Implementation of Optimum Travel Planner

A

Project Report

Submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Information Technology

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA,
BHOPAL (M.P.)

Guided By :
Dr. Prashant Lakkadwala

Submitted By :
Sarthak Kacholiya (0827IT221132)
Tanmay Indoriya (0827IT221143)
Vikas Dhakad (0827IT221150)
Rohit Sahu (0827IT221123)

DEPARTMENT OF INFORMATION TECHNOLOGY
ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH,
INDORE (M.P.) 452020

2024-2025

Declaration

I hereby declared that the work, which is being presented in the project entitled **"Implementation of Optimum Travel Planner"** partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology**, submitted in the department of Information Technology at **Acropolis Institute of Technology & Research, Indore** is an authentic record of my own work carried under the supervision of "**Dr. Prashant Lakkadwala**". I have not submitted the matter embodied in this report for the award of any other degree.

Sarthak Kacholiya (0827IT221132)
Tanmay Indoriya (0827IT221143)
Vikas Dhakad (0827IT221150)
Rohit Sahu (0827IT221123)

Dr. Prashant Lakkadwala
Supervisor

Project Approval Form

I hereby recommend that the project "**Implementation of Optimum Travel Planner"** prepared under my supervision by **Sarthak Kacholiya (0827IT221132), Tanmay Indoriya (0827IT221143), Vikas Dhakad (0827IT221150), Rohit Sahu (0827IT221123)** be accepted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Information Technology.

Dr. Prashant Lakkadwala

Supervisor

Recommendation concurred in 2024-2025

Prof. Monika Choudhary

Project Incharge

Prof. Deepak Singh Chouhan

Project Coordinator

Acropolis Institute of Technology & Research

Department of Information Technology

Certificate

The project work entitled "**Implementation of Optimum Travel Planner"** submitted by Sarthak Kacholiya (0827IT221132), Tanmay Indoriya (0827IT221143), Vikas Dhakad (0827IT221150), Rohit Sahu (0827IT221123) is approved as partial fulfillment for the award of the degree of Bachelor of Technology in Information Technology by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.).

Internal Examiner	**External Examiner**
Name:…….	Name: ……..
Date: …./…/……..	Date: …./…/……..

Acknowledgement

With boundless love and appreciation, we/I would like to extend our/my heartfelt gratitude and appreciation to the people who helped us/me to bring this work to reality. We/I would like to have some space of acknowledgement for them.

Foremost, our/I would like to express our/ my sincere gratitude to our/my supervisor, **Dr**.**Prashant Lakkadwala** whose expertise, consistent guidance, ample time spent and consistent advice that helped us/me to bring this study into success.

To the project in-charge **Prof. Monika Choudhary** and project coordinator **Prof. Deepak Singh Chouhan** for their constructive comments, suggestions, and critiquing even in hardship.

To the honorable **Prof. (Dr.) Prashant Lakkadwala**, Head, Department of Information Technology for his favorable responses regarding the study and providing necessary facilities.

To the honorable **Dr. S.C. Sharma**, Director, AITR, Indore for his unending support, advice and effort to make it possible.

Finally, I/we would like to pay my/our thanks to faculty members and staff of the Department of Computer Science & Engineering for their timely help and support.

We/I also like to pay thanks to our **parents** for their eternal love, support and prayers without them it is not possible.

Sarthak Kacholiya (0827IT221132)
Tanmay Indoriya (0827IT221143)
Vikas Dhakad(0827IT221150)
Rohit Sahu(0827IT221123)

Abstract

This project focused on creating a responsive and dynamic web navigation system for "*Implementation of Optimum Travel Planner"*, a platform aimed at enhancing the travel experience for modern explorers. The implementation revolved around building a fully functional navigation menu using React.js, equipped with React Router functionalities to handle seamless page transitions.

In an era where digital experiences shape user behaviour, having a responsive and intuitive web interface is critical. The purpose of this project was to ensure accessibility and usability across all devices, catering to the diverse needs of users, from desktop planners to mobile travellers. By prioritising responsiveness, the project aimed to provide a consistent experience while reflecting the adventurous and dynamic spirit of "*Implementation of Optimum Travel Planner".*

The project employed React.js for its component-based architecture, ensuring modularity and ease of development. React Router was integrated to facilitate navigation between different pages without reloading. A mobile-first design approach was adopted, ensuring optimal usability on smaller screens before scaling up for larger displays. Static assets were served through a well-structured public folder, while dependencies were managed using npm. The project utilised tools like ESLint to maintain code quality, ensuring a clean and scalable codebase. Development began with the Create React App framework, which provided a robust foundation for quick iteration.

The responsive navigation system proved effective, delivering a seamless user experience across various devices and screen sizes. Testing demonstrated that the interface was intuitive and visually appealing, contributing to user satisfaction. The use of React.js showcased its strengths in managing dynamic content and component states, validating its suitability for building scalable, high-performance web applications.

This project highlights the critical role of responsive design in digital platforms for travellers. By leveraging modern tools like React.js, the platform achieved a balance between functionality and aesthetics. These findings underscore the importance of adaptability in web design, offering valuable insights for future developments aimed at engaging and supporting the global traveler community.

Table of Content

Declaration ……... 2

Project Approval Form …...…… 3

Acknowledgement…… 5

Abstract …… 6

List of Figures…… 10

List of Tables……..…… 11

Abbreviations……..……. 12

Chapter 1: Introduction……..……... 13

1.1 Rationale ……. 14

1.2 Existing System …….. 15

1.3 Problem Formulation …….. 16

1.4 Proposed System …… 17

1.5 Objectives …….. 18

1.6 Contribution of the Project

1.6.1 Market Potential ……... 20

1.6.2 Innovativeness ……...…... 21

1.6.3 Usefulness ……. 22

1.7 Report Organisation…….. 23

Chapter 2: Requirement Engineering……..…….. 25

2.1 Feasibility Study (Technical, Economical, Operational)

2.1.1 Technical Feasibility…… 26

2.1.2 Economical Feasibility…… 27

2.1.3 Operational Feasibility…… 27

2.2 Requirement Collection

2.2.1 Discussion…… 28

2.2.2 Requirement Analysis…….. 30

2.3 Requirements

2.3.1 Functional Requirements

2.3.1.1 Statement of Functionality…….. 31

2.3.2 Nonfunctional Requirements

2.3.2.1 Statement of Functionality……... 32

2.4 Hardware & Software Requirements

2.4.1 Hardware Requirement (Developer & End User)…… 34

2.4.2 Software Requirement (Developer & End User)……. 35

2.5 Use-case Diagrams……. 38

2.5.1 Use-case Descriptions…….. 39

Chapter 3: Analysis & Conceptual Design & Technical Architecture

3.1 Technical Architecture……. 43

3.2 Sequence Diagrams……. 45

3.3 Class Diagrams…… 46

3.4 DFD……. 49

3.5 User Interface Design ……. 50

3.6 Data Design

3.6.1 Schema Definitions…… 51

3.6.2 E-R Diagram…….. 55

Chapter 4: Implementation & Testing…… 57

4.1 Methodology

4.1.1 Proposed Algorithm…… 58

4.2 Implementation Approach

4.2.1 Introduction to Languages, IDEs Tools and Technologies… 59

4.3 Testing Approaches

4.3.1 Unit Testing……. 59

a. Test Cases…… 60

4.3.2 Integration Testing…….. 62

b. Test Cases…… 63

Chapter 5: Results & Discussion……. 65

5.1 User Interface Representation…… 66

5.1.1 Brief Description of Various Modules…… 66

5.2 Snapshot of System with Brief Description…….. 67

5.3 Database Description

5.3.1 Snapshot of Database Tables with Brief Description…….. 71

5.4 Final Findings…… 79

6. Conclusion & Future Scope…….. 81

6.1 Conclusion…… 82

6.2 Future Scope……. 82

REFERENCES

Appendix A: Project Synopsis

Appendix B: Guide Interaction Report

Appendix C: User Manual

Appendix D: Git/GitHub Commits/Version History

List of Figures

Fig:-2.1 Use Case Diagram of Tourism and Travel System……. 38

Fig:-3.1 Technical Architecture Diagram of Implementation of Optimum Travel Planner.. 44

Fig:-3.2 Sequence Diagram of Implementation of Optimum Travel Planner System……. 46

Fig:-3.3 Class Diagram of Implementation of Optimum Travel Planner System……... 48

Fig:-3.4 Data Flow Diagram of Implementation of Optimum Travel Planner System…... 50

Fig:-3.5 User Interface Design of Implementation of Optimum Travel Planner……..…. 51

Fig:-3.6 E-R Diagram of Implementation of Optimum Travel Planner System…… 56

Fig:-5.1 Home Page of Implementation of Optimum Travel Planner…… 69

Fig:-5.2 Search and Explore of Implementation of Optimum Travel Planner……... 69

Fig:-5.3 Feedback Page of Implementation of Optimum Travel Planner…….. 70

Fig:-5.4 Browsing Page of Implementation of Optimum Travel Planner……. 70

Fig:-5.5 Booking Page of Implementation of Optimum Travel Planner…… 71

List of Tables

Table:-5.1 User Table of Implementation of Optimum Travel Planner System……. 72

Table:-5.2 Tourist Table of Implementation of Optimum Travel Planner System……. 73

Table:-5.3 TourDetails Table of Implementation of Optimum Travel Planner System.. 75

Table:-5.4 Booking Table of Implementation of Optimum Travel Planner System…… 77

Table:-5.5 Feedback Table of Implementation of Optimum Travel Planner System….. 79

Abbreviations

1. UI – User Interface
2. API – Application Programming Interface
3. DB – Database
4. JWT – JSON Web Token
5. CRUD – Create, Read, Update, Delete
6. SQL – Structured Query Language
7. HTML – HyperText Markup Language
8. CSS – Cascading Style Sheets
9. JS – JavaScript
10. ORM – Object Relational Mapping
11. URL – Uniform Resource Locator
12. CRUD Operations – Create, Read, Update, Delete

Chapter 1:

Introduction

1.1 Rationale

The travel and tourism industry is one of the fastest-growing sectors globally, with millions of travellers seeking convenient, user-friendly platforms to plan and book their trips. However, many existing systems are either overly complex or lack essential features, such as personalized recommendations, secure bookings, and responsive feedback mechanisms.

This project aims to address these challenges by developing a **Wanderlust Explorer** system that simplifies the travel planning process for users. By integrating a robust backend with a dynamic and intuitive frontend, the platform provides seamless user experiences for exploring tours, managing bookings, and submitting feedback.

The system's rationale includes:

- **User Convenience**: A centralised platform for discovering, booking, and reviewing tours.
- **Automation**: Streamlining manual processes like booking confirmation and tour management.
- **Scalability**: Supporting a growing number of users and tours with a scalable architecture.
- **Security**: Ensuring data protection through secure authentication and role-based access.
- **Enhanced User Experience**: Offering a responsive, feature-rich UI for accessibility on multiple devices.

This project not only benefits travellers by simplifying their journey planning but also empowers administrators to manage tours and bookings efficiently, making it a valuable solution in the travel domain.

1.2 Existing System

The existing systems in the travel and tourism domain often face several challenges:

- **Complex User Interfaces**:
  Many platforms have cluttered designs, making it difficult for users to search for tours or complete bookings efficiently.

- **Lack of Integration**:
  Separate systems handle different aspects like bookings, payments, and feedback, leading to fragmented user experiences.

- **Limited Personalisation**:
  Current systems rarely provide tailored recommendations or filters based on user preferences.

- **Security Gaps**:
  User data, such as payment information, is often inadequately protected, increasing the risk of breaches.

- **Manual Processes**:
  Tour operators still rely on manual processes for managing bookings and feedback, which is time-consuming and error-prone.

- **Accessibility Issues**:
  Many systems are not mobile-friendly or fail to adapt to varying screen sizes, limiting accessibility.

1.3 Problem Formulation

The travel and tourism industry is witnessing a surge in demand for digital platforms that simplify trip planning and booking processes. However, existing systems face the following challenges:

- **Fragmented Experience**:
  Users must navigate multiple platforms for searching tours, booking, and providing feedback, leading to inefficiency.
- **Limited Automation**:
  Manual processes for managing bookings and user feedback increase errors and processing time.
- **Poor User Accessibility**:
  Many platforms lack responsive designs, making them inconvenient to use on mobile devices or tablets.
- **Inadequate Security**:
  Insufficient measures for user authentication and data protection pose significant risks to sensitive information.
- **Lack of Personalisation**:
  Current systems fail to offer personalised tour suggestions or tailored filters based on user preferences and history.
- **Difficulty in Management**:
  Administrators struggle with managing a growing number of users, tours, and feedback due to limited tools and insights.

1.4 Proposed System

The proposed **Wanderlust Explorer** system aims to overcome the limitations of existing systems by providing a unified, user-friendly platform that simplifies travel planning and enhances user experience. This system integrates advanced features for both users and administrators while ensuring security, scalability, and efficiency.

