Advanced Programming Concepts

Project Report

Semester-V (Batch-2023)

Project ID – APC-2030

Hotel Booking System

Supervised By:

Dr. Mohammad Imran

Submitted By:

Jatil Garg, 2310990615(G8)
Manvir Singh, 2310990652(G8)
Osheen Mahajan, 2310990675(G8)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

The Hotel Booking System is designed to modernize and simplify the process of managing hotel reservations, offering a seamless digital solution for both guests and hotel staff. Traditional methods of booking rooms, maintaining records, and tracking reservations are often time-consuming, error-prone, and lack transparency. This system addresses these challenges by providing a centralized, secure, and efficient platform that streamlines all hotel operations. Guests can effortlessly browse available rooms, check details, make reservations, and cancel booking when necessary.

The system is built using **Spring Boot** and **Spring Core**, which provide a robust and modular framework for application development. **Hibernate** is used for object-relational mapping, ensuring smooth interaction with the **Oracle database**, where all critical data such as user profiles, booking information, and room details are securely stored. The platform also incorporates role-based dashboards, enabling administrators and guests to access relevant features and information according to their roles.

With a responsive design, the system is accessible across desktops ensuring convenience and flexibility for users on the go. Security measures are integrated throughout the platform to protect sensitive user and booking information, while automated processes reduce manual errors and improve operational efficiency.

Overall, the Hotel Booking System bridges the gap between traditional manual booking practices and modern digital requirements in the hospitality industry. By combining efficiency, accuracy, security, and user-friendliness, the system not only simplifies hotel management for staff but also offers guests a convenient and reliable way to book rooms. This project represents a future-ready solution capable of adapting to evolving user needs while maintaining high standards of operational excellence, providing a robust framework for the digital transformation of hotel services.

INDEX

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The **Hotel Booking System** is a web-based application designed to simplify and digitalize the process of searching, managing, and booking hotel rooms. In traditional scenarios, hotel bookings often involve lengthy phone calls, manual registers, or depending on third-party travel agents. These methods are not only time-consuming but also prone to errors such as double bookings, miscommunication, or lack of transparency regarding availability.

Our project resolves these issues by creating a **centralized digital platform** that connects **admins** (who manage hotels, rooms, and locations) with **users** (who search and book rooms). Admins can add multiple hotels, update locations, manage the number of available rooms. Users, on the other hand, get a simple and intuitive interface to register, log in, browse hotels, and book rooms according to their needs.

The platform is built to be **secure, scalable, and user-friendly**, ensuring smooth functionality for both stakeholders. By integrating features like real-time updates, role-based dashboards, and responsive design, the system provides a reliable solution that improves efficiency and creates a seamless experience for hotel administrators and customers alike.

1.2 PROJECT GOALS

The main goals of the project are centered on solving real-world booking challenges while ensuring simplicity and reliability. The key objectives include:

- **Digitalizing Hotel Management:** Replace traditional manual systems with a modern digital platform where admins can easily add hotels, define their locations, update room availability, and manage bookings without confusion.

- **Enhancing User Experience:** Provide customers with a smooth and transparent process where they can view available hotels, check room availability in real-time, and book rooms effortlessly.
- **Ensuring Security:** Implement role-based authentication so that admins and users have separate levels of access, keeping sensitive information secure.
- **Promoting Efficiency:** Reduce booking errors, mismanagement, and delays by using automated record-keeping and instant updates.
- **Scalability for Future Growth:** Design the system in a way that allows easy integration of advanced features in the future, such as payment gateways, booking reminders, or AI-driven hotel suggestions.

In summary, the project aims to bridge the gap between traditional booking practices and modern digital expectations, while setting the foundation for future improvements.

1.3 MOTIVATION BEHIND THE PROJECT

The motivation for building this project arises from the **real-world challenges faced in hotel booking and management**. In today's fast-paced lifestyle, people expect quick and reliable solutions at their fingertips. Traditional booking methods, such as calling hotels directly or visiting in person, are inconvenient and often lack accuracy. Customers may face problems like unavailable rooms, hidden charges, or incomplete information.

Similarly, hotel administrators struggle with maintaining records manually, which can lead to overbookings, poor customer experiences, and loss of revenue. With digital technology becoming a necessity in every sector, it is only natural to bring hotel booking systems into the **modern, automated, and transparent environment**.

On a personal level, this project is also motivated by the opportunity to apply **technical knowledge and skills** in creating something practical that mirrors real-world applications. Building such a system provides exposure to solving real problems, designing user interfaces, handling databases, and ensuring security—all essential skills for professional growth in the field of software development.

1.4 IMPORTANCE OF THE SYSTEM

This project is not only a demonstration of technical abilities but also provides educational value and practical applications for both developers and end-users.

- **Educational Benefits:**
    - Offers hands-on experience with technologies like Spring Boot, HTML, CSS, which are widely used in industry.
    - Enhances problem-solving and analytical skills by addressing issues like role-based authentication, database management, and responsive UI design.
    - Strengthens understanding of real-world concepts such as scalability, modular design, and usability testing.
- **Practical Benefits:**
    - Helps hotel administrators by providing them with a centralized dashboard to manage hotels, rooms, and bookings with accuracy and ease.
    - Offers users a convenient platform to book rooms anytime, anywhere, without the hassles of manual methods.
    - Encourages transparency in hotel operations, reducing confusion about room availability or booking status.
    - Saves time, effort, and resources for both admins and users, making it a cost-effective and efficient solution.

