PROJECT REPORT

ON

TRAVEL BOOKING PLATFORM USING AGILE METHODOLOGY

Submitted By:

Nikhil Harwani

1. ABSTRACT

The travel industry has witnessed a paradigm shift towards digital solutions, necessitating robust online platforms for booking and itinerary management. This project report outlines the development of a comprehensive Travel Booking Platform utilizing the Agile software development methodology. The primary objective is to create a seamless, user-centric interface that allows travelers to search, compare, and book flights and hotels efficiently.

By adopting the Agile Scrum framework, the development team effectively managed the complexities of integrating third-party travel APIs and evolving user requirements. The system architecture prioritizes scalability and security, featuring real-time inventory updates, secure payment processing, and personalized user dashboards.

This report documents the software development lifecycle, from initial problem identification to final testing and deployment. It highlights how iterative development cycles facilitated rapid delivery of high-value features, resulting in a reliable and competitive travel solution.

2. INTRODUCTION

2.1 Introduction

A Travel Booking Platform is an online aggregation system that connects users with travel service providers such as airlines and hotel chains. In an era where convenience is key, these platforms enable users to plan entire trips from a single interface. This project focuses on building a web-based Online Travel Agency (OTA) tailored for modern travelers, developed using the MERN stack (MongoDB, Express, React, Node.js) under an Agile framework.

2.2 Problem Identification

Travelers often face significant hurdles when planning trips using traditional or fragmented methods:

- **Platform Fragmentation:** Users must visit separate websites for flights and accommodation.
- **Pricing Opacity:** Lack of real-time comparison makes it difficult to find the best deals.
- **Complex Navigation:** Many existing legacy systems have cluttered, unintuitive interfaces.
- **Slow Response Times:** Delayed confirmations can lead to booking errors and lost opportunities.

2.3 Need of the Project

To solve these issues, the proposed platform aims to:

1. Centralize flight and hotel bookings into one unified dashboard.
2. Integrate real-time APIs to ensure accurate pricing and availability.
3. Provide a streamlined, mobile-responsive user experience.
4. Implement secure, instant payment and booking confirmation workflows.

2.4 Project Scheduling (Agile Timeline)

The project was executed in four 2-week sprints:

Phase	Duration	Deliverable
Sprint 1	2 Weeks	Environment Setup, UI Prototyping, Flight Search Logic
Sprint 2	2 Weeks	User Auth, Hotel Integration, Booking Engine
Sprint 3	2 Weeks	Payment Gateway, User Profile, Itinerary History
Sprint 4	2 Weeks	Admin Dashboard, Testing, Deployment

2.5 Objectives

- To develop a full-stack web application for travel reservations.
- To utilize Agile methodologies for flexible and adaptive development.
- To integrate Global Distribution Systems (GDS) for live travel data.
- To ensure data security and compliance with payment standards.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose

This SRS document details the functional and non-functional requirements for the Travel Booking Platform. It acts as a roadmap for the development team and a validation guide for stakeholders.

3.2 Scope

The scope includes a **User Portal** for searching and booking flights/hotels, and an **Admin Portal** for managing bookings and viewing revenue reports. Advanced features like car rentals, cruise bookings, and AI-based dynamic pricing are slated for future releases.

3.3 System Requirements

Hardware Requirements (Server)

- **Processor:** Quad-core 2.5 GHz+
- **RAM:** 8 GB Minimum
- **Storage:** 100 GB SSD
- **Bandwidth:** High throughput for API traffic

Software Requirements

- **OS:** Linux (Ubuntu/CentOS)
- **Frontend:** React.js, Redux, Bootstrap/Tailwind
- **Backend:** Node.js, Express.js
- **Database:** MongoDB (NoSQL)
- **External APIs:** Amadeus/Sabre (Travel Data), Stripe (Payments)

3.4 Tools Used

- **VS Code:** Code Editing.
- **Git/GitHub:** Source Control.

- **Jira:** Agile Sprint Management.
- **Postman:** REST API Testing.

3.5 Software Process Model

Agile Scrum: The project adheres to the Agile Scrum model. This approach is ideal for the travel domain where third-party API specifications can change, and user feedback on the booking flow is critical for conversion rates. It allows for continuous integration and regular delivery of working software modules.

