Tourism Management System

PROJECT REPORT

For

SEC 160 – FULL STACK DEVELOPMENT-1

Submitted by

Name	**Registration-No**
A S Vittal	AP23110010231
S Ashwin	AP23110010847
N Karthik	AP23110010914
K Balaji	AP23110010195

SEM: V

SECTION: O

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

SCHOOL OF ENGINEERING AND SCIENCES

SRM University-AP, Neerukonda, Andhra Pradesh 522240
December 2025

TABLE OF CONTENTS

Introduction

The Travel Booking Web Application is a modern, fully responsive platform designed to simplify the experience of exploring destinations, viewing travel packages, and managing bookings online.
Built using **React.js** , the application prioritizes performance, usability, and scalability through a component-based architecture.

This project focuses on delivering a seamless user journey with intuitive navigation, well-organized layouts, and fast-loading UI components. The backend API is integrated using Axios, ensuring smooth communication between the frontend interface and server data.

The system also includes an **admin module** designed for managing bookings, offering essential CRUD functionalities. Route protection ensures that only verified administrators can access restricted areas, maintaining system security and data integrity.

Scenario-Based Introduction

Imagine a user planning a vacation. They open the Travel Booking Web Application and instantly view a beautifully structured Home Page with featured destinations and travel packages.
The user explores the *Destinations* section, scrolling through curated cards showing various places, each with images, pricing, and brief descriptions.

Curious about a particular place, the user selects it, navigating to the *Destination Details* page. Here, they see detailed information, including itinerary, activities, and package highlights.

From there, the user browses travel packages and selects a package that suits their budget and preferences. The platform prompts them to log in or register to proceed with the booking.

Upon successful login, the user books the package with a single click, and the booking details are saved securely.

At the same time, an admin logs into the secure admin dashboard, where they can view all bookings, edit any incorrect details, or delete bookings that are no longer valid. The admin module ensures smooth backend operations, contributing to the overall management of the system.

Every part of this application is designed to deliver clarity, accessibility, and convenience to both users and admins.

Target Audience

The application primarily targets the following groups:

- **General users**

  People who want a simple and intuitive way to browse destinations and book travel packages online.

- **Travel agencies / administrative staff**

  Administrators who need easy access to booking records and need tools for editing, updating, or managing user bookings efficiently.

- **Students & developers**

  Learners exploring how modern frontends interact with backend APIs using React.

Project Goals and Objectives

Main Goal

To build an accessible, efficient, and streamlined platform for users to browse travel options and place bookings while providing administrators with the tools needed for backend management.

Objectives

- Develop a **fully responsive and visually appealing** UI using Tailwind CSS
- Ensure smooth navigation using **React Router**
- Enable users to view destinations and packages through clean structured pages
- Implement **authentication** for login and registration
- Integrate backend APIs for dynamic content retrieval
- Provide admin-only features, including **secure CRUD operations**

- Apply reusable components to maintain consistency and reduce code duplication
- Maintain a scalable and easily extendable architecture

Technologies Used

Frontend Technologies

React.js

Used to build the entire user interface with a component-based approach that improves maintainability and reusability.

React Router

Handles client-side navigation without reloading pages, enabling a smooth user experience.

Tailwind CSS

Provides utility-first styling to ensure responsive, modern, and customizable UI components.

Axios

Used for making HTTP requests to communicate with backend APIs.

