TRƯỜNG ĐẠI HỌC CMC

**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**BÁO CÁO DỰ ÁN CUỐI KỲ**

**MÔN KỸ NĂNG LẬP TRÌNH NÂNG CAO**

ĐỀ TÀI: QUẢN LÝ KHÁCH SẠN

| Sinh viên : | Nguyễn Đình Việt ( BIT220169 ) |
| --- | --- |
|  | Phùng Quang Trà ( BIT220152 ) |
|  | Phạm Đình Đức ( BIT220034 ) |
| Lớp: | 22IT2 |
| Giảng viên: | Nguyễn Đức Giang |

Hà Nội, tháng 8 năm 2024

**Giới thiệu**

**Mục tiêu:** Mục tiêu của đề tài là phát triển một hệ thống quản lý khách sạn hiện đại, sử dụng công nghệ Spring Boot để xây dựng phần backend, nơi xử lý các yêu cầu logic nghiệp vụ và quản lý dữ liệu, và Bootstrap cho phần frontend, giúp tạo ra giao diện người dùng thân thiện và dễ sử dụng. Hệ thống sẽ cho phép quản lý toàn bộ thông tin về phòng khách sạn, người dùng, và đơn đặt phòng một cách dễ dàng và hiệu quả, đồng thời đảm bảo an toàn thông tin và khả năng mở rộng trong tương lai.

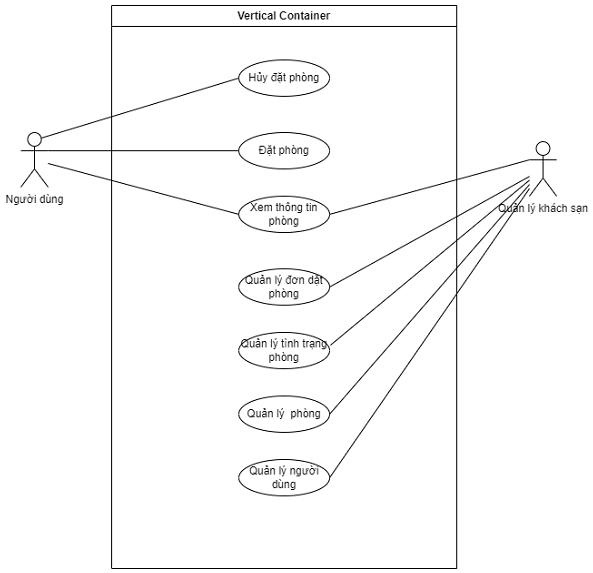
**Đối tượng sử dụng:** Hệ thống quản lý khách sạn này được thiết kế để phục vụ nhu cầu của các quản trị viên khách sạn, những người cần quản lý thông tin về phòng và khách hàng, những người chịu trách nhiệm quản lý đặt phòng và hỗ trợ khách hàng. Ngoài ra, hệ thống còn hỗ trợ các bên liên quan khác, như quản lý tài chính, đội ngũ bảo trì, và bộ phận nhân sự trong việc theo dõi và báo cáo hoạt động của khách sạn.

**Tổng quan về hệ thống**

#### **Mục tiêu người dùng**

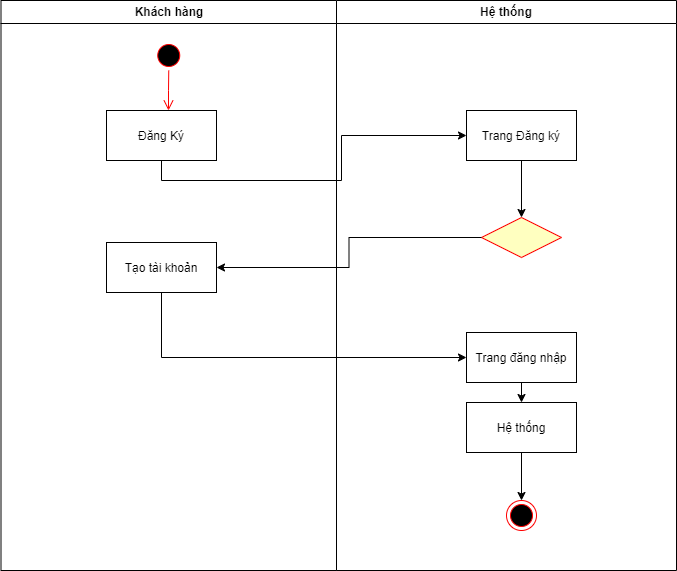
Hệ thống được thiết kế cho hai nhóm người dùng chính:

* Quản trị viên khách sạn (Admin): Có toàn quyền quản lý hệ thống, bao gồm quản lý phòng, người dùng, đơn đặt phòng, và xem báo cáo tổng hợp.
* Khách hàng (User): Có thể xem danh sách phòng, tạo đơn đặt phòng, và quản lý các đơn đặt phòng của mình.

****

**Hoạt động hệ thống:**

**Đăng ký tài khoản :**



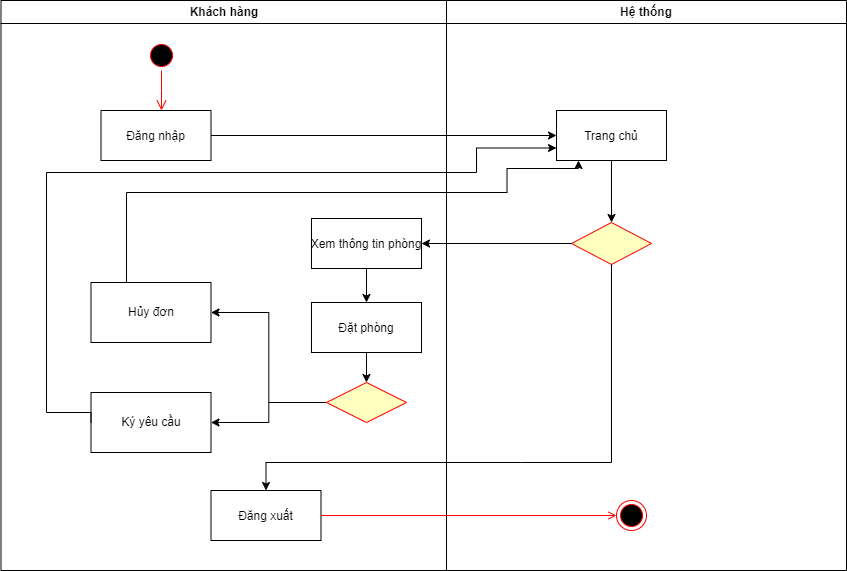
Khách hàng:

Start → Đăng ký → Trang đăng ký:

* →Tạo tài khoản →Trang đăng nhập .

