Travel Management System

Capstone Project

Case Study

By

RASHMI R

INTRODUCTION

The **Travel Management System** is a web-based application designed to automate and simplify the process of managing travel packages, hotel bookings, and user reservations.

It provides a centralized platform where **users can browse travel packages, book trips, and track booking status**, while **administrators manage packages, hotels, and booking approvals**.

The system reduces manual work, improves accuracy, and ensures a smooth booking workflow between users and administrators.

This project is developed using **Spring Boot**, **MySQL**, **HTML, CSS, JavaScript**, and **REST APIs**.

ABSTRACT

The **Travel Management System (TMS)** is developed to manage travel-related activities such as package creation, hotel management, user bookings, and approval workflows.

The system allows users to **register, log in, view packages, book trips, and monitor booking status**, while administrators can **add hotels, create packages, approve or cancel bookings, and monitor system statistics**.

The main goal of this project is to demonstrate **real-world CRUD operations, role-based access control, and client–server interaction** using Spring Boot and MYSQL. The project follows a layered architecture consisting of **Controller, Service, DAO, and Database layers**.

CLIENT REQUIREMENTS

The client requires a Travel Management System with the following features:

- Admin should be able to:
  - Add, update, delete, and view travel packages
  - Add and manage hotels
  - View all bookings
  - Approve or cancel user bookings
  - View dashboard statistics (total users, bookings, hotels, packages)
- User should be able to:
  - Register and log in
  - View available travel packages
  - Book travel packages
  - View booking history and status
  - Cancel bookings (before approval)

TECHNICAL FEATURES

- Proper **Login and Registration**
- **Role-based authentication** (Admin / User)
- **REST APIs** for data communication
- CRUD operations on:
  - ✓ Users
  - ✓ Hotels
  - ✓ Travel Packages
  - ✓ Bookings
- Booking approval workflow

- Dashboard with real-time statistics
- Exception handling and validation
- Clean UI with HTML, CSS, and JavaScript

TECHNOLOGIES AND TOOLS USED

- ✓ **Programming Language**: Java (JDK 17)
- ✓ **Framework**: Spring Boot
- ✓ **Frontend**: HTML, CSS, JavaScript
- ✓ **Database**: MySQL 8.0
- ✓ **IDE**: Spring Tool Suite (STS)
- ✓ **Web Server**: Apache Tomcat (embedded)
- ✓ **Build Tool**: Maven
- ✓ **API Testing**: Postman
- ✓ **Browser**: Google Chrome

PROJECT MODULES

- **Admin Module**
- **User Module**
- **Hotel Module**
- **Travel Package Module**
- **Booking Module**

MODULE DESCRIPTION

1. Admin Module

The **Admin Module** is responsible for managing and monitoring the entire Travel Management System. It provides administrative control over users, travel packages, hotels, and bookings.

Features:

- Admin login with authentication
- Dashboard overview showing:
  - Total users
  - Total hotels
  - Total travel packages
  - Total bookings
- Add new travel packages
- Add hotel details
- View all user bookings
- Approve or cancel user bookings
- Monitor pending bookings

Purpose:

This module ensures that all system activities are properly controlled and managed. Only authorized admin users can access this module.

2. User Module

The **User Module** allows customers to interact with the system, explore travel options, and make bookings.

Features:

- User registration
- User login and logout
- View available travel packages
- Book a travel package
- View personal booking history
- Check booking status (BOOKED / APPROVED / CANCELLED)
- Cancel booking (if booking is still pending)

Purpose:

This module provides a smooth interface for users to plan and manage their travel bookings.

3. Travel Package Module

The **Travel Package Module** manages all travel packages offered by the system.

Features:

- Add new travel packages (Admin only)
- View list of all travel packages
- Display package details:
  - Destination
  - Price
  - Number of days
  - Description
  - Associated hotel
- Delete travel packages (Admin only)

Purpose:

This module acts as the core of the application by defining travel plans that users can browse and book.

4. Hotel Module

The **Hotel Module** manages hotel details linked to travel packages.

Features:

- Add hotel details (Admin only)
- Store hotel information such as:
  - Hotel name
  - Location
  - Price per day
- Associate one hotel with a travel package
- Display hotel details in bookings

Purpose:

This module ensures that each travel package is associated with proper accommodation details, enhancing the realism of the system.

5. Booking Module

The **Booking Module** handles the entire booking workflow between users and administrators.

Features:

- User books a travel package
- Booking status initially set to **BOOKED**
- Admin reviews and updates booking status:

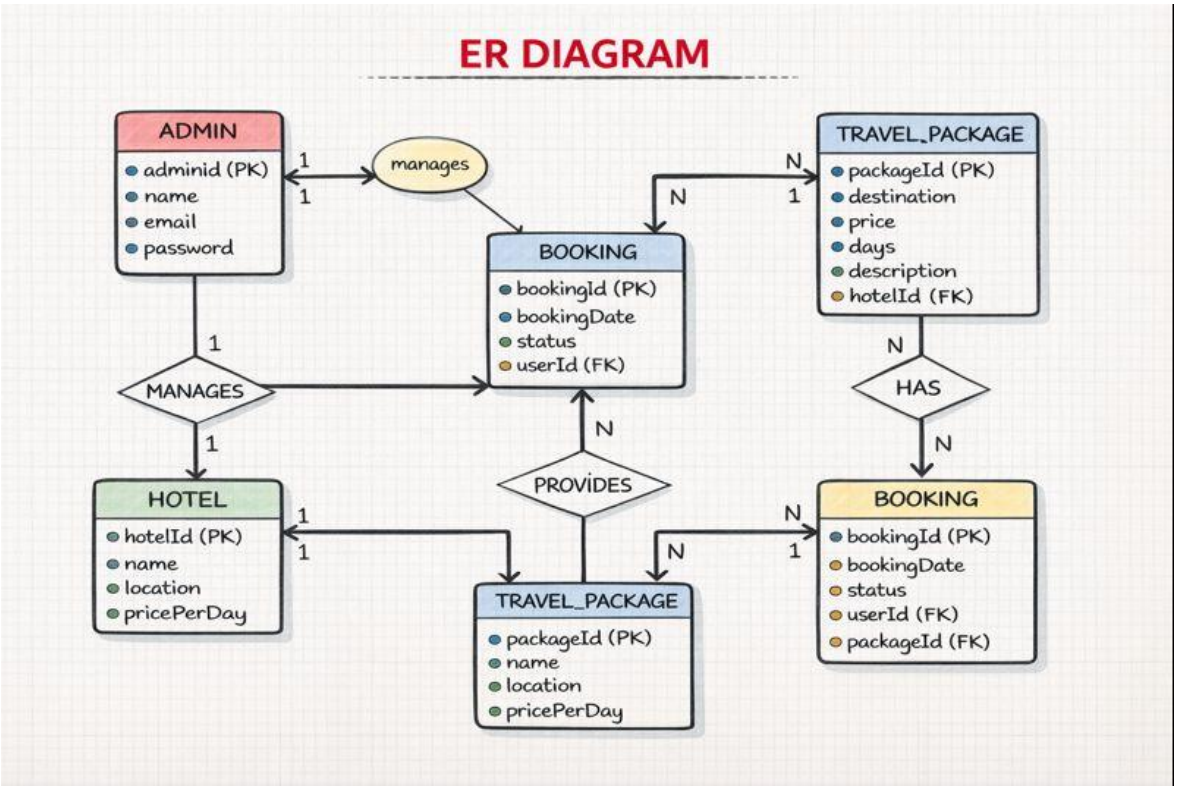
- APPROVED
- CANCELLED

- User views booking history and status
- Admin views all bookings and pending requests