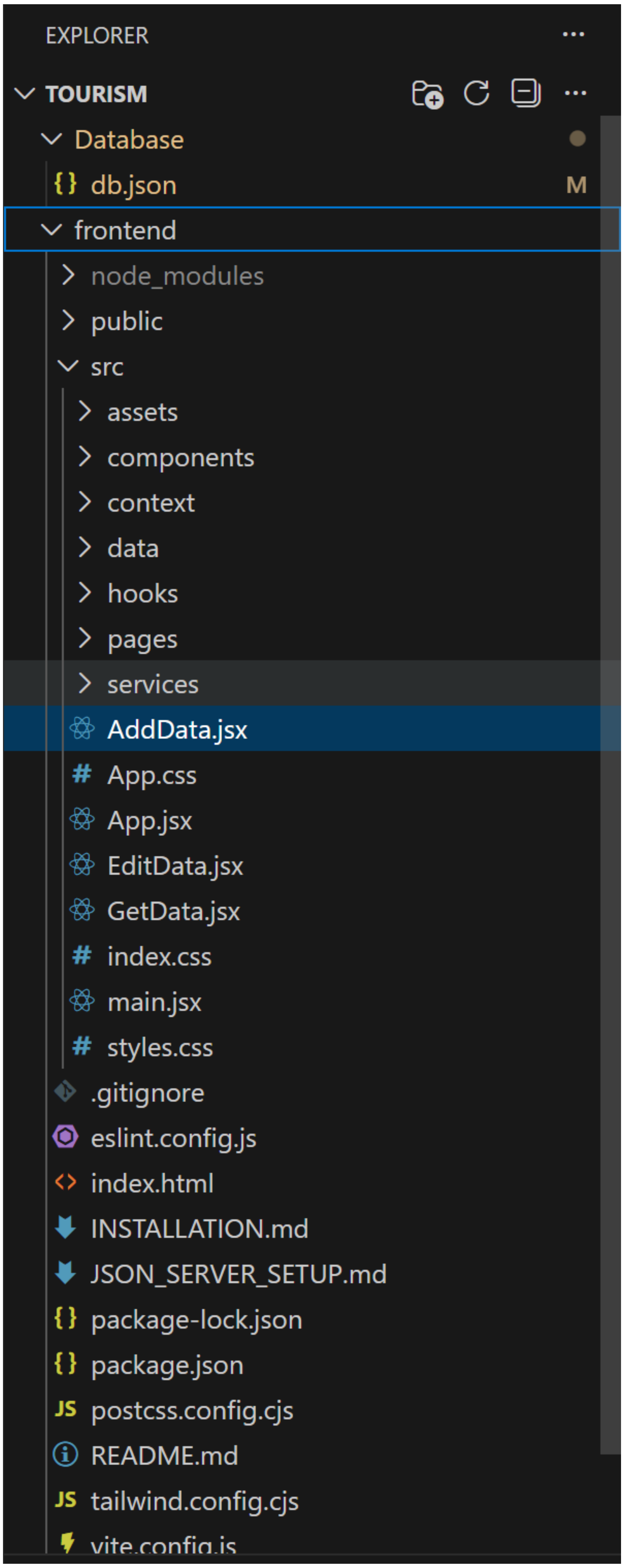
Backend Technology

- REST API for handling authentication, destinations, packages, and booking endpoints.
- Token-based authentication to secure admin routes.

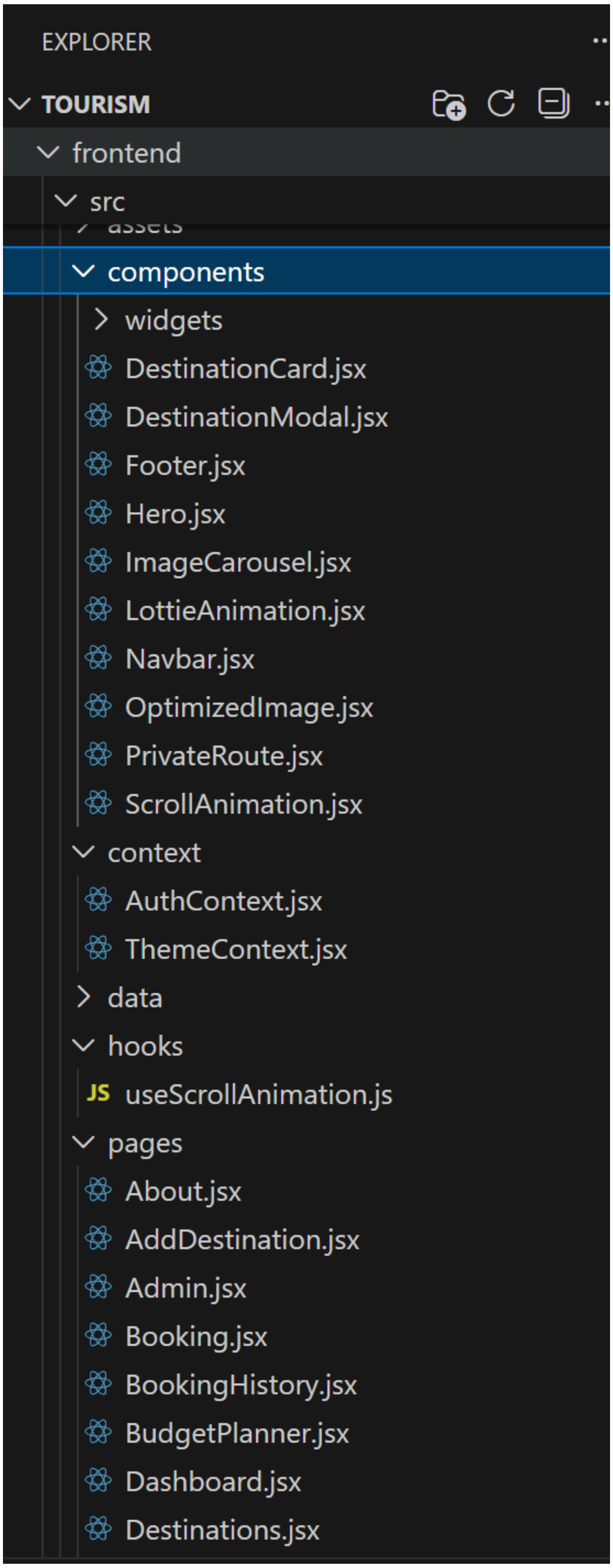
Tools

- **VS Code** – Code editor
- **Git / GitHub** – Version control and collaboration