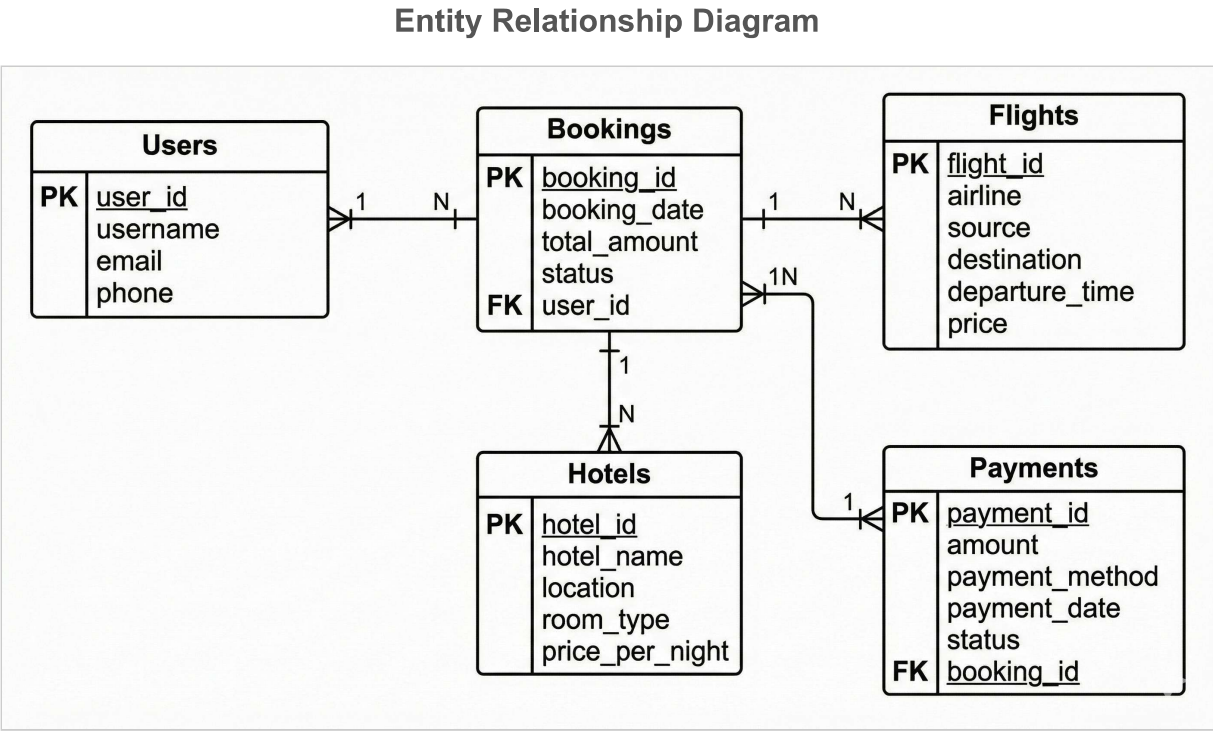
4. SYSTEM DESIGN

4.1 Data Dictionary

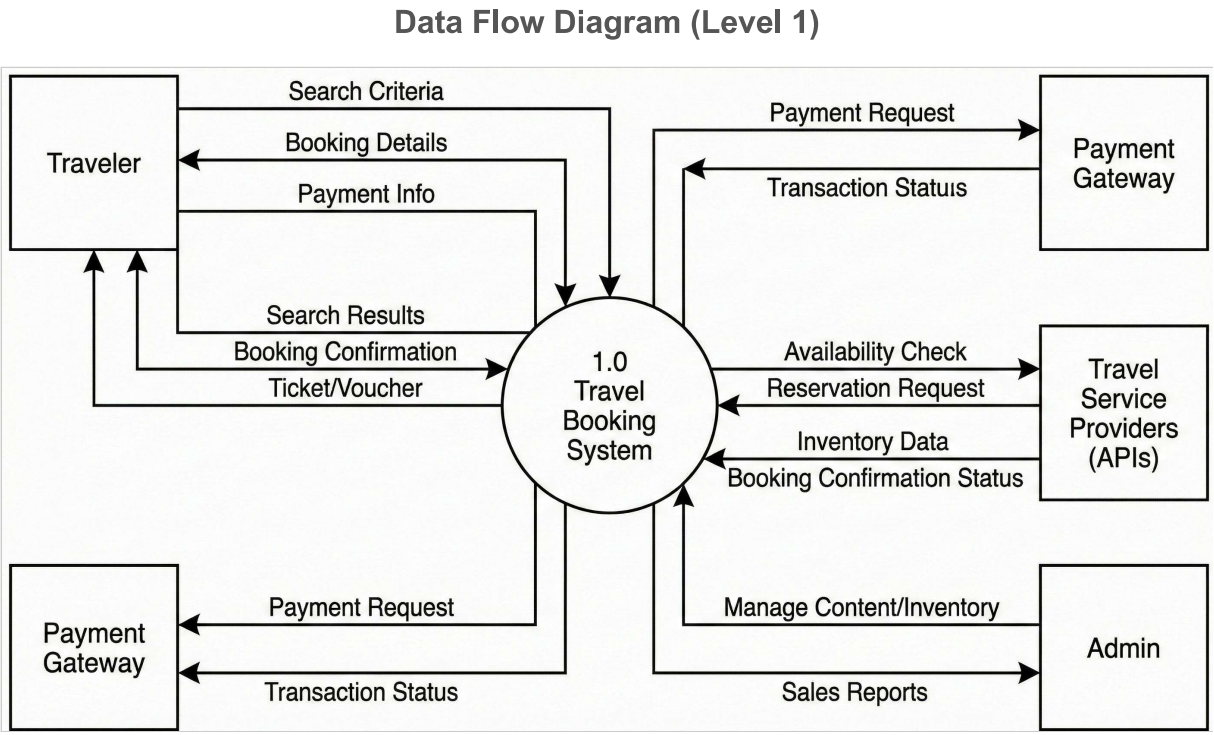
Table: Bookings

Field	Type	Description
booking_id	ObjectId (PK)	Unique Booking Reference
user_id	ObjectId (FK)	Reference to User
type	ENUM	'FLIGHT', 'HOTEL'
total_price	DECIMAL	Final transaction amount
status	ENUM	'CONFIRMED', 'CANCELLED', 'PENDING'