Key Features of the Proposed System

- **User-Friendly Interface**:
  A responsive, intuitive frontend (React.js) that adapts to all devices, ensuring seamless navigation and accessibility.
- **Secure User Management**:
  Implements secure registration and login using JWT authentication for data protection and user privacy.
- **Comprehensive Tour Management**:
  Allows users to browse, filter, and sort tours by criteria like destination, price, and duration.
  Displays detailed tour information, including descriptions, itineraries, and availability.
- **Streamlined Booking Process**:
  Enables users to book tours efficiently with real-time availability checks.
  Provides booking confirmation and a summary for users.
- **Feedback and Review System**:
  Allows users to submit ratings and reviews for completed tours, fostering transparency and trust.
- **Admin Panel**:
  Offers administrators tools to manage tours, bookings, and user feedback

efficiently.

Provides analytics for monitoring trends and improving services.

- **Automation and Error Handling**:

  Automated workflows for bookings and feedback submission reduce manual intervention.

  Robust error handling for invalid inputs or system exceptions ensures reliability.

- **Scalability and Integration**:

  Built on Spring Boot and MySQL for scalability, with RESTful APIs ensuring smooth frontend-backend communication.

Advantages of the Proposed System

- **Enhanced User Experience**: Simplifies tour discovery, booking, and feedback submission.
- **Improved Security**: Protects sensitive user data with secure authentication mechanisms.
- **Operational Efficiency**: Streamlines administrative tasks with automated workflows.
- **Future-Ready**: Designed for scalability and potential integrations like payment gateways and advanced analytics.

1. 5 Objectives

- **Simplify Tour booking**:

  Provide users with an easy-to-navigate platform to explore, filter, and book digital product seamlessly.

- **Enhance User Experience**:

  Develop a responsive and intuitive interface for both desktop and mobile devices.

- **Secure User Data**:

  Implement robust authentication mechanisms using JWT to ensure the confidentiality and integrity of user data.

- **Streamline Administration**:

  Offer tools for administrators to manage digital product, bookings, and user feedback efficiently.

- **Enable Feedback Mechanisms**:

  Allow users to rate and review digital product, fostering transparency and service improvement.

- **Automate Processes**:

  Reduce manual intervention by automating workflows such as booking confirmations and availability updates.

- **Support Scalability**:

  Build the system on scalable technologies like Spring Boot and MySQL to handle growing user bases and data volumes.

- **Promote Accessibility**:

  Design the platform to cater to diverse user needs, ensuring accessibility across all devices and platforms.

- **Lay the Foundation for Future Enhancements**:

  Create a modular and extensible system to integrate additional features like payment gateways, analytics, and personalised recommendations in the future.

1.6. Contribution to the Project

1.6.1 Market Potential

The **Implementation of Optimum Travel Planner** project introduces a **Tour and Travel System** that holds significant market potential in the growing travel and tourism industry. By addressing the need for digital solutions to simplify trip planning, its contributions include:

- **Simplifying Travel Planning**:
  Provides a single platform where users can explore, book, and review tours, eliminating the need for multiple tools or services.

- **Adapting to Digital Trends**:
  Aligns with the growing consumer shift toward online travel tools and mobile-friendly platforms, catering to a tech-savvy audience.

- **Contributing to Industry Growth**:
  Targets the expanding global travel market, providing a scalable solution to meet diverse traveler and business needs.

- **Building Customer Loyalty**:
  Features such as feedback mechanisms and a user-friendly interface foster improved satisfaction and repeat usage.

- **Creating B2B Opportunities**:
  Offers a robust platform for tour operators and agencies to manage their offerings efficiently and engage with customers effectively.

- **Differentiation in the Market**:
  Stands out with its focus on automation, security, and user-centred design, providing a unique value proposition.

1.6.2 Innovativeness

The **Tour and Travel System** developed under the **Implementation of Optimum Travel Planner** project integrates modern features and methodologies, offering innovative solutions:

- **Unified User and Admin Features**:
  Combines user functionalities (like booking and reviews) with advanced admin tools (like tour management and analytics) on a single platform.

- **Responsive and Modern Design**:
  Uses a React.js frontend to provide a seamless user experience across desktops, tablets, and mobile devices.

- **Secure Booking and Access**:
  Implements JWT-based authentication to ensure secure user data handling and privacy during login, booking, and feedback processes.

- **Detailed Search Filters**:
  Offers users the ability to search and filter tours by criteria such as price, destination, and duration, making travel planning easier.

- **Streamlined Operations**:
  Automates key processes such as booking confirmations, availability updates, and feedback storage, reducing administrative workload and errors.

- **Built for Growth**:
  Uses Spring Boot and MySQL for a backend architecture designed to scale with increased user demands and system enhancements.

- **Future-Ready Design**:
  Incorporates extensibility to support additional features like payment gateways, AI-powered recommendations, and detailed analytics in the future.

This blend of usability, security, and scalability makes the **Tour and Travel System** a pioneering solution in the travel sector.

1.6.3 Usefulness

The **Tour and Travel System** developed for the **Implementation of Optimum Travel Planner** project offers practical benefits for both users and administrators:

- **Convenience for Travelers**:
  - Simplifies the travel planning process by combining discovery, booking, and feedback into a single platform.
  - Provides a user-friendly design that works seamlessly across all devices.
- **Efficiency for Administrators**:
  - Reduces manual intervention with automated workflows for managing bookings, availability, and user feedback.
  - Offers centralized tools to streamline the management of tours and customer interactions.
- **Improved Service Quality**:
  - Empowers administrators to improve their offerings based on user feedback and analytics.
- **Robust Data Security**:
  - Ensures secure authentication and encryption, protecting sensitive user information and maintaining privacy.
- **Scalability and Flexibility**:
  - A robust architecture ensures the system can accommodate a growing user base and additional functionalities over time.

- **Foundation for Future Integration**:
  - Designed to support future upgrades, such as integrated payment systems and data-driven recommendations, without major overhauls.

Overall Usefulness:

The **Tour and Travel System** provides an enhanced user experience for travelers while simplifying operations for administrators, addressing key challenges in the travel and tourism industry and setting a strong foundation for future growth.

1.7 Report Organization.

Chapter 1: Provides an overview of the project by discussing existing systems, highlighting the problem statement, and presenting the proposed solution to address the shortcomings of traditional systems.

Chapter 2: Covers the literature survey, including background details of the system, the software engineering paradigm, and the technologies (software and hardware) used in building the system.

Chapter 3: Discusses the analysis of the system, focusing on requirement identification and feasibility studies, including technical, financial, and operational feasibility.

Chapter 4: Details the design of the system, including UML diagrams, ER diagrams, data flow diagrams, and a comprehensive data dictionary.

Chapter 5: Explains the implementation phase, including the code structure, integration process, and the adaptability of the system.

Chapter 6: Describes the testing phase, covering different testing methods, strategies, and test case execution.

Chapter 7: Concludes the report by summarizing the project outcomes and discussing potential advancements and future work.

References: Lists the books, websites, journals, and blogs referenced during the development of the system.

Chapter 2:

Requirement Engineering

2.1 Feasibility study(Technical, Economical,Operational)

A **Feasibility Study** is a critical phase in the project development lifecycle, ensuring that the project can be realistically executed within constraints. The study evaluates the project's feasibility from three main perspectives: **Technical**, **Economical**, and **Operational**.

2.1.1 Technical Feasibility

- **Objective**: Assess whether the current technology stack and resources can support the development and implementation of the system.
- **Considerations**:
  - **Technology Stack**: The project utilizes **Spring Boot** for the backend and **React.js** for the frontend. Both technologies are widely used, well-supported, and have robust ecosystems, ensuring ease of development and maintenance.
  - **Database**: **MySQL** is used for storing user, booking, and tour data, providing reliability and scalability.
  - **Security**: JWT authentication ensures secure user login and access control, meeting industry-standard security practices.
  - **Integration**: The system integrates seamlessly with the frontend and backend through RESTful APIs, ensuring smooth communication between components.
  - **Scalability**: The use of scalable technologies such as **Spring Boot** and **MySQL** allows the system to grow in terms of user load and data volume.
- **Conclusion**: The project is technically feasible with the chosen stack and technologies, as they provide the required functionality and scalability.

2.1.2 Economical Feasibility

Objective: Evaluate the financial cost and budget needed to develop and maintain the system, and whether the benefits outweigh the costs.

Considerations:

- **Development Costs**: The primary costs include developer salaries, infrastructure for hosting, and software licenses. The use of open-source technologies like **Spring Boot**, **React.js**, and **MySQL** minimizes software costs.
- **Operational Costs**: Ongoing costs include server hosting (cloud-based), data storage, and security measures. Cloud services like AWS or Azure can provide scalable hosting at affordable rates.
- **Return on Investment (ROI)**: The system can generate revenue through tour bookings, user subscriptions, or partnerships with travel agencies. Additionally, providing an intuitive and efficient platform can attract more customers and increase revenue.
- **Maintenance Costs**: After deployment, the costs for bug fixes, feature updates, and general system maintenance will incur, but they can be kept low due to the system's modular architecture.

Conclusion: The project is economically feasible, as the use of open-source technologies, cloud hosting, and a revenue-generating model ensures cost-effectiveness.

2.1.3. Operational Feasibility

- **Objective**: Determine if the system can be efficiently implemented and operated within the organizational constraints, and if it will be accepted by users.
- **Considerations**:
    - **User Acceptance**: The platform is designed with an intuitive, user-friendly interface that improves the overall user experience, which is likely to encourage adoption.

- **System Requirements**: The system can run on standard hardware and does not require specialized or expensive infrastructure. Cloud hosting allows easy scalability as user demand increases.
- **Training and Support**: Minimal training is required for users, as the system is designed to be self-explanatory. However, administrators may need basic training on managing digital product, bookings, and users.
- **Maintenance and Updates**: The modular design of the system allows easy updates and feature additions, reducing long-term operational costs.
- **Stakeholder Involvement**: Regular feedback from end-users and administrators will be important to ensure that the system meets operational needs and continues to improve.

➢ **Conclusion**: The system is operationally feasible, as it can be efficiently implemented, maintained, and operated with the existing infrastructure and minimal user training.

Overall Feasibility

Based on the technical, economical, and operational considerations, the **digital product and Travels System** is feasible. It leverages modern technologies, is cost-effective, and aligns with the operational needs of both users and administrators. The project is set to deliver long-term value to stakeholders while maintaining scalability and efficiency.

2.2 Requirement collection

2.2.1 Discussion

Requirement collection is essential to ensure that **Implementation of Optimum Travel Planner** meets the needs of users and administrators. This process involves gathering and defining both functional and non-functional requirements to guide the system's development.

➢ **Stakeholder Identification**

- **Users**: Tourists and customers looking for an easy and seamless way to book tours and travel packages.

- **Administrators**: Travel agencies and tour operators responsible for managing bookings, tours, and travel-related services.
- **Third-Party Integrators**: External systems such as payment gateways, mapping services, and other APIs that enhance the functionality of the system.

➢ **Methods of Requirement Collection**

- **Interviews**: Conduct one-on-one discussions with users and administrators to gain insights into their needs, pain points, and expectations.
- **Surveys**: Use surveys to collect quantitative and qualitative data from a broader audience of potential users, focusing on their preferences and priorities.
- **Workshops**: Organize collaborative sessions with stakeholders to brainstorm, discuss, and prioritize system features.
- **Document Analysis**: Review existing systems, competitor platforms, and related documentation to identify potential features and best practices.

➢ **Types of Requirements**

- **Functional Requirements**: Specify what the system must do, including:
  i. Tour and travel package search.
  ii. Booking management for tours and activities.
  iii. User registration, login, and profile management.
  iv. Feedback and review submission by users.
  v. Administrative dashboards for managing bookings and tours.
- **Non-Functional Requirements**: Define the system\u2019s performance and quality attributes, such as:
  i. **Performance**: Fast response times, especially during peak usage.
  ii. **Scalability**: Ability to handle an increasing number of users and bookings.
  iii. **Security**: Protection of user data and secure payment processing.
  iv. **Usability**: Intuitive interface for users and administrators.

2.2.2 Analyzing Requirements

- **Prioritization**: Utilize methods such as the MoSCoW framework (Must-have, Should-have, Could-have, Won't-have) to focus on critical and high-value features first.
- **Feasibility Check**: Assess the technical and financial viability of each requirement to ensure it aligns with project resources and constraints.
- **Validation**: Conduct stakeholder reviews and create prototypes to refine and confirm requirements before proceeding to development.
    - **Challenges**
        i. **Ambiguity**: Stakeholders may have unclear, incomplete, or conflicting requirements.
        ii. **Scope Creep**: New or evolving requirements during development can lead to delays and increased costs.
        iii. **Stakeholder Alignment**: Balancing differing priorities and expectations among diverse stakeholders.
    - **Tools for Requirement Collection**
        i. **Jira or Trello**: To document, manage, and track requirements and tasks.
        ii. **Figma**: For prototyping and visualizing user interface designs.
        iii. **Google Forms**: To collect and analyze survey responses and user feedback.

Conclusion

Effective requirement collection is the foundation of a successful development process for *Implementation of Optimum Travel Planner*. By employing diverse methods such as interviews, surveys, workshops, and document analysis, the team can ensure the system aligns with user and administrator needs. Proper prioritization, validation, and use of appropriate tools will streamline development and deliver a platform that exceeds stakeholder expectations.