Project Structure:



```
EXPLORER
TOURISM
  Database
    db.json                M
  frontend
    node_modules
    public
    src
      assets
      components
      context
      data
      hooks
      pages
      services
      AddData.jsx
      App.css
      App.jsx
      EditData.jsx
      GetData.jsx
      index.css
      main.jsx
      styles.css
    .gitignore
    eslint.config.js
    index.html
    INSTALLATION.md
    JSON_SERVER_SETUP.md
    package-lock.json
    package.json
    postcss.config.cjs
    README.md
    tailwind.config.cjs
    vite.config.js
```



```
EXPLORER
∨ TOURISM
  ∨ frontend
    ∨ src
      > assets
      ∨ components
        > widgets
        DestinationCard.jsx
        DestinationModal.jsx
        Footer.jsx
        Hero.jsx
        ImageCarousel.jsx
        LottieAnimation.jsx
        Navbar.jsx
        OptimizedImage.jsx
        PrivateRoute.jsx
        ScrollAnimation.jsx
      ∨ context
        AuthContext.jsx
        ThemeContext.jsx
      > data
      ∨ hooks
        JS useScrollAnimation.js
      ∨ pages
        About.jsx
        AddDestination.jsx
        Admin.jsx
        Booking.jsx
        BookingHistory.jsx
        BudgetPlanner.jsx
        Dashboard.jsx
        Destinations.jsx
```

System Architecture Overview

The system consists of the following major segments:

User Interface Layer

The UI layer interacts directly with the user and displays dynamic data. Key responsibilities include:

- Rendering destination and package lists
- Displaying individual details pages
- Handling login/register and user interaction
- Navigating using a clean, structured layout

Application Logic Layer

This layer processes all user actions such as:

- Form validation
- Conditional rendering
- Handling API results

Admin Module

The admin module provides:

- Restricted access to authorized users
- Tools for managing bookings
- CRUD functionality (Create, Read, Update, Delete)
- Protected pages using AdminGuard

Service Layer

A dedicated services folder contains:

- Axios configuration
- API request functions
- Token handling
- Error management
  This provides a centralized communication channel between frontend and backend.

Application Modules and Features

Reusable Components Module

All UI components in the system follow the React philosophy of reusability. This ensures a consistent look and feel across the application.

Components include:

- **Navbar.jsx** — Displays global navigation links
- **Footer.jsx** — Shown on every page for consistency
- **Layout.jsx** — Wraps page content with a common structure
- **DestinationCard.jsx & PackageCard.jsx** — Render dynamic content as reusable cards
- **AdminGuard.jsx** — Protects admin routes from unauthorized users

These components reduce code repetition and allow effortless expansion of the UI in the future.

Admin Module

The Admin Module is a core part of the project:

- **Bookings.jsx** displays all user bookings
- **EditBooking.jsx** allows modification of booking details
- CRUD operations make it easy to manage records
- Admin routes are protected by **AdminGuard.jsx**

This ensures complete control over the backend data from the admin interface.

Services Module

Located in **services/api.js**, this module manages:

- API calls using Axios
- Token-based authentication
- Error handling (e.g., expired tokens, failed requests)

Project Documentation:

Milestone 1: Project Setup and Configuration

1 . Installation of Required Tools and Software

To begin the project, the required tools and libraries are installed. The project is built using:

- React.js — For building the user interface.
- React Icons — For adding icons inside components.
- Material-UI (MUI) — For UI styling and layout.
- react-codemirror2 — For the interactive code editor functionality.

After creating/opening the project folder, the dependencies are installed.
Reference links used during setup:

- React Installation Guide: https://react.dev/learn/installation
- UI Component Setup Reference:
https://react-bootstrap-v4.netlify.app/getting-started/introduction/

