DREAM TRIP

ABSTRACT :

The **Tourist Management Application (DREAM TRIP)** is a comprehensive web-based platform designed to streamline and enhance the tourism experience for both travelers and administrators. Built using a robust tech stack including Java (Java 17), Struts, Oracle Database, and Apache Tomcat 8, this application offers a wide array of functionalities that cater to the diverse needs of the tourism industry.

For travelers, the application provides an intuitive interface to book tourist places, flights, and hotels, manage bookings, top up their wallets, and rate their experiences. The user management system ensures a seamless registration and login process, allowing users to easily manage their profiles and view their booking history.

For administrators, the application includes a powerful backend system that allows for the efficient management of tourist packages, hotels, and flights. Administrators can add, update, or delete records and view detailed reports on user activities, bookings, and ratings, ensuring smooth and effective management of tourism services.

The architecture of the application follows a multi-tier design, separating the presentation, application, and data layers, which enhances scalability, maintainability, and security. The system requirements are modest, making it accessible for deployment on standard server configurations.

Key features of the application include comprehensive booking management, a robust rating and review system, an administrative panel for managing tourism services, and a secure wallet management system for handling transactions.

This project aims to improve the efficiency and user experience in the tourism sector, providing a one-stop solution for all tourism-related needs. Future enhancements could include integration with third-party APIs, advanced search functionalities, and enhanced security measures to further enrich the user experience and operational capabilities.

INDEX

CHAPTER	TITLE	PAGE NO
	ABSTRACT	i
1	**INTRODUCTION**	
	Introduction	1
	Project Description	2
2	**PROFILE**	
	Software Profile	1
	Project Profile	1
3	**SYSTEM ANALYSIS**	
	Existing System	1
	Proposed System	1
	About the Project	1
	Feasibility Study	1

4	**SYSTEM DESIGN**	
	Input Design	**1**
	Output Design	**1**
	System Architecture	**1**
	Data Flow Diagram	**1**
	E-R Diagram	**1**
	Table Design	**1**
5	**SYSTEM REQUIREMENTS**	
	Software and Hardware Requirements	**1**
	Installation and Setup	**1**
6	**TESTING**	
	Unit Testing	**1**
	Integration Testing	**1**
	User Acceptance Testing (UAT)	**1**
7	**CONCLUSION**	**1**
8	**FUTURE ENHANCEMENT**	**1**
9	**APPENDIX**	
	Source Code	**1**
	Screenshots	**1**
	Bibliography	**1**

CHAPTER 1

INTRODUCTION

INTRODUCTION

Tourism is a rapidly growing industry that plays a crucial role in the global economy. As the number of travelers continues to increase, the need for efficient and user-friendly platforms to manage tourism-related activities has become more evident. The Tourist Management Application is designed to address this need by providing a comprehensive web-based solution for both travelers and administrators.

The Tourist Management Application aims to enhance the overall tourism experience by offering a wide range of functionalities, including booking of tourist places, flights, and hotels, managing bookings, handling payments through wallet top-ups, and collecting user ratings and reviews. The application also includes an administrative panel that enables efficient management of tourist packages, hotels, flights, and user details.

Built using modern web technologies such as Java (Java 17), Struts, Oracle Database, and Apache Tomcat 8, the application ensures robustness, scalability, and security. The multi-tier architecture of the application separates the presentation, application, and data layers, making it easier to maintain and extend.

The primary objectives of the Tourist Management Application are:

- ➢ To provide a seamless and intuitive booking experience for travelers.
- ➢ To offer a centralized platform for managing various tourism-related services.
- ➢ To enhance the efficiency of tourism operations through a robust administrative panel.
- ➢ To ensure secure and reliable transactions through a dedicated wallet management system.
- ➢ To gather and display user feedback through ratings and reviews.

By achieving these objectives, the Tourist Management Application aims to improve the satisfaction of both travelers and service providers, ultimately contributing to the growth and success of the tourism industry.

PROJECT DESCRIPTIONS

The Tourist Management Application is a web-based platform designed to enhance and streamline the experience of managing tourism-related activities. The application caters to both travelers and administrators, providing a comprehensive suite of functionalities to facilitate bookings, manage profiles, handle payments, and administer tourism services efficiently.

Key Objectives

User Experience Enhancement: Offer a user-friendly interface for travelers to book tourist places, flights, and hotels effortlessly.

Administrative Efficiency: Provide robust tools for administrators to manage tourism services, including tourist packages, hotels, flights, and user data.

Secure Transactions: Implement a secure wallet management system for handling user transactions.

Feedback Collection: Enable users to rate and review their experiences, providing valuable feedback for service improvement.

CHAPTER 2

PROFILE

SOFTWARE PROFILE

Oracle Database

Oracle Database is a powerful, enterprise-grade database management system known for its scalability, performance, and robustness. It provides a comprehensive and integrated approach to data management and is widely used in various industries.

Key Features:

- Advanced data security and encryption
- High availability and disaster recovery
- Comprehensive data warehousing and analytics
- Support for SQL, PL/SQL, and Java
- Scalability to handle large amounts of data
- Multi-model database support (relational, JSON, XML, etc.)

Use Case in Project:

- **Data Storage and Management:** Storing user information, booking details, ratings, and reviews.
- **Transaction Management:** Ensuring ACID properties for transactions such as booking and wallet top-ups.

Java

Java is a widely-used, platform-independent, object-oriented programming language known for its portability, performance, and scalability. Java is used to develop web applications, enterprise applications, and various other types of software.

Key Features:

- Write Once, Run Anywhere (WORA) capability
- Strong memory management
- Multi-threading support
- Rich standard library (Java Standard Edition, Java EE)
- Robust security features
- High performance with Just-In-Time (JIT) compilation

Use Case in Project:

- **Backend Development:** Implementing business logic, data processing, and interaction with the database.
- **Server-Side Programming:** Handling HTTP requests and responses, session management, and transaction management.

Struts

Apache Struts is an open-source framework for developing Java EE web applications. It extends the Java Servlet API to encourage developers to adopt a model–view–controller (MVC) architecture.

Key Features:

- **MVC Architecture:** Separates application logic from user interface.
- **Action-based framework:** Facilitates creation of actions to handle requests.
- **Integration with other Java technologies:** Easily integrates with JSP, JSTL, and other Java EE components.
- **Rich tag libraries:** Simplifies the creation of web pages.
- **Extensible:** Allows custom components and plugins.

Use Case in Project:

- **Web Application Framework:** Structuring the application using MVC pattern, handling user inputs, and directing them to the appropriate business logic.
- **Form Handling and Validation:** Managing form submissions and validations.

HTML (HyperText Markup Language)

HTML is the standard markup language used to create web pages. It describes the structure of a webpage and is used in conjunction with CSS and JavaScript to build user interfaces.

Key Features:

- Document structure and layout
- Support for multimedia elements (images, videos, audio)
- Hyperlinking capabilities
- Forms for user input
- Semantic elements for better SEO and accessibility

Use Case in Project:

- **User Interface:** Structuring the content of web pages, creating forms for user inputs, and displaying data.

CSS (Cascading Style Sheets)

CSS is a stylesheet language used to describe the presentation of a document written in HTML or XML. It controls the layout, colors, fonts, and overall visual appearance of web pages.

Key Features:

- Style rules for elements (selectors, properties, and values)
- Responsive design with media queries
- Flexbox and Grid for layout management
- Animation and transition effects
- Custom properties (CSS variables)

Use Case in Project:

- **Styling:** Enhancing the look and feel of web pages, ensuring responsive design, and improving user experience with animations.

JavaScript

JavaScript is a versatile, high-level programming language primarily used to create dynamic and interactive effects within web browsers. It is an essential part of web development, enabling client-side scripting.

Key Features:

- Client-side scripting for interactive web pages
- DOM manipulation to dynamically change HTML and CSS
- Asynchronous programming with Promises and async/await
- Rich standard library with APIs for various functionalities
- Event-driven programming model

Use Case in Project:

- **Interactivity:** Implementing dynamic features such as form validation, AJAX calls for asynchronous data fetching, and DOM manipulation for a responsive user experience.

PROJECT PROFILE

The Tourist Management System has been divided into several modules.

User Management Module

User Registration

Description:

This feature allows new users to create an account by providing necessary details such as username, password, email, and personal information.

Functionality:

- **Input Validation:** Ensures that the provided data meets the required format (e.g., email format, password strength).
- **Data Storage:** Stores user details in the Oracle database.
- **Email Confirmation:** Sends a confirmation email to verify the user's email address.

User Login

Description:

Enables existing users to log in using their username and password.

Functionality:

- **Authentication:** Verifies user credentials against the stored data.
- **Session Management:** Starts a user session upon successful login.

Profile Management

Description:

Allows users to view and update their profile information, including personal details and contact information.

Functionality:

- **Data Retrieval:** Fetches user details from the database.
- **Data Update:** Allows users to update their information and saves the changes in the database.

Booking System Module

Book Single Place

Description:

Enables users to book a specific tourist place with count of adults and count of nights.

Functionality:

- **Availability Check:** Displays available places.
- **Booking Process:** Takes user input for booking details and processes the booking by storing details in the database.

Book Flight

Description:

Allows users to book flights for selected date in case flights not available on the selected day are shown to user **Flight not Available** .

Functionality:

- **Flight Availability:** Shows available flights based on user criteria.
- **Booking Form:** Collects booking details such as flight selection, passenger information, and payment details.
- **Data Storage:** Saves the booking information in the database.

Book Hotel

Description:

Facilitates hotel room bookings for selected date.

Functionality:

- **Hotel Listings:** Displays available hotels and room types.
- **Reservation Form:** Takes user input for check-in and check-out dates, room selection, and payment.
- **Data Storage:** Stores the booking information in the database.

Cancel Booking

Description:

Allows users to cancel their bookings with a particular percentage of cancellation amount.

Functionality:

- **Booking Retrieval:** Fetches the list of user bookings.
- **Cancellation Process:** Enables users to select and cancel a booking, updating the database accordingly.

Change Planning Date

Description:

Enables users to modify the date of their planned bookings.

Functionality:

- **Booking Selection:** Allows users to select a booking to modify.
- **Date Update:** Takes new date input and updates the booking details in the database.

View All Bookings

Description:

Allows users to view a list of all their bookings.