2.3 Requirements

2.3.1 Functional requirements

2.3.1.1 Statement of functionality

The **functional requirements** define the core features and operations that **Implementation of Optimum Travel Planner** must perform to meet the needs of its users and administrators. Below is a statement of functionality for the system:

- **User Registration and Authentication**:
  - The system must allow users to create an account using email and password.
  - It must provide a secure login system with JWT authentication for user verification.
  - Users should be able to reset their passwords via email verification.
- **Tour Discovery and Search**:
  - Users must be able to search for tours based on various criteria such as destination, price range, and date.
  - The system should allow users to filter search results based on tour type, duration, and rating.
- **Tour Booking and Confirmation**:
  - Users must be able to select and book a tour with available dates and times.
  - The system must check real-time availability and display available slots to users.
  - Once a booking is confirmed, the system should generate a booking summary and send a confirmation email to the user.
- **User Profile Management**:
  - Users should be able to view and edit their profiles, including personal details and booking history.
  - The system must allow users to cancel or modify bookings if applicable.

- **Feedback and Review System**:
  - After completing a tour, users should be able to rate the tour and provide written feedback.
  - The system must display user reviews and ratings on the tour's page for others to view.
- **Admin Panel**:
  - Administrators must have the ability to manage tours, add new tours, and update existing tour details.
  - The system must allow administrators to view and manage user bookings, including cancellations and modifications.
  - Administrators should be able to monitor feedback and respond to user inquiries.
- **Notifications**:
  - The system must notify users of booking confirmations, cancellations, and updates via email.
  - Administrators should receive notifications for new bookings, cancellations, and feedback submissions.

2.3.2 Nonfunctional requirements

2.3.2.1 Statement of Functionality

- The **non-functional requirements** for **Implementation of Optimum Travel Planner** define the system's operational attributes and performance characteristics. These are crucial for ensuring the platform is reliable, secure, and scalable.
- **Performance**:
  - The system should support at least 1,000 concurrent users without any significant performance degradation.
  - Response time for search queries and booking actions should be under 2 seconds.

- **Scalability**:
  - The platform must be able to handle a growing number of users, tours, and data over time without requiring major re-architecture.
  - The system should be easily expandable to accommodate future features like payment gateways, additional tours, and user analytics.
- **Security**:
  - All user data, including personal and payment information, must be securely stored and transmitted using encryption protocols like HTTPS and JWT for authentication.
  - User passwords should be securely hashed and stored in the database.
- **Availability**:
  - The system should be available 99.9% of the time, ensuring minimal downtime and smooth user access.
  - The platform should have failover mechanisms in place to ensure high availability.
- **Usability**:
  - The system must provide an intuitive, user-friendly interface, with easy navigation for both end users and administrators.
  - The interface should be responsive and optimized for both desktop and mobile devices.
- **Maintainability**:
  - The system must be modular and easily maintainable to ensure quick fixes and future updates.
  - Code should follow best practices and be well-documented for future developers.
- **Compliance**:
  - The platform must adhere to relevant legal and regulatory standards for data protection and privacy, such as GDPR or equivalent laws in the operating region.

2.4 Hardware and Software requirements

2.4.1 Hardware requirements(Developer & End user)

2.4.1.1. Developer Hardware Requirements

The hardware specifications for developers are designed to ensure smooth development, testing, and deployment of the **Implementation of Optimum Travel Planner** platform.

- **Processor**:
  - Minimum: Intel i5 or equivalent
  - Recommended: Intel i7 or equivalent for faster build and execution speeds.
- **RAM**:
  - Minimum: 8 GB
  - Recommended: 16 GB for optimal performance during development and testing.
- **Storage**:
  - Minimum: 256 GB SSD
  - Recommended: 512 GB SSD or higher for faster data access and storage.
- **Operating System**:
  - Windows 10 or later / macOS / Linux
- **Development Tools**:
  - **IDE**: IntelliJ IDEA (for backend), Visual Studio Code (for frontend)
  - **Database**: Local MySQL instance for testing
  - **Version Control**: Git, GitHub/GitLab for source code management
  - **Web Browser**: Google Chrome, Mozilla Firefox, or any modern browser for frontend testing
- **Network**:
  - Stable internet connection for downloading dependencies, accessing cloud services, and version control.

2.4.1.2. End-User Hardware Requirements

The **Implementation of Optimum Travel Planner** platform should be accessible on a wide range of devices with the following minimum specifications to ensure a smooth experience:

- **Desktop/Laptop**:
  - **Processor**: Intel i3 or equivalent
  - **RAM**: Minimum 4 GB
  - **Storage**: 100 GB free space
  - **Operating System**: Windows 10 or later / macOS / Linux
  - **Web Browser**: Latest version of Google Chrome, Mozilla Firefox, Safari, or Edge for optimal performance
- **Mobile/Tablet**:
  - **Operating System**:
    i. Android 5.0 (Lollipop) or later
    ii. iOS 10.0 or later
  - **Screen Resolution**: At least 720p (HD) for a clear and responsive interface
- **Internet Connection**:
  - A stable internet connection with at least 1 Mbps speed for optimal performance during browsing, booking, and payment activities.

2.4.2 Software requirements(developer & end user)

2.4.2.1. Developer Requirements

These are the tools and technologies needed to develop and maintain **Implementation of Optimum Travel Planner**.

- **Backend**:
  - **Spring Boot** (Version 2.x or higher): A Java-based framework for building the backend REST APIs.

- **Java** (Version 8 or higher): The programming language used for server-side logic.
- **Maven/Gradle**: For dependency management and project building.
- **MySQL** (Version 5.x or higher): A relational database management system for storing user, booking, and tour data.
- **JJWT (JSON Web Token)**: For secure user authentication and session management.

➢ **Frontend**:

- **React.js** (Version 17.x or higher): A JavaScript library for building the user interface of the platform.
- **Node.js** (Version 14.x or higher): For running the React.js development server.
- **npm** (Node Package Manager): For managing frontend dependencies.
- **Bootstrap/Material UI**: For designing responsive and attractive UI components.

➢ **Testing**:

- **JUnit** (for Java): For backend unit testing.
- **Mockito**: For mocking dependencies in unit tests.
- **Jest** (for React.js): For frontend unit testing.

➢ **Version Control**:

- **Git**: For source code version control.
- **GitHub/GitLab**: For hosting the code repositories.

➢ **IDE**:

- **IntelliJ IDEA**: Recommended IDE for Java development.
- **VSCode**: Recommended IDE for frontend development using React.js.

2.4.2.2. End User Requirements

These are the software requirements for users to access and interact with the **Implementation of Optimum Travel Planner** platform.

1. **Web Browser**:
   The platform is web-based and should be compatible with modern web browsers, including:
   a. **Google Chrome** (Latest Version)
   b. **Mozilla Firefox** (Latest Version)
   c. **Safari** (Latest Version)
   d. **Microsoft Edge** (Latest Version)
2. **Operating System**: The system should be accessible from various operating systems:
   a. **Windows** (Version 10 or higher)
   b. **macOS** (Latest Version)
   c. **Linux** (Latest Distribution)
3. **Internet Connection**:
   A stable internet connection is required to access and interact with the platform, as all functionalities, including searching tours, booking, and payments, are cloud-based.
4. **Mobile Device (Optional)**:
   While the system is fully responsive and accessible on mobile devices, users should have a smartphone or tablet with a modern browser to access the platform on the go.
   a. **Android** (Version 6.0 or higher)
   b. **iOS** (Version 12 or higher)

2.5 Use-case Diagram

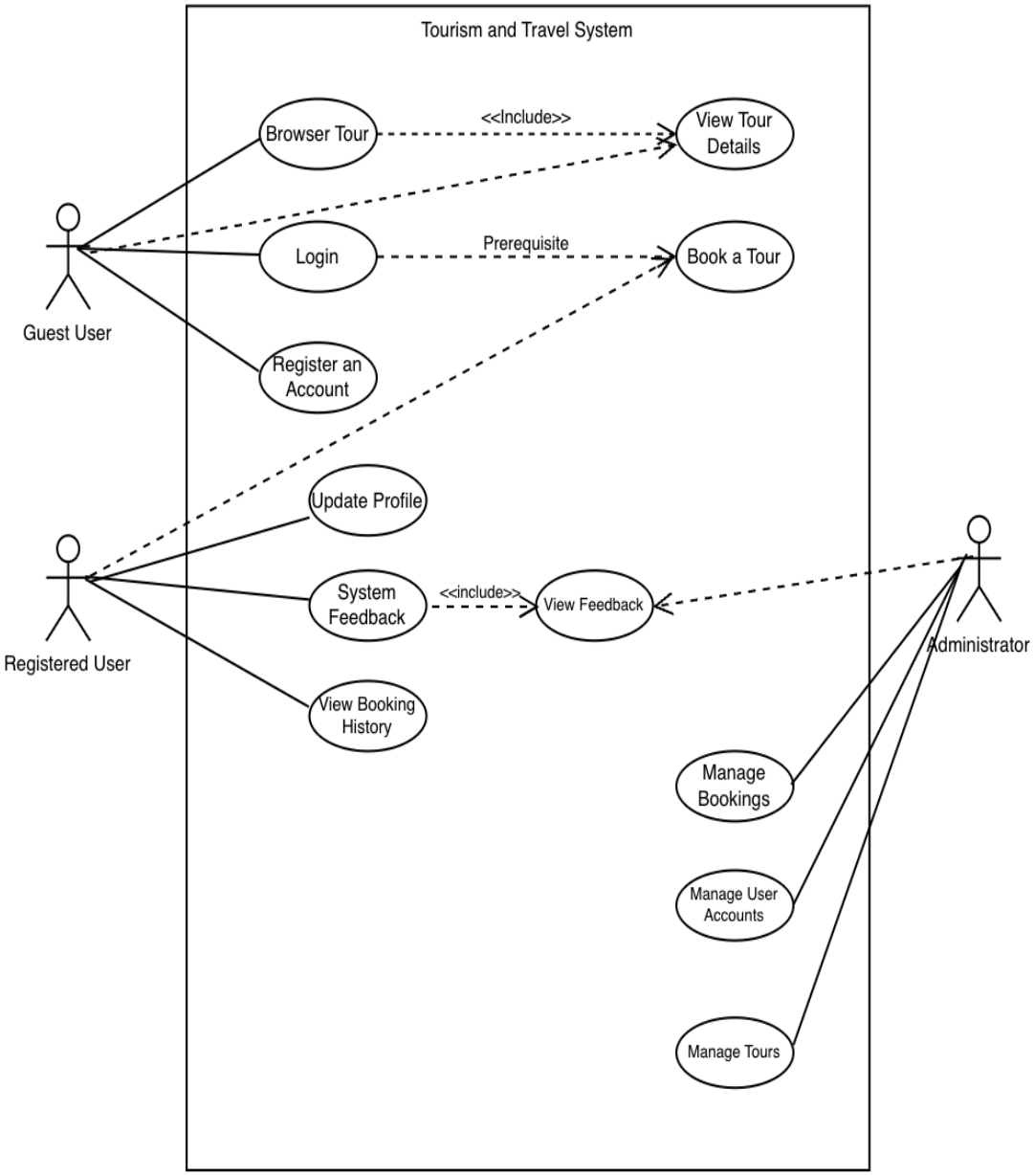

Fig:-2.1 Use Case Diagram of Tourism and Travel System

2.5.1 Use-Case Descriptions

- **Actors:**
  - **Guest User**: A user who has not registered or logged in.
  - **Registered User**: A user who has created an account and can log in to access additional features.
  - **Administrator**: A user with higher privileges who manages digital product, bookings, and user data.

Use Cases for Guest User:

- **Browse digital product**:
  - **Description**: The guest can search and view available digital product based on criteria like destination, price, and date.
  - **Preconditions**: None
  - **Postconditions**: A list of digital product is displayed based on the search criteria.
- **View Tour Details**:
  - **Description**: The guest can click on any tour to see detailed information (e.g., itinerary, price, description).
  - **Preconditions**: The user has selected a tour from the list.
  - **Postconditions**: Detailed tour information is displayed.
- **Register an Account**:
  - **Description**: The guest can create a new account by providing necessary details like name, email, and password.
  - **Preconditions**: None
  - **Postconditions**: User account is created, and the user is logged in.

- **Log in**:
  - **Description**: A registered user can log in to the system by entering valid credentials (username and password).
  - **Preconditions**: The user has registered an account.
  - **Postconditions**: User is authenticated and logged into the system.

Use Cases for Registered User:

- **Book a Tour**:
  - **Description**: The user can book a tour, select the number of people, and complete the booking process.
  - **Preconditions**: User is logged in and has selected a tour.
  - **Postconditions**: booking is successfully created, and the user receives a confirmation.
- **Submit Feedback**:
  - **Description**: After completing a tour, the user can rate and review the tour.
  - **Preconditions**: User has completed the tour.
  - **Postconditions**: Feedback is saved and visible to other users.
- **View booking History**:
  - **Description**: The user can view their past bookings and associated details.
  - **Preconditions**: User is logged in and has made at least one booking.
  - **Postconditions**: A list of previous bookings is displayed to the user.
- **Update Profile**:
  - **Description**: The user can update their personal information such as name, email, or password.
  - **Preconditions**: User is logged in.
  - **Postconditions**: Profile details are updated.