Overall, the project is valuable both as an academic exercise that enhances learning and as a practical tool that can be scaled into a real-world application.

2. PROBLEM DEFINITION AND REQUIREMENTS

2.1 INTRODUCTION TO THE PROBLEM

In today’s world, hotel booking is an essential service for travellers, tourists, and even local residents. However, many hotels still rely on outdated methods such as manual registers, phone calls, or basic spreadsheets to handle bookings. These traditional practices often result in errors such as double-booking, mismanagement of room availability, and lack of transparency for users. Customers frequently face inconvenience due to incomplete or incorrect information about available rooms, while hotel staff struggle to maintain accurate records.

Additionally, with the rise of online platforms in almost every sector, users now expect instant, reliable, and user-friendly digital services. The absence of a centralized hotel booking system that ensures accuracy, security, and transparency creates frustration for customers and inefficiency for hotel administrators.

This highlights the need for a Hotel Booking System that automates core operations, provides real-time updates, and ensures smooth interaction between admins and users.

2.2 CHALLENGES ADDRESSED BY THE SYSTEM

The proposed Hotel Booking System aims to overcome several key challenges faced by both hotel admins and users:

1. **Room Availability Management**
   - Problem: Manual systems make it difficult to keep track of which rooms are available, occupied, or reserved.
   - Solution: The system updates room availability in real-time, ensuring accuracy for both admins and users.
2. **Booking Errors and Double Reservations**
   - Problem: Without digital tracking, double-booking or misallocation of rooms is common.
   - Solution: Automated booking records prevent overlaps and ensure that every reservation is accurately logged.
3. **Limited Transparency for Customers**
   - Problem: Users often cannot check available hotels or rooms directly and must rely on phone calls or agents.
   - Solution: A user-friendly portal allows customers to log in, browse hotels, and view real-time availability before booking.
4. **Inefficient Hotel Management for Admins**
   - Problem: Admins struggle to manage multiple hotels, locations, and room details using manual registers or simple spreadsheets.
   - Solution: The system centralizes hotel and room management with an intuitive admin dashboard.
5. **Security Concerns**
   - Problem: Sensitive data like customer details or booking records are at risk in poorly managed manual systems.
   - Solution: Role-based authentication ensures only authorized admins can add hotels or manage rooms, while users securely access their accounts.

2.3 FUNCTIONAL REQUIREMENTS

Functional requirements define what the system must do to achieve its goals. For the Hotel Booking System, they include:

1. **User Registration & Login**
    - Users should be able to register, log in, and access their accounts securely.
2. **Admin Authentication & Access Control**
    - Admins should have separate login access with privileges to add hotels, manage locations, and update room availability.
3. **Hotel & Room Management (Admin)**
    - Admin can add new hotels, specify their locations, and define the number of rooms available in each hotel.
    - Admin can update or delete hotels and rooms as needed.
4. **Room Booking (User)**
    - Users view available rooms, and book them through the platform.
5. **Booking Records**
    - The system should store and display booking records for both admins and users.
6. **Dashboards**
    - Admin Dashboard: Displays hotels, locations, room availability, and all user bookings.
    - User Dashboard: Displays available hotels, personal bookings

2.4 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements define how the system should perform rather than what it should do. For the Hotel Booking System:

1. **Performance**
   - The system should process bookings and updates quickly, with minimal delay, even when multiple users are active simultaneously.
2. **Scalability**
   - The platform should support multiple hotels, locations, and a growing number of users without compromising performance.
3. **Security**
   - All user and admin data must be securely stored, with encrypted passwords and role-based access control.
4. **Usability**
   - The system should have an intuitive and responsive interface accessible across desktops, tablets, and smartphones.
5. **Reliability**
   - The application should ensure consistent availability and prevent data loss through proper database management.
6. **Maintainability**
   - The system should be modular, making it easy to add new features like payment integration, notifications, or AI-based recommendations in the future.
7. **Compatibility**
   - It should work on all major browsers (Chrome, Firefox, Edge, Safari) and operating systems (Windows, Linux, macOS).

3. PROPOSED DESIGN AND METHODOLOGY

3.1 Data Storage and Management

At the heart of the Hotel Management System lies efficient data storage. The system stores details such as hotel names, locations, number of rooms, room availability, bookings, and user information. A well-structured **relational database** is used to ensure data accuracy and consistency.

- **Hotel Information**: Includes hotel name, address.
- **Room Management**: Stores the number of rooms, availability status.
- **User Data**: Contains user registration details and their booking history.

By organizing data into tables with relationships (e.g., one hotel → many rooms), the system avoids redundancy and ensures smooth retrieval. Real-time updates make sure that room availability is accurate and visible instantly to users.

3.2 Version Control System

To manage project files, code, and updates, a **Version Control System (VCS)** such as Git is used. This ensures that every update made by developers is tracked and can be reverted if necessary.

- Developers can work on different features (like hotel module, booking module) in separate branches.
- Changes are merged systematically to avoid conflicts.
- A record of all updates is maintained for transparency and collaboration.

This is particularly useful if the project grows, as multiple team members can work together without overwriting each other's contributions.

3.3 Backup Strategy and Scheduling

Since hotel and booking data is sensitive, losing it could create major problems. Therefore, a robust **backup strategy** is implemented:

- **Daily Backups**: Databases are backed up automatically at the end of each day.