Functionality:

- **Data Retrieval:** Fetches all booking records for the user from the database.
- **Display:** Presents the booking details in a user-friendly format.

View Single Booking

Description:

Allows users to view the details of a single booking.

Functionality:

- **Data Retrieval:** Fetches detailed information of a specific booking.
- **Display:** Presents detailed booking information, including date, time, location, and payment details.

Ratings and Reviews Module

Show All Ratings

Description:

Displays all user ratings and reviews for various services.

Functionality:

- **Data Retrieval:** Fetches ratings and reviews from the database.
- **Display:** Shows the ratings and reviews in a readable format.

Add Ratings

Description:

Enables users to add ratings and reviews for places, flights, and hotels they have used.

Functionality:

- **Input Form:** Provides a form for users to submit their ratings and reviews.
- **Data Storage:** Saves the ratings and reviews in the database.

Administration Panel Module

Manage Tourist Packages

Add Tourist Packages

Description:

Allows administrators to add new tourist packages.

Functionality:

- **Input Form:** Collects details about the new package.
- **Data Storage:** Saves the new package details in the database.

Update Tourist Packages

Description:

Enables administrators to update existing tourist packages.

Functionality:

- **Data Retrieval:** Fetches existing package details.
- **Update Form:** Allows administrators to modify package information.
- **Data Storage:** Saves the updated package details in the database.

Delete Tourist Packages

Description:

Allows administrators to delete tourist packages.

Functionality:

- **Package Selection:** Enables administrators to select a package to delete.
- **Data Removal:** Deletes the selected package from the database.

Show All Tourist Packages

Description:

Displays all tourist packages available in the system.

Functionality:

- **Data Retrieval:** Fetches all package details from the database.
- **Display:** Presents the package information in an organized format.

Manage Hotels

Add Hotels

Description:

Allows administrators to add new hotels to the system.

Functionality:

- **Input Form:** Collects details about the new hotel.
- **Data Storage:** Saves the new hotel details in the database.

Update Hotels

Description:

Enables administrators to update existing hotel information.

Functionality:

- **Data Retrieval:** Fetches existing hotel details.
- **Update Form:** Allows administrators to modify hotel information.
- **Data Storage:** Saves the updated hotel details in the database.

Delete Hotels

Description:

Allows administrators to delete hotels from the system.

Functionality:

- **Hotel Selection:** Enables administrators to select a hotel to delete.
- **Data Removal:** Deletes the selected hotel from the database.

Show All Hotels

Description:

Displays all hotels available in the system.

Functionality:

- **Data Retrieval:** Fetches all hotel details from the database.
- **Display:** Presents the hotel information in an organized format.

Manage Flights

Add Flights

Description:

Allows administrators to add new flights to the system.

Functionality:

- **Input Form:** Collects details about the new flight.
- **Data Storage:** Saves the new flight details in the database.

Update Flights

Description:

Enables administrators to update existing flight information.

Functionality:

- **Data Retrieval:** Fetches existing flight details.
- **Update Form:** Allows administrators to modify flight information.
- **Data Storage:** Saves the updated flight details in the database.

Delete Flights

Description:

Allows administrators to delete flights from the system.

Functionality:

- **Flight Selection:** Enables administrators to select a flight to delete.
- **Data Removal:** Deletes the selected flight from the database.

Show All Flights

Description:

Displays all flights available in the system.

Functionality:

- **Data Retrieval:** Fetches all flight details from the database.
- **Display:** Presents the flight information in an organized format.

View All User Details

Description:

Allows administrators to view detailed information about all registered users.

Functionality:

- **Data Retrieval:** Fetches user details from the database.
- **Display:** Presents user information in an organized format.

View All Bookings

Description:

Allows administrators to view all bookings made by users.

Functionality:

- **Data Retrieval:** Fetches booking details from the database.
- **Display:** Presents booking information in a comprehensive manner.

View All Ratings

Description:

Allows administrators to view all user ratings and reviews.

Functionality:

- **Data Retrieval:** Fetches ratings and reviews from the database.
- **Display:** Shows ratings and reviews for analysis and feedback.

Wallet Management Module

Wallet Top-Up

Description:

Allows users to top up their wallet balance to make payments for bookings.

Functionality:

- **Top-Up Form:** Collects payment information for topping up the wallet.
- **Transaction Processing:** Processes the payment and updates the wallet balance in the database.

CHAPTER 3

SYSTEM ANALYSIS

Existing System

The current tourism management process is often fragmented and manual, leading to inefficiencies and a subpar user experience.

Common issues in the existing system include:

- **Manual Bookings:** Users often have to visit travel agencies or manually book flights, hotels, and tourist packages through different platforms, leading to a time-consuming and inconvenient process.

- **Lack of Integration:** The absence of an integrated system results in users managing multiple accounts and bookings separately for flights, hotels, and tourist attractions.

- **Poor User Experience:** The manual and fragmented approach often results in a lack of personalized user experience, with users needing to provide the same information multiple times across different platforms.

- **Limited Accessibility:** Users may have difficulty accessing booking information and making changes to their plans, especially if they need to interact with multiple service providers.

- **Inadequate Management Tools:** Administrators face challenges in managing bookings, user data, and ratings efficiently due to the lack of a centralized system.

Proposed System

The Tourist Management Application aims to address the shortcomings of the existing system by providing an integrated, user-friendly platform that streamlines the entire process of managing tourism-related services.

Key improvements include:

- **Integrated Booking System:** Users can book flights, hotels, and tourist packages from a single platform, enhancing convenience and efficiency.
- **Centralized User Management:** Users have a single account to manage all their bookings, personal information, and preferences, leading to a more personalized experience.
- **Enhanced User Experience:** The application provides a seamless and intuitive user interface, making it easier for users to browse, book, and manage their travel plans.
- **Real-time Updates:** Users receive real-time updates on their bookings, availability of services, and any changes to their travel plans.
- **Comprehensive Administration Tools:** Administrators have access to robust tools for managing bookings, user data, ratings, and reviews, leading to better operational efficiency.

About the Project

The Tourist Management Application is a web-based platform designed to streamline the management of tourism-related services. It integrates user management, booking systems, ratings and reviews, and administrative tools into a single cohesive system. The application is built using a robust tech stack, including Java, Oracle Database, Struts, HTML, CSS, and JavaScript, to ensure high performance, security, and scalability.

Project Objectives:

- To provide users with a convenient and efficient way to book and manage flights, hotels, and tourist packages.
- To enhance user experience through a unified and intuitive platform.
- To enable administrators to manage tourism services efficiently through comprehensive tools.

Project Scope:

- User registration, login, and profile management.
- Booking system for flights, hotels, and tourist packages.
- Ratings and reviews module for user feedback.
- Administrative panel for managing services and user data.
- Wallet management for easy payment processing.

Feasibility Study

A feasibility study was conducted to evaluate the viability of the Tourist Management Application from various perspectives:

Technical Feasibility

- **Technology Stack:** The selected technology stack (Java, Oracle Database, Struts, HTML, CSS, JavaScript) is well-established, widely supported, and capable of handling the requirements of the application.

- **Development Expertise:** The development team possesses the necessary expertise and experience with the chosen technologies to successfully implement the project.

- **Infrastructure:** The required infrastructure (servers, databases, development tools) is readily available and can support the development and deployment of the application.

Economic Feasibility

- **Cost-Benefit Analysis:** The benefits of developing the application, such as improved user satisfaction, operational efficiency, and potential revenue from increased bookings, outweigh the development and maintenance costs.

- **Budget Availability:** Sufficient budget is available to cover development, testing, deployment, and ongoing maintenance costs.

Operational Feasibility

- **User Adoption:** The integrated and user-friendly nature of the application is expected to drive high user adoption rates, reducing the reliance on fragmented systems.

- **Administrative Efficiency:** The comprehensive tools provided to administrators will streamline operations and improve management efficiency.

- **Support and Maintenance:** A robust plan is in place for ongoing support and maintenance to ensure the application remains functional and up-to-date.

Legal Feasibility

- **Compliance:** The application will adhere to relevant legal and regulatory requirements, including data protection and privacy laws.

- **Licensing:** Necessary licenses for the technologies and third-party components used in the application will be obtained to avoid any legal issues.

CHAPTER 4

SYSTEM DESIGN

Input Design

Input design is crucial for ensuring that the data entered into the system is accurate, complete, and usable. In the Tourist Management Application, input design focuses on the following key forms and interfaces:

User Registration Form

- **Fields:** Username, Password, Email, First Name, Last Name, Date of Birth, Address, Phone Number.
- **Validation:** Email format, password strength, mandatory fields check.

User Login Form

- **Fields:** Username, Password.
- **Validation:** Mandatory fields check, credentials verification.

Profile Update Form

- **Fields:** First Name, Last Name, Date of Birth, Address, Phone Number.
- **Validation:** Mandatory fields check, format checks.

Booking Forms

- **Fields:** Destination, Check-in/Check-out Dates, Number of Guests, Flight Details (if booking a flight), Hotel Selection (if booking a hotel), Payment Information.
- **Validation:** Date validation, availability checks, mandatory fields check.

Rating and Review Form

- **Fields:** Service Type (Flight/Hotel/Place), Rating (1-5), Review Text.
- **Validation:** Rating range check, mandatory fields check.

Admin Forms

- **Fields:** Tourist Package Details, Hotel Details, Flight Details, User Details.
- **Validation:** Mandatory fields check, format checks.

Output Design

Output design focuses on the presentation of data to users and administrators, ensuring that the information is clear, concise, and accessible.

User Dashboards

- **Components:** Upcoming Bookings, Recent Bookings, Profile Information, Wallet Balance.
- **Format:** Tables, cards, and charts.

Booking Confirmation

- **Components:** Booking Details, Payment Confirmation, Booking Reference Number.
- **Format:** Printable document, email confirmation.

Rating and Review Displays

- **Components:** Service Name, Rating, Review Text, Reviewer Name.
- **Format:** List view, star ratings.

Admin Dashboards

- **Components:** User Management, Booking Management, Ratings Overview, Service Management.
- **Format:** Tables, graphs, and interactive elements.

System Architecture

The system architecture of the Tourist Management Application follows a multi-tier approach to ensure scalability, maintainability, and security.

Presentation Layer

- **Technologies:** HTML, CSS, JavaScript.
- **Components:** User interfaces, input forms, dashboards.

Business Logic Layer

- **Technologies:** Java, Struts.
- **Components:** Business logic, application services, controllers.