→ Hệ thống → End

**Đặt phòng :**



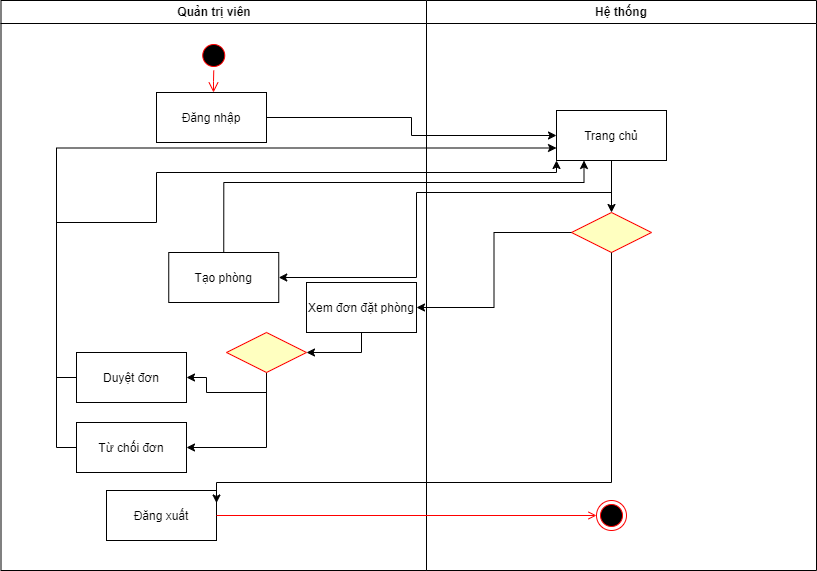
Khách hàng:

Start → Đăng nhập → Hệ thống:

* →Xem thông tin phòng →Đặt phòng .
* →Ký yêu cầu → Quay lại hệ thống.
* →Hủy Đơn → Quay lại hệ thống

→ Đăng xuất → End

**quản trị viên:**



Quản trị viên:

Start → Đăng nhập → Hệ thống:

* →Tạo phòng →Quay lại hệ thống .
* →Xem đơn đặt phòng → Từ chối đơn →Quay lại hệ thống
* →Xem đơn đặt phòng → Duyệt đơn →Quay lại hệ thống

→ Đăng xuất → End

**1.Kiến trúc hệ thống.**

**A.Kiến trúc hệ thống**

Frontend:

Công nghệ sử dụng: Angular

* Mô tả: Angular được sử dụng để xây dựng giao diện người dùng (UI). Đây là một framework mạnh mẽ giúp phát triển các ứng dụng web một trang (SPA) với trải nghiệm người dùng tốt, phản hồi nhanh và dễ dàng quản lý trạng thái ứng dụng. Angular sẽ giao tiếp với API do backend cung cấp để hiển thị và quản lý dữ liệu trong thời gian thực.

Backend:

Công nghệ sử dụng: Spring Boot

* Mô tả: Spring Boot được sử dụng để xây dựng backend, cung cấp các RESTful API cho các chức năng của hệ thống. Spring Boot xử lý logic nghiệp vụ, quản lý kết nối cơ sở dữ liệu, và thực thi các quy trình liên quan đến bảo mật, xác thực, và ủy quyền người dùng.

Cơ sở dữ liệu:

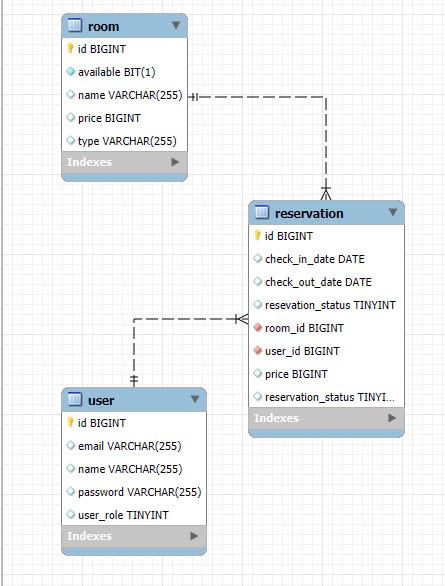
Công nghệ sử dụng: MySQL

* Mô tả: MySQL được sử dụng để lưu trữ tất cả dữ liệu của hệ thống, bao gồm thông tin phòng, người dùng, và đơn đặt phòng. Cơ sở dữ liệu này sẽ được kết nối với Spring Boot để xử lý các truy vấn và lưu trữ dữ liệu một cách hiệu quả.

**B.Mô hình dữ liệu**

Hệ thống quản lý khách sạn bao gồm ba thực thể chính: Room, User, và Reservation. Dưới đây là mô tả các thực thể và mối quan hệ giữa chúng:

**\*.Sơ đồ thực thể:**



**1.Thực thể Room :**

Thuộc tính chính:

⦁ id: Mã định danh duy nhất cho mỗi phòng.

⦁ name: Tên của phòng, ví dụ: "Deluxe Room", "Standard Room".

⦁ type: Loại phòng, ví dụ: Deluxe, Standard, Suite.

⦁ price: Giá thuê phòng tính theo đêm.

⦁ available: Trạng thái của phòng, xác định liệu phòng có sẵn để đặt hay không.

**2.Thực thể User:**

Thuộc tính chính:

⦁ id: Mã định danh duy nhất cho mỗi người dùng.

⦁ email: Địa chỉ email của người dùng, đóng vai trò như một định danh duy nhất.

⦁ name: Tên của người dùng.

⦁ password: Mật khẩu được mã hóa của người dùng.

⦁ userRole: Vai trò của người dùng trong hệ thống, ví dụ: quản trị viên, lễ tân, khách hàng.

**3.Thực thể Reservation:**

Thuộc tính chính:

⦁ id: Mã định danh duy nhất cho mỗi đơn đặt phòng.

⦁ checkInDate: Ngày khách nhận phòng.

⦁ checkOutDate: Ngày khách trả phòng.

⦁ price: Tổng chi phí của đơn đặt phòng.

⦁ reservationStatus: Trạng thái của đơn đặt phòng, ví dụ: đã xác nhận, đã hủy.

⦁ room\_id: Khóa ngoại liên kết với thực thể Room.

⦁ user\_id: Khóa ngoại liên kết với thực thể User.