4.2 ER Diagram

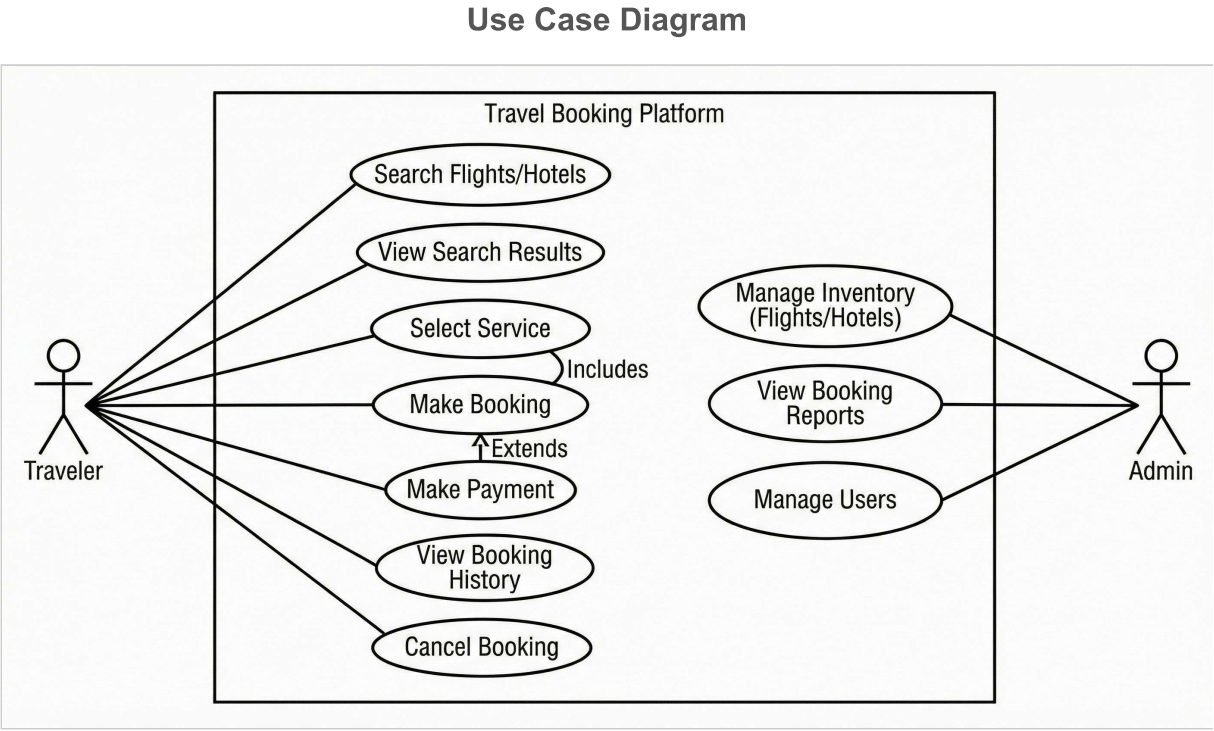


4.3 Data Flow Diagram (DFD)

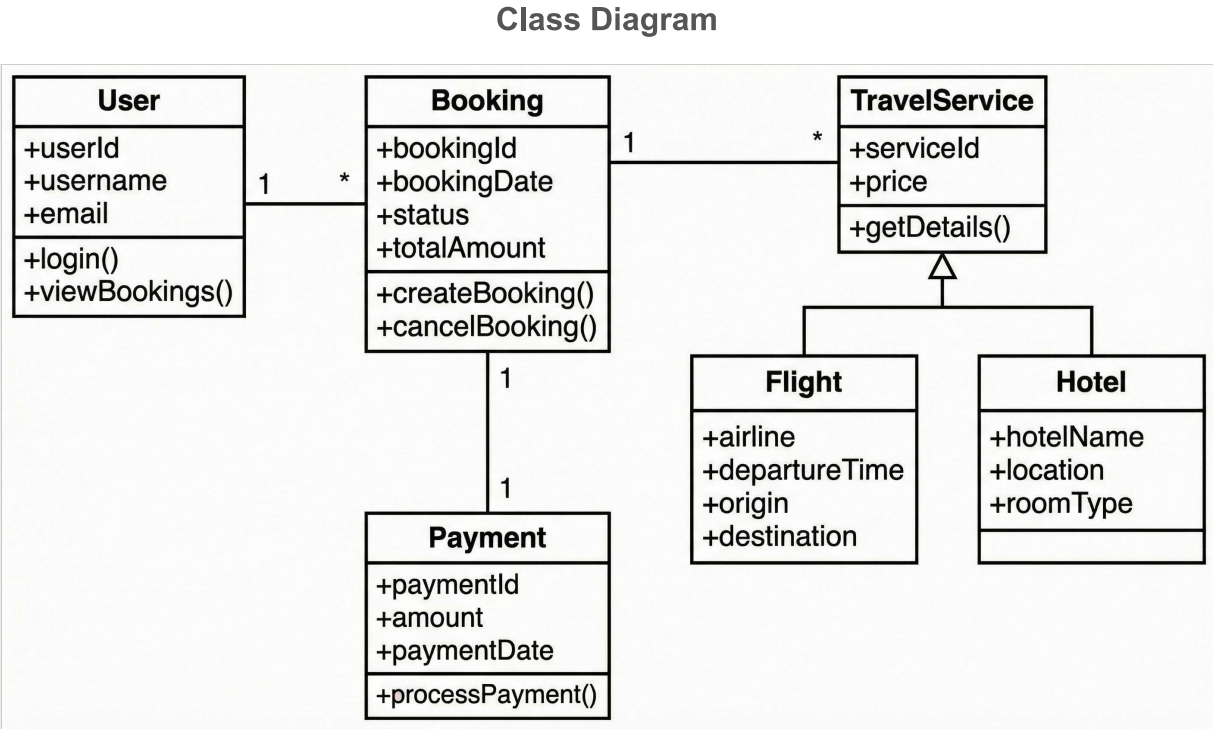


4.4 UML Diagrams

Use Case Diagram



Class Diagram



5. IMPLEMENTATION

5.1 Program Code

Key logic for the booking engine and API integration.

Backend: Booking Controller

```
// controllers/bookingController.js
const Booking = require('../models/Booking');
const Payment = require('../services/payment');

exports.createBooking = async (req, res) => {
    try {
        const { userId, tripDetails, paymentToken } = req.body;

        // 1. Validate Inventory (Mock API)
                                    const     isAvailable     =     await
checkAvailability(tripDetails.serviceId);
                if   (!isAvailable)   return   res.status(400).send("Service
Unavailable");

        // 2. Process Payment
            const  transaction  =  await  Payment.charge(tripDetails.price,
paymentToken);

        // 3. Save Booking
        const newBooking = await Booking.create({
            user: userId,
            details: tripDetails,
            transactionId: transaction.id,
            status: 'CONFIRMED'
        });

        res.status(201).json({ success: true, data: newBooking });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
};
```