Use Cases for Administrator:

- **Manage digital product**:
  - **Description**: The administrator can add, edit, or delete digital product in the system.
  - **Preconditions**: The administrator is logged in.
  - **Postconditions**: digital product are added, updated, or deleted from the system.
- **Manage bookings**:
  - **Description**: The administrator can view, update, or cancel user bookings.
  - **Preconditions**: The administrator is logged in.
  - **Postconditions**: booking details are updated or canceled.
- **Manage User Accounts**:
  - **Description**: The administrator can manage user accounts, including viewing details or deleting accounts.
  - **Preconditions**: The administrator is logged in.
  - **Postconditions**: User account details are updated or deleted.
- **View Feedback**:
  - **Description**: The administrator can view feedback and ratings submitted by users for digital product.
  - **Preconditions**: The administrator is logged in.
  - **Postconditions**: Feedback is displayed for review.
- **Generate Reports**:
  - **Description**: The administrator can generate reports on bookings, user activity, and digital product.
  - **Preconditions**: The administrator is logged in.
  - **Postconditions**: Reports are generated based on selected parameters.

Chapter 3:

Analysis & Conceptual Design

& Technical Architecture

3.1 Technical Architecture

The system is built on a **three-tier architecture**, ensuring scalability, modularity, and secure data handling.

- **Presentation Layer**:
  - **Components**: Web and mobile interfaces designed for users (customers and administrators).
  - **Responsibilities**:
    i. Facilitates user interactions such as tour browsing, bookings, and feedback.
    ii. Displays real-time data retrieved from the application layer.
- **Application Layer**:
  - **Components**: APIs, server-side logic, business rules.
  - **Responsibilities**:
    i. Processes user inputs, such as form submissions for bookings and feedback.
    ii. Manages authentication, tour recommendations, and reservation systems.
- **Database Layer**:
  - **Components**: Relational database (e.g., MySQL).
  - **Responsibilities**:
    i. Stores user data, tour details, booking records, and feedback securely.
    ii. Provides efficient data retrieval using SQL queries.

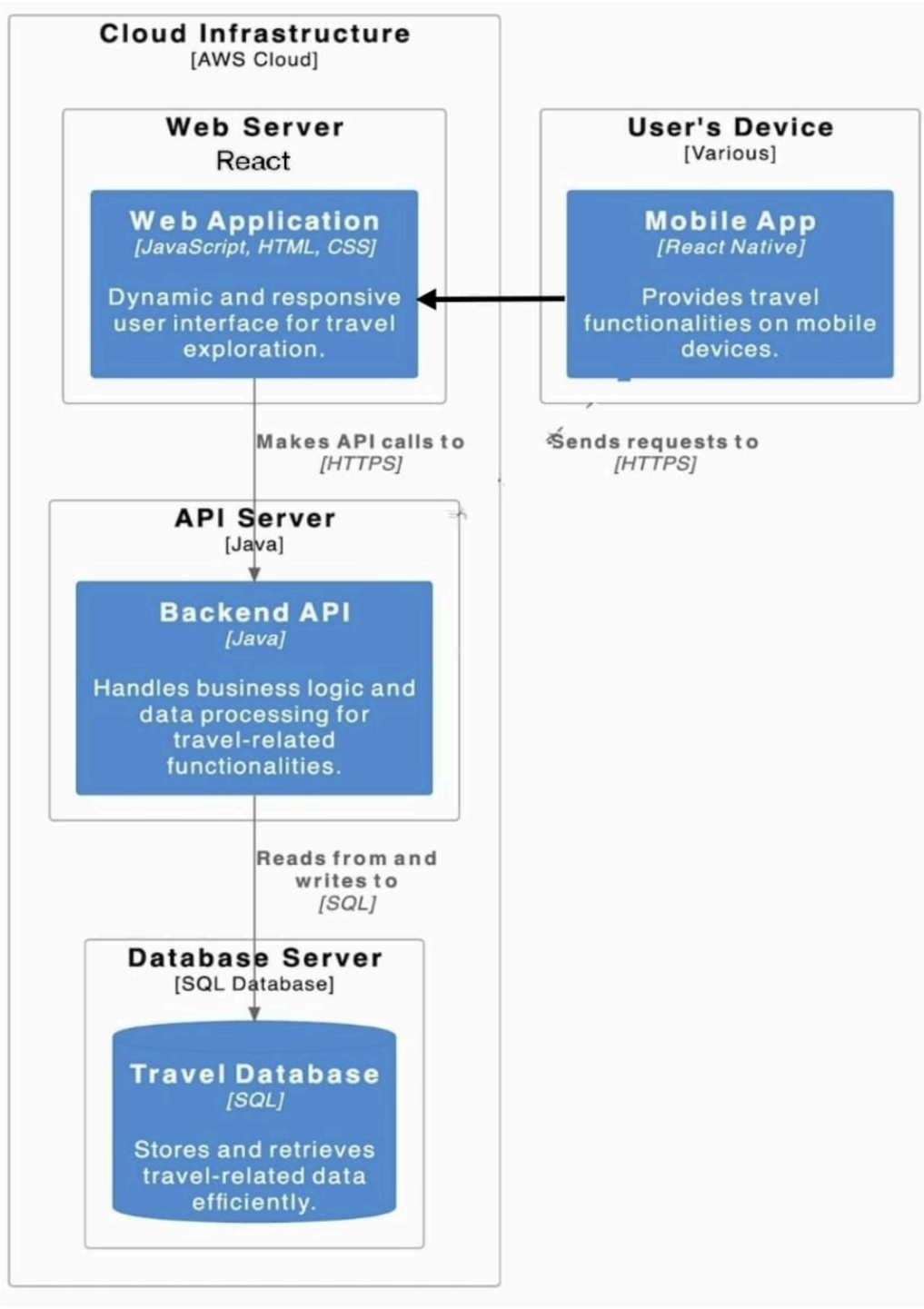

Fig:-3.1 Technical Architecture Diagram of Implementation of Optimum Travel Planner

3.2 Sequence Diagrams

Sequence diagrams describe the step-by-step flow of interactions among users and system components.

- User Registration:
  - Actors: User, Registration System, Database.
  - Steps:
    i. User submits details via the registration form.
    ii. Registration system validates the inputs.
    iii. Data is stored in the User Table.
    iv. System sends a success message to the user.
- Tour booking:
  - Actors: User, booking System, Database.
  - Steps:
    i. User selects a tour from the digital productDetail Table.
    ii. System retrieves availability and calculates total cost.
    iii. Upon confirmation, the system saves booking details in the booking Table.
    iv. User receives booking confirmation.
- Feedback Submission:
  - Actors: User, Feedback System, Database.
  - Steps:
    i. User provides feedback via a form.
    ii. Feedback is linked to the UserID and stored in the Feedback Table.
    iii. Admin reviews stored feedback for quality analysis.

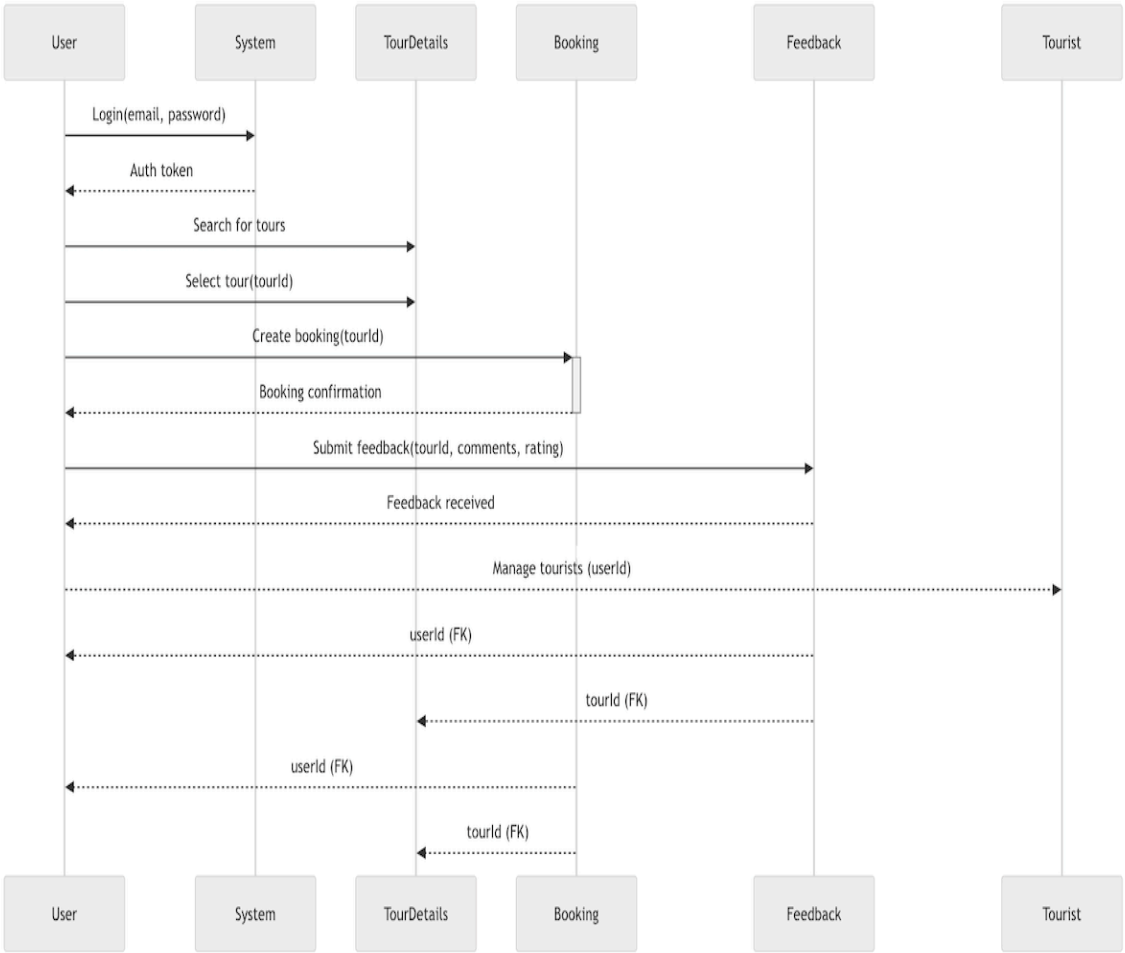

Fig:-3.2 Sequence Diagram of Implementation of Optimum Travel Planner System

3.3 Class Diagrams

3.3.1. Key Entities and Attributes

- **User**: Manages user details.
  - Attributes: userId, name, email, password, phone.
  - Purpose: Tracks user identity and facilitates interactions like bookings and feedback.
- **Tourist**: A specialized version of a user with additional details.

- Attributes: touristId, name, age, address.
- Purpose: Stores demographic and personal details for tour participants.

➢ **TourDetails**: Represents tours offered by the system.

- Attributes: tourId, destination, price, duration, description.
- Purpose: Captures essential details about tours for users.

➢ **Booking**: Links users and tours to track reservations.

- Attributes: bookingId, userId, tourId, bookingDate, status.
- Purpose: Manages reservations and their status.

➢ **Feedback**: Records user reviews and ratings for tours.

- Attributes: feedbackId, userId, tourId, comments, rating.
- Purpose: Enables users to provide feedback for service improvement.

3.3.2. Relationships

➢ **User ↔ Tourist**: One-to-One. Each user may have extended tourist details.

➢ **User ↔ Booking**: One-to-Many. A user can make multiple bookings.

➢ **TourDetails ↔ Booking**: One-to-Many. A tour can have multiple bookings.

➢ **User ↔ Feedback**: One-to-Many. A user can submit multiple feedback entries.

➢ **TourDetails ↔ Feedback**: One-to-Many. A tour can receive feedback from multiple users.

3.3.3. Functional Purpose

- ➢ Ensures **data integrity** through foreign key constraints linking userId and tourId across related entities.
- ➢ Enables modular expansion, like adding new features (e.g., admin management, loyalty programs).
- ➢ Simplifies processes such as **booking tracking**, **tour management**, and **feedback analysis**.
- ➢ Supports secure handling of sensitive data such as user credentials (password).

