**Mini Project 2: Paying Guest Accommodation**

Submitted By: Aniket Singh (Emp Id: 2605511)

**SQL**

CREATE DATABASE pg\_accommodation;

USE pg\_accommodation;

CREATE TABLE tenants (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

age INT NOT NULL CHECK (age >= 18),

contact\_number VARCHAR(15) NOT NULL,

locality VARCHAR(100) NOT NULL

);

CREATE TABLE owners (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

age INT NOT NULL CHECK (age >= 18),

contact\_number VARCHAR(15) NOT NULL

);

CREATE TABLE accommodation\_places (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

registration\_number VARCHAR(50) UNIQUE NOT NULL,

built\_up\_area DOUBLE NOT NULL,

rent\_amount DOUBLE NOT NULL,

city VARCHAR(100) NOT NULL,

locality VARCHAR(100) NOT NULL,

availability BOOLEAN NOT NULL,

owner\_id BIGINT,

FOREIGN KEY (owner\_id) REFERENCES owners(id)

);

**Owner.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**import** jakarta.persistence.\*;

**import** jakarta.validation.constraints.Email;

**import** jakarta.validation.constraints.Min;

**import** java.util.List;

@Entity

@Table(name = "owners")

**public** **class** Owner {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

@Column(nullable = **false**)

**private** String name;

@Email

@Column(nullable = **false**, unique = **true**)

**private** String email;

@Min(18)

@Column(nullable = **false**)

**private** **int** age;

@Column(name = "contact\_number", nullable = **false**)

**private** String contactNumber;

@OneToMany(mappedBy = "owner", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

**private** List<AccommodationPlace> places;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getContactNumber() {

**return** contactNumber;

}

**public** **void** setContactNumber(String contactNumber) {

**this**.contactNumber = contactNumber;

}

**public** List<AccommodationPlace> getPlaces() {

**return** places;

}

**public** **void** setPlaces(List<AccommodationPlace> places) {

**this**.places = places;

}

}

**Tenant.java**

**package** com.example.demo;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

**import** jakarta.validation.constraints.Email;

**import** jakarta.validation.constraints.Min;

@Entity

@Table(name = "tenants")

**public** **class** Tenant {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

@Column(nullable = **false**)

**private** String name;

@Email

@Column(nullable = **false**, unique = **true**)

**private** String email;

@Min(18)

@Column(nullable = **false**)

**private** **int** age;

@Column(name = "contact\_number", nullable = **false**)

**private** String contactNumber;

@Column(nullable = **false**)

**private** String locality;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getContactNumber() {

**return** contactNumber;

}

**public** **void** setContactNumber(String contactNumber) {

**this**.contactNumber = contactNumber;

}

**public** String getLocality() {

**return** locality;

}

**public** **void** setLocality(String locality) {

**this**.locality = locality;

}

}

**AccommodationPlace.java**

**package** com.example.demo;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.FetchType;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.JoinColumn;

**import** jakarta.persistence.ManyToOne;

**import** jakarta.persistence.Table;

**import** com.example.demo.\*;

@Entity

@Table(name = "accommodation\_places")

**public** **class** AccommodationPlace {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

@Column(name = "registration\_number", nullable = **false**, unique = **true**)

**private** String registrationNumber;

@Column(name = "built\_up\_area", nullable = **false**)

**private** **double** builtUpArea;

@Column(name = "rent\_amount", nullable = **false**)

**private** **double** rentAmount;

@Column(nullable = **false**)

**private** String city;

@Column(nullable = **false**)

**private** String locality;

@Column(nullable = **false**)

**private** **boolean** availability;

@ManyToOne(fetch = FetchType.***LAZY***)

@JoinColumn(name = "owner\_id", nullable = **false**)

**private** Owner owner;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getRegistrationNumber() {

**return** registrationNumber;

}

**public** **void** setRegistrationNumber(String registrationNumber) {

**this**.registrationNumber = registrationNumber;

}

**public** **double** getBuiltUpArea() {

**return** builtUpArea;

}

**public** **void** setBuiltUpArea(**double** builtUpArea) {

**this**.builtUpArea = builtUpArea;

}

**public** **double** getRentAmount() {

**return** rentAmount;

}

**public** **void** setRentAmount(**double** rentAmount) {

**this**.rentAmount = rentAmount;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getLocality() {

**return** locality;

}

**public** **void** setLocality(String locality) {

**this**.locality = locality;

}

**public** **boolean** isAvailability() {

**return** availability;

}

**public** **void** setAvailability(**boolean** availability) {

**this**.availability = availability;

}

**public** Owner getOwner() {

**return** owner;

}

**public** **void** setOwner(Owner owner) {

**this**.owner = owner;

}

}

**OwnerRepository.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **interface** OwnerRepository **extends** JpaRepository<Owner, Long> {

}

**TenantRepository.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **interface** TenantRepository **extends** JpaRepository<Tenant, Long> {

}

**AccommodationPlaceRepository.java**

**package** com.example.demo;

**import** java.util.List;

**import** com.example.demo.\*;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **interface** AccommodationPlaceRepository **extends** JpaRepository<AccommodationPlace, Long> {