**2.Phát triển hệ thống.**

**2.1 Phát triển Backend**

**2.1.1 Lớp thực thể (entity):**

**Thực thể Reservation:**

private Long id;

private LocalDate checkInDate;

private LocalDate checkOutDate;

private Long price;

private ReservationStatus reservationStatus;

**Thực thế Room :**

private Long id;

private String name;

private String type;

private Long price;

private boolean available;

**Thực thể User:**

private Long id;

private String email;

private String password;

private String name;

private UserRole userRole;

**2.1.2 Cơ sở dữ liệu :** sử dụng JPA Repository để tương tác với cơ sở dữ liệu.

spring.datasource.url=jdbc:mysql://localhost:3306/hotel\_db

spring.datasource.username=root

spring.datasource.password=1234567

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

**Mục đích :**

-Cấu hình cơ sở dữ liệu :

⦁ Tạo url kết nối csdl và giao thức kết nối

⦁ Xác thực người dùng kết nối csdl

⦁ Đặt lớp Driver JDBC cho MySQL.

-Cấu hình Hibernate và JPA:

⦁ Tạo tương tác với csdl, giúp thay đổi trong csdl khi có sự thay đổi trong các entity

⦁ Kiểm tra truy vấn csdl

⦁ Áp dụng chính xác các tương tác với csdl

**2.1.3 Controllers**

**1. RoomController**

**Chức năng:**

* Quản lý thông tin phòng (Room): thêm, cập nhật, xóa, và lấy danh sách phòng.

**API Endpoints:**

⦁ GET /api/rooms: Lấy danh sách tất cả các phòng.

⦁ GET /api/rooms/{id}: Lấy thông tin chi tiết của một phòng theo ID.

⦁ POST /api/rooms: Thêm một phòng mới.

⦁ PUT /api/rooms/{id}: Cập nhật thông tin của một phòng theo ID.

⦁ DELETE /api/rooms/{id}: Xóa một phòng theo ID.

**2. UserController**

**Chức năng:**

* Quản lý thông tin người dùng (User): thêm, cập nhật, xóa, và lấy thông tin người dùng.

**API Endpoints**:

⦁ GET /api/users: Lấy danh sách tất cả người dùng.

⦁ GET /api/users/{id}: Lấy thông tin chi tiết của một người dùng theo ID.

⦁ POST /api/users: Thêm một người dùng mới.

⦁ PUT /api/users/{id}: Cập nhật thông tin của một người dùng theo ID.

⦁ DELETE /api/users/{id}: Xóa một người dùng theo ID.

**3. ReservationController**

**Chức năng:**

* Quản lý đơn đặt phòng (Reservation): thêm, cập nhật, hủy, và lấy danh sách đơn đặt phòng.

**API Endpoints:**

⦁ GET /api/reservations: Lấy danh sách tất cả đơn đặt phòng.

⦁ GET /api/reservations/{id}: Lấy thông tin chi tiết của một đơn đặt phòng theo ID.

⦁ POST /api/reservations: Thêm một đơn đặt phòng mới.

⦁ PUT /api/reservations/{id}: Cập nhật thông tin của một đơn đặt phòng theo ID.

⦁ DELETE /api/reservations/{id}: Hủy một đơn đặt phòng theo ID.

**2.2 Phát triển Frontend**

Công nghệ sử dụng: Angular

Mô tả:

⦁ Angular là một framework mạnh mẽ được sử dụng để xây dựng giao diện người dùng (UI) cho ứng dụng quản lý khách sạn. Với Angular, bạn có thể phát triển các ứng dụng web một trang (SPA) có trải nghiệm người dùng tốt, khả năng phản hồi nhanh, và dễ dàng quản lý trạng thái của ứng dụng.

**2.2.1 Components**

**a. Room Components:**

⦁ room-list.component.ts: Hiển thị danh sách các phòng.

⦁ Lấy dữ liệu từ RoomService và hiển thị danh sách các phòng.

⦁ Cung cấp chức năng lọc, tìm kiếm, và phân trang.

⦁ room-detail.component.ts: Hiển thị chi tiết một phòng.

⦁ Nhận ID của phòng từ URL và gọi RoomService để lấy thông tin chi tiết của phòng.

⦁ room-form.component.ts: Form để thêm mới hoặc chỉnh sửa phòng.

⦁ Sử dụng ReactiveFormsModule để quản lý form và validate dữ liệu.

⦁ Gọi RoomService để lưu hoặc cập nhật thông tin phòng.

**b. User Components:**

⦁ user-list.component.ts: Hiển thị danh sách người dùng.

⦁ Hiển thị danh sách tất cả người dùng và cho phép tìm kiếm, lọc, và phân trang.

⦁ user-detail.component.ts: Hiển thị chi tiết một người dùng.

⦁ Lấy dữ liệu chi tiết của người dùng từ UserService dựa trên ID.

⦁ user-form.component.ts: Form để thêm mới hoặc chỉnh sửa người dùng.

⦁ Sử dụng reactive forms để tạo form và validate dữ liệu người dùng.

**c. Reservation Components:**

⦁ reservation-list.component.ts: Hiển thị danh sách đơn đặt phòng.

⦁ Hiển thị danh sách tất cả đơn đặt phòng, cùng với các chức năng tìm kiếm, lọc, và phân trang.

⦁ reservation-detail.component.ts: Hiển thị chi tiết một đơn đặt phòng.

⦁ Lấy chi tiết đơn đặt phòng từ ReservationService dựa trên ID.

⦁ reservation-form.component.ts: Form để thêm mới hoặc chỉnh sửa đơn đặt phòng.

⦁ Cho phép người dùng tạo mới hoặc cập nhật thông tin đơn đặt phòng.

**2.2.2 Services**

**a. RoomService:**

Quản lý các API call liên quan đến phòng, bao gồm:

⦁ getRooms(): Lấy danh sách các phòng.

⦁ getRoomById(id: number): Lấy thông tin chi tiết của một phòng.

⦁ createRoom(room: Room): Tạo một phòng mới.

⦁ updateRoom(id: number, room: Room): Cập nhật thông tin của phòng.

⦁ deleteRoom(id: number): Xóa một phòng.

**b. UserService:**

Quản lý các API call liên quan đến người dùng:

⦁ getUsers(): Lấy danh sách người dùng.

⦁ getUserById(id: number): Lấy thông tin chi tiết của một người dùng.

⦁ createUser(user: User): Tạo mới một người dùng.

⦁ updateUser(id: number, user: User): Cập nhật thông tin người dùng.

⦁ deleteUser(id: number): Xóa một người dùng.

**c. ReservationService:**

Quản lý các API call liên quan đến đơn đặt phòng:

⦁ getReservations(): Lấy danh sách đơn đặt phòng.

⦁ getReservationById(id: number): Lấy thông tin chi tiết của một đơn đặt phòng.

⦁ createReservation(reservation: Reservation): Tạo mới một đơn đặt phòng.

⦁ updateReservation(id: number, reservation: Reservation): Cập nhật thông tin đơn đặt phòng.

⦁ deleteReservation(id: number): Hủy một đơn đặt phòng.