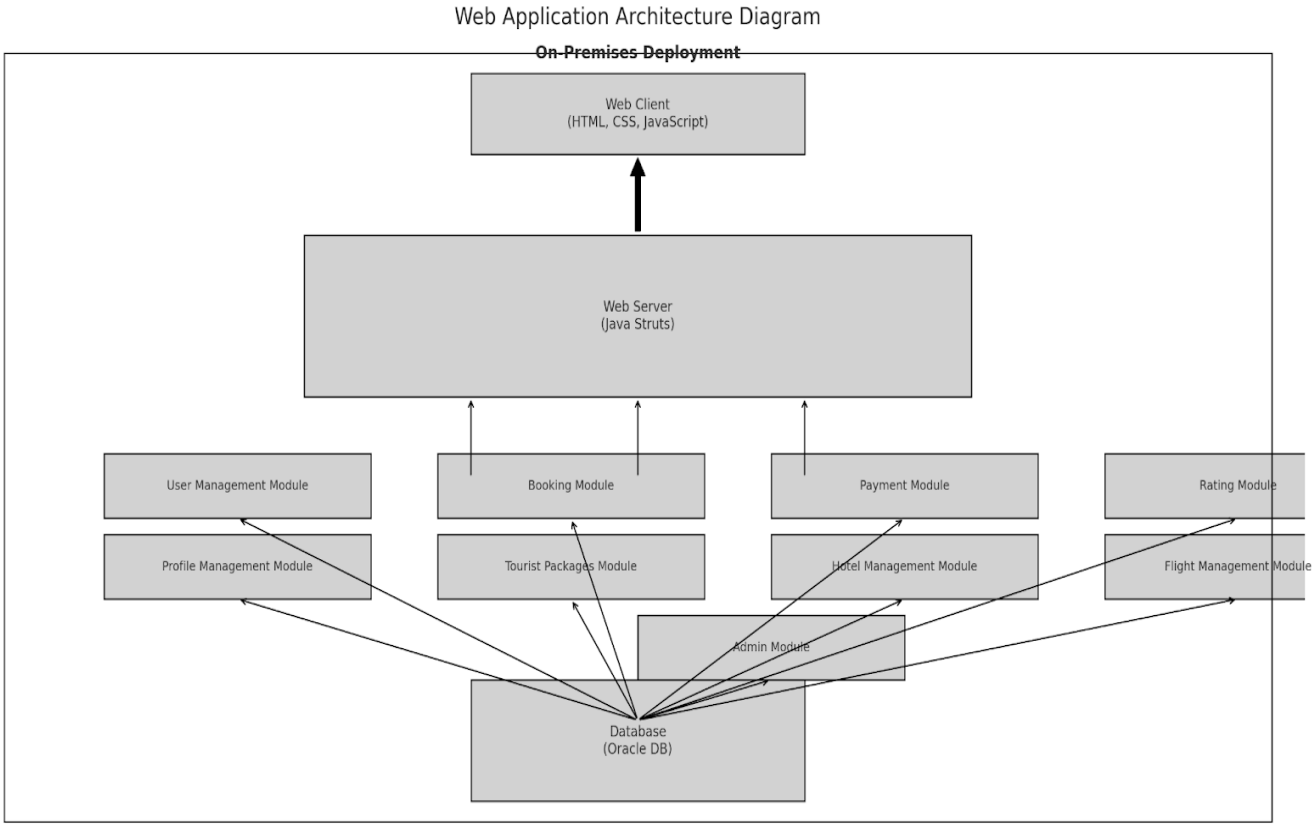
Data Access Layer

- **Technologies:** JDBC, Oracle Database.
- **Components:** Data access objects (DAOs), database connection management.

Database Layer

- **Technologies:** Oracle Database.
- **Components:** Tables, stored procedures, triggers.

SYSTEM ARCHITECTURE DIAGRAM



Data Flow Diagram

Data Flow Explanation

User Registration/Login:

- Both User and Admin interact with the Login/Register process.
- The Login/Register process validates credentials and manages sessions.
- User and Admin credentials are checked against the Oracle Database.

User Processes:

After logging in, a User can perform three main functions:

- **Bookings Management:** The user can manage bookings for places, flights, and hotels. This involves checking availability, making bookings, and viewing booking details.

- **Ratings Management:** The user can submit ratings and reviews for services like places, flights, and hotels.

- **Profile Management:** The user can view and update their profile details.

Admin Processes:

After logging in, an Admin can perform two main functions:

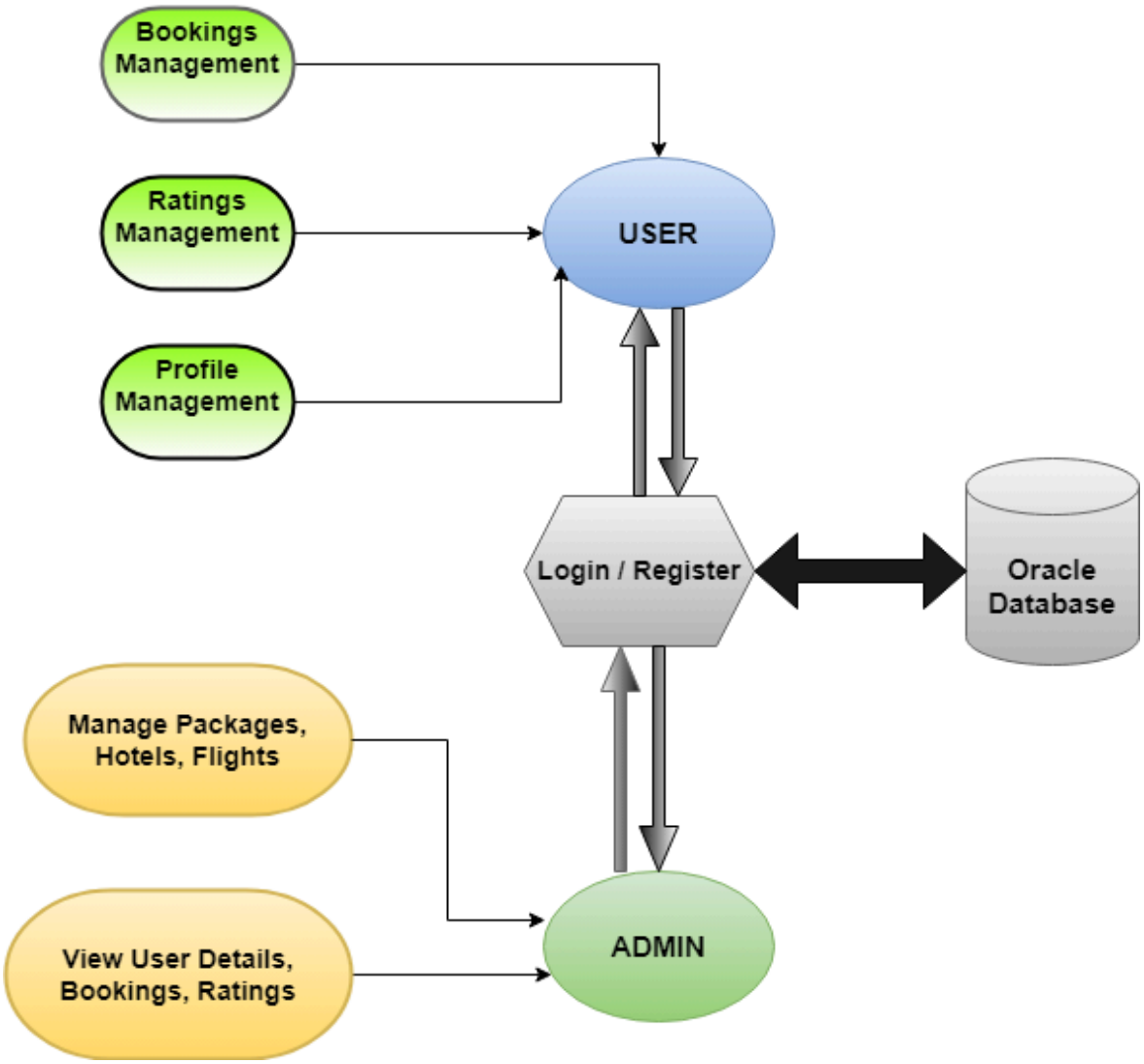
- Manage Packages, Hotels, Flights: The admin can add, update, or delete details about tourist packages, hotels, and flights. This ensures the database is up-to-date with the latest offerings.

- View User Details, Bookings, Ratings: The admin can view detailed information about users, their bookings, and their ratings. This helps in managing user activities and ensuring service quality.

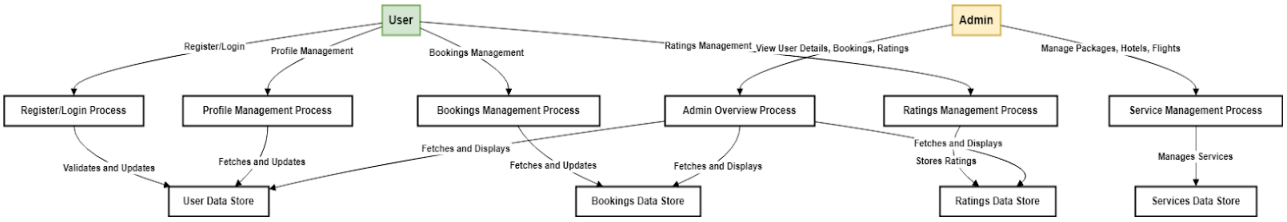
Data Storage and Retrieval:

- All the interactions with user details, bookings, ratings, packages, hotels, and flights are stored and retrieved from the Oracle Database. This ensures that all data is centralized, secure, and easily accessible for the application processes.

DFD DIAGRAM



E-R Diagram



Definition

An Entity-Relationship (E-R) Diagram is a visual representation of the entities within a system and the relationships between those entities. It is widely used in database design and development to illustrate the logical structure of a database.

Components

Entities:

- Represent objects or things in the system, such as Users, Bookings, Flights, Hotels, Packages, etc.
- Depicted as rectangles in the E-R diagram.

Attributes:

- Characteristics or properties of an entity, such as UserID, UserName, BookingID, FlightNumber, HotelName, etc.
- Depicted as ovals connected to their respective entities.