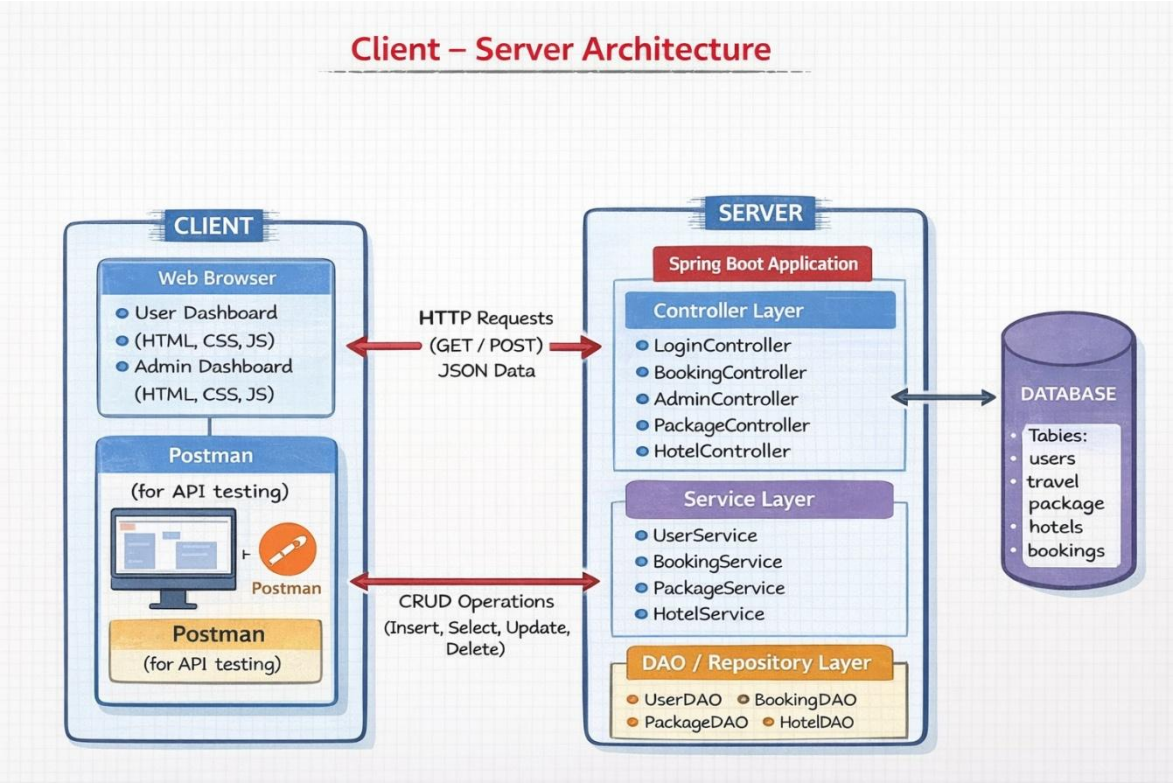
Purpose:

This module manages the interaction between users and admins, ensuring bookings are properly tracked and approved.

ER DIAGRAM



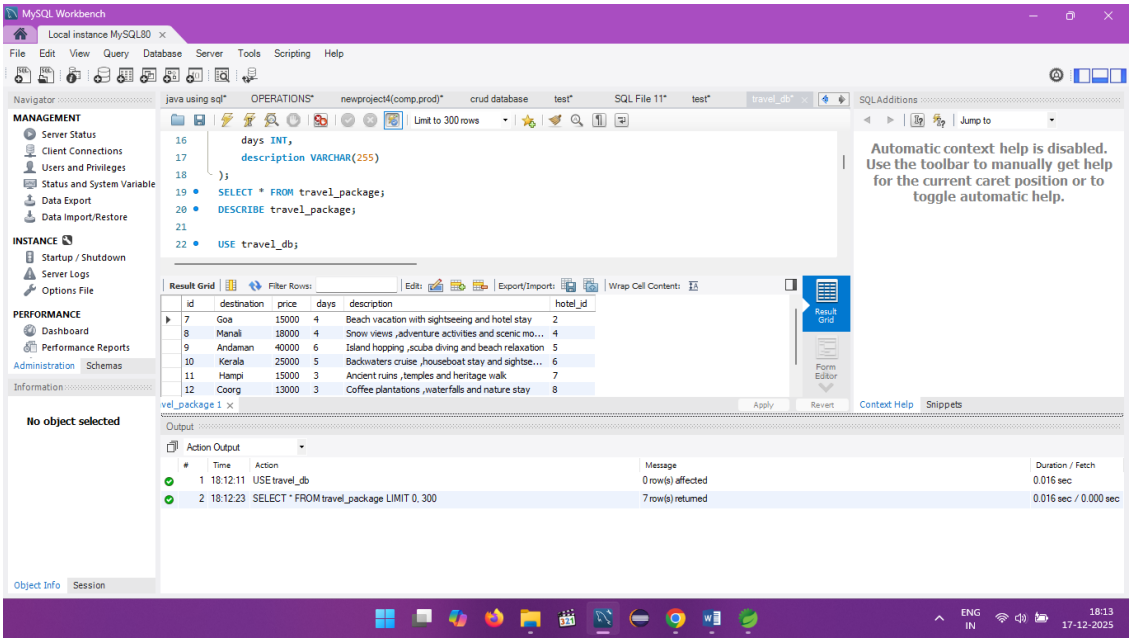
CLIENT-SERVER ARCHITECTURE



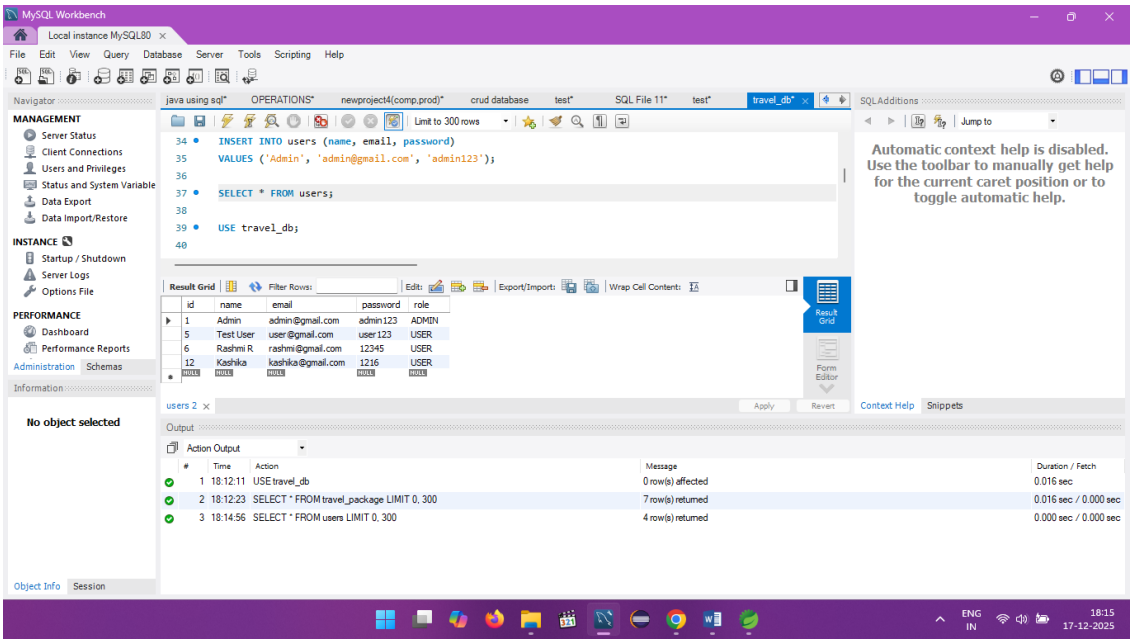
DATA DICTIONARY

TABLES OF SQL DATABASES

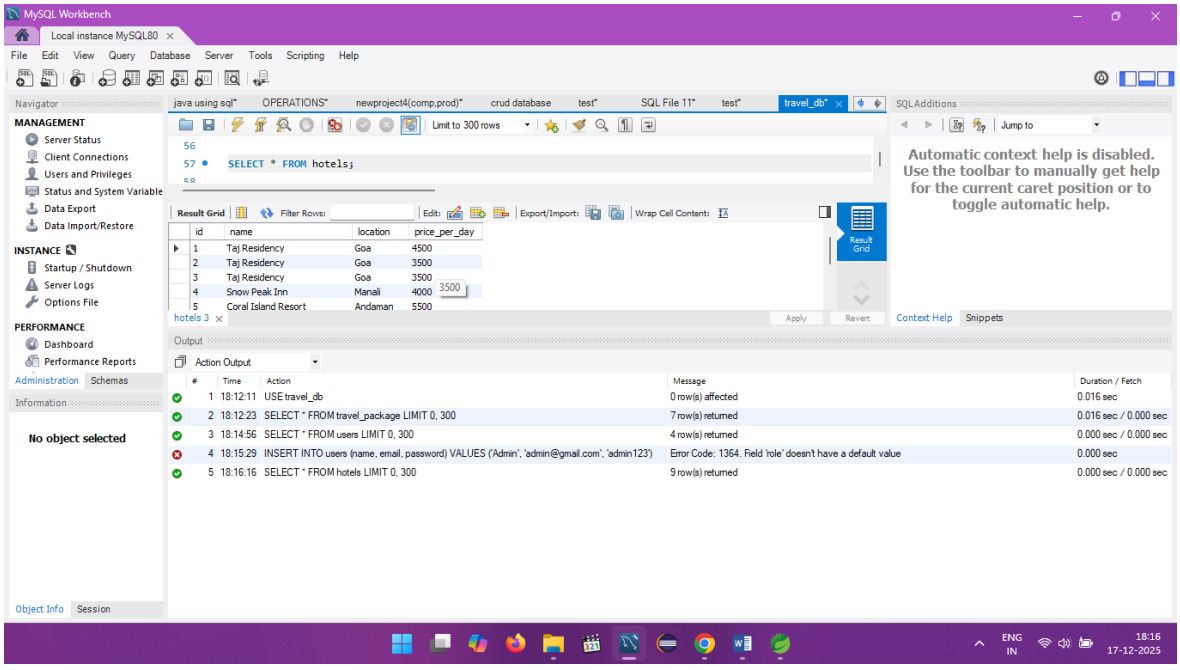
The Travel Management System uses MySQL as the backend database. The database consists of four main tables: Users, Hotels, Travel Packages, Bookings. These tables are connected using foreign key relationships to maintain data integrity.