This guarantees that even in case of accidental deletion, system crashes, or cyberattacks, the data remains safe and recoverable.

3.4 File Change and Detection Tracking

In the Hotel Management System, data accuracy is extremely important. Whenever details about a hotel, location, or room availability are updated by the admin, the system must instantly detect and reflect those changes so that users always see the most up-to-date information.

- **Real-Time Updates**: If the admin changes the number of available rooms, adds a new hotel, or updates the hotel location, the system automatically refreshes the records in the database. This prevents situations where users might see incorrect availability.
- **Change Detection Mechanism**: The system continuously checks for modifications in key fields such as hotel name, room count, or availability status. As soon as a change is detected, the database is updated, and the front-end user interface displays the new data without delay.
- **Consistency Across Modules**: For example, if the admin marks 2 rooms as booked in one hotel, the change is immediately reflected in both the "Available Rooms" section for users and the "Hotel Management Dashboard" for the admin. This synchronization avoids mismatches or duplication.

3.5 Security and Data Integrity

Security is one of the most important aspects of the system. Since sensitive data (user details, booking records) is stored, strong measures are adopted:

- **Role-Based Access Control (RBAC)**: Admins can add/manage hotels, while users only view and book.
- **Authentication & Authorization**: Secure login with encrypted passwords.
- **Data Validation**: Inputs (like hotel name, room numbers, or user details) are validated to avoid incorrect entries.
- **Encryption**: Sensitive data is encrypted to prevent misuse even if a breach occurs.
- **Integrity Checks**: The system ensures that no booking overlaps occur and that room availability remains accurate at all times.

This not only builds trust among users but also ensures compliance with modern security standards.