Milestone 2: Web Development

1. Setting up the React Application

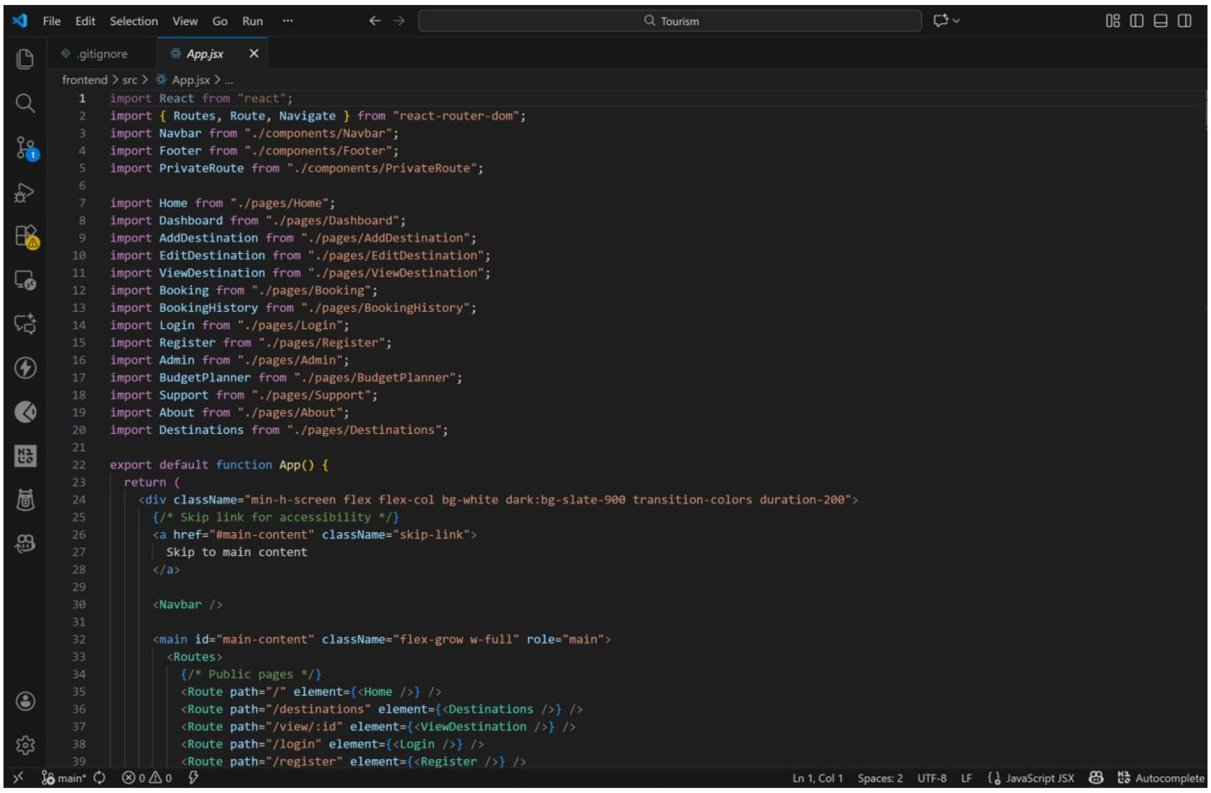
The React application is created and configured to support routing and UI components.

App.jsx Component Overview

- ./App.css — Provides styling for the App component.
- Home (./components/Home) — Main UI component of the application.
- DataProvider (./context/DataProvider) — Provides global state using React Context API.

The <DataProvider> wraps the <Home/> component so the entire application can access shared state values like HTML, CSS, and JavaScript code.

Finally, App is exported as the default component.



```jsx
import React from "react";
import { Routes, Route, Navigate } from "react-router-dom";
import Navbar from "./components/Navbar";
import Footer from "./components/Footer";
import PrivateRoute from "./components/PrivateRoute";

import Home from "./pages/Home";
import Dashboard from "./pages/Dashboard";
import AddDestination from "./pages/AddDestination";
import EditDestination from "./pages/EditDestination";
import ViewDestination from "./pages/ViewDestination";
import Booking from "./pages/Booking";
import BookingHistory from "./pages/BookingHistory";
import Login from "./pages/Login";
import Register from "./pages/Register";
import Admin from "./pages/Admin";
import BudgetPlanner from "./pages/BudgetPlanner";
import Support from "./pages/Support";
import About from "./pages/About";
import Destinations from "./pages/Destinations";

export default function App() {
  return (
    <div className="min-h-screen flex flex-col bg-white dark:bg-slate-900 transition-colors duration-200">
      {/* Skip link for accessibility */}
      <a href="#main-content" className="skip-link">
        Skip to main content
      </a>

      <Navbar />

      <main id="main-content" className="flex-grow w-full" role="main">
        <Routes>
          {/* Public pages */}
          <Route path="/" element={<Home />} />
          <Route path="/destinations" element={<Destinations />} />
          <Route path="/view/:id" element={<ViewDestination />} />
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
```

Description:

This is the main routing file (App.jsx) of the Tourist Management System.
It imports all pages, admin components, layout components, and authentication guards.
Using **React Router v6** , different routes are defined for Login, Register, Home, Destinations, Packages, Bookings, About, Contact, and Admin pages.

Each route uses the <Route> component, and most pages are wrapped inside the <Layout> component so that the navbar and footer appear consistently.
Admin routes are protected using AdminGuard, ensuring only authenticated admin users can access those pages.

2. Designing UI Components

During this phase:

- All required React components are created.
- Layout and styling are implemented using **Material-UI** and custom CSS.