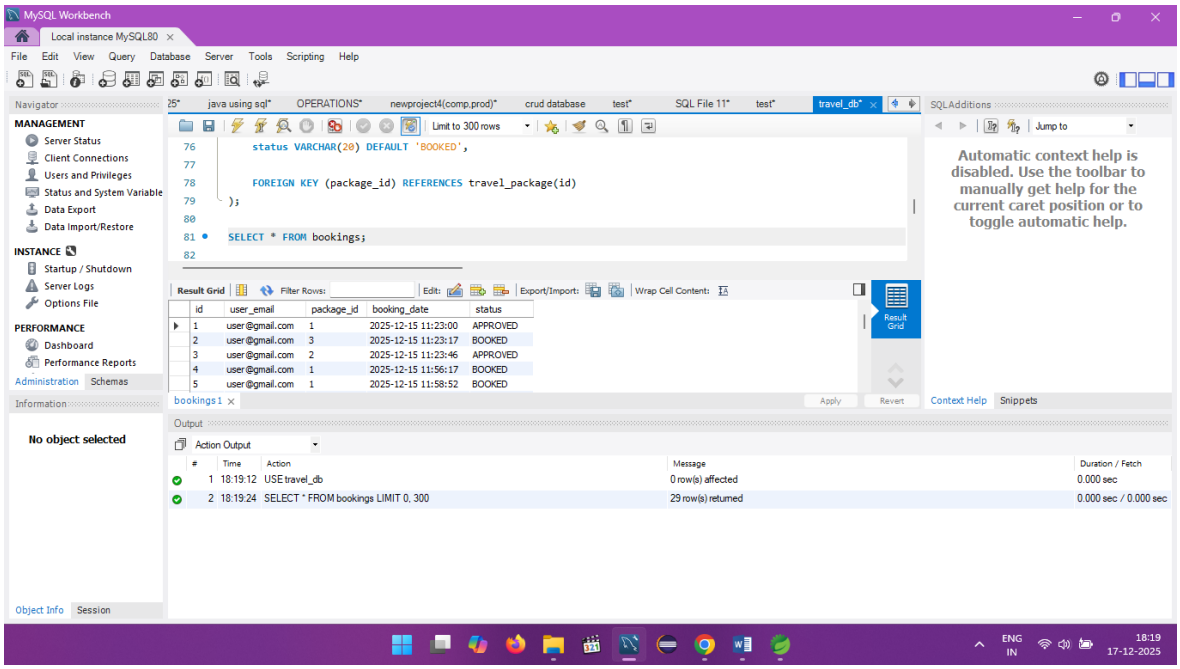
Travel-Packages Table



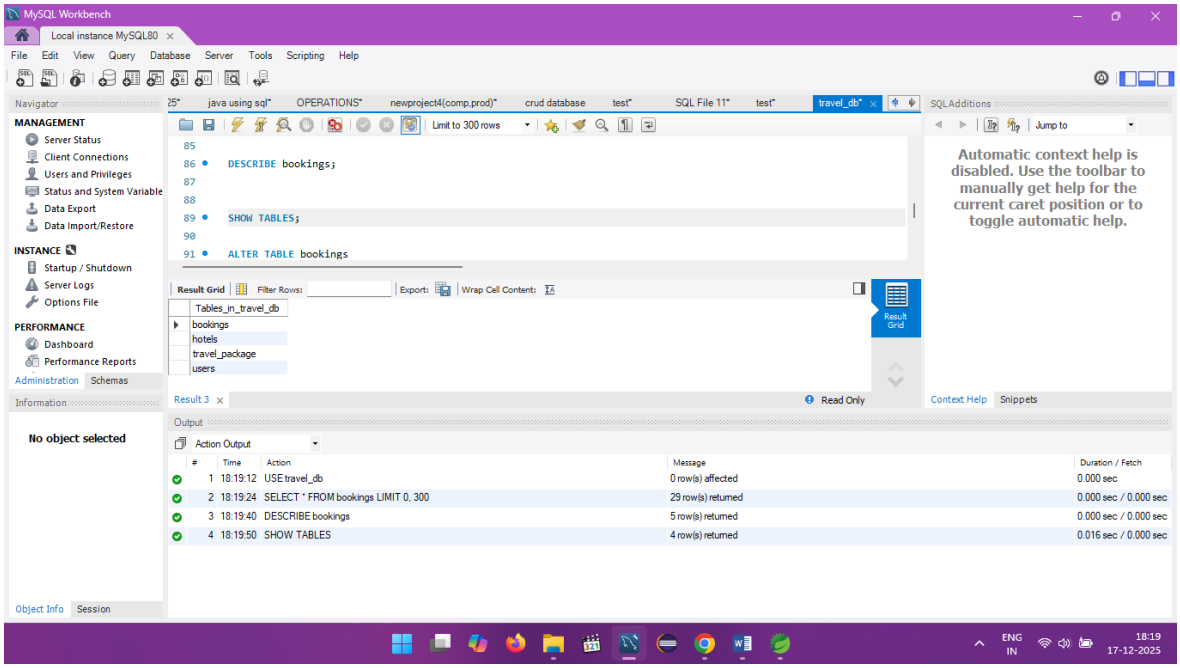
Users Table



Hotels Table



Bookings Table



Tables in travel_db

HTTP Request Methods

Definition:

HTTP defines a set of request methods to indicate the action to be performed on a given resource.

HTTP Method	URI	Purpose
GET	http://localhost:8088/view-packages	Fetch all travel packages
GET	http://localhost:8088/bookPackage?packageId=3	Book a selected package
GET	http://localhost:8088/my-bookings	Retrieve user booking history
POST	http://localhost:8088/doLogin	Login authentication
POST	http://localhost:8088/register	Register new user
POST	http://localhost:8088/addPackage	Create new travel package
POST	http://localhost:8088/addHotel	Create new hotel
DELETE	http://localhost:8088/deletePackage?id=7	Delete travel package

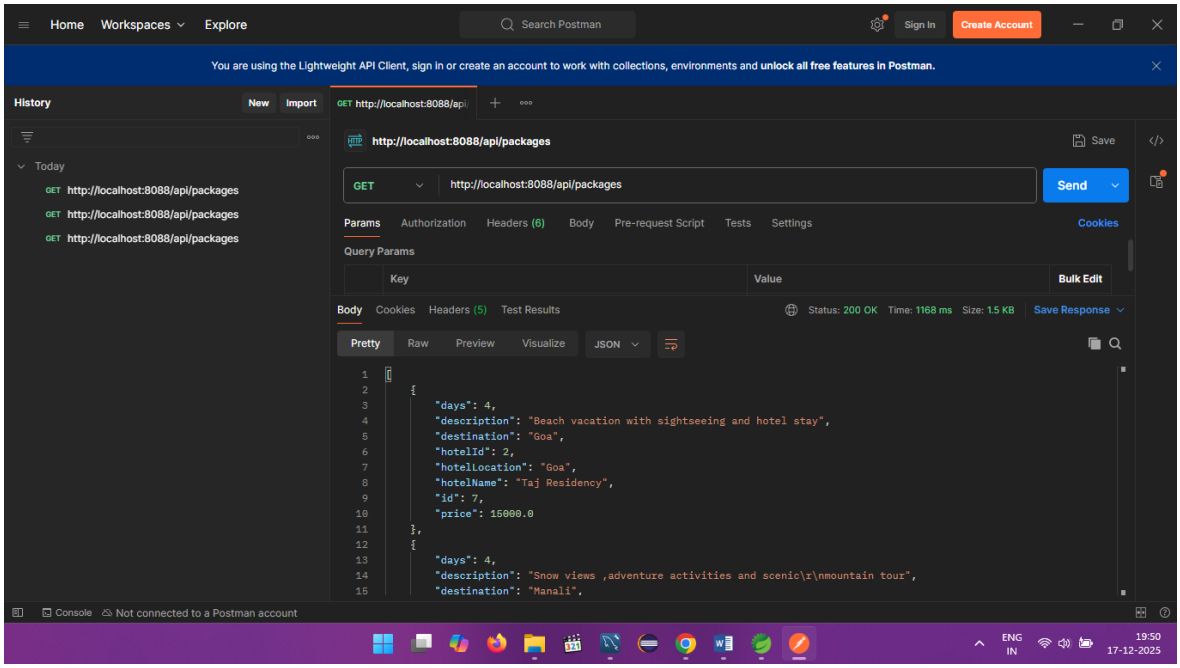
URI (Uniform Resource Identifier)