Relationships:

- Illustrate how entities interact with each other. For example, a User books a Flight or a Hotel.
- Depicted as diamonds connecting related entities.

Primary Key:

- A unique identifier for each entity instance, such as UserID for Users, BookingID for Bookings.
- Underlined in the entity rectangle.

Foreign Key:

- An attribute in one entity that is a primary key in another entity, establishing a link between the entities.
- Often depicted using a line with an arrow pointing to the primary key.

Table Design

User Details (user_details) :

Column Name	Null ?	Type
USER_ID (PK)	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(100)
EMAIL_ID (UNIQUE)	NOT NULL	VARCHAR2(100)
MOBILE_NO	NOT NULL	NUMBER(20)
PASSWORD	NOT NULL	VARCHAR2(30)
WALLET		NUMBER(30)
STATUS		VARCHAR2(30)
REGISTER_DATE		TIMESTAMP(6)

Admin Details (admin_details) :

Column Name	Null ?	Type
ADMIN_ID (PK)	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(100)
EMAIL_ID (UNIQUE)	NOT NULL	VARCHAR2(100)
MOBILE_NO	NOT NULL	NUMBER(20)
PASSWORD	NOT NULL	VARCHAR2(30)

Package modes (package_modes) :

Column Name	Null ?	Type
PACKAGE_ID (PK)	NOT NULL	NUMBER
PACKAGE_NAME	NOT NULL	VARCHAR2(500)
PACKAGE_PRICE_1N	NOT NULL	NUMBER(30,2)
SEASON	NOT NULL	VARCHAR2(60)
PROTOCOLS	NOT NULL	VARCHAR2(3999)
DESCRIPTION	NOT NULL	VARCHAR2(3999)
STATUS		VARCHAR2(30)
IMAGE		VARCHAR2(4000)

Flight Details (flights_details) :

Column Name	Null ?	Type
FLIGHT_NO (PK)	NOT NULL	NUMBER
FLIGHT_NAME	NOT NULL	VARCHAR2(100)
DEPARTURE	NOT NULL	VARCHAR2(100)
DESTINATION	NOT NULL	VARCHAR2(100)
DEPATURE_DATE_TIME	NOT NULL	TIMESTAMP(6)
ARRIVAL_DATE_TIME	NOT NULL	TIMESTAMP(6)
BUSINESS_CLASS_FARE	NOT NULL	NUMBER(30,2)
ECONOMIC_CLASS_FARE	NOT NULL	NUMBER(30,2)
STATUS	NOT NULL	VARCHAR2(60)
BUSINESS_CLASS_SEAT_STATUS	NOT NULL	NUMBER
ECONOMIC_CLASS_SEAT_STATUS	NOT NULL	NUMBER

Hotel Details (hotel_details) :

Column Name	Null ?	Type
HOTEL_ID (PK)	NOT NULL	NUMBER
LOCATION	NOT NULL	VARCHAR2(100)
HOTEL_NAME	NOT NULL	VARCHAR2(500)
ROOM_TYPE_MID_RANGE_PRICE	NOT NULL	NUMBER(30,2)
ROOM_TYPE_PREMIUM_PRICE	NOT NULL	NUMBER(30,2)
STATUS		VARCHAR2(30)
IMAGE		VARCHAR2(4000)

Booking Details (booking_details) :

Column Name	Null ?	Type
BOOKING_ID (PK)	NOT NULL	NUMBER
USER_ID (FK)	NOT NULL	NUMBER
PACKAGE_ID (FK)	NOT NULL	NUMBER
FLIGHT_NO (FK)	NOT NULL	NUMBER
HOTEL_ID (Fk)	NOT NULL	NUMBER
NUMBER_OF_PERSON	NOT NULL	NUMBER
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
TOTAL_PRICE	NOT NULL	NUMBER(30,2
STATUS	NOT NULL	VARCHAR2(30)
BOOKING_DATE	NOT NULL	TIMESTAMP(6)
FLIGHT_CLASS	NOT NULL	VARCHAR2(30)
HOTEL_ROOM_TYPE	NOT NULL	VARCHAR2(30)
DAYS_IN_NIGHT	NOT NULL	VARCHAR2(30)
PACKAGE_NAME	NOT NULL	VARCHAR2(50)
PAYMENT_DETAILS	NOT NULL	VARCHAR2(100)
NO_OF_ROOM	NOT NULL	NUMBER(30)

User Feedback (users_feedback) :

Column Name	Null ?	Type
FEEDBACK_ID (PK)	NOT NULL	NUMBER
USER_ID (FK)	NOT NULL	NUMBER
BOOKING_ID (FK)	NOT NULL	NUMBER
PACKAGE_ID (FK)	NOT NULL	NUMBER
USER_NAME	NOT NULL	VARCHAR2(60)
PACKAGE_NAME	NOT NULL	VARCHAR2(3999)
RATING	NOT NULL	NUMBER(5,2)
DESCRIPTION		VARCHAR2(200)
STATUS		VARCHAR2(30)

CHAPTER 5

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS :

Processor	:	Intel Core i5
Processor Speed:	:	3.06 GHz
Ram	:	8 GB
Hard Disk Drive	:	250 GB

SOFTWARE REQUIREMENTS :

Operating System	:	Windows 10 / 11
Technology Used	:	Advanced Java (J2EE)
Database	:	ORACLE Database
Connectivity	:	JDBC
Web Server	:	Apache - Tomcat
Browser	:	Google Chrome

INSTALLATION AND SETUP

PREREQUISITES

- Install JDK 17
- Install Apache Tomcat 8
- Install Oracle Database

STEPS :

Clone the project repository:

- Bash (SSH) : **git clone git@github.com:ajithak003/touristmanagementapp-web.git**
- HTTP : **https://github.com/ajithak003/touristmanagementapp-web.git**

Set up the database:

- Create the required tables and insert initial data using the provided SQL scripts.

Configure the project:

- Update database connection settings in the configuration files.

Deploy the application on Tomcat:

- Build the project using Maven or Gradle.

- Deploy the generated WAR file to the Tomcat server.

Start the Tomcat server:

- Access the application via the browser at **http://localhost:8080/touristmanagementapp-web/index.jsp**

CHAPTER 6

TESTING

Testing is crucial for ensuring the reliability and performance of your web application.

UNIT TESTING

Objective:

Test individual components or methods in isolation to ensure they function correctly.

Components to Test:

User Registration and Login:

- **Registration:** Validate input fields, password strength, user existence check, and successful user creation.
- **Login:** Verify correct authentication, handling of invalid credentials, and session management.

Booking Operations:

- **Place Booking:** Test logic for selecting and reserving a place, handling edge cases like double booking.
- **Flight and Hotel Booking:** Ensure that booking processes correctly handle available inventory and booking constraints.

Profile Management:

- **Update Profile:** Validate correct processing of profile updates (e.g., changing contact details).
- **Wallet Top-Up:** Test the addition of funds, balance updates, and error handling for invalid transactions.

CRUD Operations:

- **Packages, Hotels, Flights:** Ensure that create, read, update, and delete operations work correctly for each of these entities.

Tools:

JUnit, Mockito (for mocking), AssertJ (for fluent assertions).

Integration Testing

Objective:

Test interactions between different components and systems to ensure they work together seamlessly.

Components to Test:

Database Integration:

- **Data Access Layer:** Verify that the application correctly interacts with the Oracle database for CRUD operations.

- **Transaction Management:** Ensure that transactions are handled correctly, including rollbacks and commits.

Booking Flow:

- **End-to-End Booking:** Test the full flow from place selection through flight and hotel booking, including interactions with the database and any external services.

API Endpoints:

- **REST Endpoints:** Validate that API endpoints return expected results and handle various request scenarios (e.g., valid/invalid inputs).

- **Integration with External Systems:** If applicable, test integration with any third-party services (though you mentioned no third-party APIs).

Tools:

JUnit with Spring Test, TestNG, Postman (for API testing).

User Acceptance Testing (UAT)

Objective:

Ensure the application meets business requirements and is ready for real-world use.

Components to Test:

End-to-End User Scenarios:

- **User Journeys:** Test complete workflows such as user registration, searching and booking places, and managing bookings.

- **Error Handling:** Validate that error messages are user-friendly and provide clear instructions for resolving issues.

Usability Testing:

- **User Interface:** Ensure the interface is intuitive and easy to navigate.
- **Accessibility:** Verify that the application is accessible to users with disabilities (e.g., screen readers, keyboard navigation).

Performance Testing:

- **Load Testing:** Simulate multiple users to ensure the application can handle expected loads without performance degradation.
- **Stress Testing:** Test the application's limits to identify potential breaking points.

Tools:
Selenium (for UI testing), Cucumber (for BDD), Apache JMeter (for performance testing).

Test Strategy Summary:

- **Unit Testing:** Focus on individual components to ensure correctness.
- **Integration Testing:** Validate interactions between components and external systems.
- **UAT:** Ensure the application meets user requirements and performs well in real-world scenarios.

CHAPTER 7

CONCLUSION

The Tourist Management Application successfully delivers a comprehensive and user-friendly platform for managing various aspects of travel planning and booking. By incorporating key features such as user registration, login, profile management, and an extensive booking system for places, flights, and hotels, the application offers a seamless experience for users.

Key Achievements:

User-Friendly Interface:

- Simplified navigation and intuitive design ensure a positive user experience.
- Easy access to booking options and user profile management enhances usability.

Comprehensive Booking System:

- Efficiently handles bookings for places, flights, and hotels.
- Provides options for booking modifications and cancellations, offering flexibility to users.

Robust Administrative Tools:

- Allows administrators to manage tourist packages, hotels, and flights effectively.
- Enables viewing of all user details, bookings, and ratings, aiding in operational oversight.

Security and Performance:

- Ensures data security through secure registration and login processes.
- Performance testing guarantees that the application can handle high traffic volumes.

Detailed Testing:

- Comprehensive unit, integration, and user acceptance testing ensure a reliable and stable application.

The **Tourist Management Application (DREAM TRIP)** stands as a robust and versatile solution for managing travel-related activities. Its well-thought-out design and extensive feature set make it an invaluable tool for both users and administrators. Continuous improvements and future enhancements will further solidify its position as a premier travel management platform.

CHAPTER 8

FUTURE ENHANCEMENT

Mobile Application Development:

- Native apps for iOS and Android.
- Responsive design for seamless experience.