- Navigation structure is defined as per the application flow.

Description:

This is the root HTML file of the project used by Vite.
It contains the basic HTML document structure with a <div id="root"></div> where the entire React application is rendered.

The <script type="module" src="/src/main.jsx"></script> line loads the React entry file, which initializes the application.

The <meta> tags define character encoding and enable responsive layout for all screen sizes.

3. Implementing Frontend Logic

Frontend logic includes:

- Integration of UI components with context state.
- API integration (if used in future expansion).
- Data binding with CodeMirror editors through React state and context.

DataProvider Component

Code Description

- A **DataContext** is created using createContext(null).
- The **DataProvider** component stores three state variables:

o

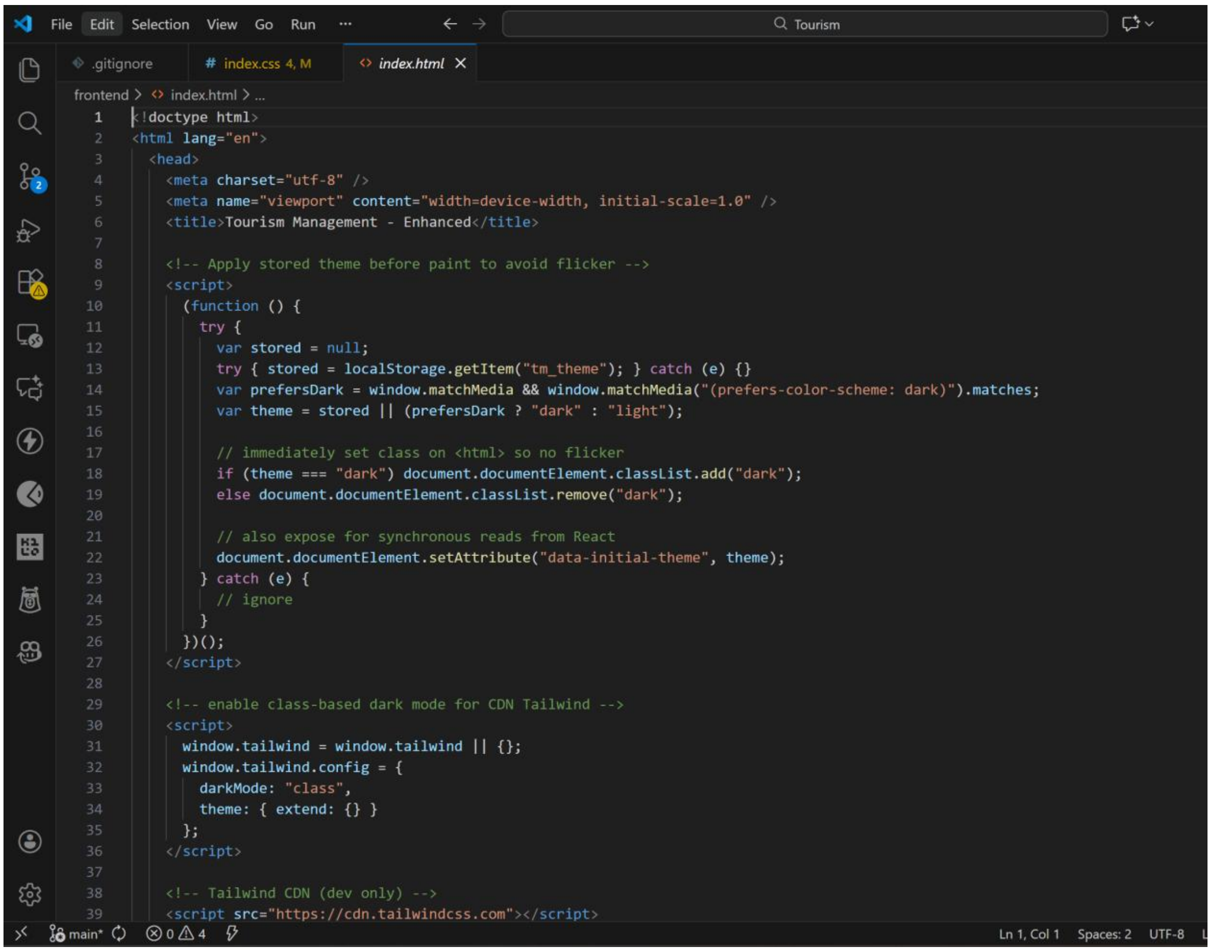
html o

css

o js

- These are managed using useState and shared with all nested components.
- DataContext.Provider returns these values and setter functions.
- Any child component can read or update the values using useContext(DataContext).