List<AccommodationPlace> findByCityAndAvailability(String city, **boolean** availability);

List<AccommodationPlace> findByLocalityAndAvailability(String locality, **boolean** availability);

}

**OwnerService.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**import** java.util.List;

**public** **interface** OwnerService {

AccommodationPlace addPlace(AccommodationPlace place);

**void** updatePlace(Long placeId, AccommodationPlace updatedPlace);

**void** deletePlace(Long placeId);

AccommodationPlace changeAvailability(Long placeId, **boolean** availability);

List<AccommodationPlace> listPlacesByOwner(Long ownerId);

}

**TenantService.java**

**package** com.example.demo;

**import** java.util.List;

**import** com.example.demo.\*;

**public** **interface** TenantService {

Tenant registerTenant(Tenant tenant);

List<AccommodationPlace> searchPlacesByCity(String city);

List<AccommodationPlace> searchPlacesByLocality(String locality);

AccommodationPlace getPlaceDetails(Long placeId);

Owner getOwnerDetails(Long placeId);

}

**OwnerServiceImpl.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** OwnerServiceImpl **implements** OwnerService {

@Autowired

**private** OwnerRepository ownerRepository;

@Autowired

**private** AccommodationPlaceRepository placeRepository;

@Override

**public** AccommodationPlace addPlace(AccommodationPlace place) {

**return** placeRepository.save(place);

}

@Override

**public** **void** updatePlace(Long placeId, AccommodationPlace updatedPlace) {

AccommodationPlace existingPlace = placeRepository.findById(placeId)

.orElseThrow(() -> **new** ResourceNotFoundException("Place not found with id " + placeId));

existingPlace.setBuiltUpArea(updatedPlace.getBuiltUpArea());

existingPlace.setRentAmount(updatedPlace.getRentAmount());

existingPlace.setCity(updatedPlace.getCity());

existingPlace.setLocality(updatedPlace.getLocality());

existingPlace.setAvailability(updatedPlace.isAvailability());

placeRepository.save(existingPlace);

}

@Override

**public** **void** deletePlace(Long placeId) {

placeRepository.deleteById(placeId);

}

@Override

**public** AccommodationPlace changeAvailability(Long placeId, **boolean** availability) {

AccommodationPlace place = placeRepository.findById(placeId)

.orElseThrow(() -> **new** ResourceNotFoundException("Place not found with id " + placeId));

place.setAvailability(availability);

**return** placeRepository.save(place);

}

@Override

**public** List<AccommodationPlace> listPlacesByOwner(Long ownerId) {

Owner owner = ownerRepository.findById(ownerId)

.orElseThrow(() -> **new** ResourceNotFoundException("Owner not found with id " + ownerId));

**return** owner.getPlaces();

}

}

**TenantServiceImpl.java**

**package** com.example.demo;

**import** java.util.List;

**import** com.example.demo.\*;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** TenantServiceImpl **implements** TenantService {

@Autowired

**private** TenantRepository tenantRepository;

@Autowired

**private** AccommodationPlaceRepository placeRepository;

@Autowired

**private** OwnerRepository ownerRepository;

@Override

**public** Tenant registerTenant(Tenant tenant) {

**return** tenantRepository.save(tenant);

}

@Override

**public** List<AccommodationPlace> searchPlacesByCity(String city) {

**return** placeRepository.findByCityAndAvailability(city, **true**);

}

@Override

**public** List<AccommodationPlace> searchPlacesByLocality(String locality) {

**return** placeRepository.findByLocalityAndAvailability(locality, **true**);

}

@Override

**public** AccommodationPlace getPlaceDetails(Long placeId) {

**return** placeRepository.findById(placeId)

.orElseThrow(() -> **new** ResourceNotFoundException("Place not found with id " + placeId));

}

@Override

**public** Owner getOwnerDetails(Long placeId) {

AccommodationPlace place = placeRepository.findById(placeId)

.orElseThrow(() -> **new** ResourceNotFoundException("Place not found with id " + placeId));

**if** (!place.isAvailability()) {

**throw** **new** IllegalStateException("Place is currently occupied");

}

**return** place.getOwner();

}

}

**OwnerController.java**

**package** com.example.demo;

**import** java.util.List;

**import** com.example.demo.\*;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** jakarta.validation.Valid;

@RestController

@RequestMapping("/owner")

**public** **class** OwnerController {

@Autowired

**private** OwnerService ownerService;

@PostMapping("/places/add")

**public** ResponseEntity<AccommodationPlace> addPlace(@Valid @RequestBody AccommodationPlace place) {

AccommodationPlace savedPlace = ownerService.addPlace(place);

**return** ResponseEntity.*ok*(savedPlace);

}

@PutMapping("/places/edit/{id}")

**public** ResponseEntity<AccommodationPlace> editPlace(@PathVariable Long id, @Valid @RequestBody AccommodationPlace updatedPlace) {

ownerService.updatePlace(id, updatedPlace);

**return** ResponseEntity.*ok*(updatedPlace);

}

@DeleteMapping("/places/delete/{id}")

**public** ResponseEntity<Void> deletePlace(@PathVariable Long id) {

ownerService.deletePlace(id);