4. PROPOSED RESULTS

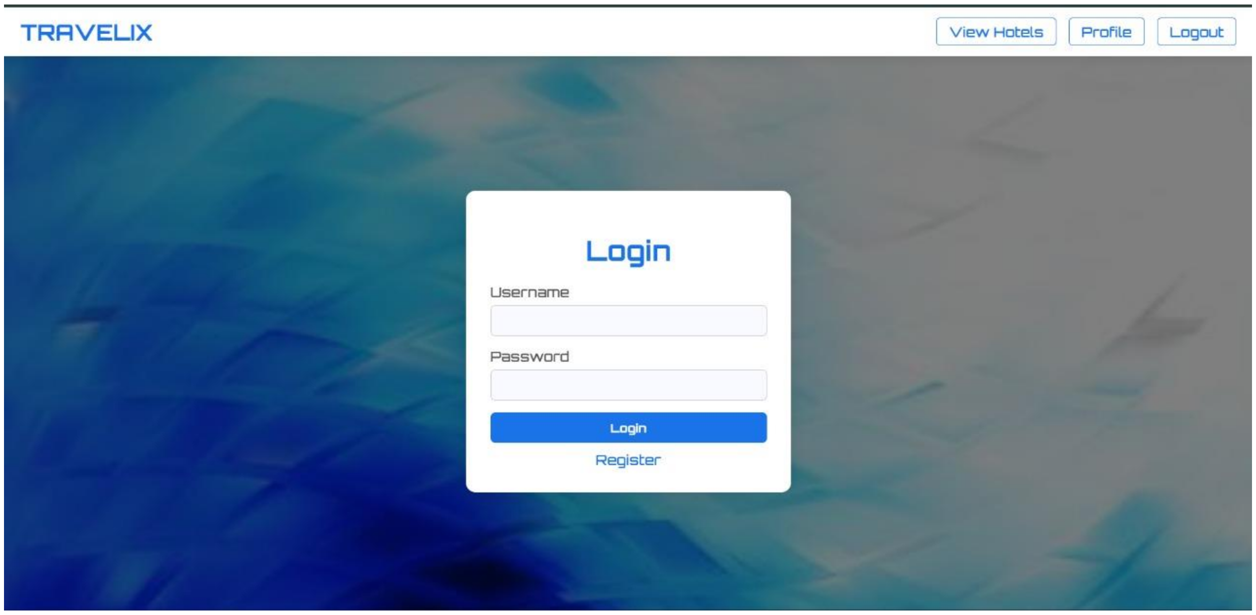


Fig. 1 – Login Page

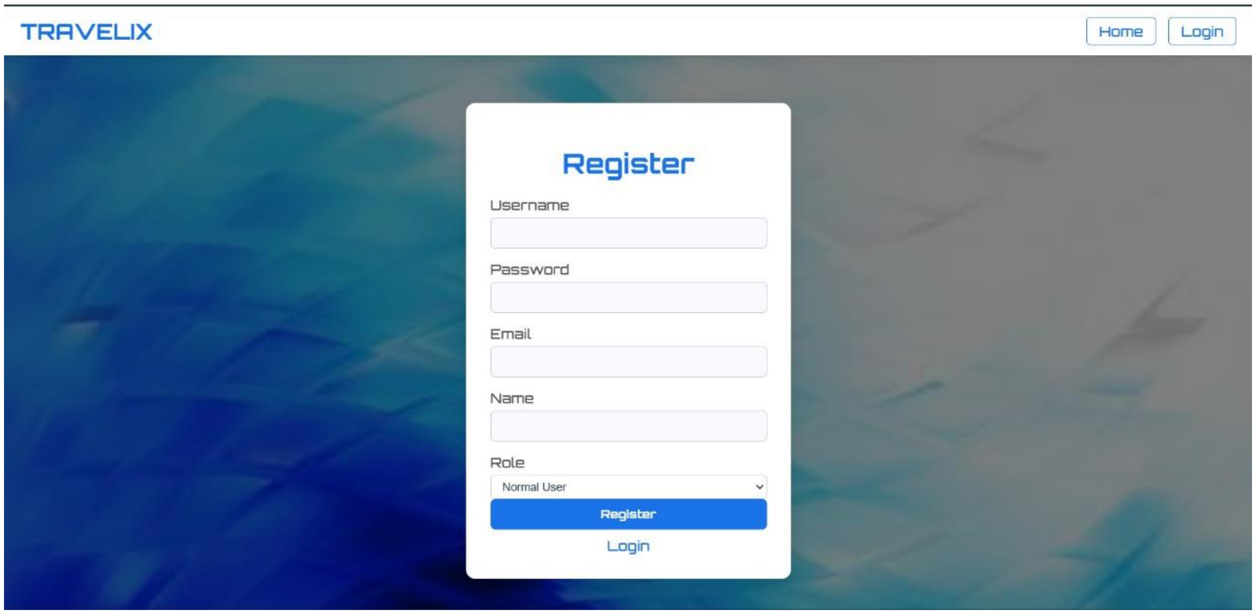
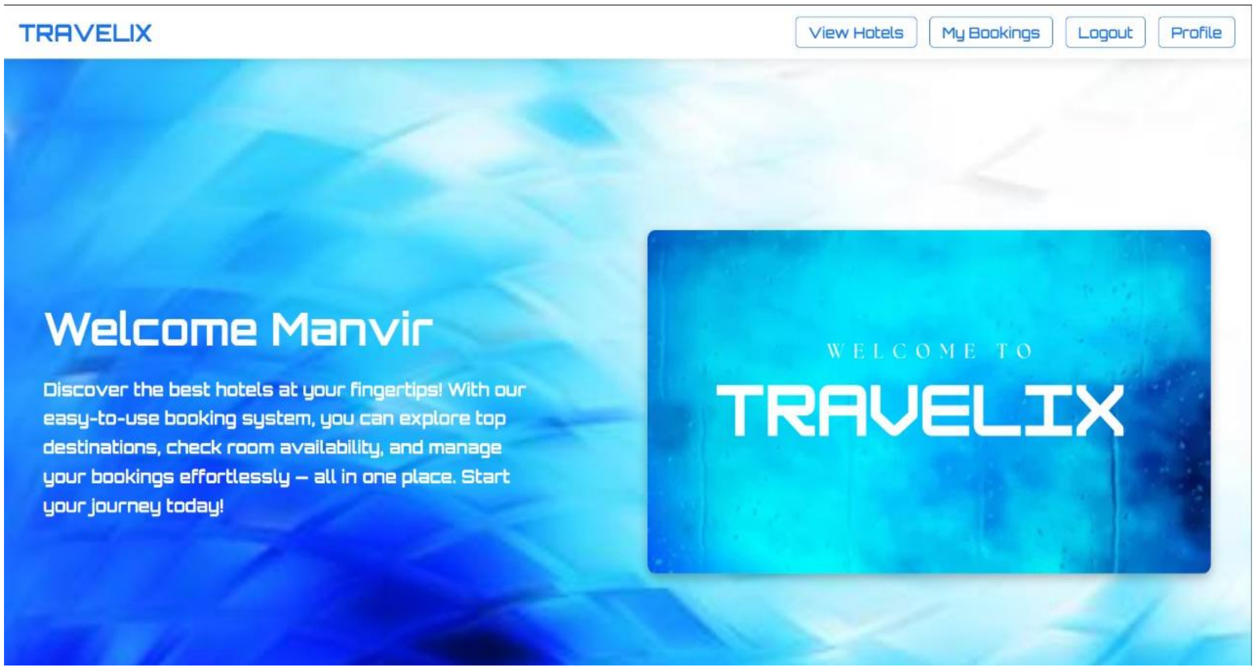