This enables synchronized editing and output generation for HTML, CSS, and JavaScript code throughout the app.



```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Tourism Management - Enhanced</title>

    <!-- Apply stored theme before paint to avoid flicker -->
    <script>
      (function () {
        try {
          var stored = null;
          try { stored = localStorage.getItem("tm_theme"); } catch (e) {}
          var prefersDark = window.matchMedia && window.matchMedia("(prefers-color-scheme: dark)").matches;
          var theme = stored || (prefersDark ? "dark" : "light");

          // immediately set class on <html> so no flicker
          if (theme === "dark") document.documentElement.classList.add("dark");
          else document.documentElement.classList.remove("dark");

          // also expose for synchronous reads from React
          document.documentElement.setAttribute("data-initial-theme", theme);
        } catch (e) {
          // ignore
        }
      })();
    </script>

    <!-- enable class-based dark mode for CDN Tailwind -->
    <script>
      window.tailwind = window.tailwind || {};
      window.tailwind.config = {
        darkMode: "class",
        theme: { extend: {} }
      };
    </script>

    <!-- Tailwind CDN (dev only) -->
    <script src="https://cdn.tailwindcss.com"></script>
```

Description:

This preview shows the same index.html file as it appears inside the editor. It confirms that the React app will mount inside the <div id="root">, and Vite will compile and serve the project using the entry file main.jsx.

This file is essential because it acts as the **single HTML page** for the entire React application (SPA — Single Page Application).

Code-Editor Component

Code Description

- The component uses useState to maintain editor open/close toggle.
- Imports:
  - o CodeMirror styles and modes o ControlledEditor from react-codemirror2 o Material-UI components (Box, Typography, Button, etc.) o Icons for editor heading and fullscreen toggle

Styled components include:

1. **Container** — Manages layout and spacing.
2. **Header** — Displays the editor title.
3. **Heading** — Shows the icon, title, and color styling.

The editor functionality includes:

- Syntax highlighting
- Linting
- Line numbers
- Theme setup

The handleChange function updates the code and passes it back to the parent using onChange.

Finally, the Editor component is exported for use inside the main Code component.

Project Execution

After writing all components, the application is run

using: npm start Or if using Vite: npm run dev

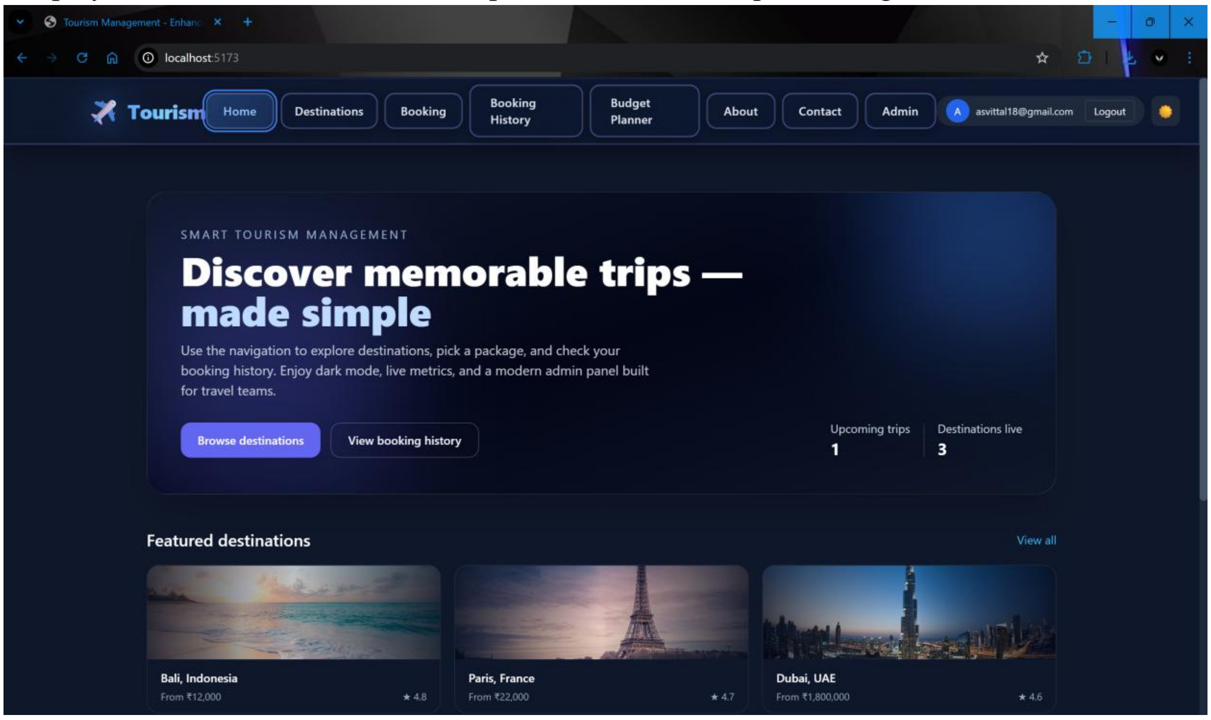
This starts the development server and launches the application.