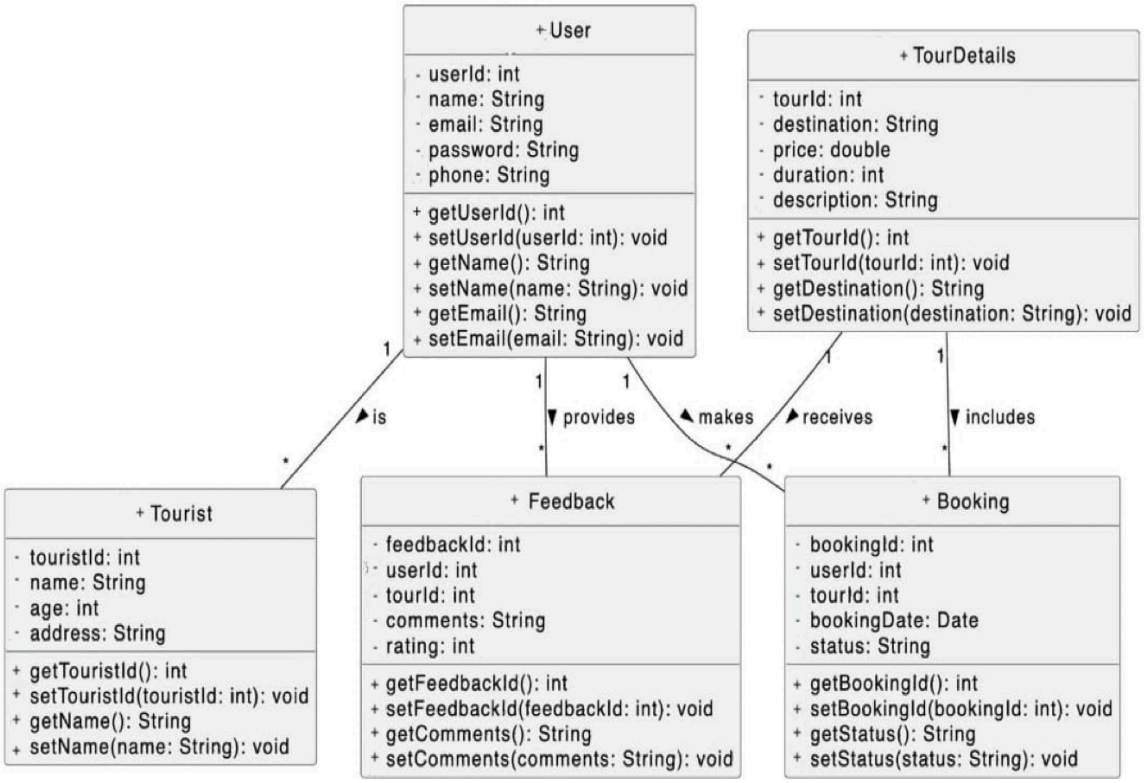

Fig:-3.3 Class Diagram of Implementation of Optimum Travel Planner System

3.4 Data Flow Diagram (DFD)

The DFD represents the flow of data between the system's processes, data stores, and external entities.

Level 0 (Context Diagram)

- ➢ Purpose: Displays the system as a single process interacting with external entities like users and administrators.
- ➢ Processes:
    - User inputs (registration, booking, feedback).
    - Admin inputs (tour management).
    - System outputs (feedback acknowledgment).

Level 1 (Detailed DFD)

- ➢ Sub-processes:
    - User Management: Handles registration, login, and profile updates.
    - Tour Management: Retrieves tour details and availability.
    - Booking Management: Processes reservations and updates availability in the database.
    - Feedback Management: Collects and stores user feedback for analysis.
- ➢ Data Stores:
    - User Table for user credentials and profiles.
    - digital productDetail Table for tour information.
    - booking Table for reservation records.
    - Feedback Table for user reviews.

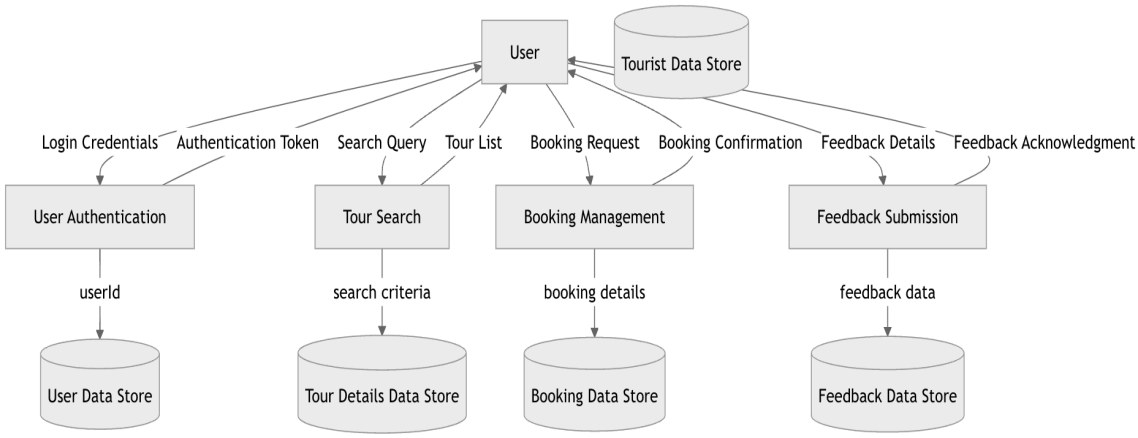

Fig:-3.4 Data Flow Diagram of Implementation of Optimum Travel Planner System

3.5 User Interface Design

The system's User Interface (UI) focuses on simplicity and usability.

Core UI Elements:

- ➢ Homepage:
  - Featured digital product.
  - Search and filter options for tour categories.
- ➢ Registration/Login Page:
  - User registration form with fields for personal and login details.
  - Login page for authentication.
- ➢ User Dashboard:
  - Displays:
    - i. Active bookings.
    - ii. Personalized recommendations.
    - iii. Past travel history.
- ➢ Tour Details Page:
  - Detailed description, pricing, and availability.
  - "Book Now" button linking to the booking form.

➢ Feedback Page:

- Star rating system.
- Text box for additional comments.

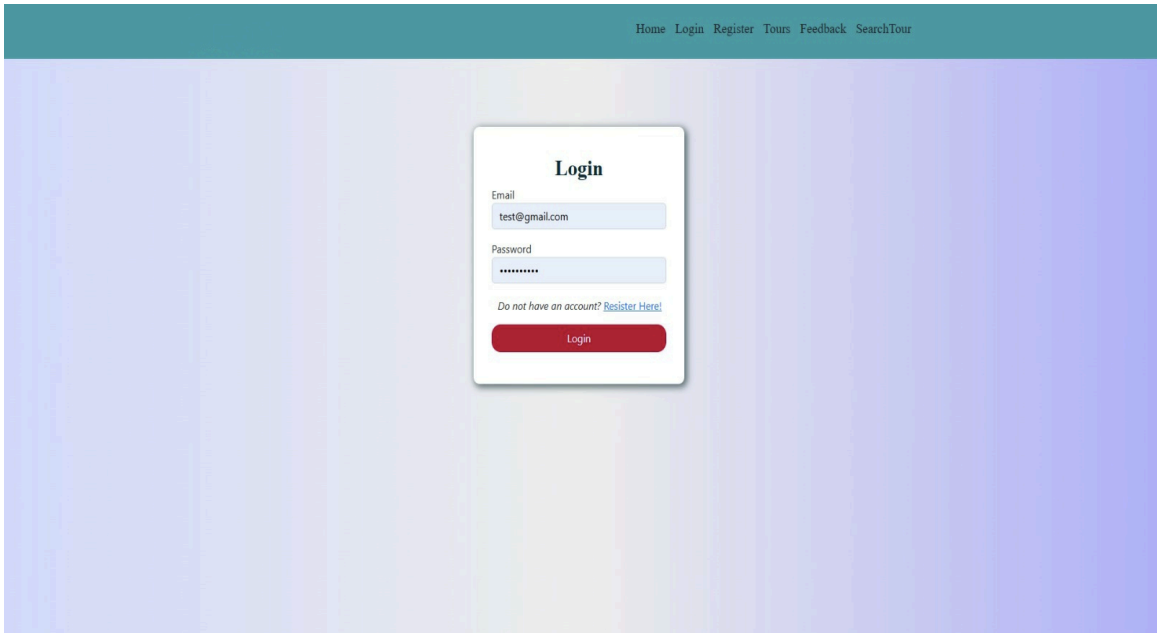

Fig:-3.5 User Interface Design of Implementation of Optimum Travel Planner

3.6 Data Design

This section explains the structure of the database and its role in system functionality.

3.6.1 Schema Definitions

The schema defines tables used in the system:

➢ **User:**

```
CREATE TABLE `users` (
  `user_id` bigint NOT NULL AUTO_INCREMENT,
  `address` varchar(100) NOT NULL,
  `date_of_birth` date NOT NULL,
```

```
  `email` varchar(255) NOT NULL,
  `first_name` varchar(20) NOT NULL,
  `last_name` varchar(20) NOT NULL,
  `password` varchar(255) NOT NULL,
  `phone_no` bigint NOT NULL,
  `role` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`user_id`)
);
```

➢ **Tourist Table:**

```
CREATE TABLE `tourist` (
  `tourist_id` bigint NOT NULL AUTO_INCREMENT,
  `age` int NOT NULL,
  `id_proof` varchar(255) NOT NULL,
  `id_proof_no` varchar(255) NOT NULL,
  `tourist_name` varchar(20) NOT NULL,
  `booking_id` bigint DEFAULT NULL,
  PRIMARY KEY (`tourist_id`),
  KEY `FKegiqgndkbel8rlanb6mk661i9` (`booking_id`),
CONSTRAINT `FKegiqgndkbel8rlanb6mk661i9` FOREIGN KEY (`booking_id`) REFERENCES `booking` (`booking_id`)
```

```
);
```

- **TourDetails:**

```
CREATE TABLE `tour_details` (
  `tour_id` bigint NOT NULL AUTO_INCREMENT,
  `activities` varchar(30) DEFAULT NULL,
  `booking_amount` double NOT NULL,
  `destination` varchar(30) NOT NULL,
  `max_seats` int DEFAULT NULL,
  `source` varchar(30) NOT NULL,
  `tour_detail_info` varchar(50) DEFAULT NULL,
  `tour_end_date` date NOT NULL,
  `tour_name` varchar(30) NOT NULL,
  `tour_start_date` date NOT NULL,
  `tour_type` varchar(255) DEFAULT NULL,
  `transportation_mode` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`tour_id`)
);
```

- **Booking Table:**

```
CREATE TABLE `booking` (
  `booking_id` bigint NOT NULL AUTO_INCREMENT,
```

```
  `booking_date` date NOT NULL,
  `payment_status` varchar(255) NOT NULL,
  `seat_count` int NOT NULL,
  `total_amount` bigint NOT NULL,
  `tour_id` bigint DEFAULT NULL,
  `user_id` bigint DEFAULT NULL,
  PRIMARY KEY (`booking_id`),
  KEY `FKhavlcxrs7dfxpgrr2w3jfgsg3` (`tour_id`),
  KEY `FK7udbel7q86k041591kj6lfmvw` (`user_id`),
  CONSTRAINT `FK7udbel7q86k041591kj6lfmvw` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
  CONSTRAINT `FKhavlcxrs7dfxpgrr2w3jfgsg3` FOREIGN KEY (`tour_id`) REFERENCES `tour_details` (`tour_id`)
);
```

- **Feedback Table:**

```
CREATE TABLE `feedback` (
  `feedback_id` bigint NOT NULL AUTO_INCREMENT,
  `comment` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `first_name` varchar(255) NOT NULL,
  `rating` int DEFAULT NULL,
```

```
  `user_id` bigint DEFAULT NULL,

  PRIMARY KEY (`feedback_id`),

  KEY `FKpwwmhguqianghvi1wohmtsm8l` (`user_id`),

CONSTRAINT `FKpwwmhguqianghvi1wohmtsm8l` FOREIGN KEY (`user_id`)
REFERENCES `users` (`user_id`)

);
```

3.6.2 E-R Diagram

The E-R Diagram visually represents the relationships between entities in the database:

- Entities:
  - User: Tracks personal and login details.
  - customer: Stores travel preferences and history.
  - digital productDetail: Contains tour-specific information.
  - booking: Links users to their reservations.
  - Feedback: Stores user feedback.
- Relationships:
  - User ↔ customer: One-to-One (each user is a customer).
  - User ↔ Feedback: One-to-Many (users can submit multiple feedback).
  - digital productDetail ↔ booking: One-to-Many (each tour can have multiple bookings).
  - User ↔ booking: One-to-Many (each user can make multiple bookings).