AI-Powered Recommendations:

- Personalized travel suggestions.
- Predict popular destinations and dynamic pricing.

Enhanced Payment Options:

- Multiple payment gateways.
- Support for digital wallets and cryptocurrencies.

Social Media Integration:

- Share bookings on social media.
- Social media login options.

Advanced Analytics and Reporting:

- Analytics dashboards for tracking and optimizing.
- Data-driven marketing strategies.

Multilingual Support:

- Offer multiple languages.
- Real-time translation for user content.

Enhanced Security Measures:

- Two-factor authentication (2FA).
- Advanced encryption for data protection.

CHAPTER 9

APPENDIX

SOURCE CODE

Register.java

```
package com.touristmgntapp.controller;

import java.io.IOException;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.touristmgntapp.dao.Impl.UserTableDaoImplement;
import com.touristmgntapp.exception.UserDefineException;
import com.touristmgntapp.model.UserClass;

@WebServlet("/register")
public class UserRegister extends HttpServlet {

	private static final long serialVersionUID = 1L;

	@Override
	public void doPost(HttpServletRequest request, HttpServletResponse response) {

		try {

			UserTableDaoImplement userDao = new UserTableDaoImplement();

			String name = request.getParameter("FullName");

			String email = request.getParameter("regemail");
			email = email.trim().toLowerCase();

			long mboilNo = Long.parseLong(request.getParameter("regmobile"));
```

```
                        String password = request.getParameter("regpsw");

                        boolean verifi = true;
                        verifi = userDao.emailvalid(email);

                        if (!verifi) {
                                throw new UserDefineException();

                        }

                        else {
                                if (email.contains("@admin")) {

                                        response.sendRedirect("register.jsp?notallow=not allowed
to use @admin");

                                } else {

                                        UserClass userinsert = new UserClass(name, email,
mboilNo, password);
                                        boolean boo = userDao.insertUser(userinsert);
                                        if (boo) {


response.sendRedirect("login.jsp?infomsg=successfully registered");

                                        }

                                }
                        }
                } catch (UserDefineException | IOException e) {
                        try {
                                response.sendRedirect("register.jsp?errormsg=This email id
already registered");
                        } catch (IOException e1) {
                                System.out.println(e1.getMessage());
                        }

                }

        }
}
```

Register.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="ISO-8859-1">
<link rel="icon" type="" href="Assets/logo.png">
<title>register Form</title>
<script

src="https://cdn.jsdelivr.net/npm/sweetalert2@11.3.10/dist/sweetalert2.all.min.js"></script>
<link rel='stylesheet'
	href='https://cdn.jsdelivr.net/npm/sweetalert2@10.10.1/dist/sweetalert2.min.css'>

<link rel='stylesheet' href="assets/css/register.css">

</head>
<body>

	<form action="register" id="register" method="post">
		<div class="loginbox">
			<h1>Register</h1>
			<div class="textbox">
				<input type="text" placeholder="FullName" name="FullName"
value=""
					id="" required autofocus onkeyup="remove()"
pattern="^[a-zA-Z]*$*\s"
					title="must contain text only minimum 2 characters"
					aria-label="FullName">
			</div>
			<div class="textbox">
				<input type="email" placeholder="Email" name="regemail"
value=""
					id="" required
pattern="[A-Za-z0-9]+[@][a-zA-Z]+[.][A-Za-z]{2,3}"
					title="follow this pattern 'abc@xyz.com'" aria-label="Email
id">
			</div>
```

```
                    <div class="textbox">
                        <input type="text" placeholder="Mobile No" name="regmobile"
value=""
                            id="" required pattern="[6-9][0-9]{9}"
                            title="Must contain 10 numbers only and starting 6-9 only"
                            aria-label="Mobile No">
                    </div>

                    <div class="textbox">
                        <input type="password" placeholder="Password" name="regpsw"
value=""
                            id="psw" onkeyup="checkpattern()" required

pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,15}$"
                            title="follow this pattern Ex: 'Abcd@123' or 'abCd234$'"
                            aria-label="Password">
                    </div>

                    <div>
                        <ul>
                            <li id="upper">At least one upper case[A-Z]</li>
                            <li id="lower">At least one lower case [a-z]</li>
                            <li id="number">At least one number [0-9]</li>
                            <li id="special">At least one special character
[@$!%*?&]</li>
                            <li id="char">At least 8 character</li>
                        </ul>
                    </div>

                    <button class="btn" type="submit">Sign up</button>
                    <br>

                </div>
            </form>

            <script src="assets/js/popUpMessages.js"></script>

            <c:if test="${param.notallow!=null}">
                <script type="text/javascript">
                    popupMessages('Not allowed')
                </script>
```

```
		</c:if>

		<c:if test="${param.errormsg!=null}">
			<script type="text/javascript">
				popupMessages('This email id already registered')
			</script>
		</c:if>

		<script src="assets/js/loginAndRegister.js"></script>

</body>

</html>
```

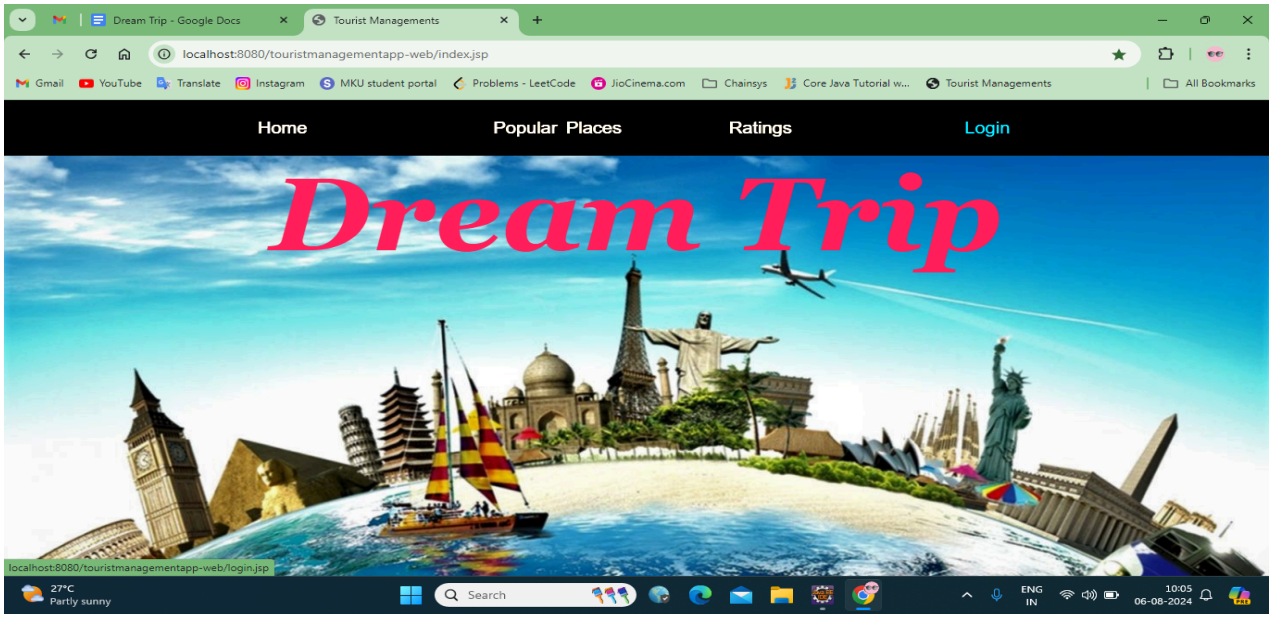
loginAndRegister.js

```
function showPassword() {
			var x = document.getElementById("psw");
			if (x.type === "password") {
				x.type = "text";
			} else {
				x.type = "password";
			}
		}

function checkpattern() {
			//console.log("function calling")
			var password = document.getElementById("psw").value;

			if (password.match(/(?=[A-Z])/)) {
				document.getElementById("upper").style.color = "rgb(31, 224,
31)";
			} else {
				document.getElementById("upper").style.color = "black";
			}
```

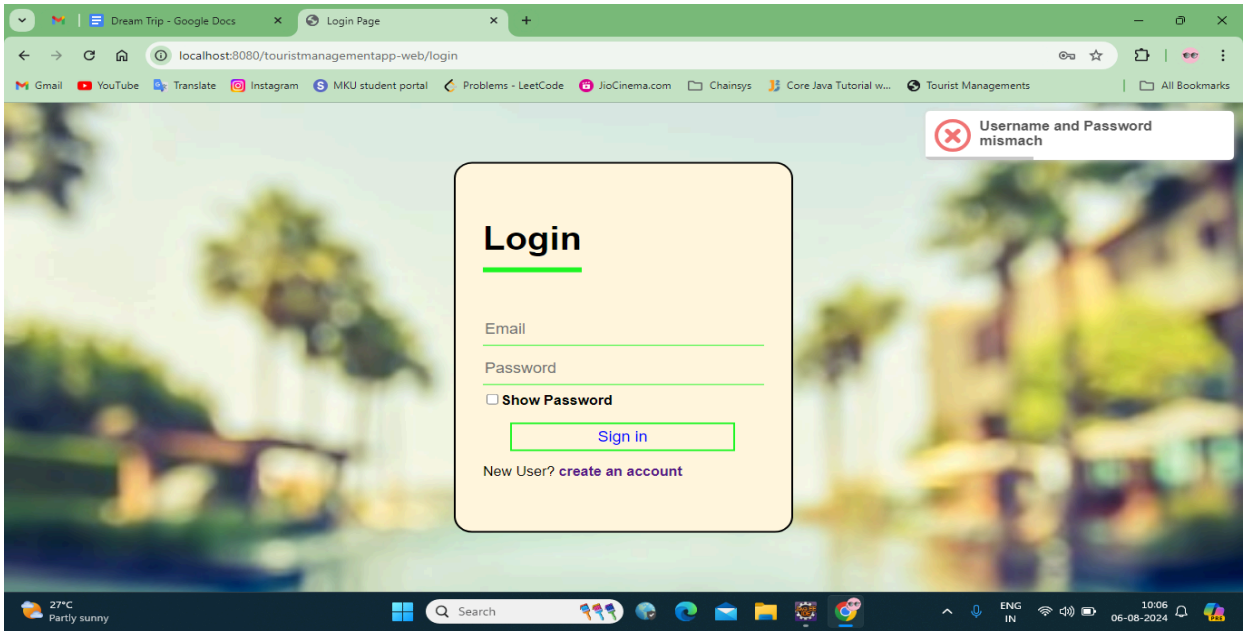
```
if (password.match(/(?=[a-z])/)) {
                document.getElementById("lower").style.color = "rgb(31, 224,
31)";
            } else {
                document.getElementById("lower").style.color = "black";
            }

            if (password.match(/(?=[0-9])/)) {
                document.getElementById("number").style.color = "rgb(31, 224,
31)";
            } else {
                document.getElementById("number").style.color = "black";
            }

            if (password.match(/(?=.*[!@#\$%\^&\*])/)) {
                document.getElementById("special").style.color = "rgb(31, 224,
31)";
            } else {
                document.getElementById("special").style.color = "black";
            }

            if (password.length > 7) {
                document.getElementById("char").style.color = "rgb(31, 224, 31)";
            } else {
                document.getElementById("char").style.color = "black";
            }