Pages Module (Output screenshots):

This module contains all the user-facing screens:

Home Page

Displays featured destinations and important sections for quick navigation



SMART TOURISM MANAGEMENT

Discover memorable trips — made simple

Use the navigation to explore destinations, pick a package, and check your booking history. Enjoy dark mode, live metrics, and a modern admin panel built for travel teams.

Browse destinations | View booking history

Upcoming trips 1 | Destinations live 3

Featured destinations — View all

Bali, Indonesia — From ₹12,000 — ★ 4.8

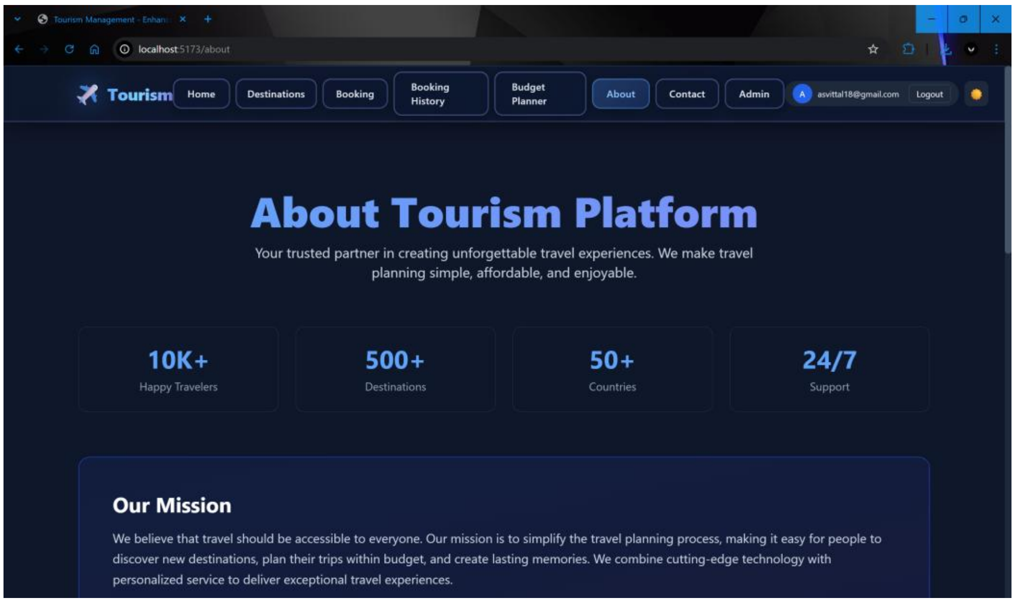
Paris, France — From ₹22,000 — ★ 4.7

Dubai, UAE — From ₹1,800,000 — ★ 4.6

.

About Page

Describes the purpose and functionality of the platform.