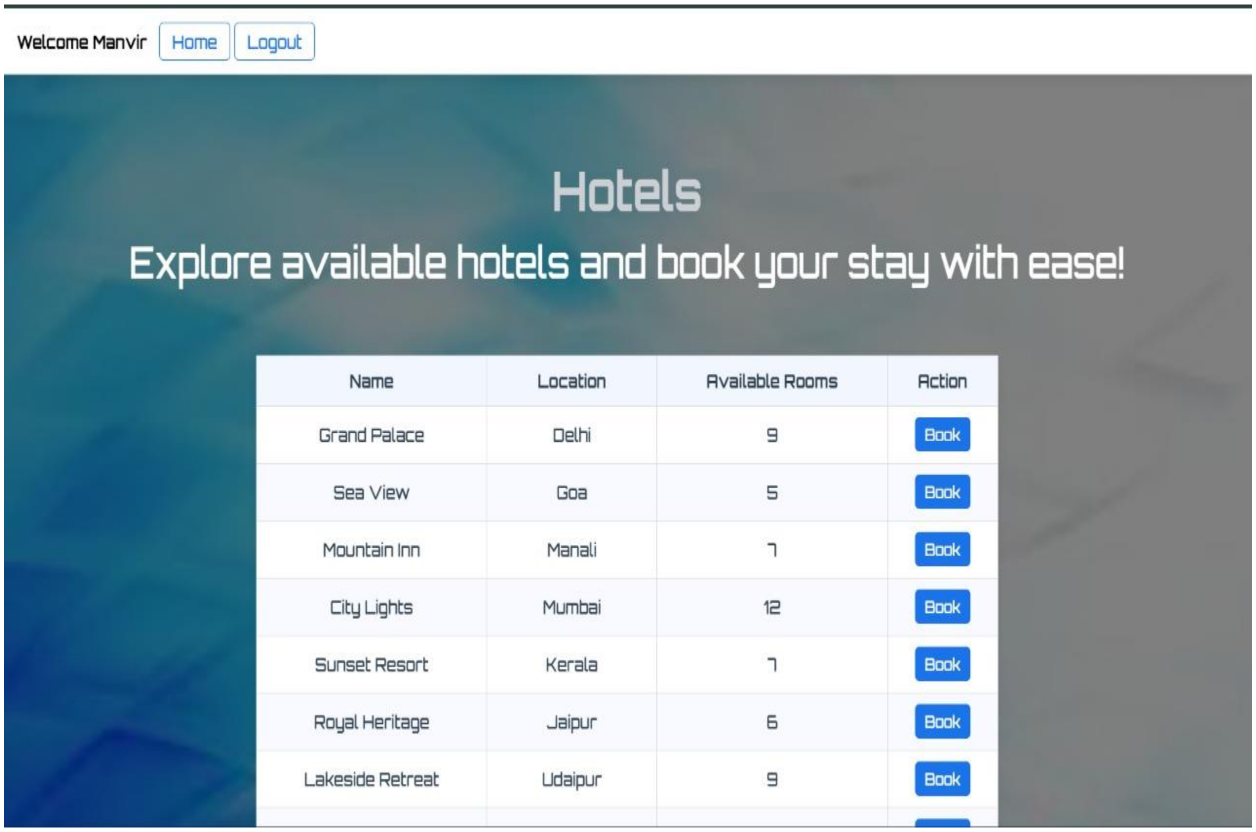
Fig. 2 – Register Page



Welcome Manvir

Discover the best hotels at your fingertips! With our easy-to-use booking system, you can explore top destinations, check room availability, and manage your bookings effortlessly – all in one place. Start your journey today!

Fig. 3 - Dashboard Page

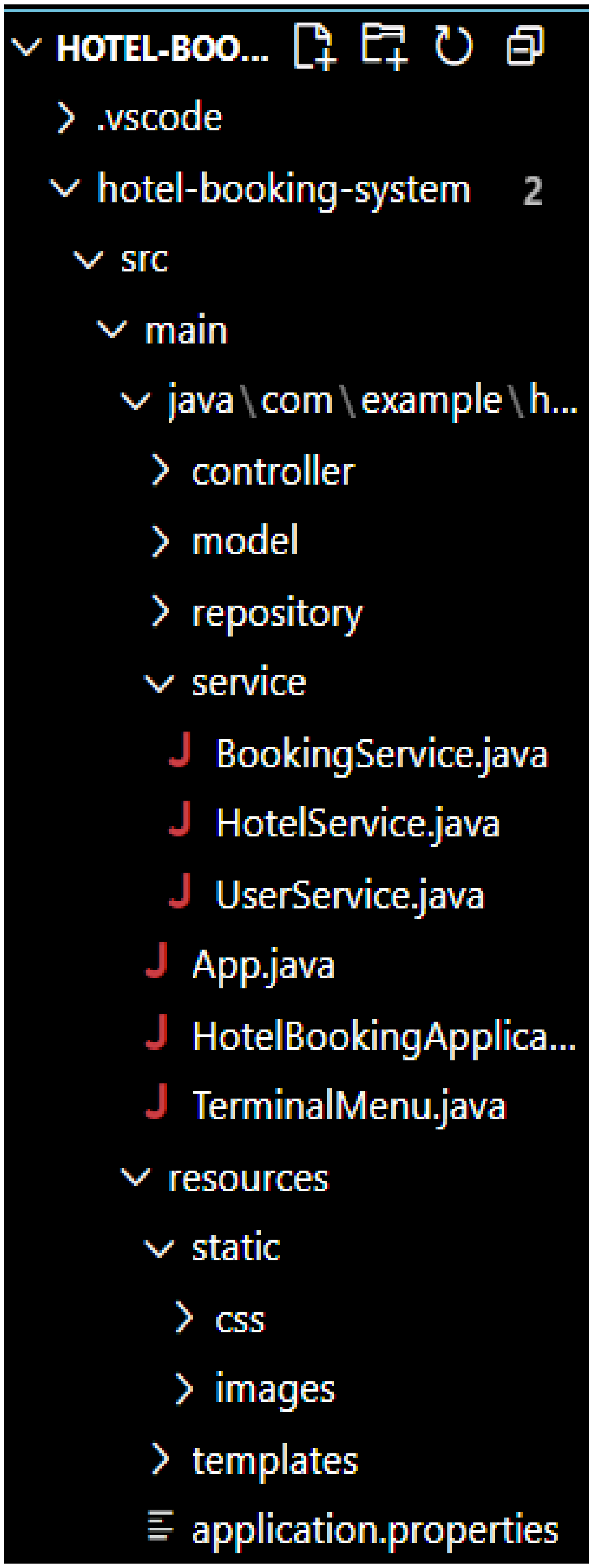


Hotels

Explore available hotels and book your stay with ease!