        }
```

SCREENSHOTS

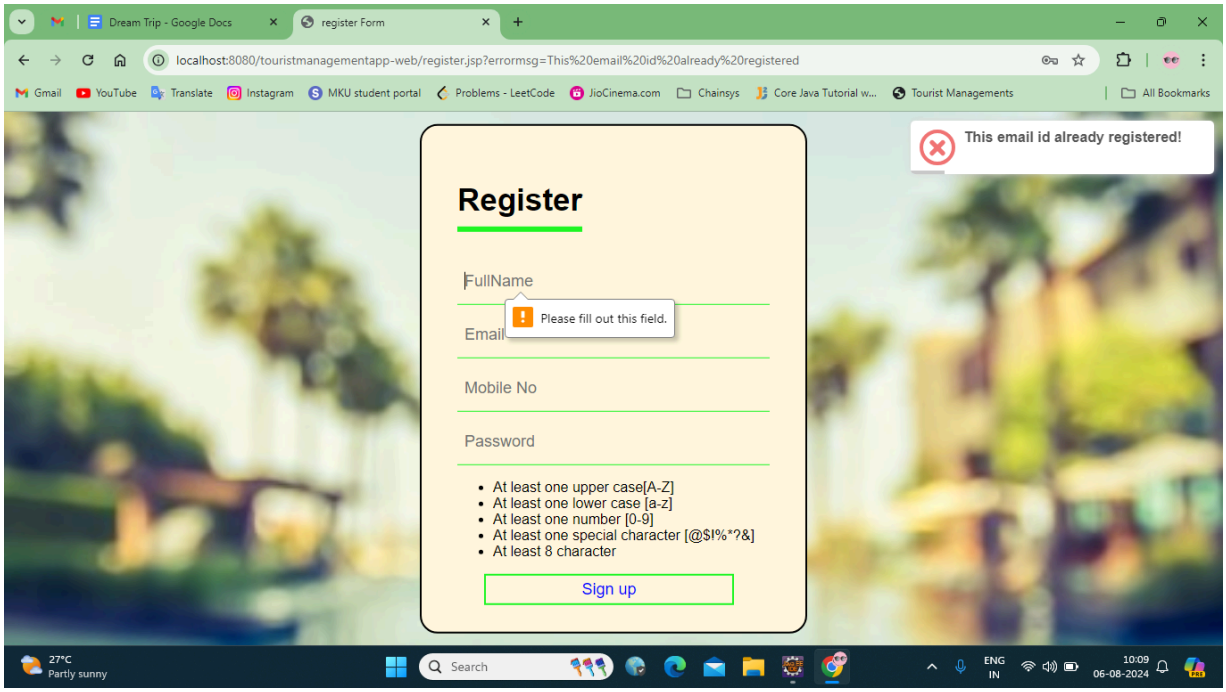
Home Page :



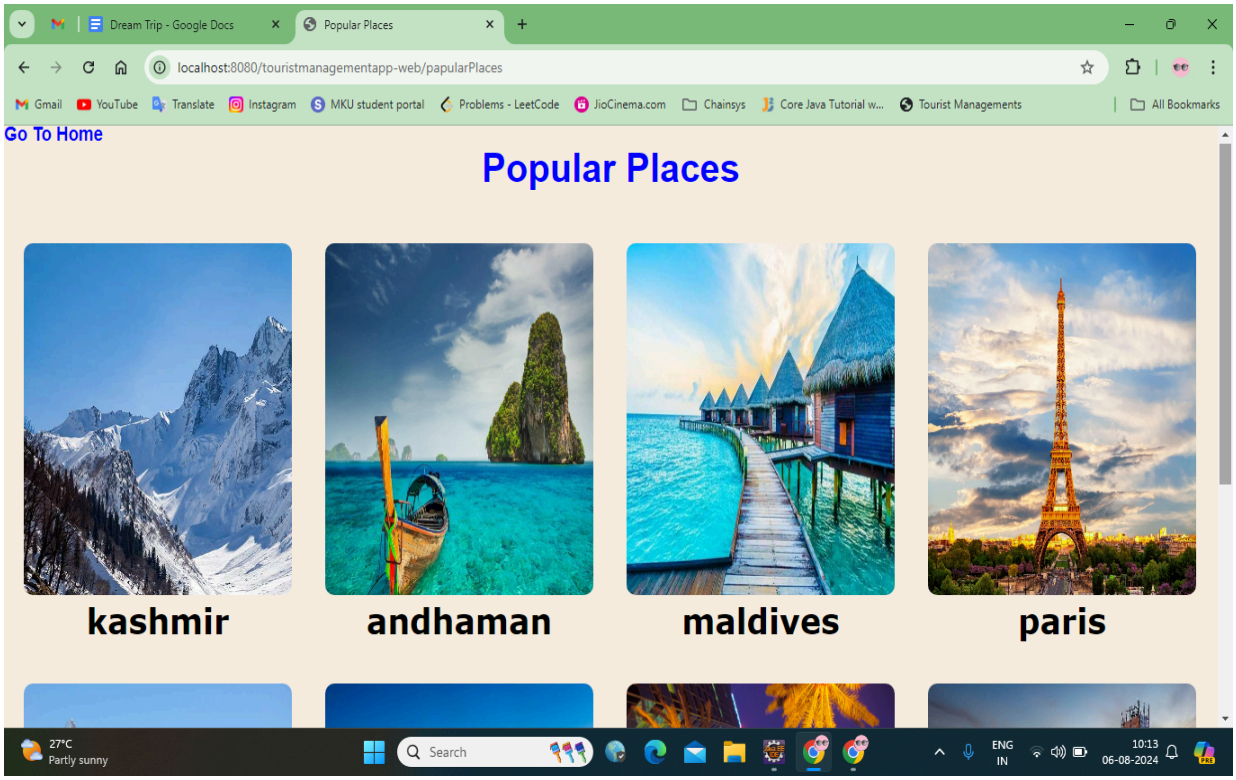
Login Page :



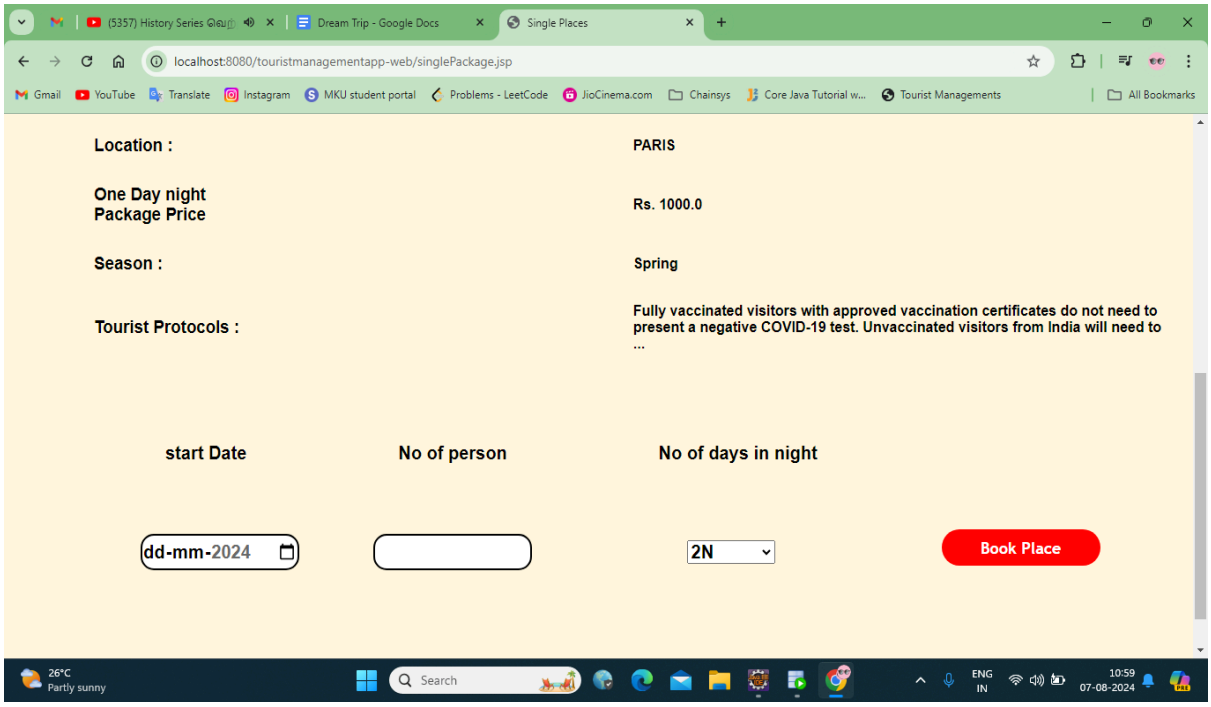
Register Page :



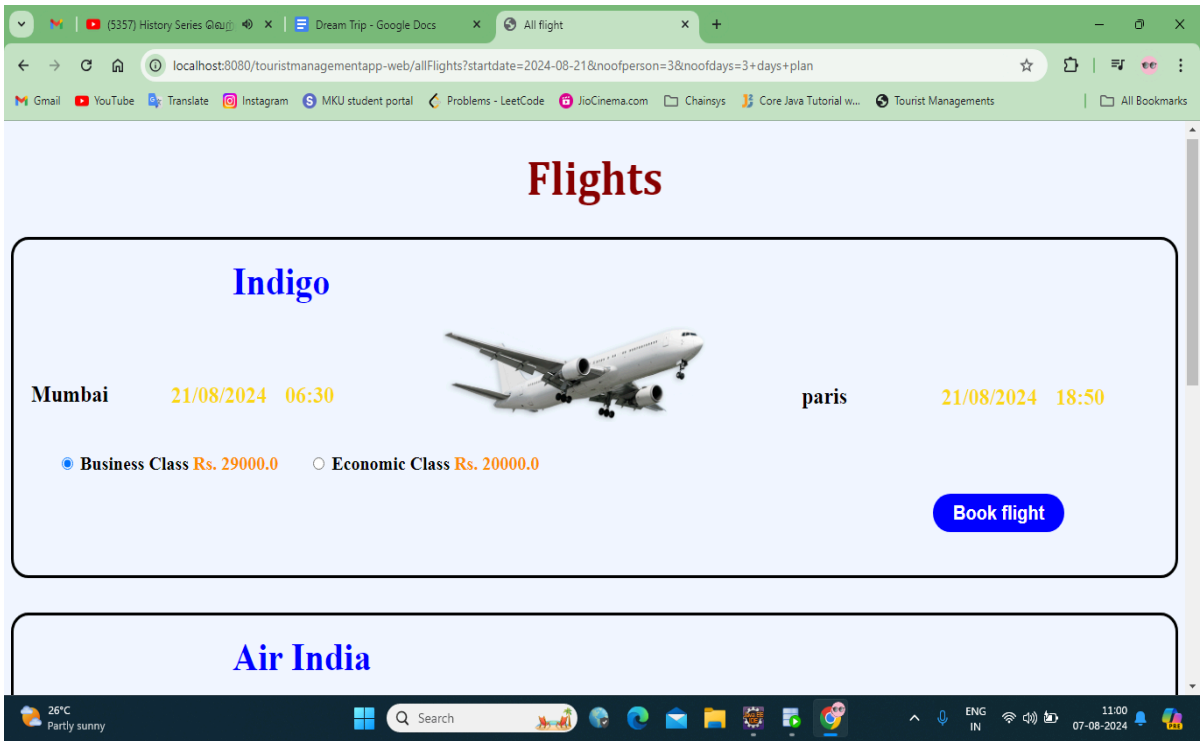
Popular Places :



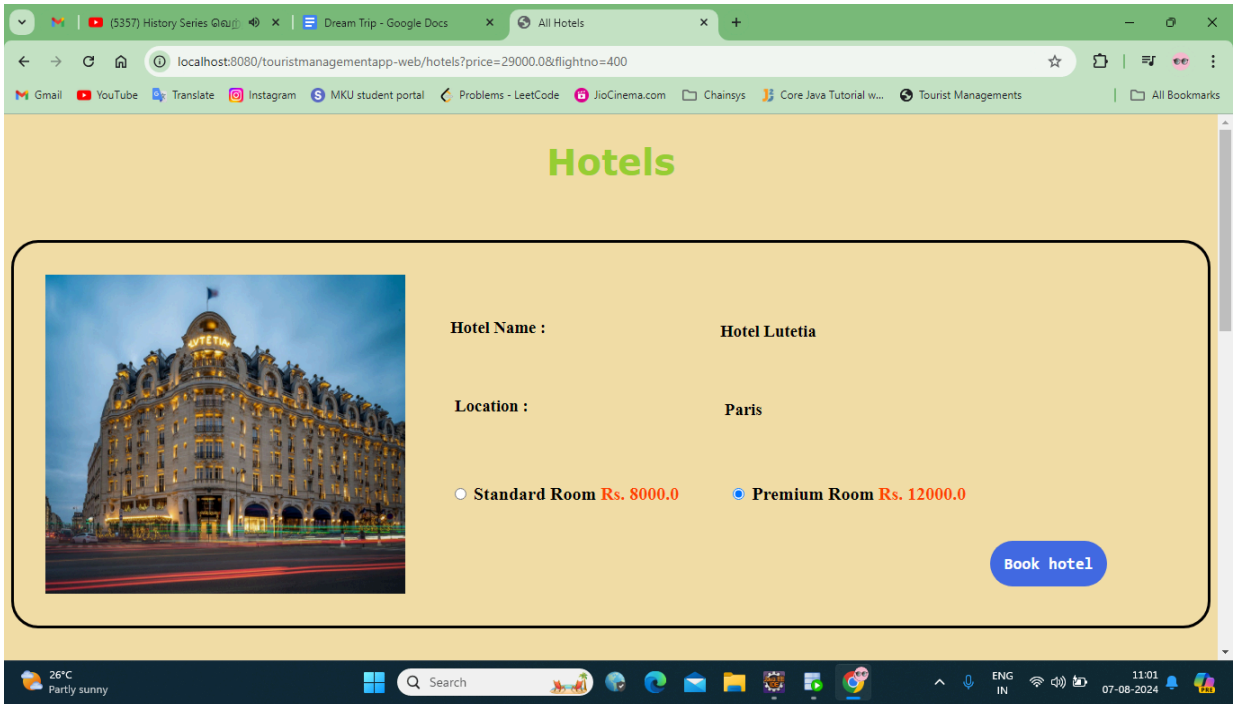
Book place page :



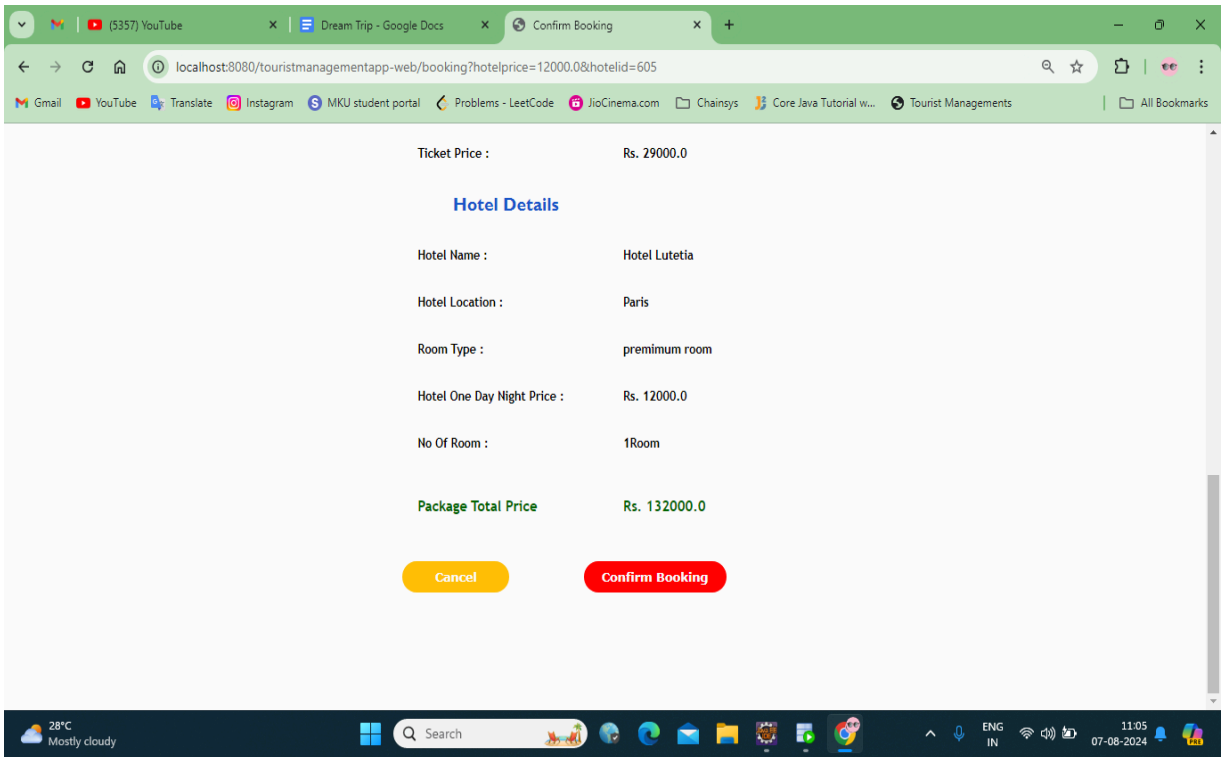
Flight Booking Page :



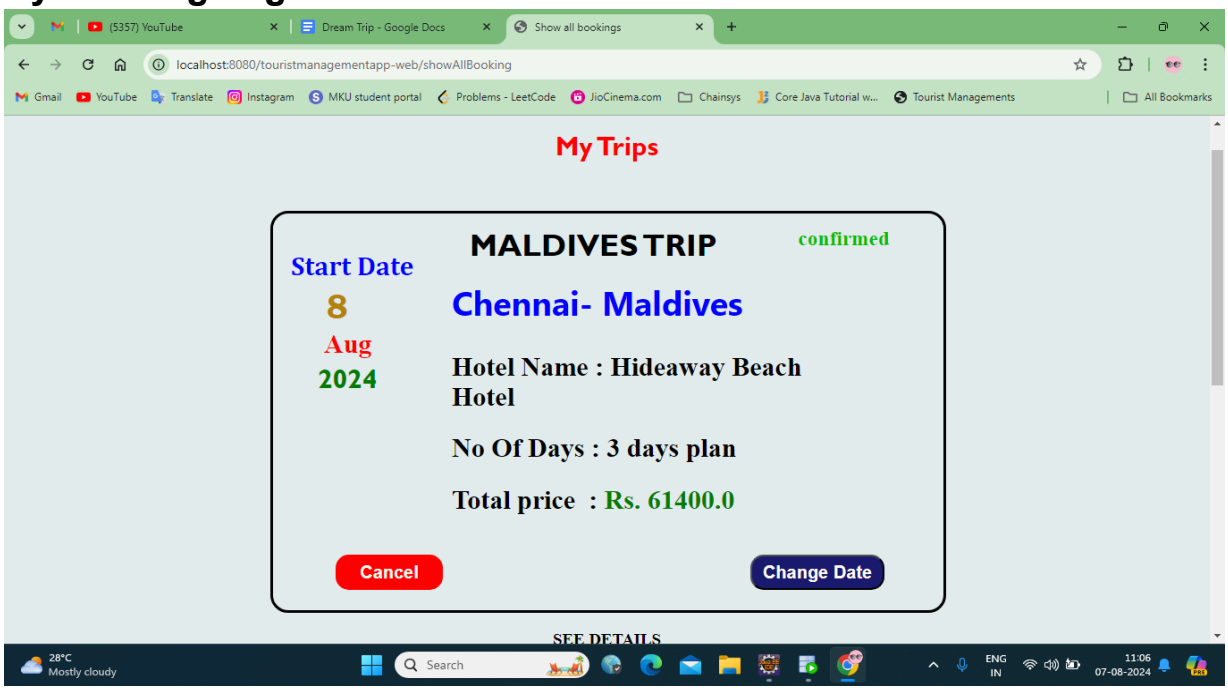
Hotel Booking Page :



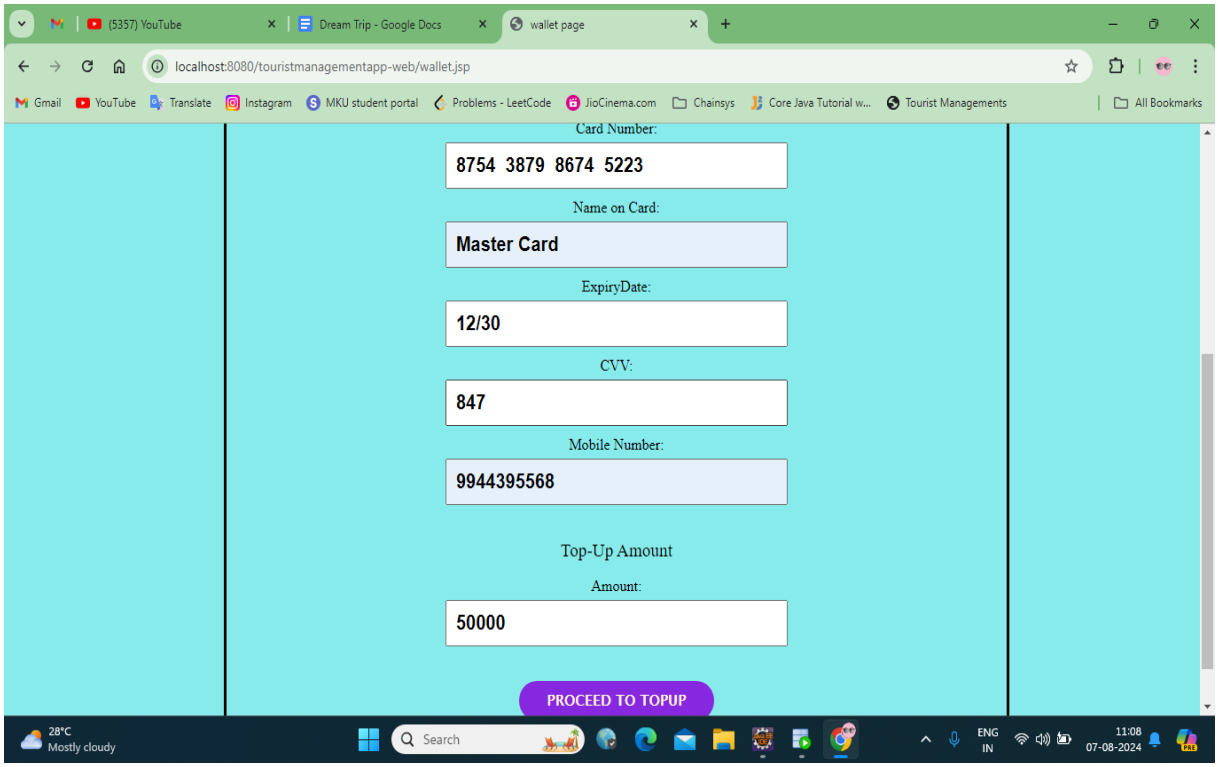
Booking Confirm Page :



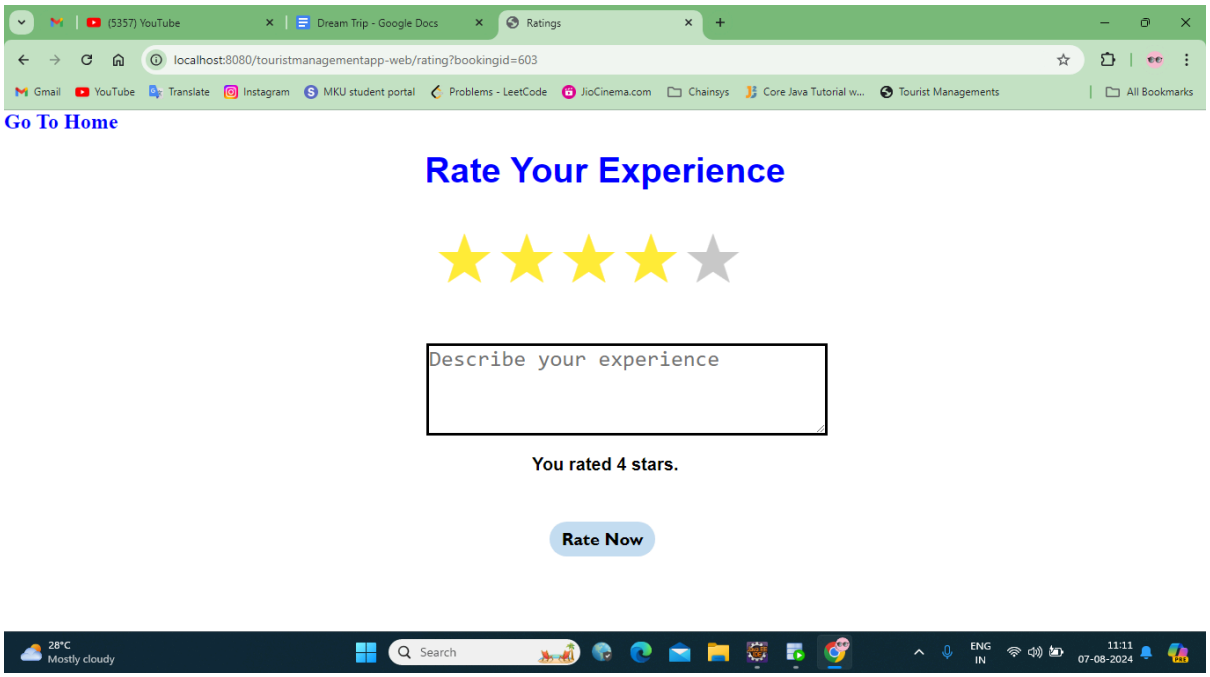