**return** ResponseEntity.*noContent*().build();

}

@PutMapping("/places/{id}/availability")

**public** ResponseEntity<AccommodationPlace> changeAvailability(@PathVariable Long id, @RequestParam **boolean** availability) {

AccommodationPlace place = ownerService.changeAvailability(id, availability);

**return** ResponseEntity.*ok*(place);

}

@GetMapping("/places")

**public** ResponseEntity<List<AccommodationPlace>> listPlacesByOwner(@RequestParam Long ownerId) {

List<AccommodationPlace> places = ownerService.listPlacesByOwner(ownerId);

**return** ResponseEntity.*ok*(places);

}

}

**TenantController.java**

**package** com.example.demo;

**import** java.util.List;

**import** com.example.demo.\*;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** jakarta.validation.Valid;

@RestController

@RequestMapping("/tenant")

**public** **class** TenantController {

@Autowired

**private** TenantService tenantService;

@PostMapping("/register")

**public** ResponseEntity<Tenant> registerTenant(@Valid @RequestBody Tenant tenant) {

Tenant savedTenant = tenantService.registerTenant(tenant);

**return** ResponseEntity.*ok*(savedTenant);

}

@GetMapping("/search/city/{city}")

**public** ResponseEntity<List<AccommodationPlace>> searchByCity(@PathVariable String city) {

List<AccommodationPlace> places = tenantService.searchPlacesByCity(city);

**return** ResponseEntity.*ok*(places);

}

@GetMapping("/search/locality/{locality}")

**public** ResponseEntity<List<AccommodationPlace>> searchByLocality(@PathVariable String locality) {

List<AccommodationPlace> places = tenantService.searchPlacesByLocality(locality);

**return** ResponseEntity.*ok*(places);

}

@GetMapping("/details/{id}")

**public** ResponseEntity<AccommodationPlace> getPlaceDetails(@PathVariable Long id) {

AccommodationPlace place = tenantService.getPlaceDetails(id);

**return** ResponseEntity.*ok*(place);

}

@GetMapping("/owner/{id}")

**public** ResponseEntity<Owner> getOwnerDetails(@PathVariable Long id) {

Owner owner = tenantService.getOwnerDetails(id);

**return** ResponseEntity.*ok*(owner);

}

}

**GlobalExceptionHandler.java**

**package** com.example.demo;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** org.springframework.context.support.DefaultMessageSourceResolvable;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

**import** org.springframework.web.bind.annotation.ControllerAdvice;

**import** org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice

**public** **class** GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.**class**)

**public** ResponseEntity<ErrorResponse> handleResourceNotFoundException(ResourceNotFoundException ex) {

ErrorResponse errorResponse = **new** ErrorResponse(HttpStatus.***NOT\_FOUND***.value(), ex.getMessage());

**return** **new** ResponseEntity<>(errorResponse, HttpStatus.***NOT\_FOUND***);

}

@ExceptionHandler(MethodArgumentNotValidException.**class**)

**public** ResponseEntity<ErrorResponse> handleValidationExceptions(MethodArgumentNotValidException ex) {

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(DefaultMessageSourceResolvable::getDefaultMessage)

.collect(Collectors.*toList*());

ErrorResponse errorResponse = **new** ErrorResponse(HttpStatus.***BAD\_REQUEST***.value(), String.*join*(", ", errors));

**return** **new** ResponseEntity<>(errorResponse, HttpStatus.***BAD\_REQUEST***);

}

@ExceptionHandler(IllegalStateException.**class**)

**public** ResponseEntity<ErrorResponse> handleIllegalStateException(IllegalStateException ex) {

ErrorResponse errorResponse = **new** ErrorResponse(HttpStatus.***BAD\_REQUEST***.value(), ex.getMessage());

**return** **new** ResponseEntity<>(errorResponse, HttpStatus.***BAD\_REQUEST***);

}

@ExceptionHandler(Exception.**class**)

**public** ResponseEntity<ErrorResponse> handleGenericException(Exception ex) {

ErrorResponse errorResponse = **new** ErrorResponse(HttpStatus.***INTERNAL\_SERVER\_ERROR***.value(), "An unexpected error occurred");

**return** **new** ResponseEntity<>(errorResponse, HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

}

**ErrorResponse.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**public** **class** ErrorResponse {

**private** **int** statusCode;

**private** String message;

**public** ErrorResponse(**int** statusCode, String message) {

**this**.statusCode = statusCode;

**this**.message = message;

}

**public** **int** getStatusCode() {

**return** statusCode;

}

**public** **void** setStatusCode(**int** statusCode) {

**this**.statusCode = statusCode;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

**ResourceNotFoundException.java**

**package** com.example.demo;

**import** com.example.demo.\*;

**public** **class** ResourceNotFoundException **extends** RuntimeException {

**public** ResourceNotFoundException(String message) {

**super**(message);

}

}

**Demo1Application.java**

**package** com.example.demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Demo1Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Demo1Application.**class**, args);

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/pg\_accommodation

spring.datasource.username=root

spring.datasource.password=1234

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

server.port=8090

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>miniproject4</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo-1</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>