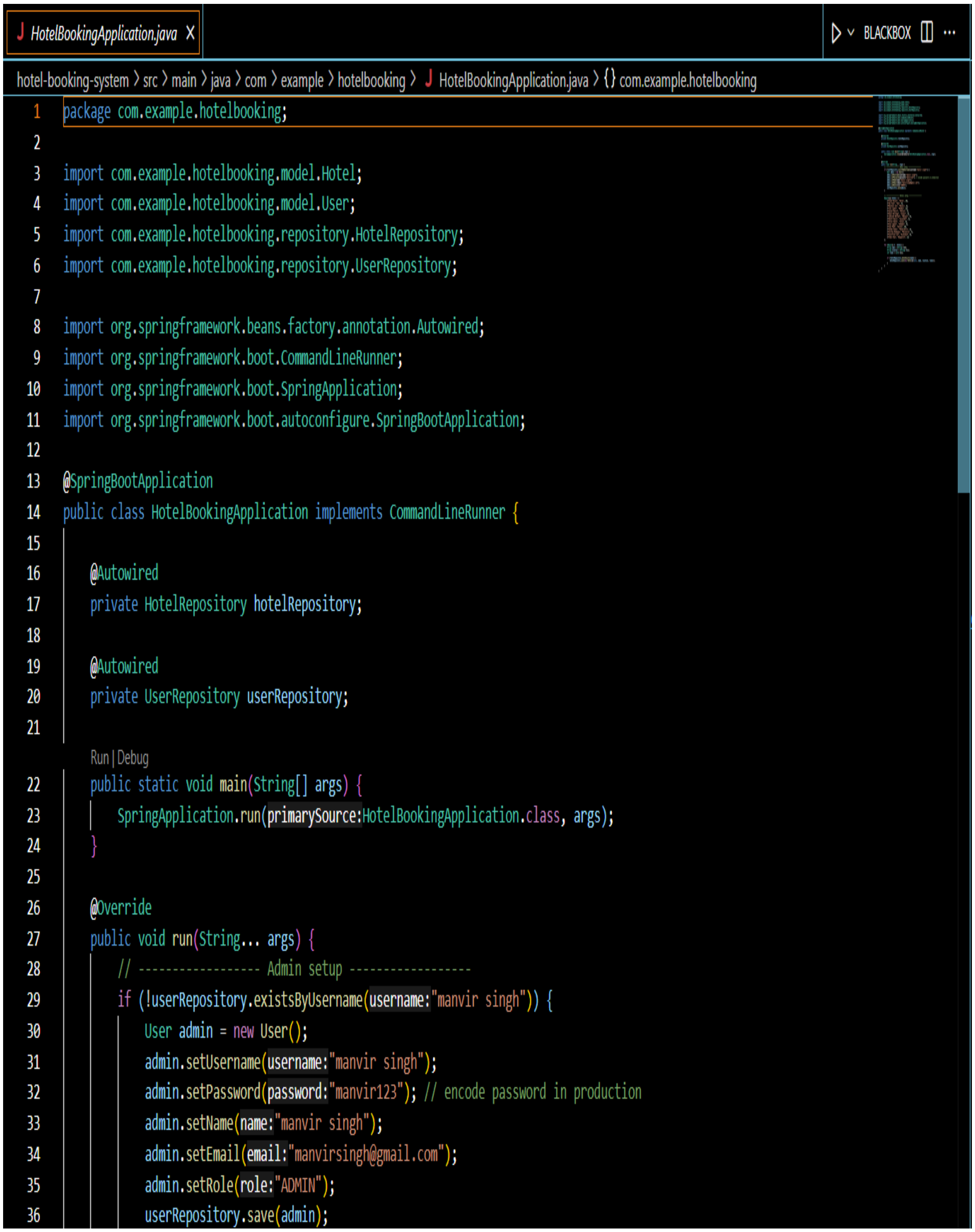
Name	Location	Available Rooms	Action
Grand Palace	Delhi	9	Book
Sea View	Goa	5	Book
Mountain Inn	Manali	7	Book
City Lights	Mumbai	12	Book
Sunset Resort	Kerala	7	Book
Royal Heritage	Jaipur	6	Book
Lakeside Retreat	Udaipur	9	Book