Definition

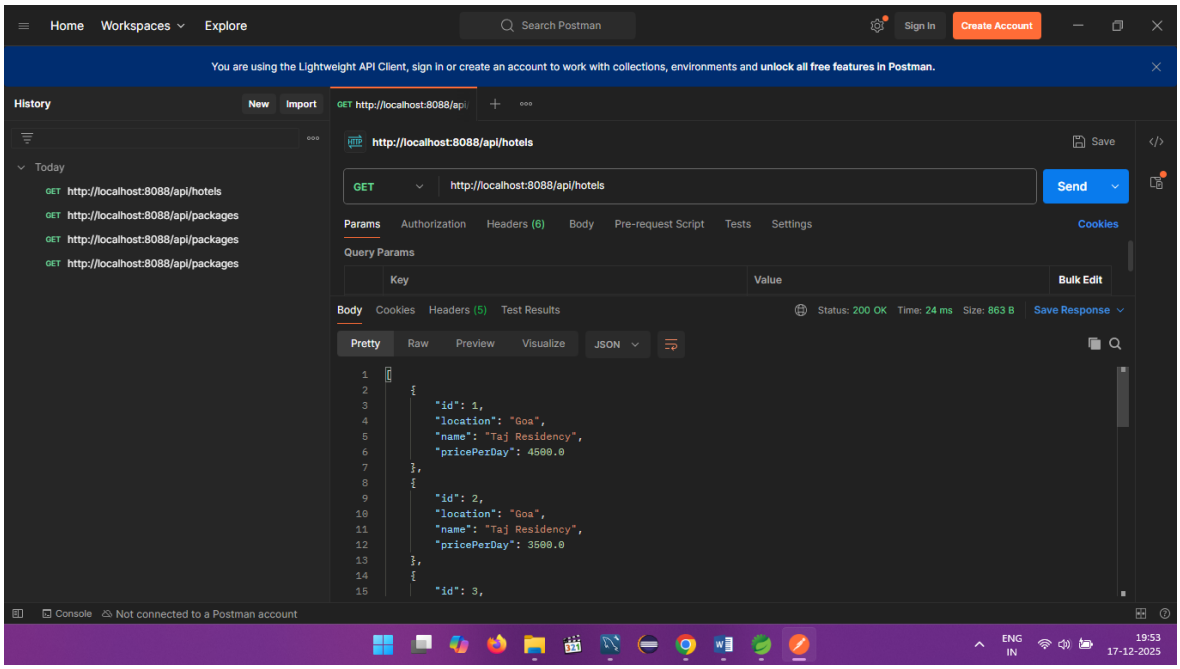
URI is used to uniquely identify resources in a web application. In this project, URI is used to identify users, travel packages, hotels, and bookings.

HTTP Method	URI	Description
GET	http://localhost:8088/login	Display login page
GET	http://localhost:8088/register	Display registration page
GET	http://localhost:8088/logout	Logout from system
GET	http://localhost:8088/admin-dashboard	Display admin dashboard
GET	http://localhost:8088/user-dashboard	Display user dashboard
GET	http://localhost:8088/view-packages	View all travel packages
GET	http://localhost:8088/my-bookings	View logged-in user bookings
GET	http://localhost:8088/add-package	Open add travel package page (Admin)
POST	http://localhost:8088/addPackage	Add new travel package
GET	http://localhost:8088/add-hotel	Open add hotel page (Admin)
POST	http://localhost:8088/addHotel	Add new hotel
GET	http://localhost:8088/bookPackage?packageId=5	Book a travel package
POST	http://localhost:8088/admin/update-booking	Approve/Reject booking