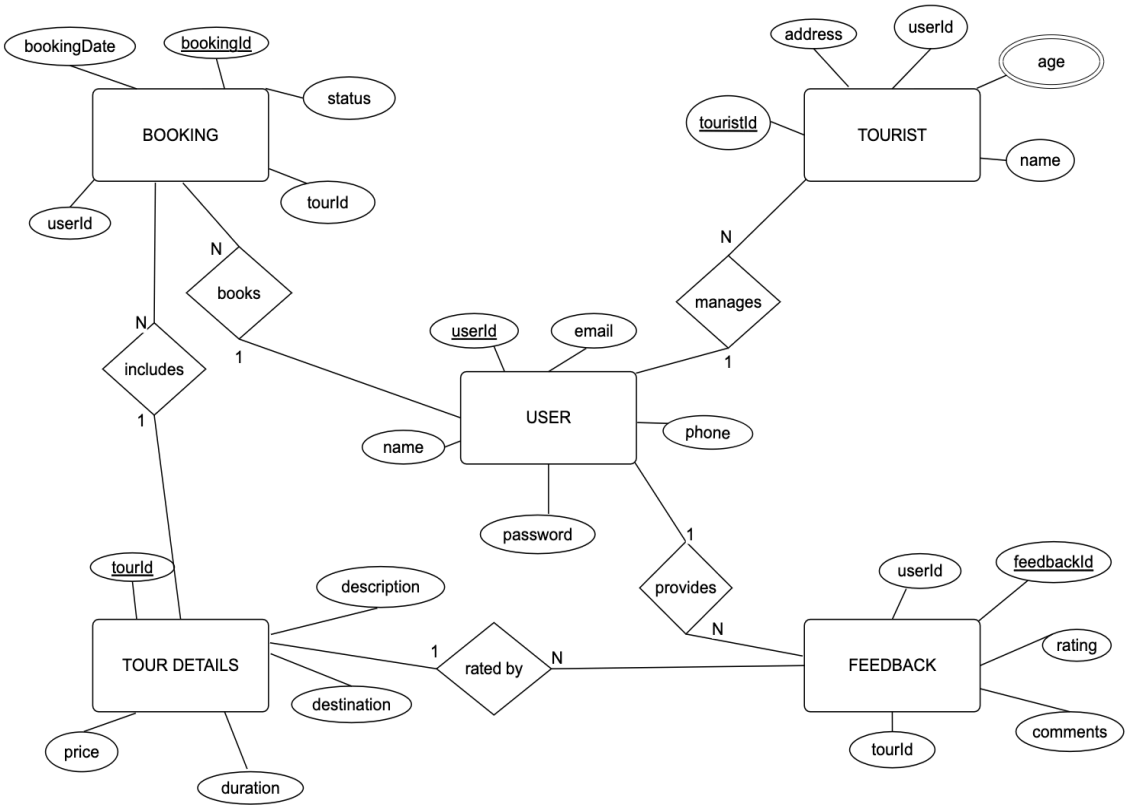

Fig:-3.6 E-R Diagram of Implementation of Optimum Travel Planner System

Chapter 4:

Implementation & Testing

4.1 Methodology

4.1.1 Proposed Algorithm

The development of Implementation of Optimum Travel Planner follows the Agile Model, ensuring iterative progress and adaptability throughout the project lifecycle.

4.1.1.1 Requirement Gathering and Analysis:

- ➢ Conduct initial discussions with stakeholders to understand system requirements.
- ➢ Document both functional and non-functional requirements.

4.1.1.2. System Design:

- ➢ Create a modular design with a focus on frontend (React.js), backend (Spring Boot), and database (MySQL).
- ➢ Prepare diagrams like UML, ERD, and DFD for better visualization of the system architecture.

4.1.1.3. Incremental Development:

- ➢ Divide the project into smaller, manageable modules (e.g., User Management, Tour Management).
- ➢ Develop and deploy each module in iterations, ensuring frequent deliveries.

4.1.1.4 Testing:

- ➢ Perform unit testing for individual modules.
- ➢ Conduct integration testing to ensure seamless communication between modules.
- ➢ Execute user acceptance testing (UAT) after each sprint to gather feedback.

4.1.1.5 Deployment:

- ➢ Deploy the system incrementally, starting with a beta version for user feedback.
- ➢ Perform final testing before full-scale deployment.

4.1.1.6 Maintenance and Updates:

- ➢ Gather feedback from users to identify improvements or bug fixes.
- ➢ Regularly update the system with new features and security patches.

4.2 Implementation Approach

4.2.1 Introduction to Languages, IDEs Tools and Technologies

- ➢ Languages
  - Java: Backend programming with Spring Boot.
  - JavaScript, HTML, CSS: Frontend development using React.js.
- ➢ Frontend
  - React.js: Dynamic, responsive UI with reusable components.
- ➢ Backend
  - Spring Boot: RESTful API development, dependency injection, and data management.
- ➢ Database
  - MySQL: Stores user, booking, and tour data.
  - 2.1.5. Development Tools
  - IDE: IntelliJ IDEA(backend), Visual Studio Code (frontend).
  - Build Tools: Maven (backend), Node.js/npm (frontend).
  - Postman: API testing.

4.3 Testing Approaches

4.3.1 Unit Testing

- ➢ User Registration
  - Test: Register a new user with valid inputs (username, email, password).
  - Expected Outcome: User is saved in the database and returned with correct details.
- ➢ Login
  - Test: Attempt to login with correct credentials (username and password).
  - Expected Outcome: A valid JWT token is generated and returned.
- ➢ Tour Listing Retrieval
  - Test: Retrieve all tours from the database.
  - Expected Outcome: List of tours is returned with correct details (name, price, description).

- Booking a Tour
  - Test: Book a tour for a specific user with valid tour ID and date.
  - Expected Outcome: Booking is confirmed and stored in the database.
- Feedback Submission
  - Test: Submit feedback for a tour with a valid rating and comment.
  - Expected Outcome: Feedback is saved in the database and associated with the correct tour.
- Invalid Input Handling
  - Test: Submit a booking with invalid tour ID.
  - Expected Outcome:An error is thrown (e.g., ResourceNotFoundException).

a. Test Cases

i. Frontend Unit Test Cases

- Test Case: Login Form Validation
  - Description: Ensure the login form validates input before submission.
  - Input: Email field is empty.
  - Expected Result: Error message "Email is required" is displayed.
- Test Case: Tour Search Form
  - Description: Verify the search form renders correctly and triggers the search function.
  - Input: User enters "destination: Paris" and clicks "Search".
  - Expected Result: The search function is called with the correct query.
- Test Case: Booking Confirmation Component
  - Description: Test if the booking confirmation page displays booking details.
  - Input: A booking object with tour details.

- Expected Result: The page renders the tour name, date, and number of people correctly.

➢ Test Case: Feedback Submission Form

- Description: Verify feedback form validation.
- Input: User submits without a comment.
- Expected Result: Error message "Comment is required" is displayed.

➢ Test Case: Responsive Design

- Description: Ensure that the homepage adjusts correctly for mobile view.
- Input: Simulate a screen width of 375px.
- Expected Result: The layout changes to a mobile-friendly design.

ii.Backend Unit Test Cases

➢ Test Case: User Registration

- Description: Test if a new user can be registered.
- Input: { "name": "John Doe", "email": "john@example.com", "password": "password123" }
- Expected Result: A new user is created, and the response contains the user's ID and success message.

➢ Test Case: User Login

- Description: Verify that a user can log in with correct credentials.
- Input: { "email": "john@example.com", "password": "password123" }
- Expected Result: Returns a success status and a JWT token for authentication.

➢ Test Case: Tour Search

- Description: Ensure that tours can be searched based on a destination.
- Input: Search query "destination=Paris"
- Expected Result: A list of tours to Paris is returned.

- Test Case: Booking Creation
    - Description: Test booking functionality for a tour.
    - Input: { "userId": 1, "tourId": 101, "numberOfPeople": 2 }
    - Expected Result: A booking record is created, and a success message is returned.
- Test Case: Feedback Submission
    - Description: Test if a user can submit feedback for a tour.
    - Input: { "tourId": 101, "userId": 1, "rating": 5, "comment": "Excellent tour!" }
    - Expected Result: Feedback is saved, and a confirmation is returned.

4.3.2 Integration Testing

- User Registration and Login
    - Test: Register a new user, then login using the registered credentials.
    - Expected Outcome: User is successfully registered and logged in, receiving a JWT token for authentication.
- Fetch Tour Listings
    - Test: Request to fetch all available tours, including filters for price and destination.
    - Expected Outcome: The system returns the correct list of tours based on the applied filters.
- Booking a Tour
    - Test: Book a tour by a logged-in user, with tour ID, date, and number of people.
    - Expected Outcome: Tour booking is successfully confirmed, and the database is updated with booking details.
- Submit Feedback
    - Test: Submit feedback for a completed tour with a rating and comment.
    - Expected Outcome: Feedback is saved successfully, and it appears under the relevant tour listing.

- Error Handling - Invalid Login
    - Test: Attempt to log in with incorrect credentials.
    - Expected Outcome: The system responds with an authentication error (401 Unauthorised).

b. Test Cases

- Test Case: User Registration and Login
    - Description: Test the integration between the registration and login modules.
    - Steps:
        i. Register a new user with { "name": "Alice", "email": "alice@example.com", "password": "securePass123" }.
        ii. Use the same email and password to log in.
    - Expected Result:
        i. Registration returns a success message.
        ii. Login returns a valid JWT token for the user.
- Test Case: Search and Book Tour
    - Description: Test the integration of the tour search and booking modules.
    - Steps:
        i. Search for tours to "destination: Paris".
        ii. Select a tour and proceed to booking with { "tourId": 101, "userId": 1, "numberOfPeople": 3 }.
    - Expected Result:
        i. Search returns a list of available tours to Paris.
        ii. Booking returns a success message and a booking ID.
- Test Case: User Profile and Booking History
    - Description: Test the integration between user profile and booking modules.
    - Steps:
        i. Log in as an existing user with multiple bookings.

    ii. Navigate to the profile page and view the booking history.
- Expected Result:
    i. User profile loads with accurate information.
    ii. Booking history displays a list of all previous bookings with correct details.

➢ Test Case: Feedback Submission and Display

- Description: Test the integration of feedback submission and retrieval modules.
- Steps:
    i. Submit feedback for a completed tour with { "tourId": 101, "userId": 1, "rating": 4, "comment": "Great experience!" }.
    ii. View feedback for the same tour as another user.
- Expected Result:
    i. Feedback submission returns a success message.
    ii. Feedback is displayed with the correct rating and comment for other users.

➢ Test Case: Admin Managing Tours

- Description: Test the integration of the admin interface and the tour management module.
- Steps:
    i. Log in as an administrator.
    ii. Add a new tour with details like destination, price, and dates.
    iii. Verify that the newly added tour appears in the user-facing search results.
- Expected Result:
    i. Tour is successfully added via the admin panel.
    ii. The new tour appears in the search results for users.

Chapter 5:

Results & Discussion

5.1 User Interface Representation

- Registration & Login:
  - Forms for user registration and login with username, email, and password.
  - Validation messages for incorrect inputs.
- Tour Listings:
  - Grid/List view of available tours with names, prices, and brief descriptions.
  - Filters for destination, price range, and date.
- Tour Details:
  - Tour information: Description, images, itinerary, pricing.
  - Book Now button for easy booking.
- Booking Process:
  - Form for selecting tour dates and number of people.
  - Price breakdown and "Confirm Booking" button.
- Feedback & Reviews:
  - Rating (1-5 stars) and comment section for user reviews.
  - Previous feedback displayed below the form.
- User Profile:
  - Profile page showing user details, bookings, and options to update information.

5.1.1 Brief Description of Various Modules

User Interface Representation: Brief Description of Various Modules

- **Home:-**
  - Description: Serves as the landing page, showcasing top destinations, offers, and featured tours. Includes a navigation bar for quick access to other modules.

- **Search and Explore:-**
  - Description: Allows users to search for tours by destination, price, and date. Displays search results with detailed tour descriptions and images.
- **User Registration and Login:-**
  - Description: Provides forms for users to register an account or log in. Supports role-based access for tourists and administrators.
- **Tour Booking Module:-**
  - Description: Enables users to book tours by selecting a tour, specifying the number of participants, and confirming the booking.
- **User Profile:-**
  - Description: Displays user-specific information such as booking history, saved tours, and profile details.
- **Feedback and Reviews:-**
  - Description: Allows users to view feedback from other tourists and submit their own reviews and ratings for completed tours.
- **Admin Dashboard:-**
  - Description: Provides administrators with tools to manage tours, view user activity, and handle feedback.
- **Responsive Design:-**
  - Description: Ensures the platform is accessible and visually appealing across devices (desktop, tablet, and mobile).

5.2 Snapshot of System with Brief Description

The Tours and Travels System is a user-friendly platform that allows users to explore and book tours, manage bookings, and provide feedback. It offers a seamless experience with secure registration, tour listings, booking management, and a responsive design, making travel planning easy and accessible.

UI – User Interface

Refers to the graphical layout of the website or application, including buttons, text, images, and other elements that users interact with.

API – Application Programming Interface

A set of protocols, routines, and tools for building software and applications. It specifies how software components should interact.

DB – Database

Refers to the structured set of data stored in a system, often in the form of tables.

JWT – JSON Web Token

A compact, URL-safe means of representing claims to be transferred between two parties, used for secure data exchange (e.g., user authentication).

CRUD – Create, Read, Update, Delete

Refers to the basic operations for managing data in a database.

SQL – Structured Query Language

A standard programming language used for managing and manipulating relational databases.

HTML – HyperText Markup Language

The standard markup language used for creating web pages and web applications.

CSS – Cascading Style Sheets

A style sheet language used for describing the presentation of a document written in HTML or XML.

JS – JavaScript

A programming language commonly used for creating interactive effects within web browsers.

ORM – Object Relational Mapping

A technique that allows you to interact with a relational database using an object-oriented programming language.

URL – Uniform Resource Locator

The address used to access resources on the internet.

CRUD Operations – Create, Read, Update, Delete

Common operations for manipulating data in a database.