**2.2.3 Routing**

Angular sử dụng RouterModule để định nghĩa các tuyến đường cho ứng dụng. Mỗi tuyến đường sẽ liên kết với một thành phần tương ứng

\***Nhánh APP.js**

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { RegisterComponent } from './auth/components/register/register.component';

import { LoginComponent } from './auth/components/login/login.component';

const routes: Routes = [

{path:'register', component:RegisterComponent},

{path:'', component:LoginComponent},

{ path: 'customer', loadChildren: () => import('./modules/customer/customer.module').then(m => m.CustomerModule) },

{ path: 'admin', loadChildren: () => import('./modules/admin/admin.module').then(m => m.AdminModule) },

];

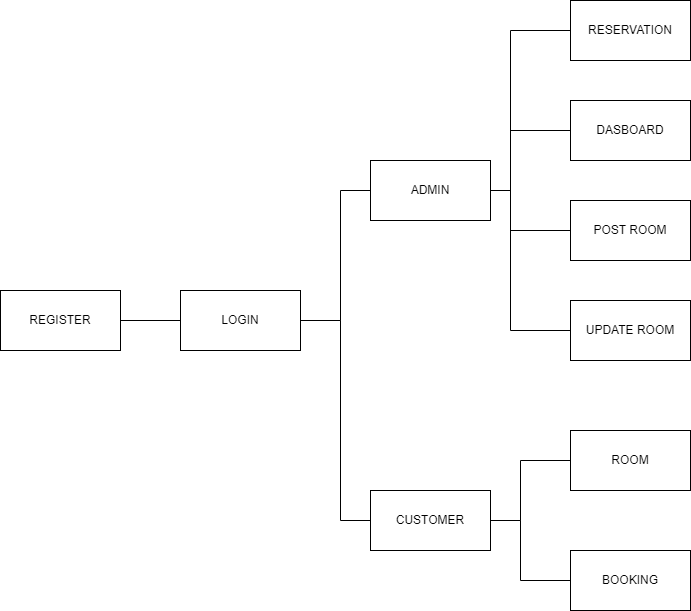
@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }



**2.2.4 Kết nối Backend**

Angular sẽ giao tiếp với backend qua các dịch vụ bằng cách sử dụng HttpClientModule để thực hiện các HTTP request tới API.

const BASIC\_URL = "<http://localhost:8080/>";

@Injectable({

providedIn: 'root'

})

export class AuthService {

constructor(private http: HttpClient) { }

register(signupRequest:any): Observable<any>{

return this.http.post(BASIC\_URL + "api/auth/signup", signupRequest );

}

login(loginRequest:any): Observable<any>{

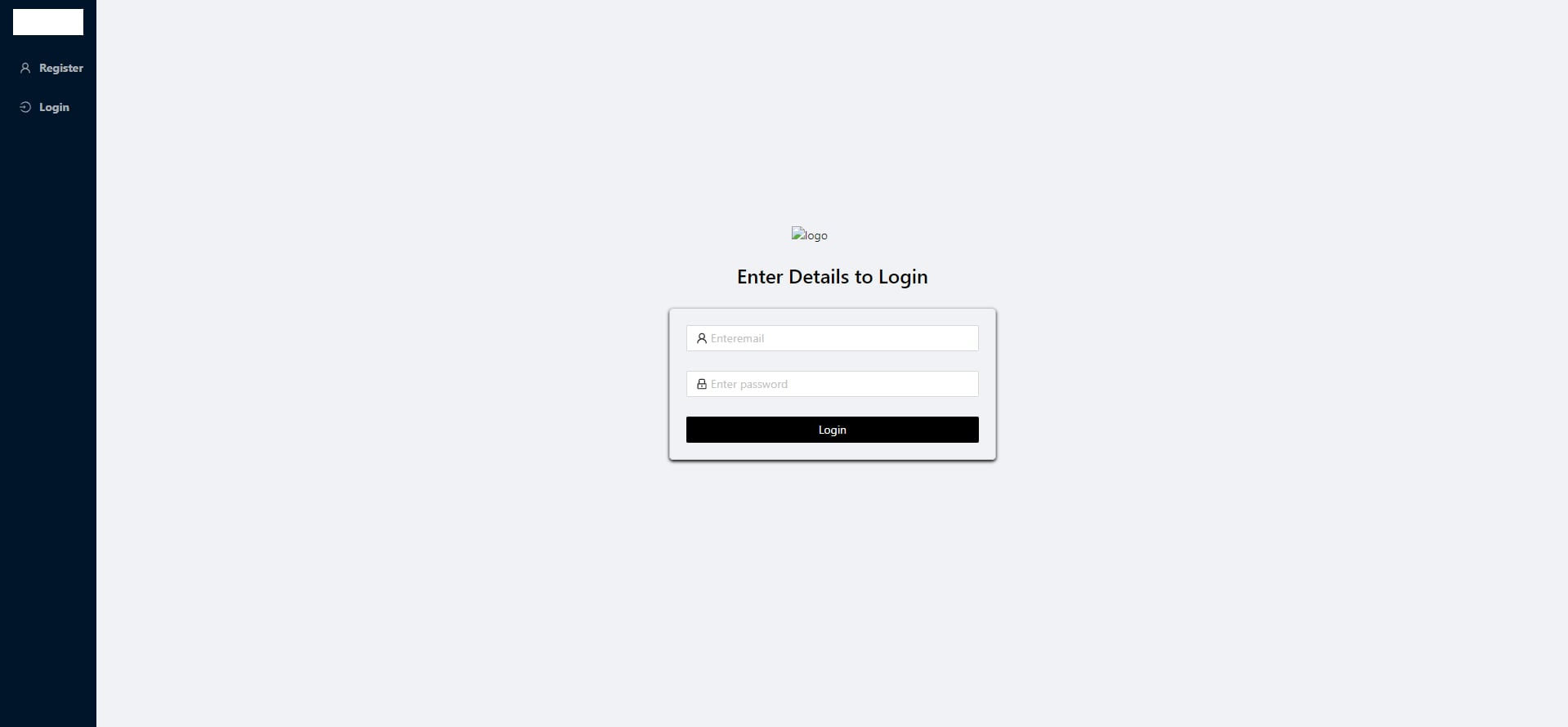
return this.http.post(BASIC\_URL + "api/auth/login", loginRequest);