POSTMAN DEMONSTRATION

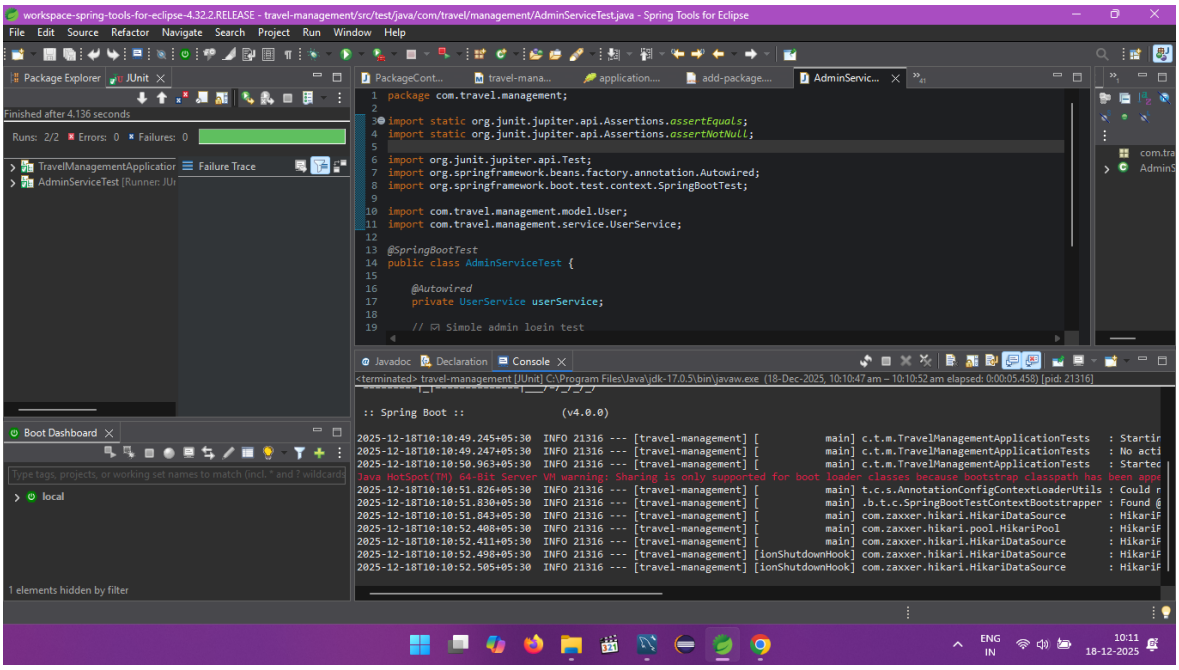


Get Method for packages

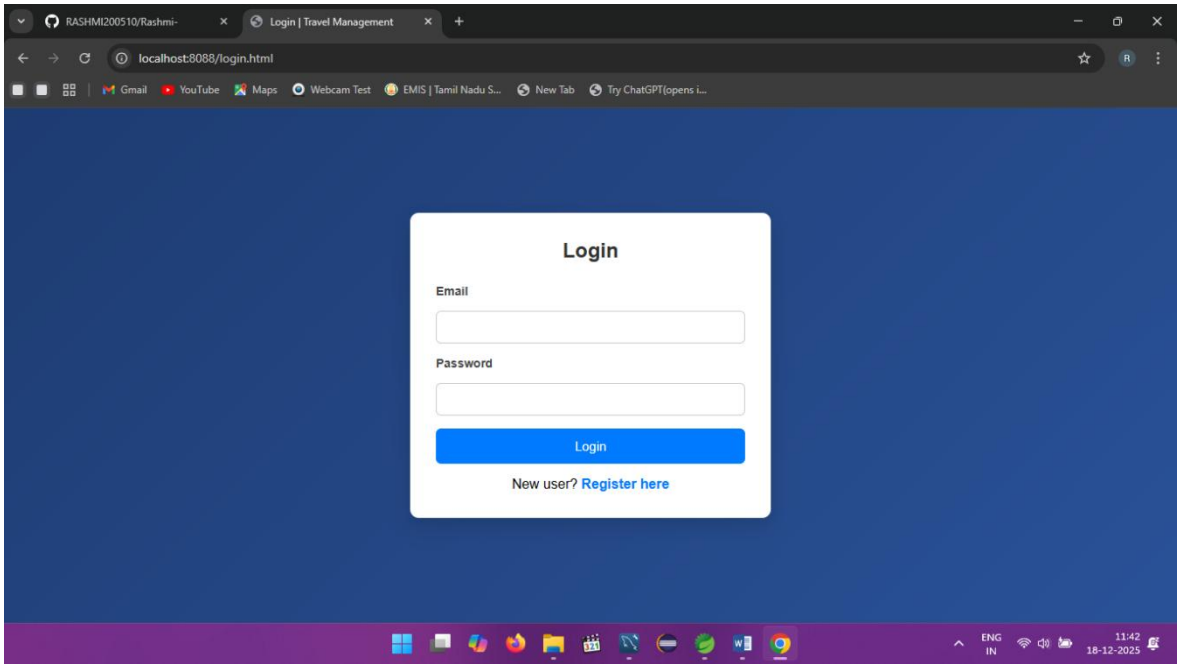


Get method for hotels

JUNIT TEST FOR ADMIN MODULE

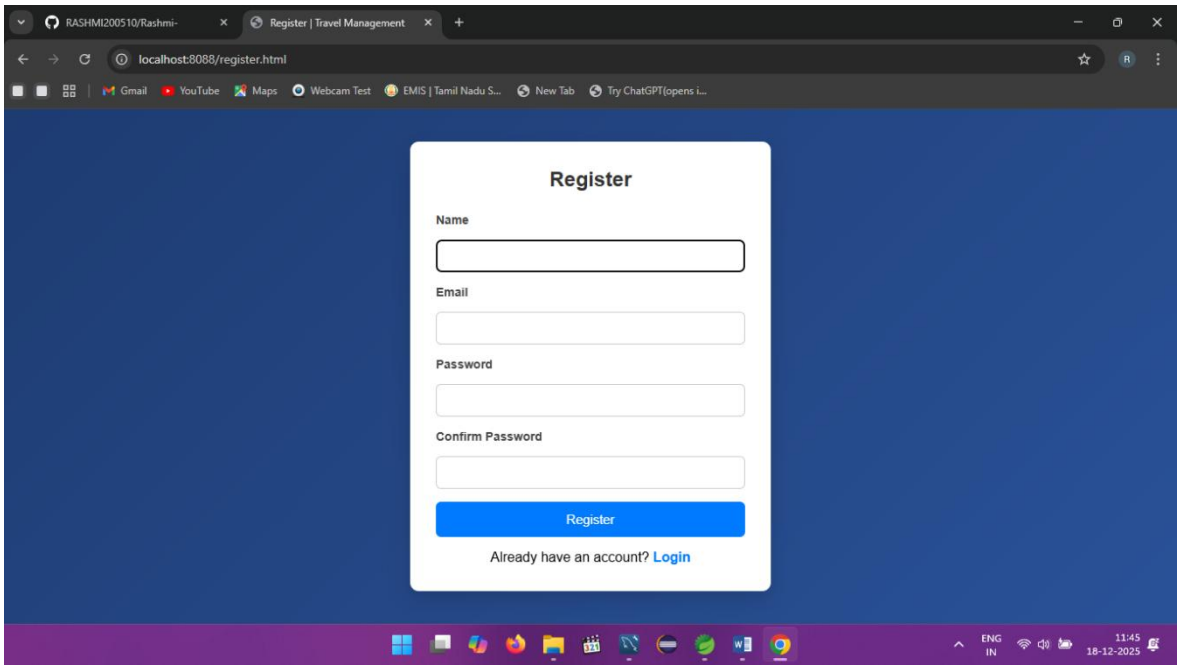


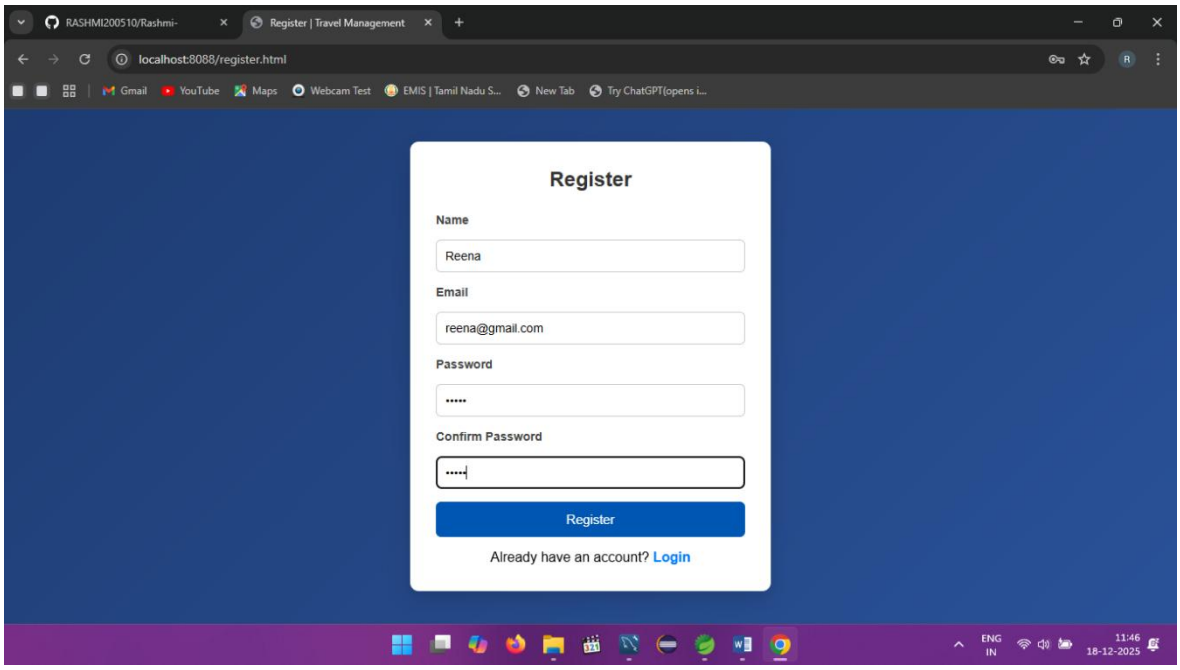
