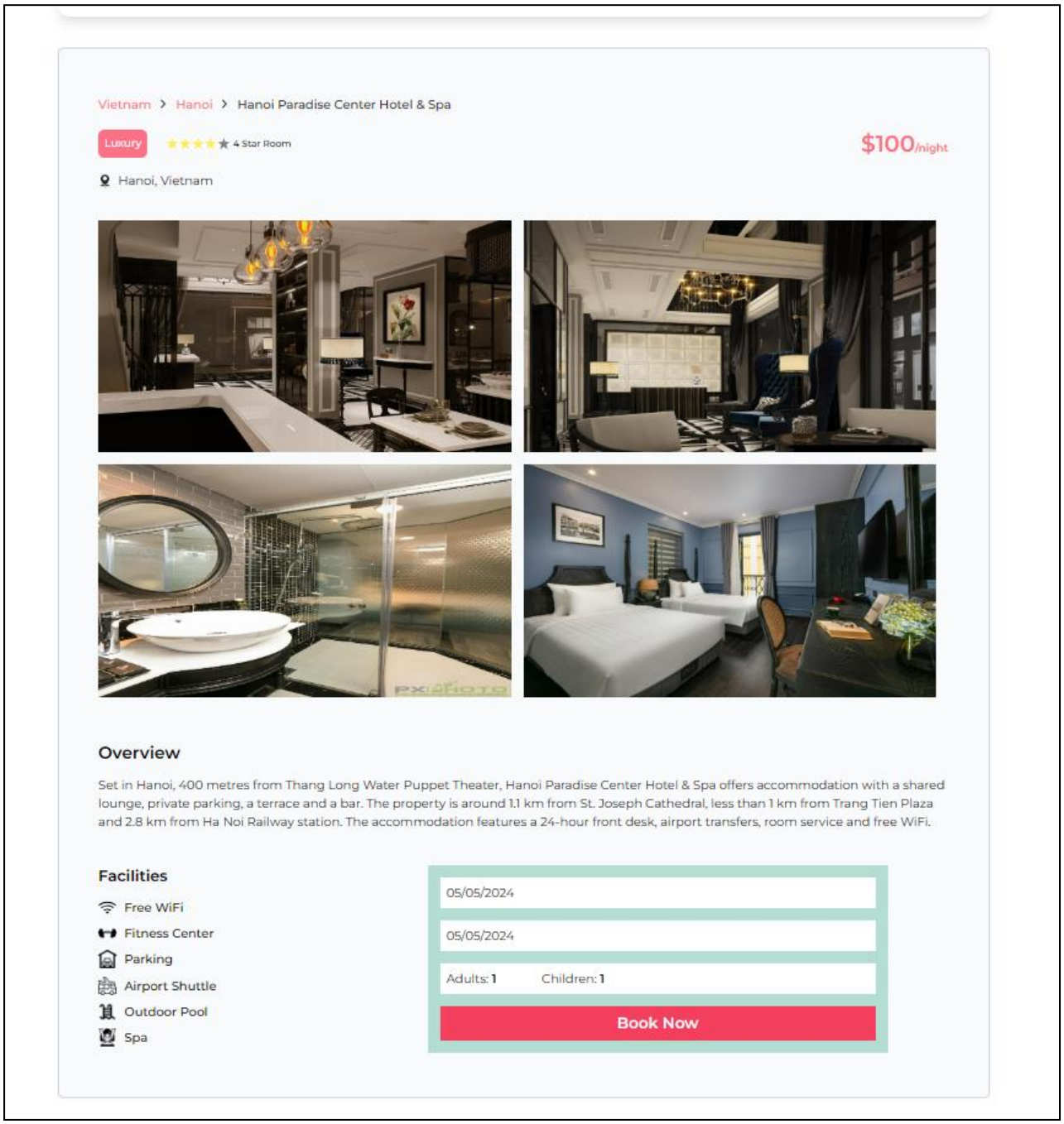
3.4 Search Page



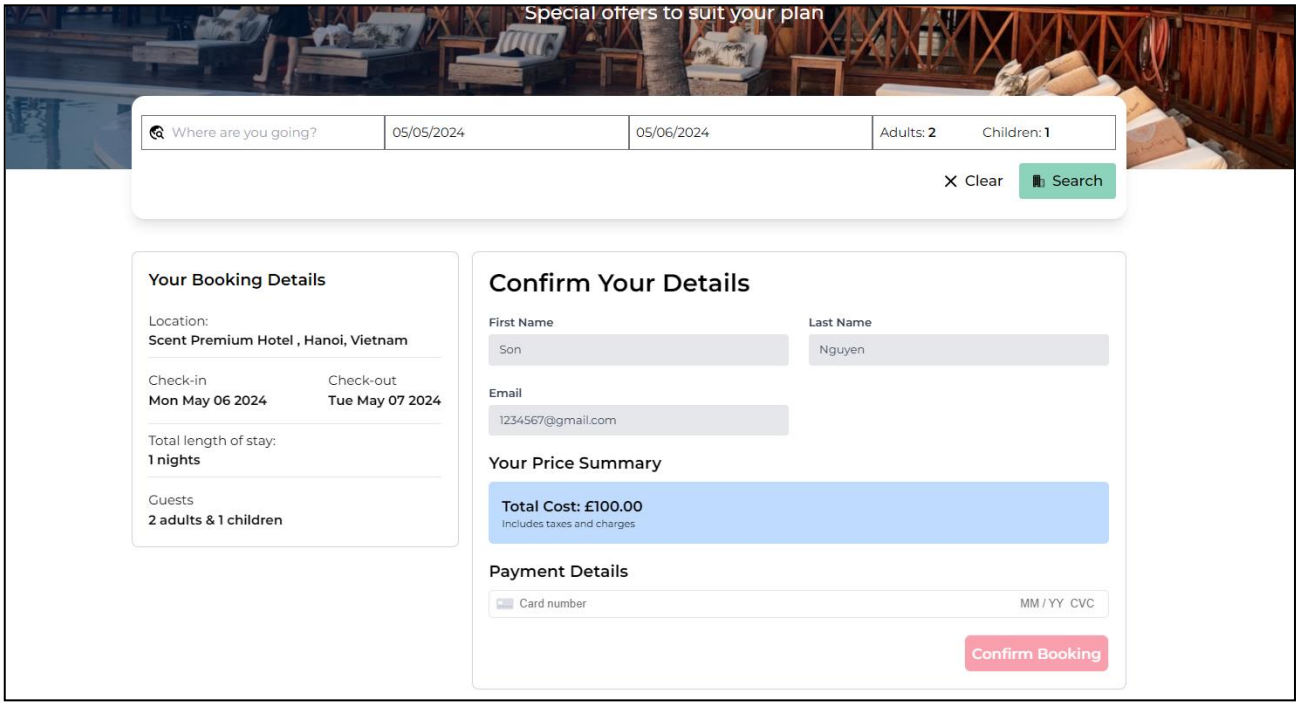
3.5 View Bookings



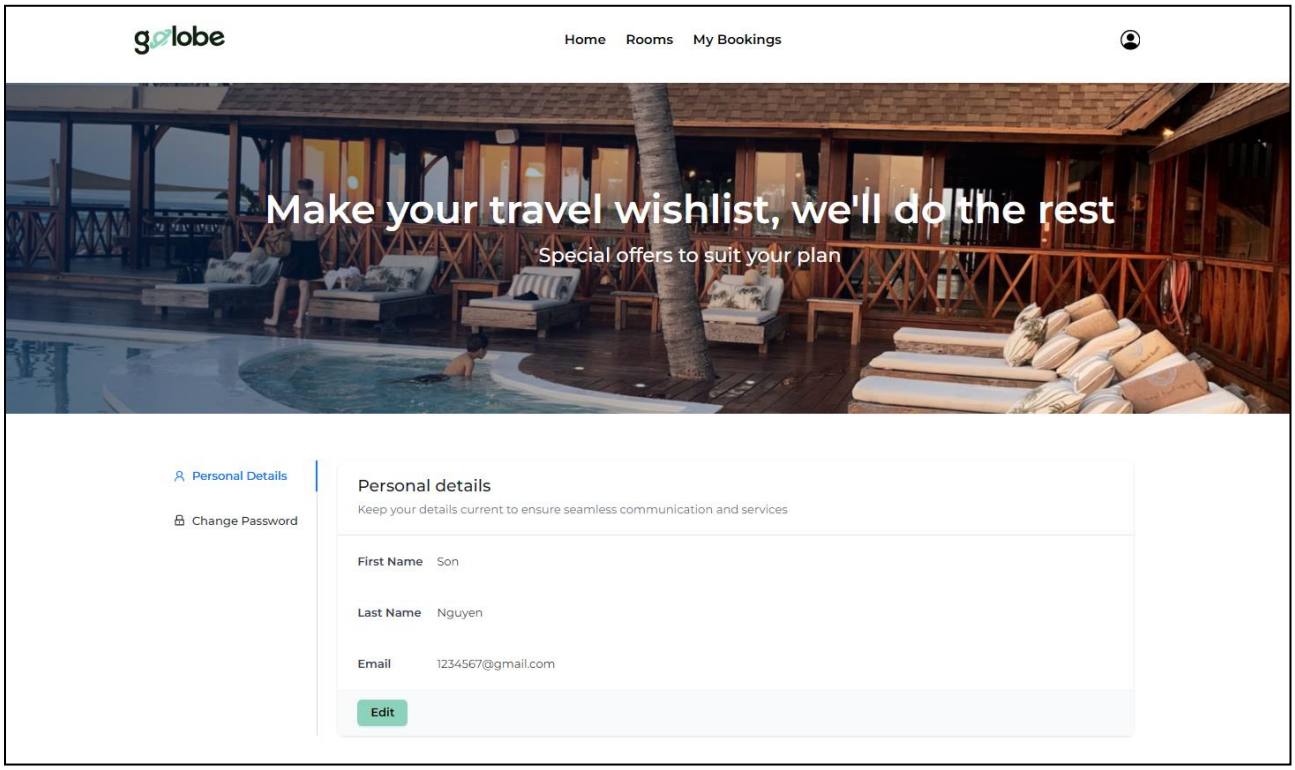
3.6 Room Detail Page



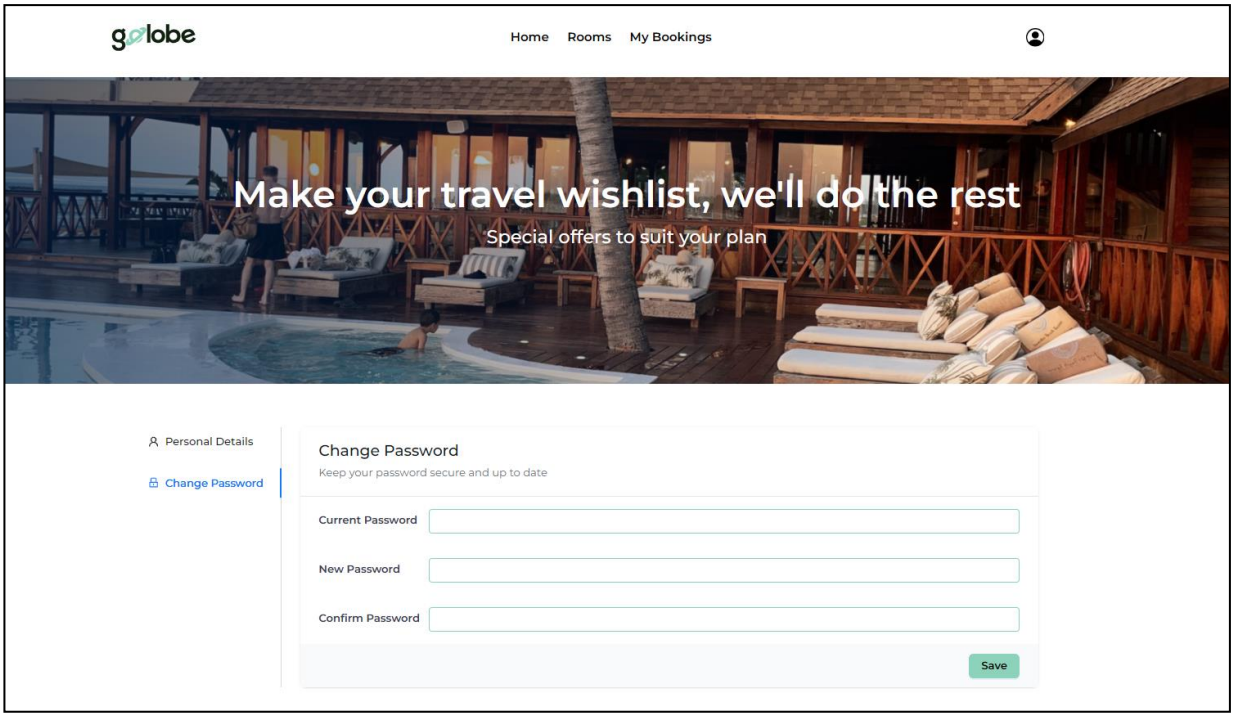
3.7 Booking Confirmation



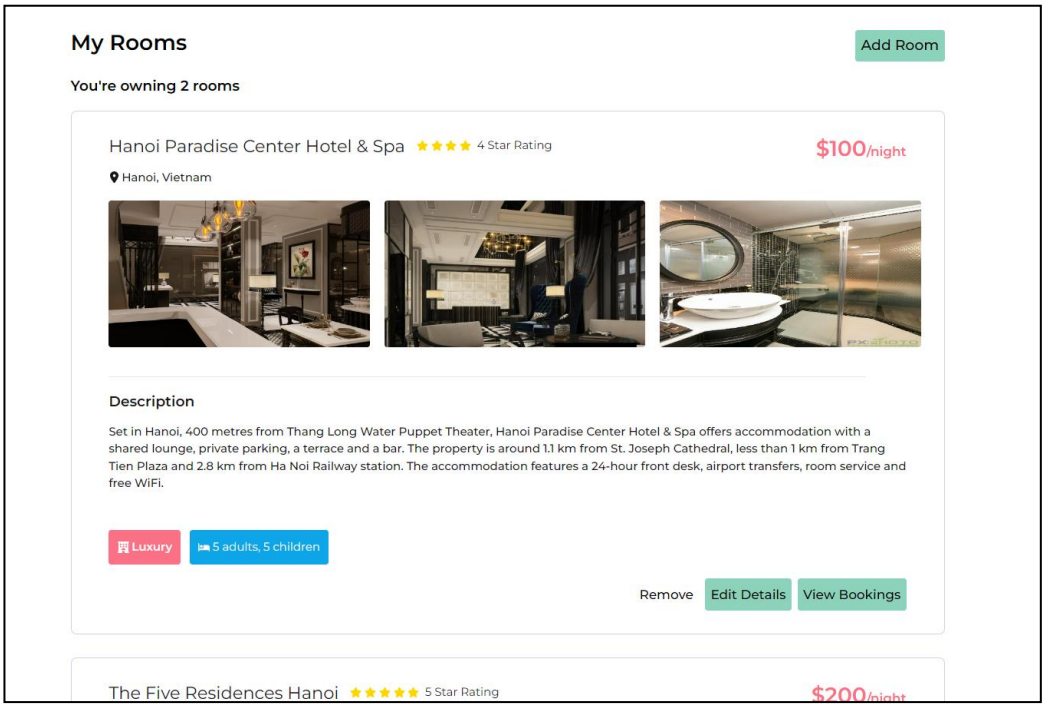
3.8 User Profile



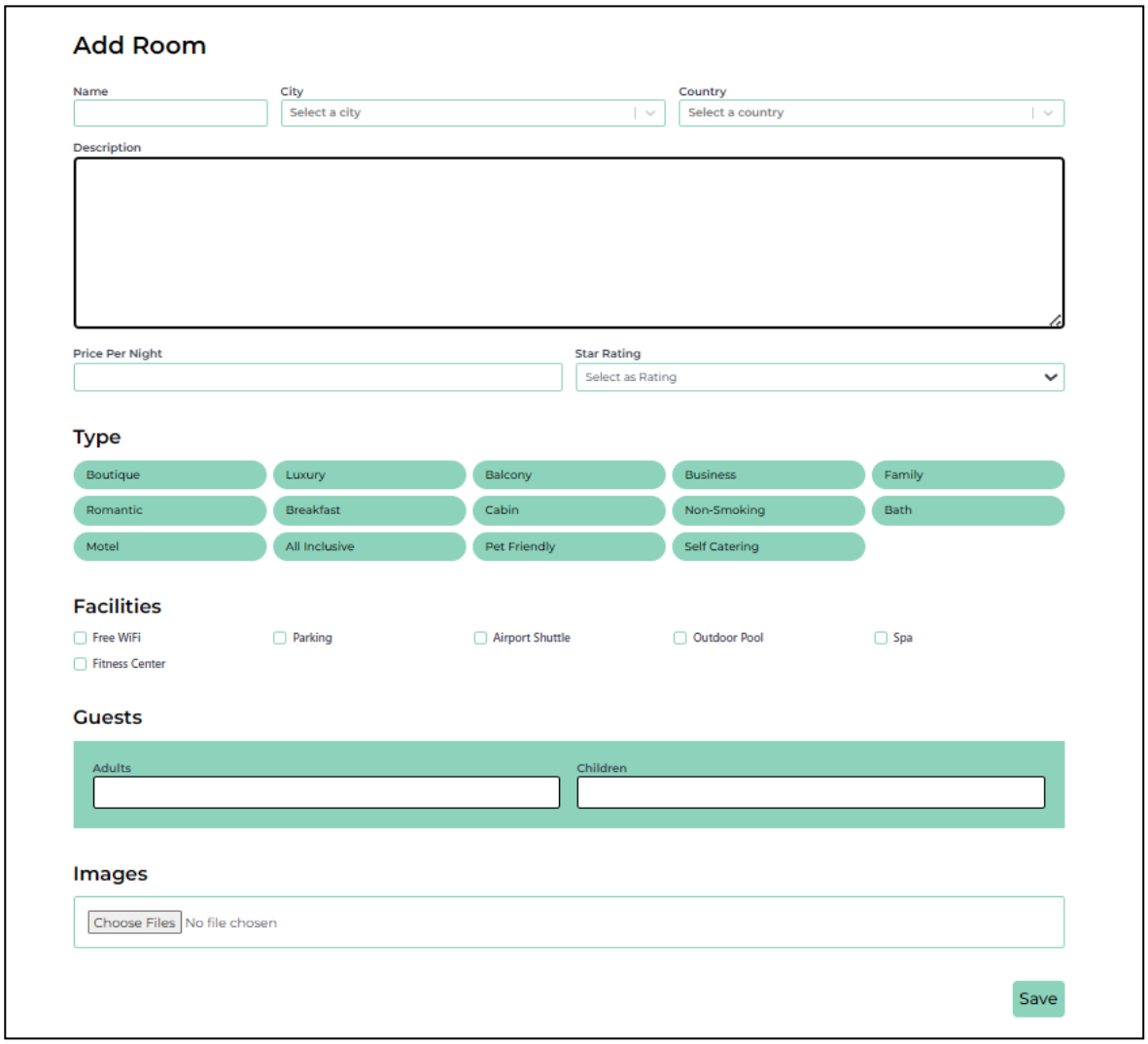
3.9 Change Pasword



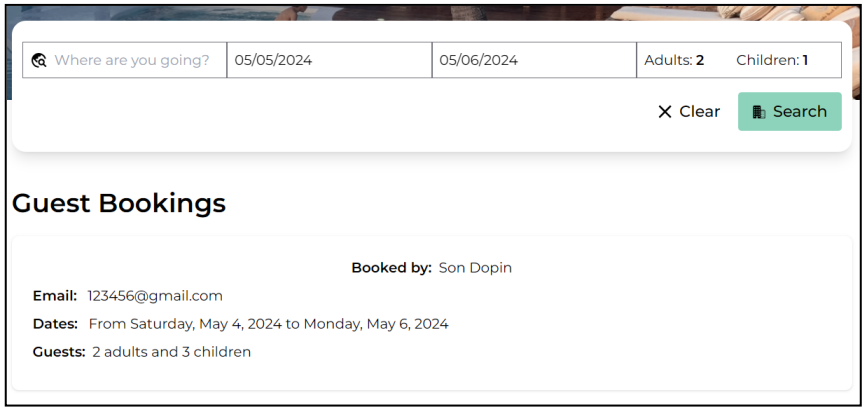
3.10 View Host's Rooms



3.11 Add Room