}

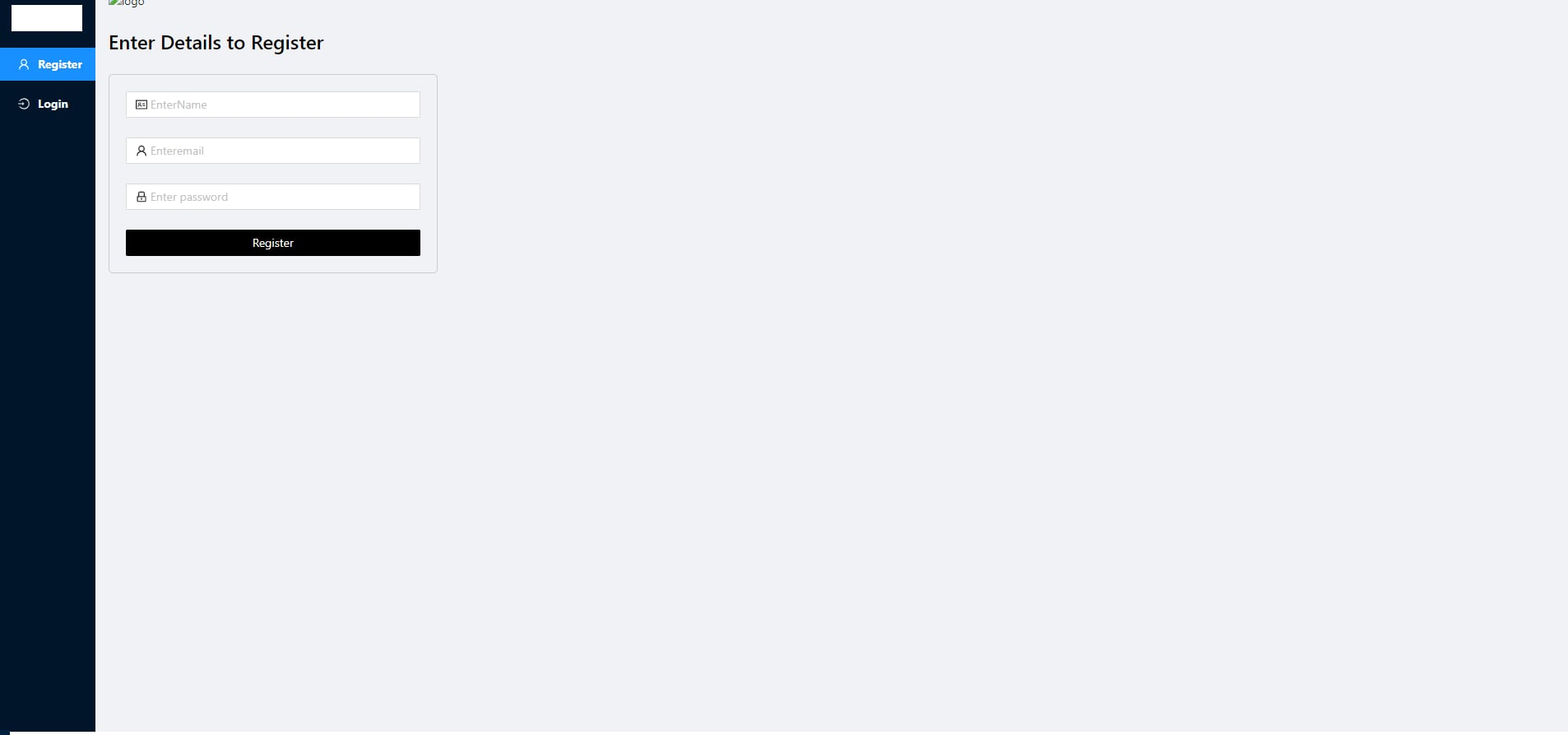
}

**2.2.5 Thiết kế giao diện (UI)**

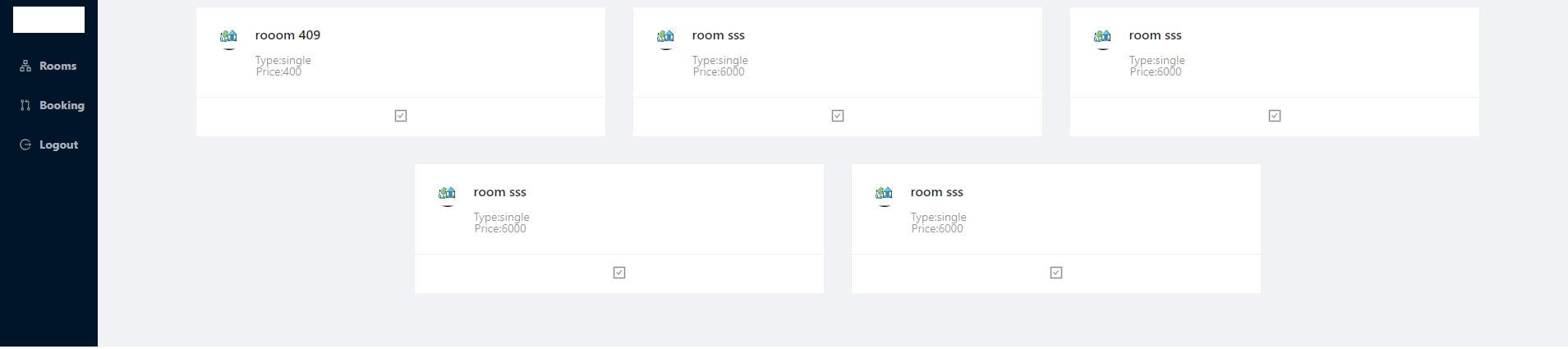
Trang đăng nhập:



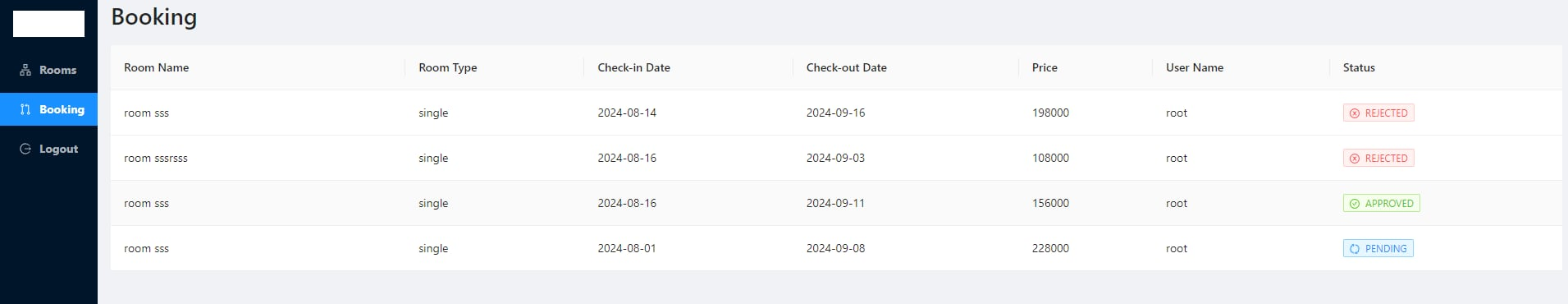
Trang đăng ký:



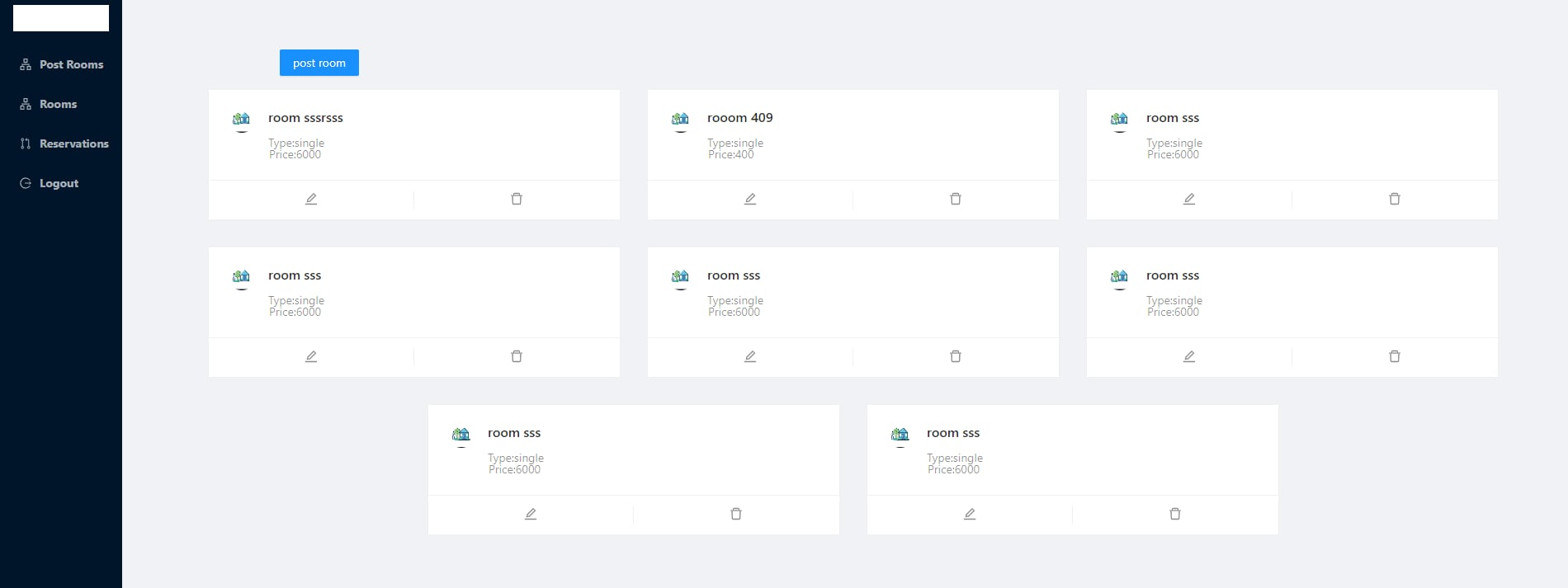
Trang chọn phòng:



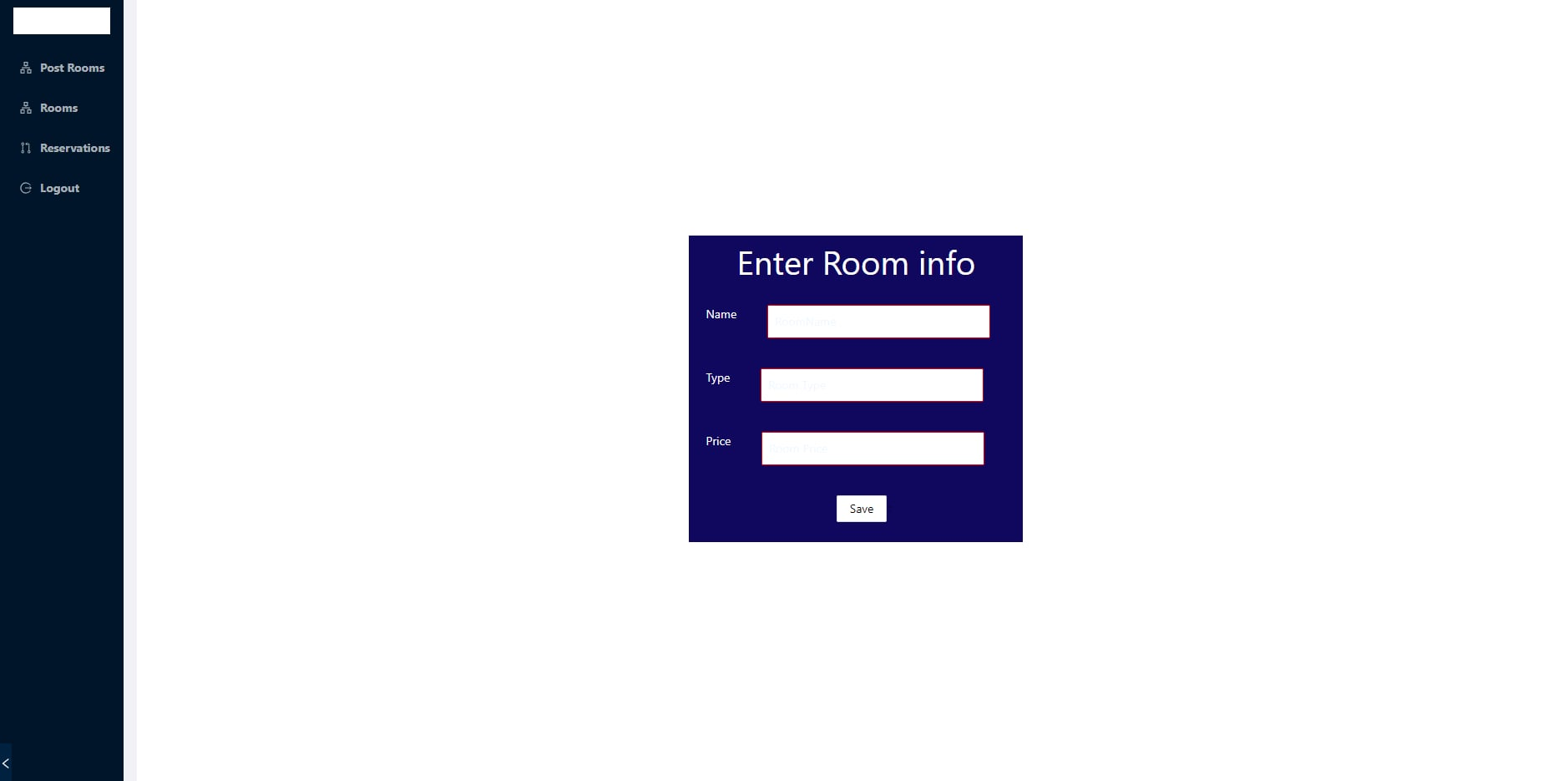
Trang đặt lịch sử đặt phòng:



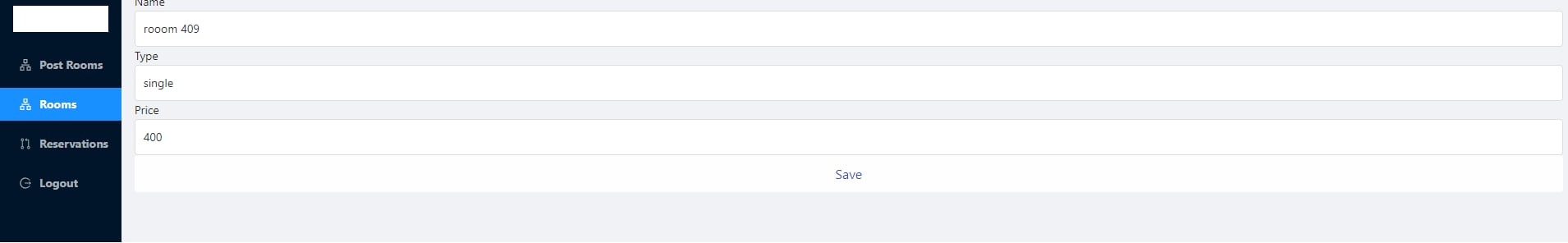
Trang quản lý phòng:



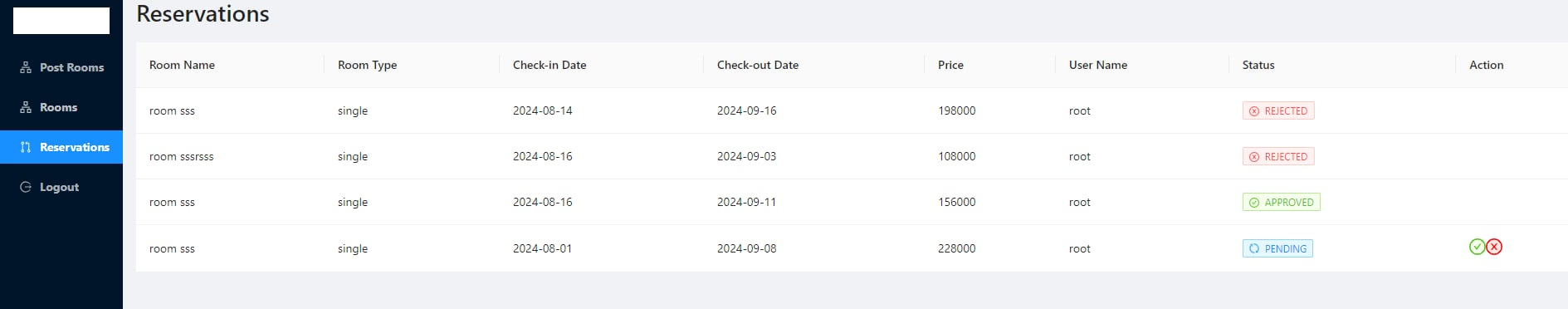
Trang tạo phòng:



Trang sửa phòng:



quản lý đặt phòng:



**2.3 Cơ sở dữ liệu (Database)**

Công nghệ sử dụng: MySQL

Mô tả:

⦁ MySQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được sử dụng rộng rãi trong các ứng dụng web để lưu trữ và quản lý dữ liệu. Trong hệ thống quản lý khách sạn, MySQL được chọn để lưu trữ tất cả các dữ liệu liên quan đến các phòng (Room), người dùng (User), và đơn đặt phòng (Reservation).

**Cấu trúc Cơ sở dữ liệu:**

**Bảng Room**: Lưu trữ thông tin về các phòng trong khách sạn.

⦁ id (Primary Key): Định danh duy nhất cho mỗi phòng.

⦁ name: Tên của phòng.

⦁ type: Loại phòng (ví dụ: Deluxe, Standard).

⦁ price: Giá của phòng.

⦁ available: Trạng thái phòng (có sẵn hoặc đã đặt).

**Bảng User**: Lưu trữ thông tin về người dùng trong hệ thống.

⦁ id (Primary Key): Định danh duy nhất cho mỗi người dùng.

⦁ email: Email của người dùng, được sử dụng để đăng nhập.

⦁ password: Mật khẩu đã được mã hóa của người dùng.

⦁ name: Tên của người dùng.

⦁ userRole: Vai trò của người dùng (quản trị viên, lễ tân, khách hàng).

**Bảng Reservation:** Lưu trữ thông tin về các đơn đặt phòng.

⦁ id (Primary Key): Định danh duy nhất cho mỗi đơn đặt phòng.

⦁ checkInDate: Ngày nhận phòng.

⦁ checkOutDate: Ngày trả phòng.

⦁ price: Giá của đơn đặt phòng.

⦁ reservationStatus: Trạng thái của đơn đặt phòng (đã xác nhận, đã hủy).

⦁ room\_id (Foreign Key): Khóa ngoại liên kết với bảng Room.

⦁ user\_id (Foreign Key): Khóa ngoại liên kết với bảng User.

**3. Triển khai và kiểm thử:**

**3.1 Quá trình triển khai**

**Bước 1: Thiết lập Môi trường Phát triển**

* **Cài đặt công cụ phát triển:** Cài đặt các công cụ cần thiết như IntelliJ IDEA hoặc Eclipse cho Spring Boot, và Visual Studio Code hoặc WebStorm cho Angular.
* **Thiết lập cơ sở dữ liệu MySQL:** Tạo cơ sở dữ liệu trong MySQL và cấu hình kết nối trong Spring Boot.
* **Cài đặt Maven:** Sử dụng Maven để quản lý các phụ thuộc cho dự án Spring Boot.

**Bước 2: Phát triển Backend với Spring Boot**

* **Xây dựng mô hình dữ liệu:** Tạo các entity cho các thực thể chính như Room, User, Reservation, và các repository tương ứng để truy xuất dữ liệu từ cơ sở dữ liệu.
* **Triển khai các API REST:** Xây dựng các RESTful API cho các chức năng quản lý phòng, người dùng, và đơn đặt phòng.
* **Bảo mật ứng dụng:** Thêm các lớp bảo mật để quản lý quyền truy cập vào các API, phân quyền cho các vai trò người dùng khác nhau.