PROJECT OUTPUTS



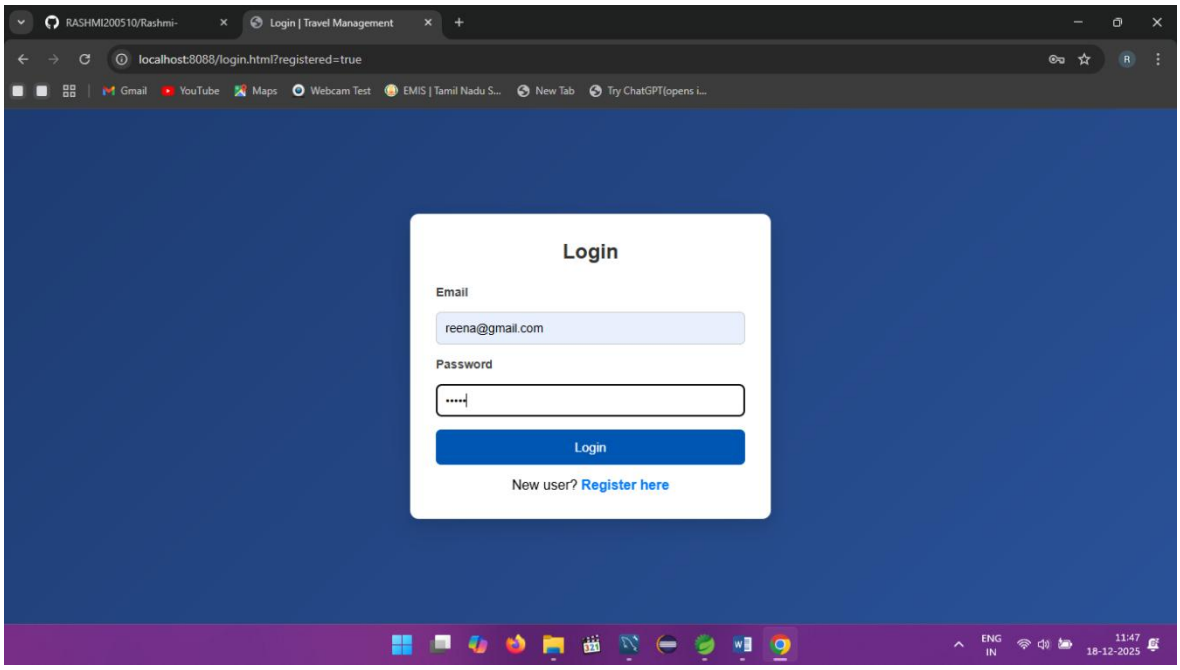
Login Page

Registration Page

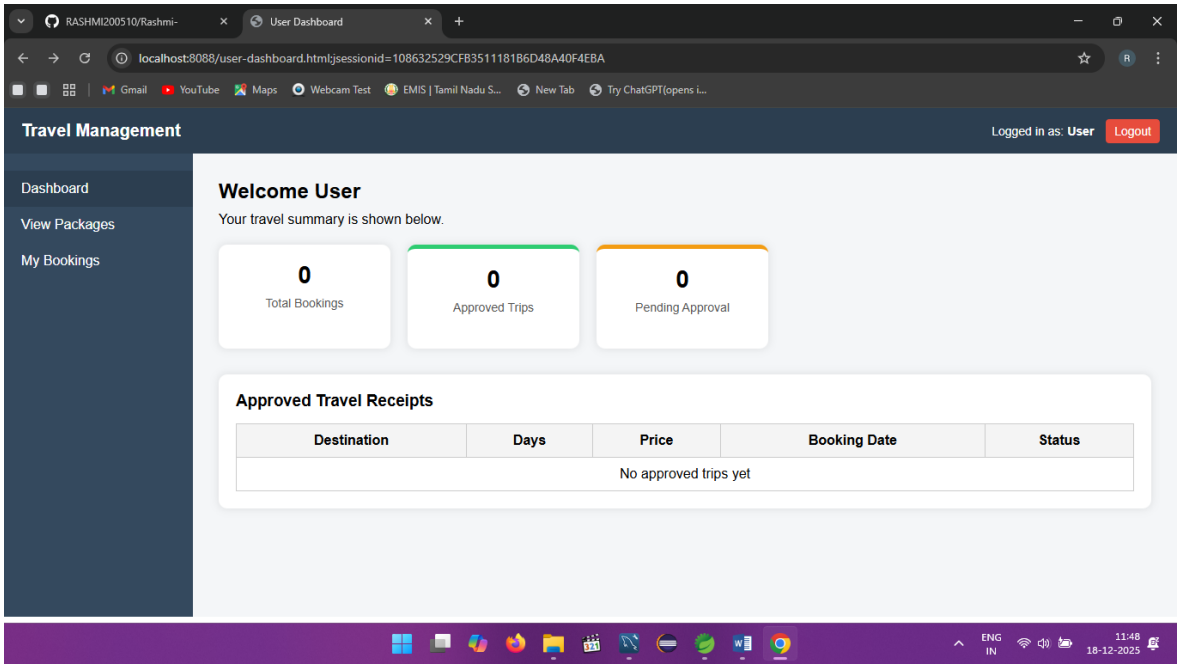




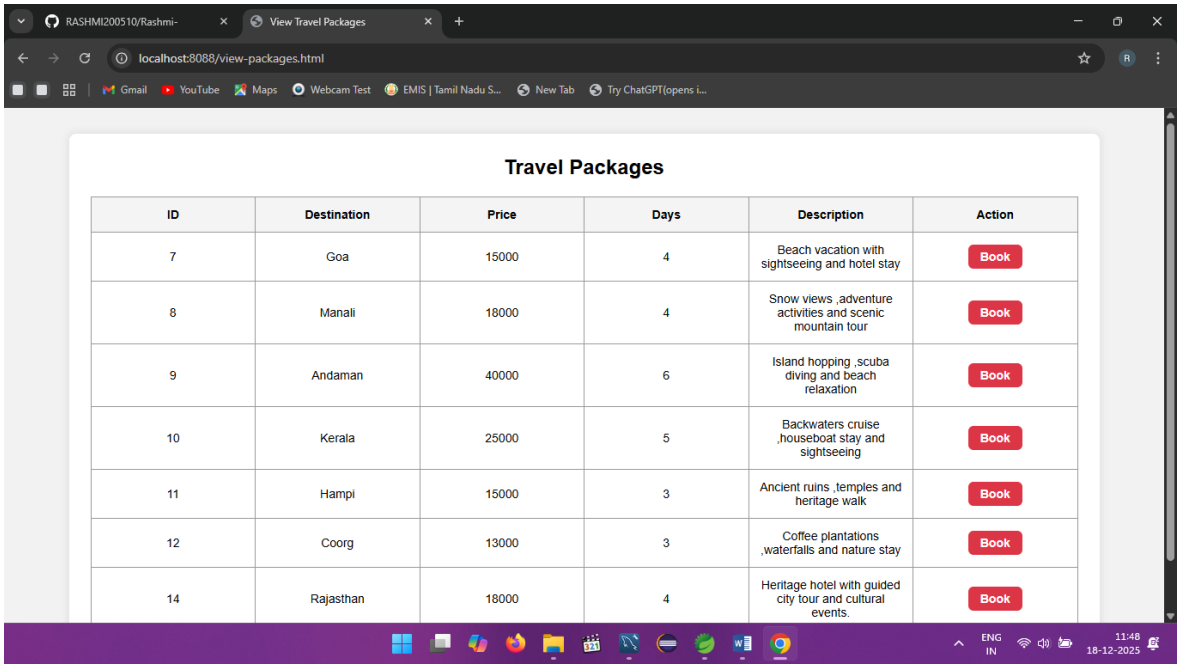
User Login



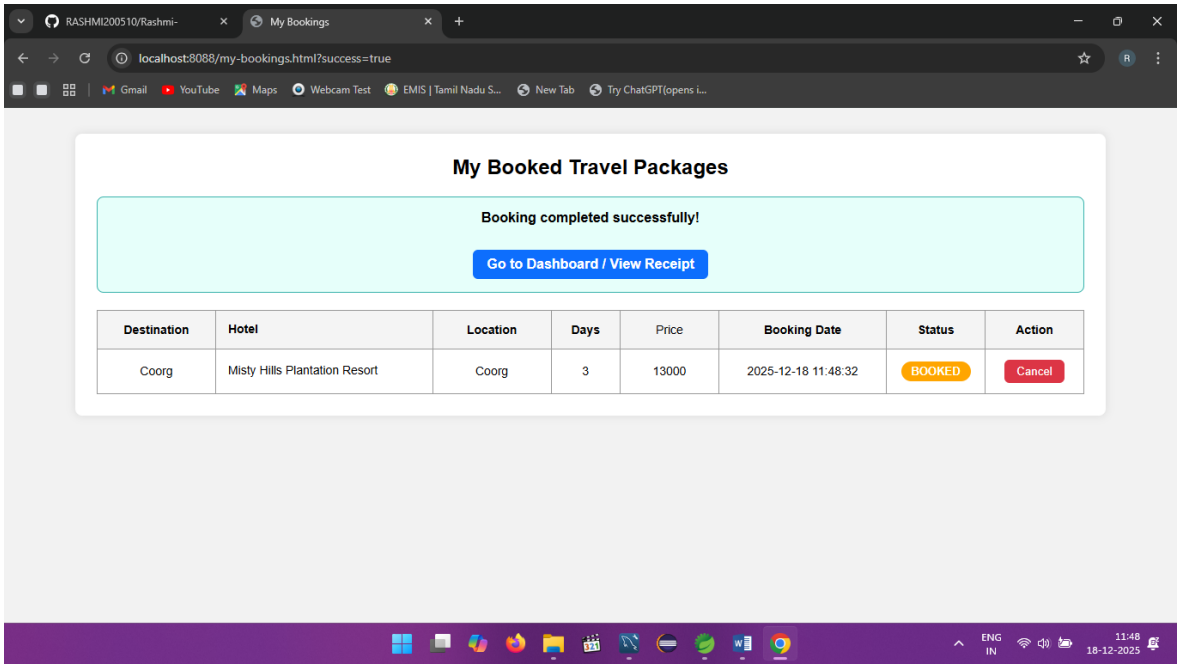
User Dashboard