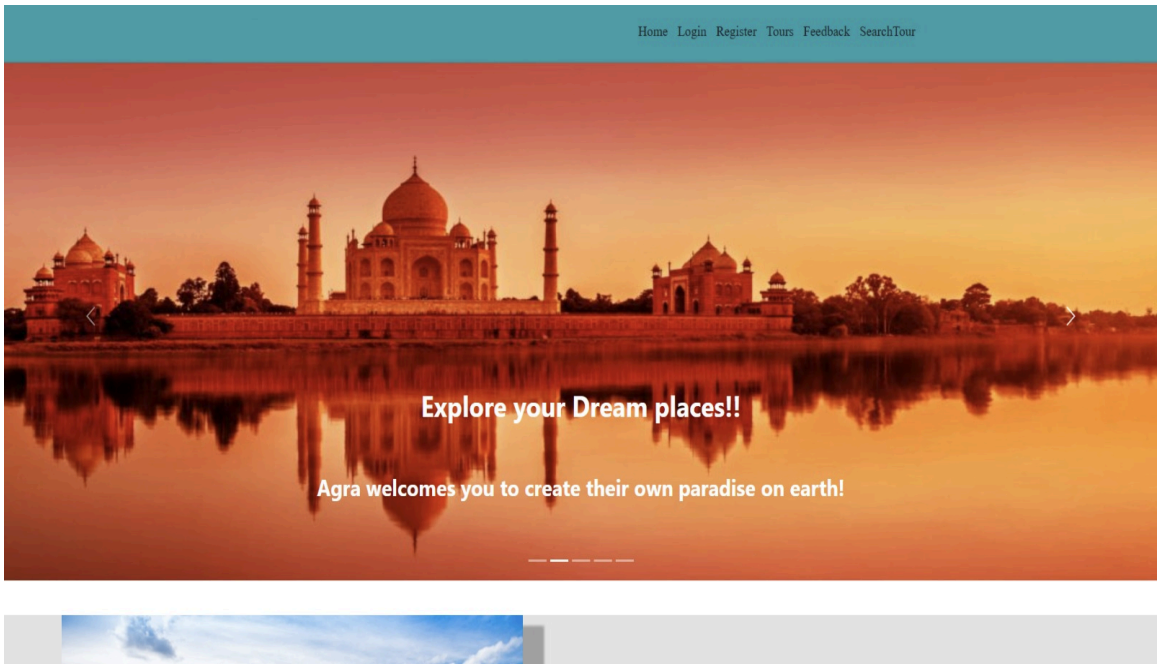

Fig:-5.1 Home Page of Implementation of Optimum Travel Planner

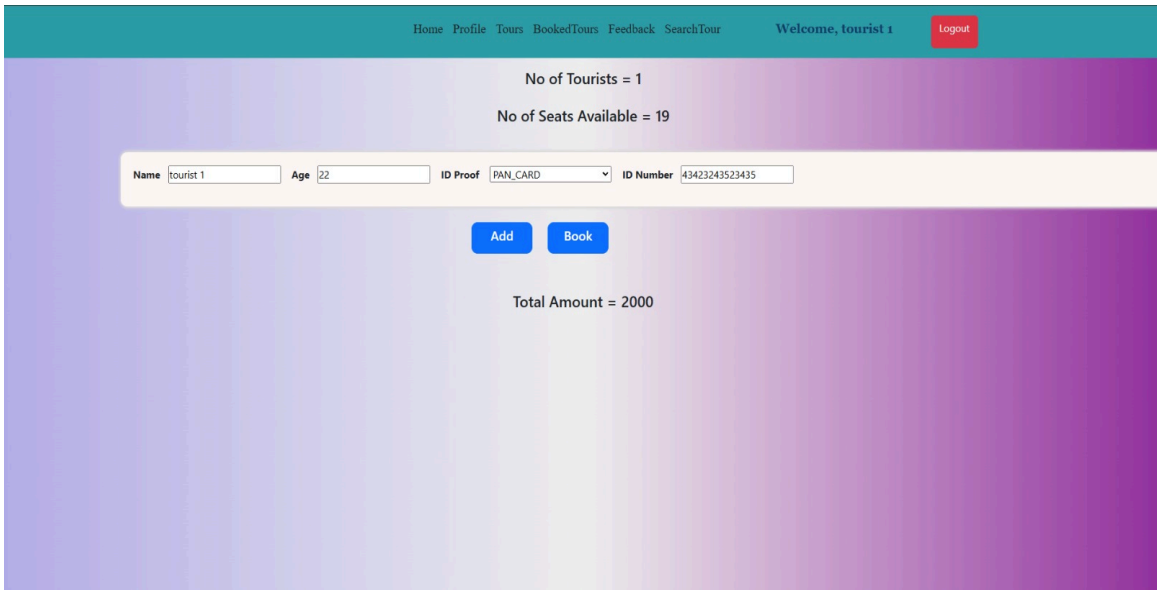

Fig:-5.2 Search and Explore of Implementation of Optimum Travel Planner

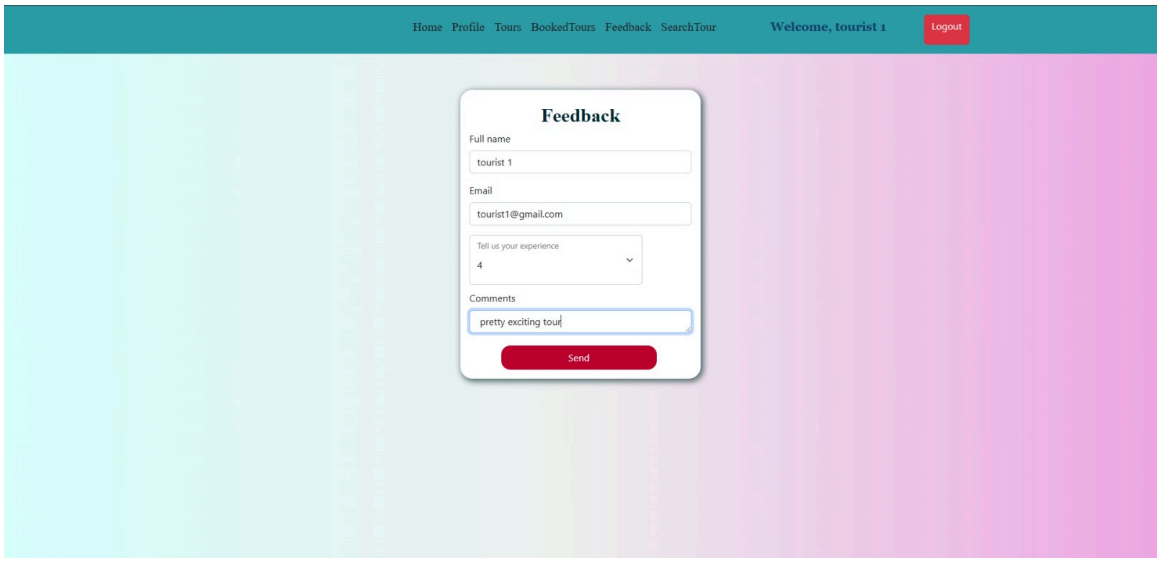

Fig:-5.3 Feedback Page of Implementation of Optimum Travel Planner

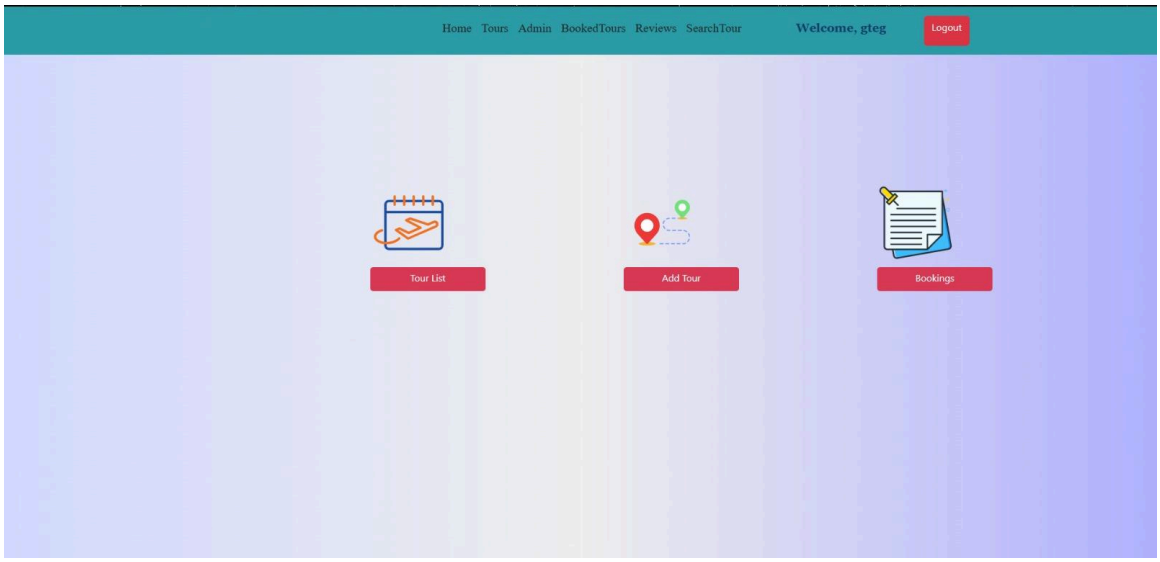

Fig:-5.4 Browsing Page of Implementation of Optimum Travel Planner

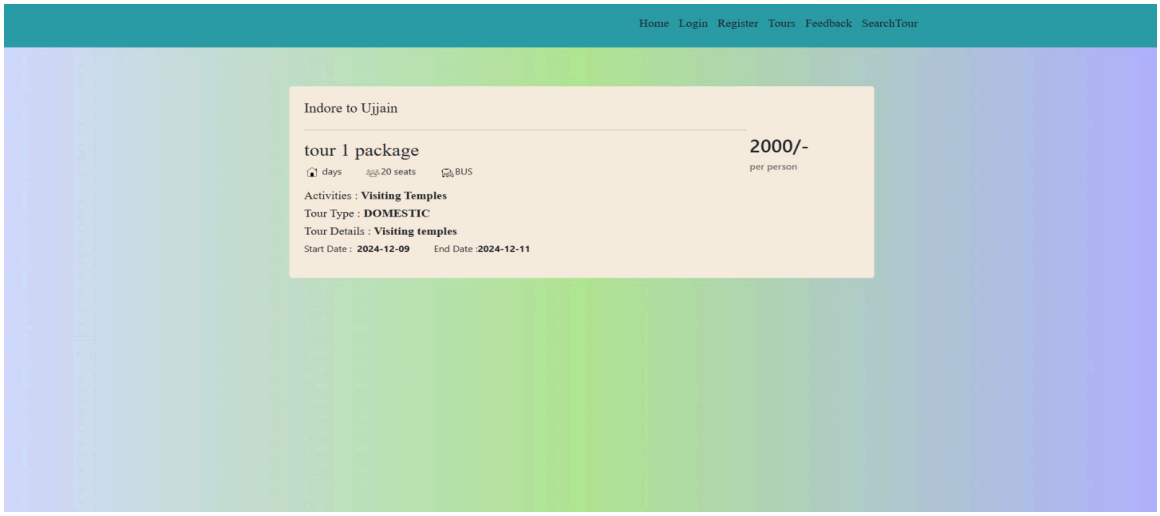

Fig:-5.5 Booking Page of Implementation of Optimum Travel Planner

5.3 Database Description

5.3.1 Snapshot of Database Tables with Brief Description

5.3.1.1 User Table

The Users table is a critical component of the database for this project, designed to store basic information about users of the system. It uniquely identifies each user using the userId column, which is the primary key. This ensures that all user-related operations, such as authentication, communication, and data management, are efficiently handled.

The table includes the following key fields:

- Name: Stores the user's full name.
- Email: Serves as a unique identifier for login and contact purposes.
- Password: Stores securely encrypted passwords to ensure the system's data security.
- Phone: Contains the user's contact number, which can be used for notifications or verification.

This table forms the foundation for implementing features such as user authentication, profile management, and interactions with other entities in the system, like orders or transactions. It ensures data integrity and acts as a bridge for user-related functionalities.

Table:-5.1 User Table of Implementation of Optimum Travel Planner System

Column Name	Data Type	Constraints	Description
userId	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each user
name	VARCHAR(100)	NOT NULL	User's name
email	VARCHAR(100)	UNIQUE, NOT NULL	User's email address
password	VARCHAR(255)	NOT NULL	Encrypted password
phone	VARCHAR(15)		User's phone number

5.3.1.2 Tourist Table

The Tourists table is designed to store detailed information about tourists using the system. It complements the Users table by providing additional details specific to tourists, such as age and address, while maintaining a connection to the corresponding user profile.

- Purpose:
  This table primarily focuses on managing data related to tourists, ensuring proper identification and linkage to the Users table through the userId column, which acts as a foreign key.
- Key Features:
  - touristId: A unique identifier for each tourist, automatically generated to ensure data integrity.
  - name: Captures the tourist's full name for identification.
  - age: Stores the age of the tourist, essential for personalizing services.
  - address: Provides the tourist's address, which may be required for correspondence or location-specific services.

- userId: Establishes a relationship with the Users table, ensuring that each tourist is linked to a valid user profile in the system.

➢ Relationships:

The userId column acts as a foreign key, creating a one-to-one or one-to-many relationship with the Users table. This linkage ensures that tourist records are associated with user credentials and accounts for seamless integration across the system.

This table is a vital component for applications in tourism management, helping manage tourist profiles and enabling features like bookings, itineraries, or personalized services.