3.12 View Guests' Bookings



4. Data Structures and Algorithms

There are some notable data structures and alogirthms used in the project:

- Bcrypt hasing function: a cryptographic algorithm that hashes passwords in a way that is secure agianst brute-force attacks.
- sessionStorage: a web storage API that’s used to store data on the client side for the duration of the session. In this project, sessionStorage is used to persist the state of destination, checkIn, checkOut variables across page reloads.

5. External Interfaces

a) GeoDB Cities API:

- Description: The GeoDB Cities API provides a comprehensive database of cities worldwide. It allows us to retrieve city-related data based on user input, such as city name or geographical coordinates.
- Functionality: The API supports various endpoints for querying cities by name, country code, or location. It returns detailed information about cities, including their name, country, population, coordinates, and more.
- Usage in the system: When a user inputs a location, we'll send a request to the GeoDB Cities API to fetch relevant city data. This data will then be used to populate the UI with available location options.

b) REST Countries API:

- Description: The REST Countries API provides information about countries worldwide. It offers data such as country name, capital, population, languages, currencies, and more.
- Functionality: This API allows us to retrieve country-related data based on the city selected by the user. We'll use it to obtain additional information about the country corresponding to the chosen city.
- Usage in the system: After the user selects a city, we'll send a request to the REST Countries API to obtain data about the corresponding country. This data can then be displayed alongside the selected city information in the UI, providing users with additional context.