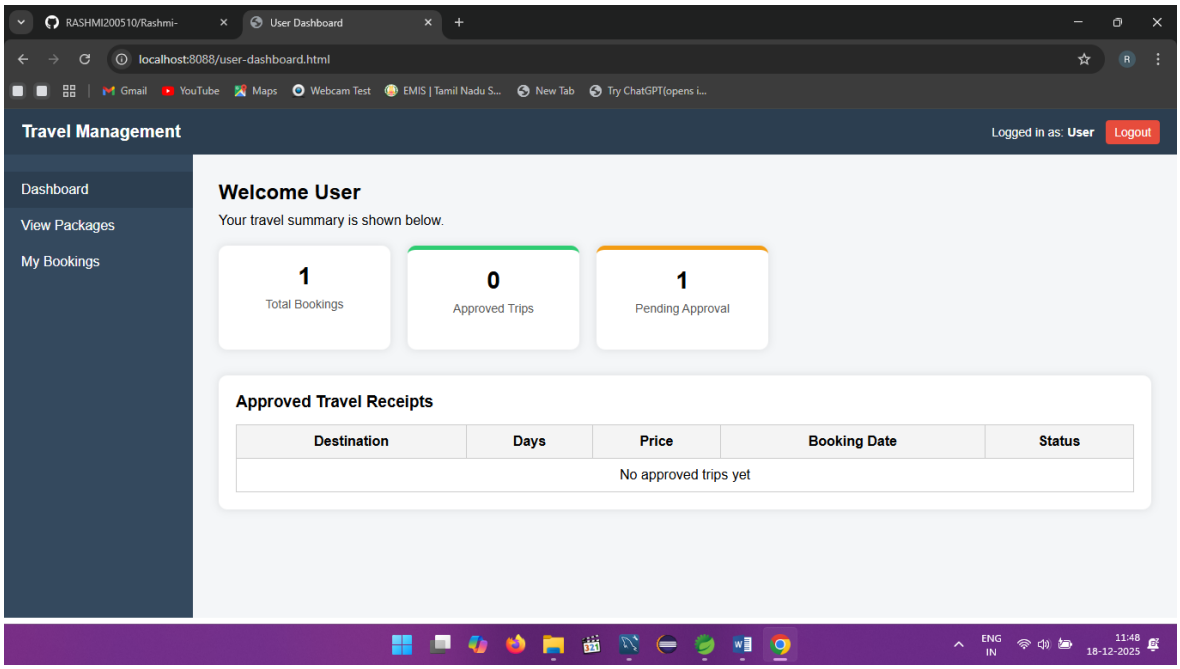
Travel Packages



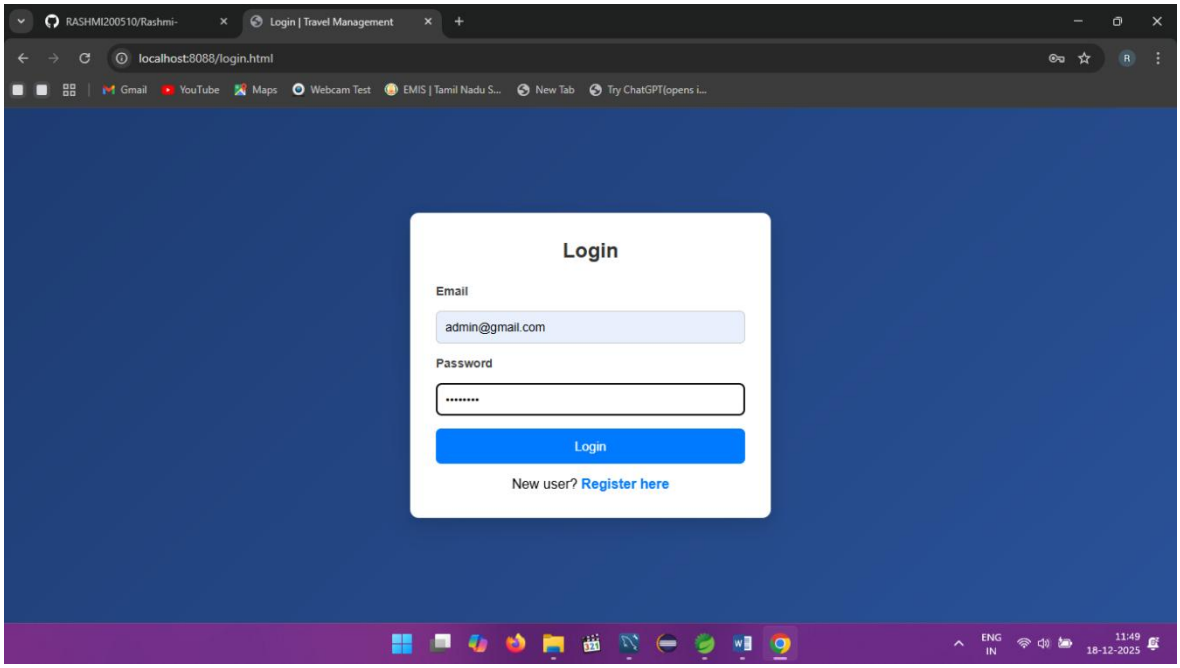
My Booked Travel Packages



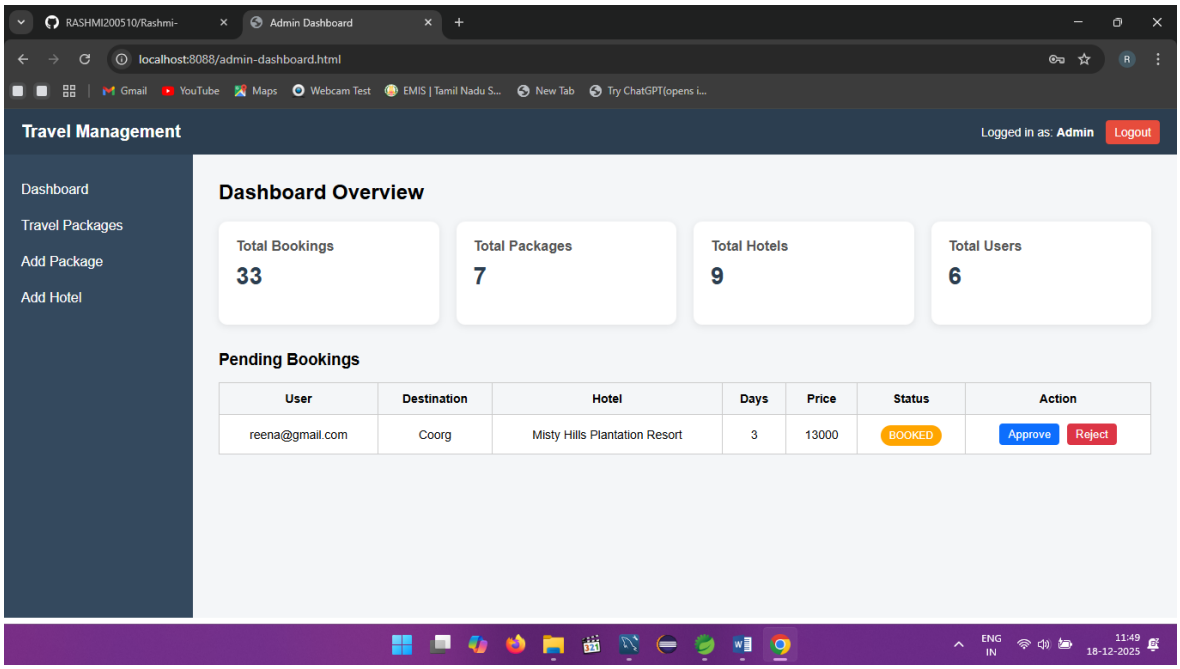
User Dashboard After Booking



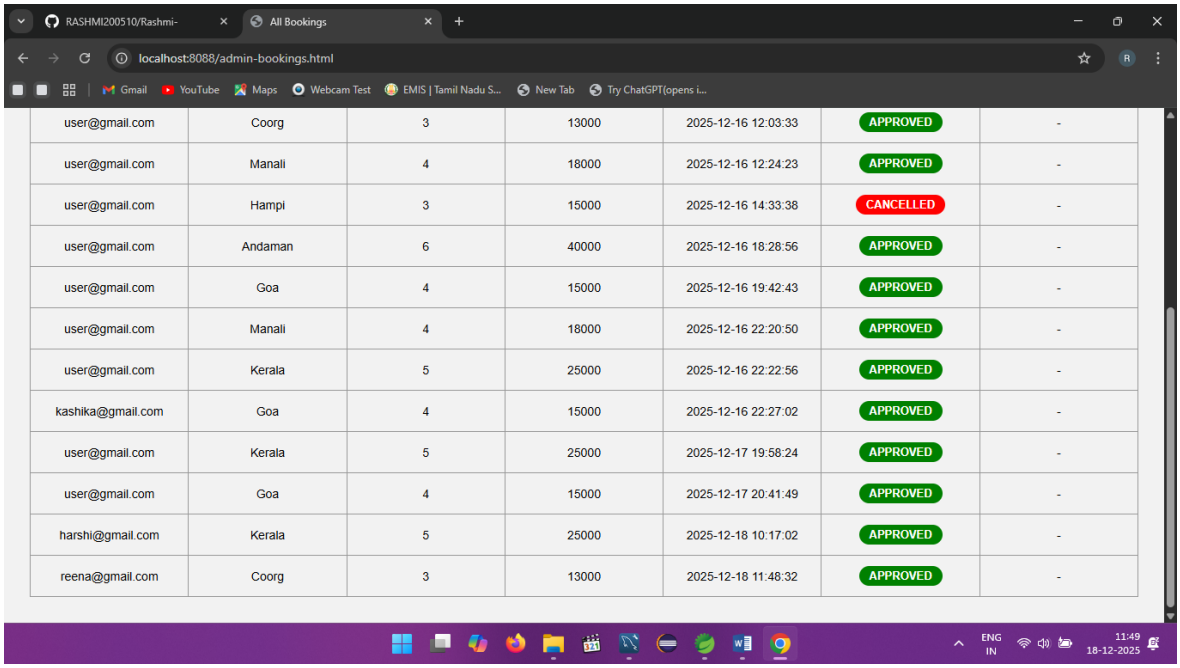
Admin Login