Fig. 4 – Available Rooms Information Page



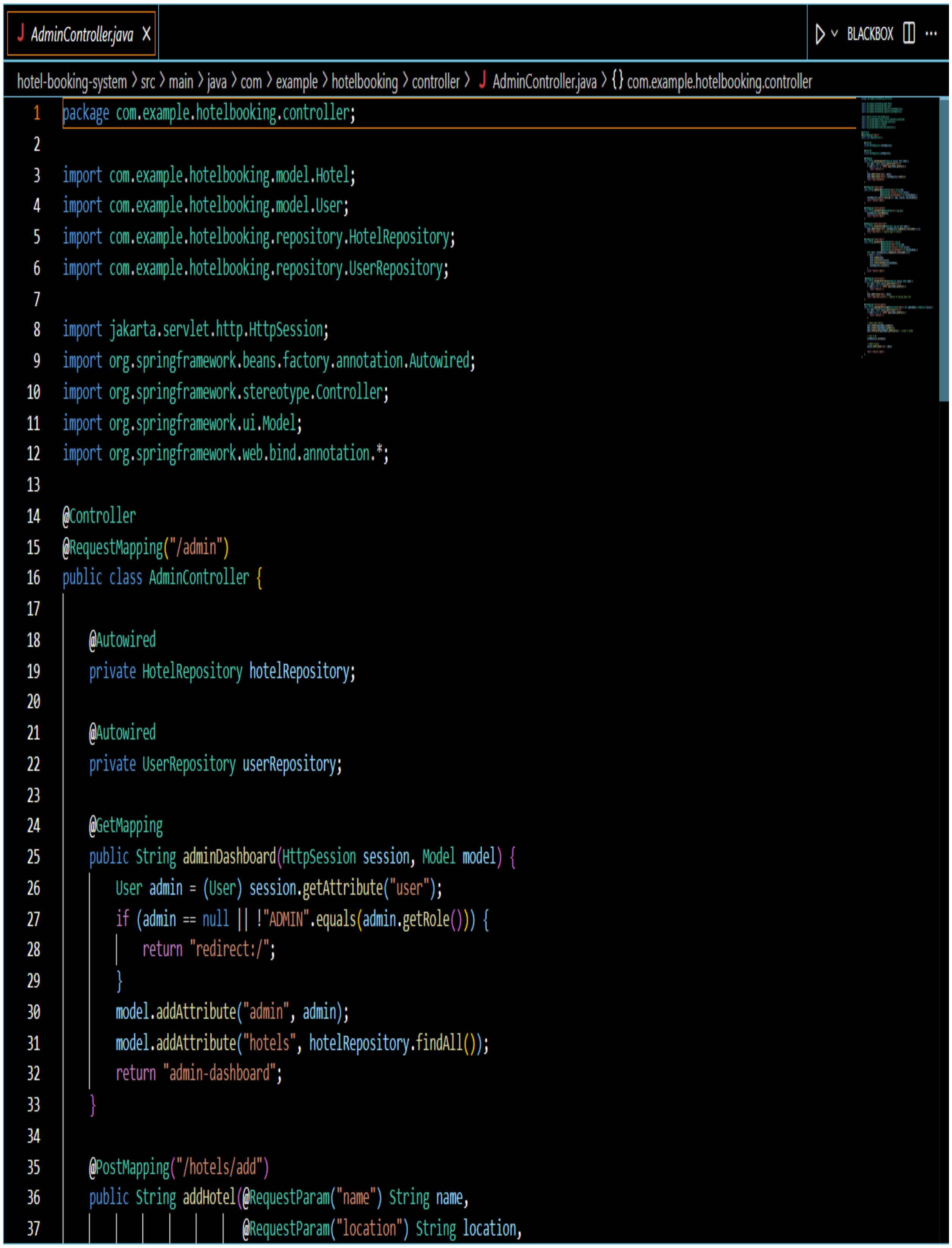
```
HOTEL-BOO...
  > .vscode
  v hotel-booking-system    2
    v src
      v main
        v java\com\example\h...
          > controller
          > model
          > repository
          v service
             BookingService.java
             HotelService.java
             UserService.java
           App.java
           HotelBookingApplica...
           TerminalMenu.java
        v resources
          v static
            > css
            > images
          > templates
           application.properties
```

Fig. 5 File Structure


HotelBookingApplication.java X
BLACKBOX
hotel-booking-system > src > main > java > com > example > hotelbooking > HotelBookingApplication.java > {} com.example.hotelbooking
Run | Debug

```java
package com.example.hotelbooking;

import com.example.hotelbooking.model.Hotel;
import com.example.hotelbooking.model.User;
import com.example.hotelbooking.repository.HotelRepository;
import com.example.hotelbooking.repository.UserRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HotelBookingApplication implements CommandLineRunner {

    @Autowired
    private HotelRepository hotelRepository;

    @Autowired
    private UserRepository userRepository;

    public static void main(String[] args) {
        SpringApplication.run(primarySource:HotelBookingApplication.class, args);
    }

    @Override
    public void run(String... args) {
        // ----------------- Admin setup -----------------
        if (!userRepository.existsByUsername(username:"manvir singh")) {
            User admin = new User();
            admin.setUsername(username:"manvir singh");
            admin.setPassword(password:"manvir123"); // encode password in production
            admin.setName(name:"manvir singh");
            admin.setEmail(email:"manvirsingh@gmail.com");
            admin.setRole(role:"ADMIN");
            userRepository.save(admin);
```