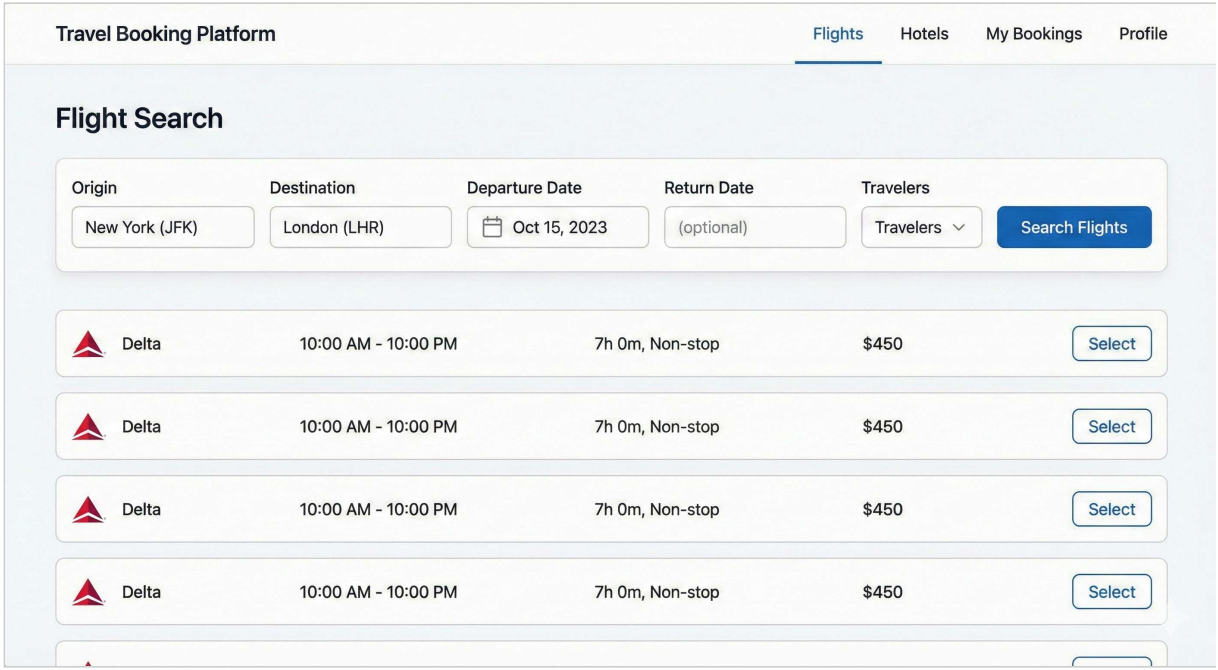
Frontend: Search Component

```
// components/SearchBox.js
import React, { useState } from 'react';

const SearchBox = ({ onSearch }) => {
    const [criteria, setCriteria] = useState({
        from: '', to: '', date: ''
    });

    const handleSubmit = (e) => {
        e.preventDefault();
        onSearch(criteria);
    };

    return (


                    From (City)  setCriteria({...criteria, from:
e.target.value})}
            />
                    To (City)  setCriteria({...criteria, to:
e.target.value})}
            />
            Find Flights


    );
};
```