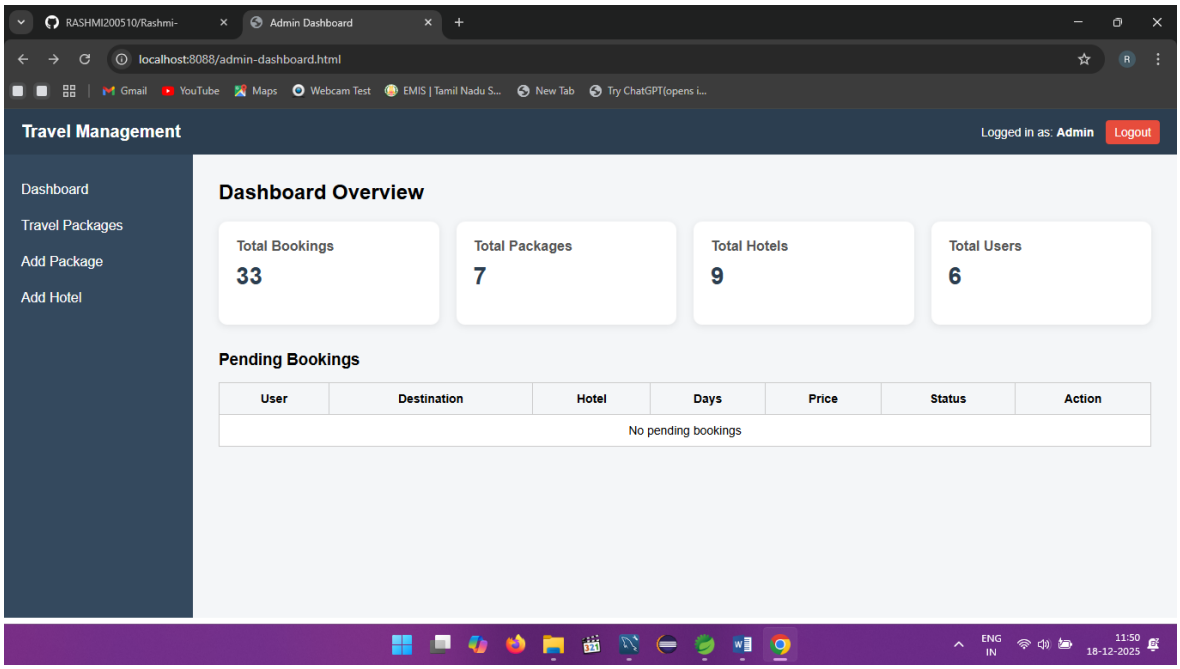
Admin Dashboard



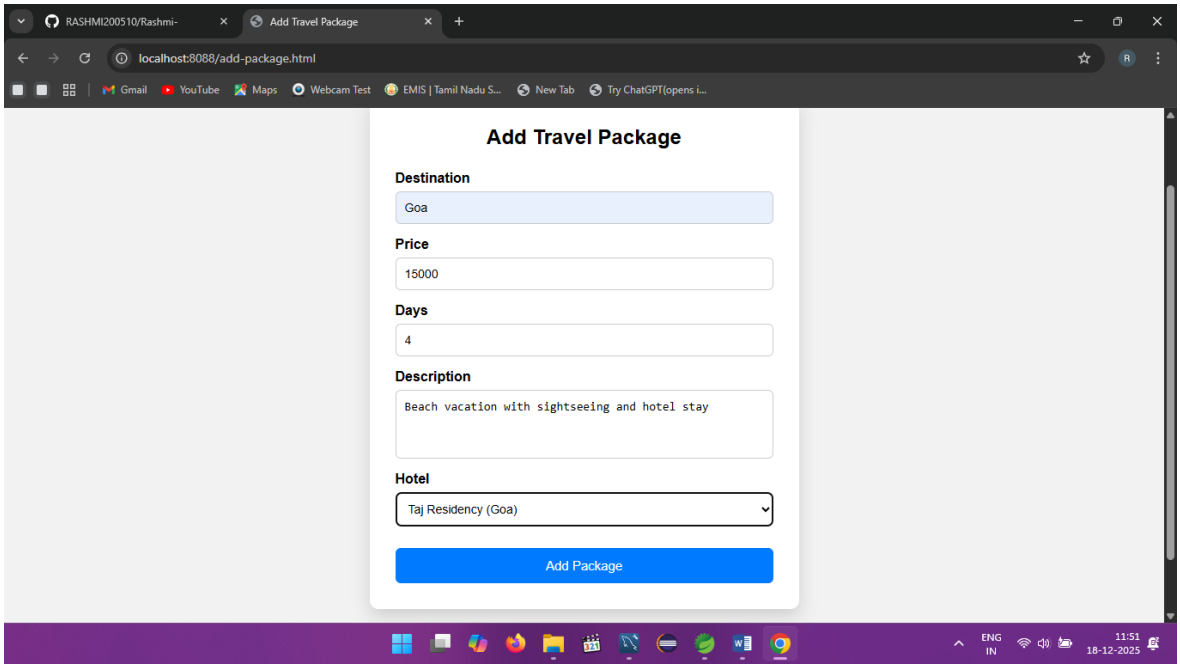
Bookings Page of Admin



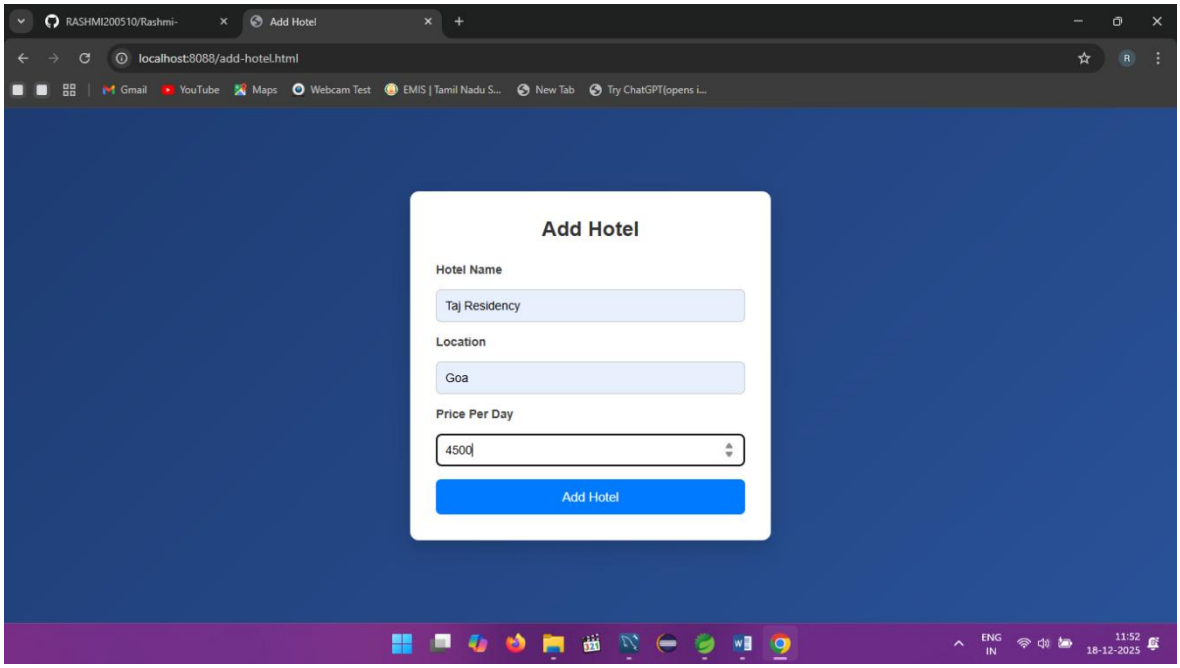
Updated Admin Dashboard



Add Travel Packages Page



Add Hotel Page



User Dashboard After Approval