Fig. 6 – HotelBookingApplication.java Code



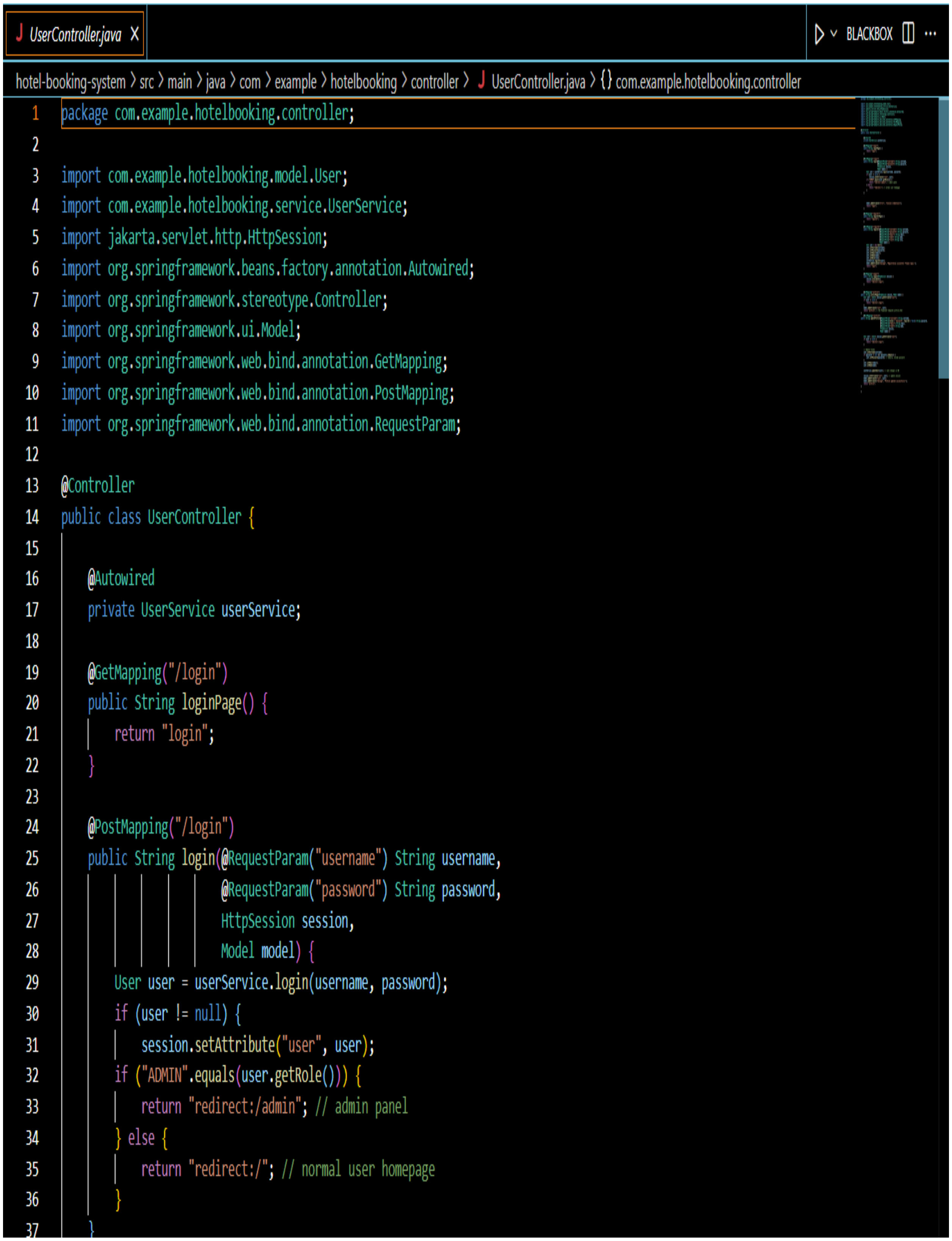
AdminController.java X

BLACKBOX

hotel-booking-system > src > main > java > com > example > hotelbooking > controller > AdminController.java > {} com.example.hotelbooking.controller

```java
package com.example.hotelbooking.controller;

import com.example.hotelbooking.model.Hotel;
import com.example.hotelbooking.model.User;
import com.example.hotelbooking.repository.HotelRepository;
import com.example.hotelbooking.repository.UserRepository;

import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private HotelRepository hotelRepository;

    @Autowired
    private UserRepository userRepository;

    @GetMapping
    public String adminDashboard(HttpSession session, Model model) {
        User admin = (User) session.getAttribute("user");
        if (admin == null || !"ADMIN".equals(admin.getRole())) {
            return "redirect:/";
        }
        model.addAttribute("admin", admin);
        model.addAttribute("hotels", hotelRepository.findAll());
        return "admin-dashboard";
    }

    @PostMapping("/hotels/add")
    public String addHotel(@RequestParam("name") String name,
                           @RequestParam("location") String location,
```

Fig. 7 – AdminController.java Code


UserController.java X
BLACKBOX
hotel-booking-system > src > main > java > com > example > hotelbooking > controller > UserController.java > {} com.example.hotelbooking.controller

```java
package com.example.hotelbooking.controller;

import com.example.hotelbooking.model.User;
import com.example.hotelbooking.service.UserService;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping("/login")
    public String loginPage() {
        return "login";
    }

    @PostMapping("/login")
    public String login(@RequestParam("username") String username,
                        @RequestParam("password") String password,
                        HttpSession session,
                        Model model) {
        User user = userService.login(username, password);
        if (user != null) {
            session.setAttribute("user", user);
        if ("ADMIN".equals(user.getRole())) {
            return "redirect:/admin"; // admin panel
        } else {
            return "redirect:/"; // normal user homepage
        }
    }
```

Fig. 8 – UserController.java Code

6.REFERENCES

- https://www.geeksforgeeks.org/ - Basic overview
- www.google.com – Project reference
- https://docs.oracle.com – Oracle Official Documentation
- https://spring.io/projects/spring-boot - Spring Boot Official Guide
- https://guides.github.com – GitHub Guides