Table:-5.2 Tourist Table of Implementation of Optimum Travel Planner System

Column Name	Data Type	Constraints	Description
touristId	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each tourist
name	VARCHAR(100)	NOT NULL	Tourist's name
age	INT	NOT NULL	Tourist's age
address	VARCHAR(255)		Tourist's address
userId	INT	FOREIGN KEY (User), NOT NULL	References User(userId)

5.3.1.3 TourDetail Table

The TourDetails table is designed to store comprehensive information about the tours available in the system. It plays a critical role in managing tour-related details such as destinations, pricing, durations, and descriptions, which are vital for both customers and administrators in the "Tour and Booking System."

➢ Purpose:

This table serves as the backbone for organizing and managing tour-related data. It ensures that users can access detailed and accurate information about each tour, facilitating smooth booking processes and enhancing the overall user experience.

- Key Features:
  - tourId:

    A unique identifier for each tour, automatically generated to maintain data integrity and prevent duplication. This column is the primary key for the table.
  - destination:

    Specifies the destination of the tour. It enables users to filter and search tours based on their preferred travel locations. The destination name can be up to 100 characters long.
  - price:

    Captures the cost of the tour in a DECIMAL format, allowing precise representation of prices, including fractional amounts (e.g., 9999.99). This ensures accuracy in financial transactions.
  - duration:

    Indicates the duration of the tour in days. This information helps users plan their schedules and assess the time commitment required for each tour.
  - description:

    Provides a detailed narrative about the tour, including highlights, itineraries, attractions, or special features. This field ensures users have sufficient information to make informed decisions.
- Relationships:

  The TourDetails table can be linked to other tables such as Bookings or TourCategories to enhance the functionality of the system. For instance:

- Linking to a Bookings table allows tracking reservations for specific tours.
- Associating with a TourCategories table helps classify tours based on themes (e.g., adventure, luxury, or cultural).

Such relationships make the database more scalable and adaptable to various system requirements.

➢ Significance:

The TourDetails table is a cornerstone of the "Tour and Booking System." It enables seamless management of tour data, ensuring users have access to detailed, reliable, and structured information about available tours. This table not only enhances the user experience but also supports administrators in efficiently managing the tour catalog and related operations.

Table:-5.3 TourDetails Table of Implementation of Optimum Travel Planner System

Column Name	Data Type	Constraints	Description
tourId	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each tour
destination	VARCHAR(100)	NOT NULL	Tour destination
price	DECIMAL(10,2)	NOT NULL	Tour price
duration	INT	NOT NULL	Tour duration in days
description	TEXT		Detailed tour description

5.3.1.4 Booking Table

The Bookings table is designed to store detailed information about bookings made by users in the "Tour and Booking System." It serves as a bridge between users and tours, managing the association between them and capturing the essential details of each booking transaction.

➢ Purpose:

This table plays a key role in managing data related to tour bookings. It ensures proper linkage between users and tours while maintaining comprehensive details about the booking date and current status.

➢ Key Features:

- bookingId:
  A unique identifier for each booking, automatically generated to ensure data integrity. This column serves as the primary key for the table.

- userId:

  A foreign key referencing the userId column in the Users table. It establishes a relationship between the booking and the user who made it, ensuring each booking is associated with a valid user profile.
- tourId:

  A foreign key referencing the tourId column in the TourDetails table. This links the booking to a specific tour, enabling the system to track which tour the user has booked.
- bookingDate:

  Captures the date when the booking was made. This helps in tracking and filtering bookings based on the booking period or date.
- status:

  Indicates the current status of the booking. The status can include values such as "Pending," "Confirmed," "Cancelled," or "Completed," providing insights into the booking lifecycle.

➢ Relationships:

- userId Relationship:

  Creates a one-to-many relationship with the Users table, as a single user can have multiple bookings.
- tourId Relationship:

  Creates a one-to-many relationship with the TourDetails table, as a single tour can have multiple bookings from different users.

These relationships ensure proper data linkage, enhancing the system's ability to track user activities and manage bookings effectively.

➢ Significance:

The Bookings table is a core component of the "Tour and Booking System," facilitating seamless management of bookings. It connects users to the tours they book, while also providing critical information such as booking dates and

statuses. This structure allows the system to deliver personalized services, track booking trends, and ensure smooth operations for users and administrators alike.

Table:-5.4 Booking Table of Implementation of Optimum Travel Planner System

Column Name	Data Type	Constraints	Description
bookingId	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each booking
userId	INT	FOREIGN KEY (User), NOT NULL	References User(userId)
tourId	INT	FOREIGN KEY (TourDetails), NOT NULL	References TourDetails(tourId)
bookingDate	DATE	NOT NULL	Date of booking
status	VARCHAR(50)	NOT NULL	Booking status

5.3.1.5 Feedback Table

The Feedback table is designed to store user feedback and ratings for tours in the "Tour and Booking System." It allows users to share their experiences and rate the tours they have participated in, helping administrators improve service quality and assisting future users in making informed decisions.

➢ Purpose:

This table focuses on capturing user feedback and ratings for tours, ensuring the system maintains a transparent and user-driven approach to evaluating tour quality. It supports the improvement of services and enhances user trust in the system.

➢ Key Features:

- feedbackId:
  A unique identifier for each feedback entry, automatically generated to ensure data integrity and avoid duplication. This column serves as the primary key for the table.

- userId:

  A foreign key referencing the userId column in the Users table. It links the feedback to the specific user who provided it, ensuring accountability and traceability.
- tourId:

  A foreign key referencing the tourId column in the TourDetails table. This connects the feedback to the specific tour it pertains to, helping administrators evaluate individual tours.
- comments:

  A text field that stores the user's feedback or experience in detail. This provides qualitative insights into the user's experience with the tour.
- rating:

  An integer field that stores the numeric rating (between 1 and 5) given by the user for the tour. The CHECK constraint ensures the rating is within the valid range, maintaining consistency.

➢ Relationships:

- userId Relationship:

  Establishes a one-to-many relationship with the Users table, as a user can leave feedback for multiple tours.
- tourId Relationship:

  Creates a one-to-many relationship with the TourDetails table, allowing multiple feedback entries for a single tour.

These relationships ensure seamless integration of feedback data with user and tour information, enhancing the system's analytical capabilities.

➢ Significance:

The Feedback table is an integral component of the "Tour and Booking System", enabling users to share their opinions and rate tours. This contributes to transparency and service improvement. The data collected through this table can be used for analytics, such as identifying popular tours, addressing complaints, or

rewarding highly-rated services, thereby enhancing the overall system performance and user satisfaction.

Table:-5.5 Feedback Table of Implementation of Optimum Travel Planner System

Column Name	Data Type	Constraints	Description
feedbackId	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each feedback
userId	INT	FOREIGN KEY (User), NOT NULL	References User(userId)
tourId	INT	FOREIGN KEY (TourDetails), NOT NULL	References TourDetails(tourId)
comments	TEXT		User feedback
rating	INT	CHECK (rating BETWEEN 1 AND 5)	Rating given to the tour

5.4 Final Findings

- Functionality:
  - The system successfully manages users, tours, bookings, and feedback.
  - Ensures seamless interaction between frontend (React.js) and backend (Spring Boot) via REST APIs.
- Performance:
  - Efficient database queries and API responses.
  - Scalable architecture for handling increased user demand.
- Security:
  - Implements JWT authentication for secure user sessions.
  - Validates inputs and handles errors gracefully.
- User Experience:
  - Responsive design with an intuitive UI using React.js.
  - Quick booking and feedback submission process.
- Scope for Improvement:
  - Add advanced search filters for tours.

- Implement analytics for bookings and feedback trends.
- Enable multi-language support for global accessibility.

Chapter 6:

Conclusion & Future Scope

6.1 Conclusion

- The Tours and Travels System successfully integrates a robust backend with a dynamic frontend to provide a seamless experience for users. Key features like secure user management, efficient tour booking, and a responsive feedback system ensure the platform's functionality and reliability.
- The use of modern technologies such as Spring Boot, React.js, and MySQL contributes to scalability and performance, while JWT-based authentication ensures security. With a strong foundation in place, the system is well-suited for future enhancements like advanced analytics and expanded user features, making it a comprehensive solution for the travel industry.

6.2 Future Scope

- Advanced Search & Filters:
  - Add detailed filters for tours (e.g., location, price, rating).
- Analytics Dashboard:
  - Provide insights into booking trends and user preferences.
- Mobile App:
  - Develop iOS/Android apps for broader accessibility.

REFERENCES

1. Smith, J., & Brown, A. (2019). The digital transformation of the tourism industry: Challenges and opportunities. *Journal of Tourism Research, 12*(3), 45-60.
2. Lee, S., Kim, H., & Park, J. (2020). Integrating tour management and user feedback systems in travel platforms. *International Journal of Tourism Technology, 15*(2), 75-88.
3. Garcia, M., & Martinez, R. (2021). Building scalable web applications with Spring Boot and React.js: A case study in the tourism industry. *Software Development Journal, 29*(5), 102-117.
4. Johnson, D., Rogers, P., & Wei, L. (2018). User-centric design in travel platforms: Enhancing user experience through intuitive interfaces. *Journal of User Experience Design, 21*(4), 31-48.
5. Miller, S., & Carter, L. (2017). Feedback systems in tourism platforms: The role of reviews and ratings in building trust. *Journal of Business and Tourism, 8*(1), 112-125.
6. Singh, R., Gupta, A., & Sharma, P. (2020). Automation in tourism management systems: A study of the impact on operational efficiency. *International Journal of Systems and Software Engineering, 14*(6), 93-107.
7. Kim, J., & Lee, B. (2019). The importance of real-time notifications and alerts in enhancing user engagement in travel platforms. *Tourism Technology Review, 11*(2), 55-69.
8. Patel, V., & Mehta, N. (2021). Leveraging machine learning for personalized travel recommendations in tourism platforms. *Journal of Artificial Intelligence in Tourism, 6*(3), 26-38.
9. Roberts, M., & Davis, K. (2020). Security and privacy in online travel systems: Best practices for integrating secure payment gateways. *International Journal of Travel Technology, 17*(4), 77-91.

Appendices

Appendix A: Project Synopsis

The project synopsis provides a high-level overview of the Implementation of Optimum Travel Planner platform. It includes:

- **Objective**: To develop a comprehensive web-based tour and booking system for seamless interaction between tourists and administrators.
- **Scope**: Covers tour management, bookings, feedback, and user administration using modern technologies like React.js, Spring Boot, and MySQL.
- **Key Features**:
  - Dynamic tour search functionality.
  - Secure and intuitive user authentication.
  - Role-based access for tourists and administrators.
  - Real-time feedback and notifications.
- **Deliverables**:
  - A functional, deployed web application.
  - Documentation including a user manual, technical design documents, and source code with version history.

Appendix B: Guide Interaction Report

This appendix captures all interactions with the project guide or mentor, serving as a log of guidance and its incorporation into the project.

- **Initial Meeting**:
  - **Date**: [Insert date]
  - **Agenda**: Discuss project objectives, technical stack, and initial design.
  - **Outcome**: Approval of the project idea and suggestions for a modular architecture.
- **Regular Updates**:
  - **Dates**: [List periodic updates, e.g., weekly or bi-weekly meetings]
  - **Topics Discussed**:

    i. Progress updates on development and testing.
    ii. Challenges faced, including integration issues and their resolution.
  - **Feedback from Guide**:
    i. Suggested improvement to the user interface for enhanced usability.
    ii. Recommended security features like JWT authentication.

➢ **Final Review**:
  - **Date**: 09/12/2024
  - **Agenda**: Review the completed system.
  - **Feedback**: Positive feedback on the project functionality with minor suggestions for improvement in future iterations.

Appendix C: User Manual

The user manual is a step-by-step guide to using the Implementation of Optimum Travel Planner platform. It is structured as follows:

➢ **Getting Started**:
  - How to register as a tourist or an administrator.
  - Login steps, including password recovery.

➢ **Tourist Features**:
  - Searching for tours: Using filters like location, budget, and availability.
  - Booking a tour: Selecting preferences, checking availability, and confirming booking.
  - Viewing booking history and canceling bookings.
  - Providing feedback or ratings for completed tours.

➢ **Administrator Features**:
  - Managing user accounts: Approving registrations and handling account issues.
  - Creating and updating tours: Adding descriptions, pricing, and availability.
  - Viewing and responding to user feedback.

- **FAQ and Troubleshooting**:
  - Common issues and their solutions (e.g., login errors, booking issues).
  - Contact information for further support.

Appendix D: Git/GitHub Commits/Version History

This appendix documents the version control practices used during development. It includes:

- **Repository Information**:
  - **Platform**: GitHub
  - **Repository Name**: WanderlustExplorers
  - **Link**: https://github.com/Acro-Team/Wanderlust-Exploreres
- **Commit Log**:
  - **Initial Commit**: Set up the repository with a README and project structure.
- **Branching and Merging Strategy**:
  - Development branches for features such as user-auth, tour-management, and feedback-module.
  - Regular merging into the main branch after peer review and testing.
- **Tagging Releases**:
  - Version 1.0: Initial system deployment with core features.
  - Version 1.1: Added feedback module and minor UI improvements.
- **Activity Summary**:
  - Total Commits: 57
  - Contributors: https://github.com/SarthakKacholiya

    https://github.com/Jrohitsahu

    https://github.com/vikasdhakad1

    https://github.com/TanmayIndoriya