**Bước 3: Phát triển Frontend với Angular**

* **Tạo cấu trúc dự án:** Sử dụng Angular CLI để tạo cấu trúc dự án và các thành phần cần thiết.
* **Kết nối với Backend:** Sử dụng HttpClient trong Angular để gọi các API từ Spring Boot, xử lý dữ liệu và hiển thị trên giao diện người dùng.
* **Xây dựng các component UI:** Phát triển các component như danh sách phòng, form đặt phòng, và bảng điều khiển quản trị viên, đảm bảo giao diện thân thiện và dễ sử dụng.

**Bước 4: Tích hợp và Triển khai**

* **Kiểm tra tích hợp:** Đảm bảo rằng frontend và backend hoạt động liền mạch cùng nhau, tất cả các API được gọi đúng cách và dữ liệu được hiển thị chính xác.
* **Triển khai trên môi trường staging:** Triển khai ứng dụng trên một máy chủ staging để thử nghiệm với dữ liệu thực tế.
* **Chuyển giao sản phẩm:** Sau khi hoàn thành kiểm tra trên môi trường staging, chuyển giao sản phẩm cho phía khách hàng hoặc người dùng cuối.

**3.2 kiểm thử**

**Kiểm thử tích hợp (Integration Testing)**

* Mục tiêu: Đảm bảo rằng các thành phần riêng lẻ hoạt động tốt khi kết hợp với nhau.
* Công cụ sử dụng: Spring Boot Test cho backend để kiểm tra các lớp controller, service cùng với cơ sở dữ liệu thật. Angular CLI và Protractor cho frontend để kiểm tra sự tương tác giữa các component.
* Phạm vi: Tích hợp giữa frontend và backend, kiểm tra các luồng chính như đặt phòng, quản lý người dùng và phòng.

**Kiểm thử chức năng (Functional Testing)**

* Mục tiêu: Đảm bảo rằng toàn bộ hệ thống hoạt động theo đúng yêu cầu đã đề ra.
* Phương pháp: Thực hiện các kịch bản sử dụng chính như quản trị viên đăng nhập, quản lý phòng, người dùng đặt phòng, và kiểm tra xem hệ thống phản hồi đúng với từng hành động.
* Phạm vi: Bao gồm toàn bộ các chức năng đã triển khai, như đăng nhập, quản lý phòng, đặt phòng, quản lý người dùng.

**Kiểm thử hồi quy (Regression Testing)**

* Mục tiêu: Đảm bảo rằng các tính năng cũ vẫn hoạt động tốt sau khi thêm hoặc sửa chữa các tính năng mới.
* Phương pháp: Chạy lại toàn bộ các bài kiểm thử đơn vị, tích hợp, và chức năng sau mỗi lần triển khai thay đổi.
* Phạm vi: Toàn bộ hệ thống, đặc biệt chú trọng vào các tính năng có liên quan đến các thay đổi mới.

**4.Kết luận:**

**Tóm tắt:**

Dự án quản lý khách sạn đã đạt được mục tiêu ban đầu là xây dựng một hệ thống quản lý phòng, người dùng, và đơn đặt phòng với khả năng hoạt động ổn định, hiệu quả. Bằng cách sử dụng Spring Boot cho phần backend và Angular cho phần frontend, hệ thống cung cấp giao diện người dùng thân thiện, phản hồi nhanh, và khả năng quản lý dữ liệu theo thời gian thực. MySQL đã được chọn làm hệ quản trị cơ sở dữ liệu, giúp đảm bảo tính toàn vẹn và khả năng mở rộng của dữ liệu.

**Khó khăn:**

Trong quá trình phát triển, nhóm dự án đã gặp phải một số khó khăn đáng kể:

Tích hợp giữa Frontend và Backend: Việc đảm bảo giao tiếp mượt mà giữa Angular và Spring Boot yêu cầu sự hiểu biết sâu về RESTful APIs và cách xử lý CORS (Cross-Origin Resource Sharing).

Quản lý trạng thái phức tạp trong Frontend: Với Angular, việc quản lý trạng thái của các thành phần UI khi dữ liệu thay đổi liên tục là một thách thức, đòi hỏi nhóm phải sử dụng các kỹ thuật như RxJS và các service quản lý trạng thái.

Cấu hình và tối ưu hóa cơ sở dữ liệu: Quản lý các kết nối với cơ sở dữ liệu MySQL, tối ưu hóa các truy vấn để đảm bảo hiệu suất khi xử lý lượng dữ liệu lớn là một phần không thể thiếu trong dự án.

Nhóm đã vượt qua các khó khăn này bằng cách tìm hiểu kỹ hơn về các công nghệ đang sử dụng, tham khảo tài liệu từ các nguồn uy tín, và thực hiện nhiều thử nghiệm để đảm bảo rằng hệ thống hoạt động hiệu quả.

**Hướng phát triển tương lai:**

Để nâng cao hệ thống trong tương lai, một số cải tiến và tính năng có thể được xem xét:

Tích hợp với các cổng thanh toán trực tuyến: Cho phép khách hàng thực hiện thanh toán trực tuyến khi đặt phòng, tăng cường trải nghiệm người dùng.

Xây dựng hệ thống thông báo tự động: Gửi email hoặc SMS tự động nhắc nhở khách hàng về thời gian nhận phòng, trả phòng, hoặc các ưu đãi đặc biệt.

Bổ sung tính năng quản lý khách hàng thân thiết: Hệ thống có thể quản lý điểm thưởng và ưu đãi dành cho khách hàng thường xuyên, giúp tăng cường sự hài lòng và trung thành của khách hàng.

Tối ưu hóa hiệu suất và bảo mật: Tiếp tục tối ưu hóa các truy vấn cơ sở dữ liệu, cải thiện hiệu suất tổng thể, và tăng cường các biện pháp bảo mật như mã hóa dữ liệu, xác thực hai yếu tố (2FA) cho người dùng.

Hỗ trợ đa ngôn ngữ: Mở rộng hệ thống để hỗ trợ nhiều ngôn ngữ khác nhau, giúp khách sạn có thể phục vụ khách hàng quốc tế dễ dàng hơn.

Với các cải tiến này, hệ thống quản lý khách sạn có thể trở nên mạnh mẽ hơn, linh hoạt hơn, và đáp ứng tốt hơn nhu cầu của các doanh nghiệp khách sạn trong tương lai.

**\*Các nguồn tham khảo:**

[Project Demo and Overview | Hotel Management System Project | Spring Boot + Angular + MySQL | Part 1](https://www.youtube.com/watch?v=1v6Rq3GmYoo&list=PLgYFT7gUQL8FXoj-jJe7pEsvgj3-klbwQ)