5.2 Output Screens

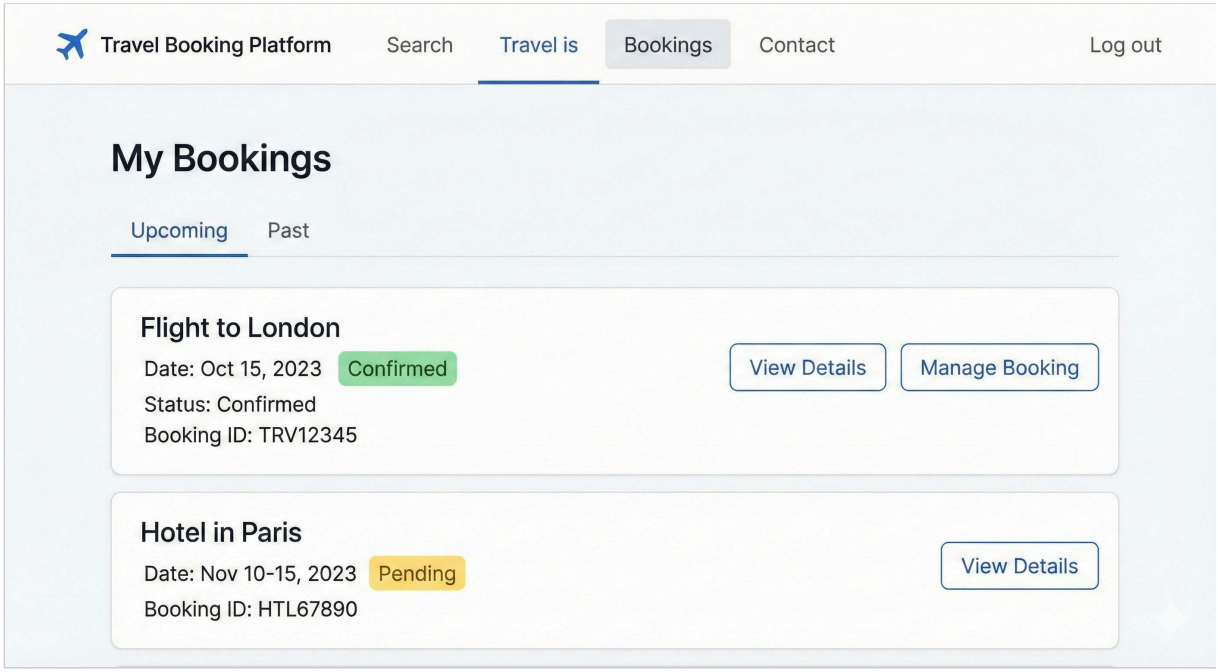
Login Screen:

Login Interface Screenshot



Travel Dashboard & Search:

Dashboard Screenshot



6. TESTING

6.1 Test Data & Strategy

A combination of Unit Testing (Jest) and Integration Testing was used. Mock data representing various travel scenarios (domestic, international, multi-passenger) was employed to validate system stability.

6.2 Test Results

Test Case ID	Description	Input Data	Expected Output	Status
TC_SEARCH_01	Valid Flight Search	Origin: DEL, Dest: BOM, Date: Future	Show Flight List	PASS
TC_BOOK_01	Booking Without Payment	Trip selected, no payment info	Error: Payment Required	PASS
TC_AUTH_01	User Registration	Valid email, strong password	Account Created & Logged In	PASS
TC_API_01	External API Failure	Simulate API Timeout	Graceful Error Message	PASS

7. USER MANUAL

7.1 How to use project guidelines

1. **Start:** Open the homepage and log in using your credentials.
2. **Search:** Enter your travel dates and destination in the search bar.
3. **Select:** Choose a flight or hotel from the results list.
4. **Pay:** Enter payment details to confirm the booking.
5. **Manage:** View your e-ticket under the "My Trips" section.

7.2 Screen Layouts and Description

Home Page: Features a large search banner and promotional offers.
Search Results: A list view with filters for price, duration, and airline/hotel ratings.
Checkout: A secure form for passenger details and payment processing.

8. PROJECT APPLICATIONS AND LIMITATIONS

Applications

- **Leisure Travel:** Simplifies vacation planning for families and solo travelers.
- **Business Travel:** Efficient booking for corporate trips.
- **Last-Minute Deals:** Quick access to immediate inventory.

Limitations

- Dependency on third-party API uptime and rate limits.
- Complex itineraries (e.g., multi-city with different classes) are not yet supported.
- Requires active internet connection; no offline mode.

9. CONCLUSION AND FUTURE ENHANCEMENT

The Travel Booking Platform project successfully demonstrates the efficacy of Agile methodologies in developing complex, consumer-facing web applications. By breaking down development into sprints, the team delivered a functional, user-friendly platform that addresses the core needs of modern travelers. The system efficiently handles search, booking, and payment processes.

Future Enhancements:

- **AI Recommendations:** Personalized trip suggestions based on user history.
- **Mobile App:** Native iOS and Android applications.
- **Crypto Payments:** Integration of cryptocurrency payment options.

10. BIBLIOGRAPHY & REFERENCES

Books:

- Martin, R. C. (2002). *Agile Software Development, Principles, Patterns, and Practices*. Pearson.
- Banks, A., & Porcello, E. (2017). *Learning React*. O'Reilly Media.

Websites:

- React Documentation (https://reactjs.org/)
- Amadeus for Developers (https://developers.amadeus.com/)
- Agile Alliance (https://www.agilealliance.org/)