My Booking Page :



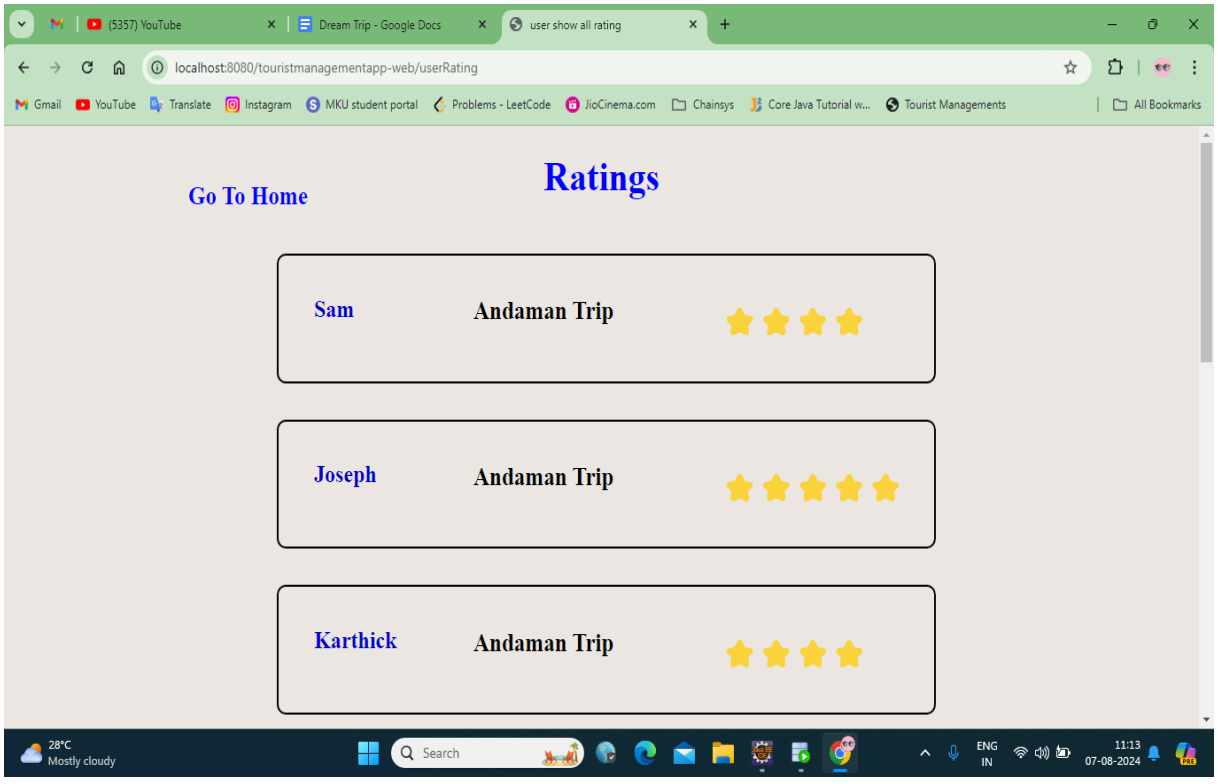
Wallet Top-up Page :



Rating Page :



Show All Rating Page :



BIBLIOGRAPHY

Books

Designing Data-Intensive Applications by Martin Kleppmann :

This book provides a comprehensive guide to designing robust, scalable, and maintainable data-intensive applications, covering key concepts that are essential for building a tourist management application.

Research Papers and Articles

"User Experience in Tourism Management Systems" by Lisa Brown :

An article that discusses the importance of user experience design in tourism management applications and provides guidelines for improving usability.

Online Resources

Struts 2 Documentation :

- The official documentation for the Struts 2 framework, including guides, tutorials, and examples.
- **URL:** Struts 2 Documentation

Stack Overflow :

- A community-driven Q&A site where you can find answers to specific technical questions related to your project's tech stack.
- **URL:** Stack Overflow

Websites and Blogs

Java Code Geeks :

- A website with articles and tutorials on Java and related technologies.
- **URL:** Java Code Geeks