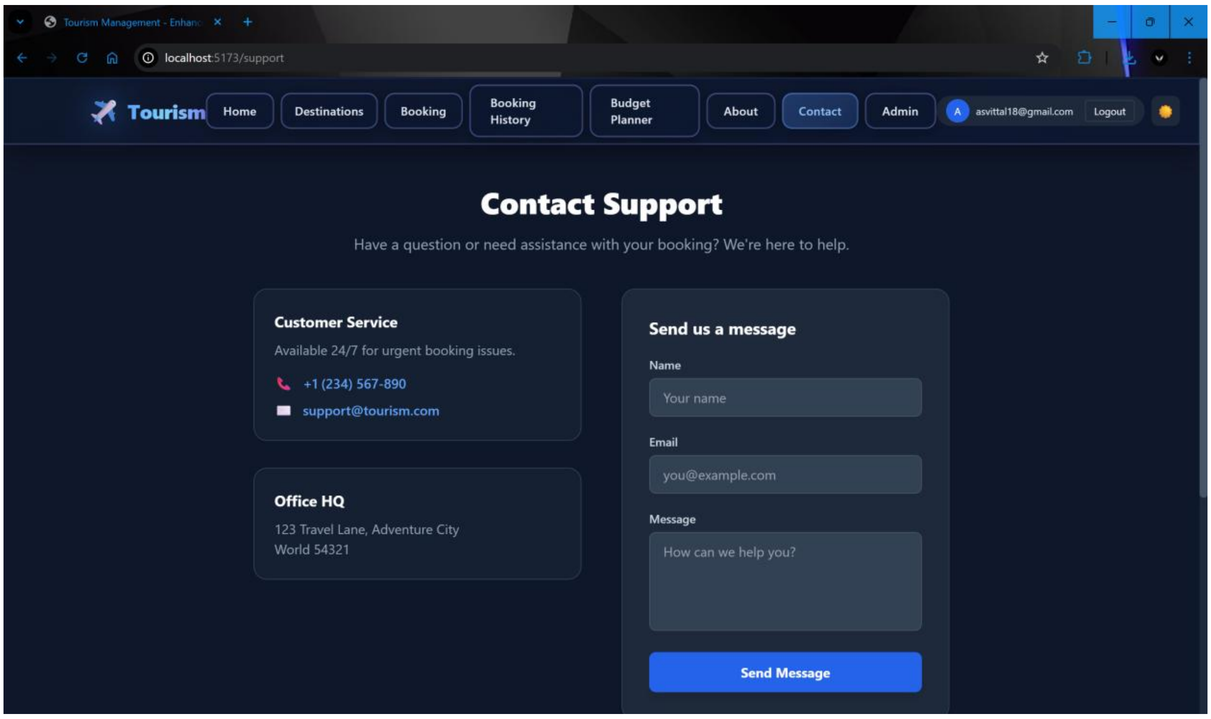
About Tourism Platform

Your trusted partner in creating unforgettable travel experiences. We make travel planning simple, affordable, and enjoyable.

10K+ Happy Travelers | 500+ Destinations | 50+ Countries | 24/7 Support

Our Mission

We believe that travel should be accessible to everyone. Our mission is to simplify the travel planning process, making it easy for people to discover new destinations, plan their trips within budget, and create lasting memories. We combine cutting-edge technology with personalized service to deliver exceptional travel experiences.



Tourism Management - Enhanc

localhost:5173/support

Tourism | Home | Destinations | Booking | Booking History | Budget Planner | About | Contact | Admin | asvittal18@gmail.com | Logout

Contact Support

Have a question or need assistance with your booking? We're here to help.

Customer Service

Available 24/7 for urgent booking issues.

+1 (234) 567-890

support@tourism.com

Office HQ

123 Travel Lane, Adventure City
World 54321

Send us a message

Name: Your name

Email: you@example.com

Message: How can we help you?

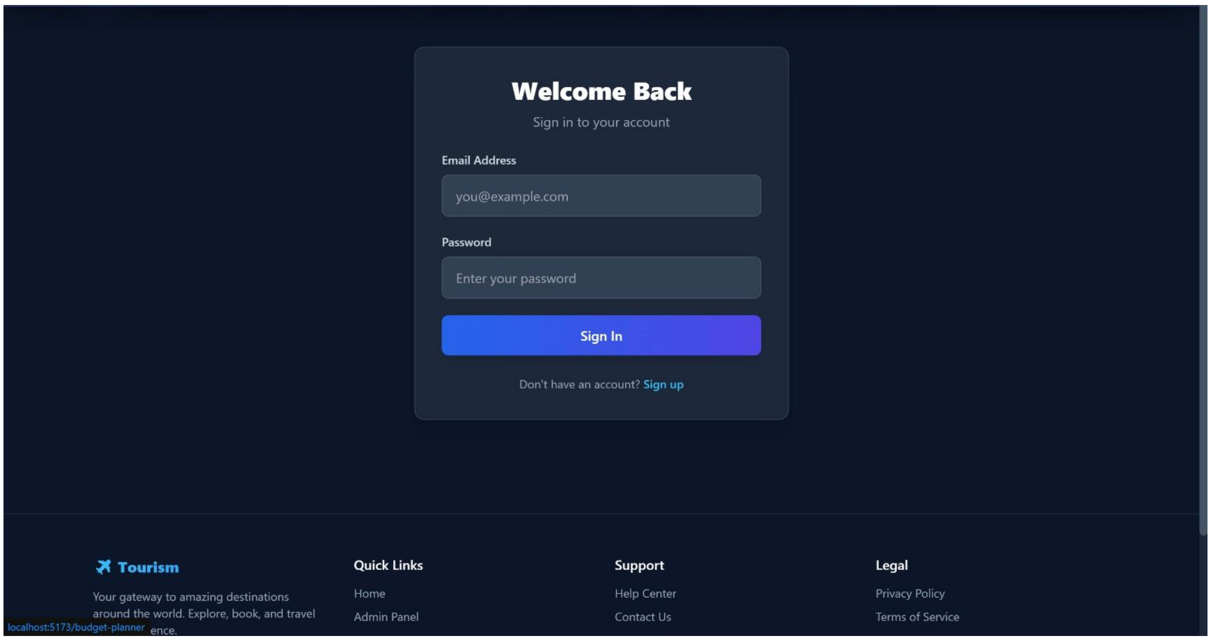
Send Message

Contact Page

Login & Register Pages

Contain authentication forms with frontend validation.

Successful login generates a secure token for protected navigation.



Welcome Back

Sign in to your account

Email Address: you@example.com

Password: Enter your password

Sign In

Don't have an account? Sign up

Tourism

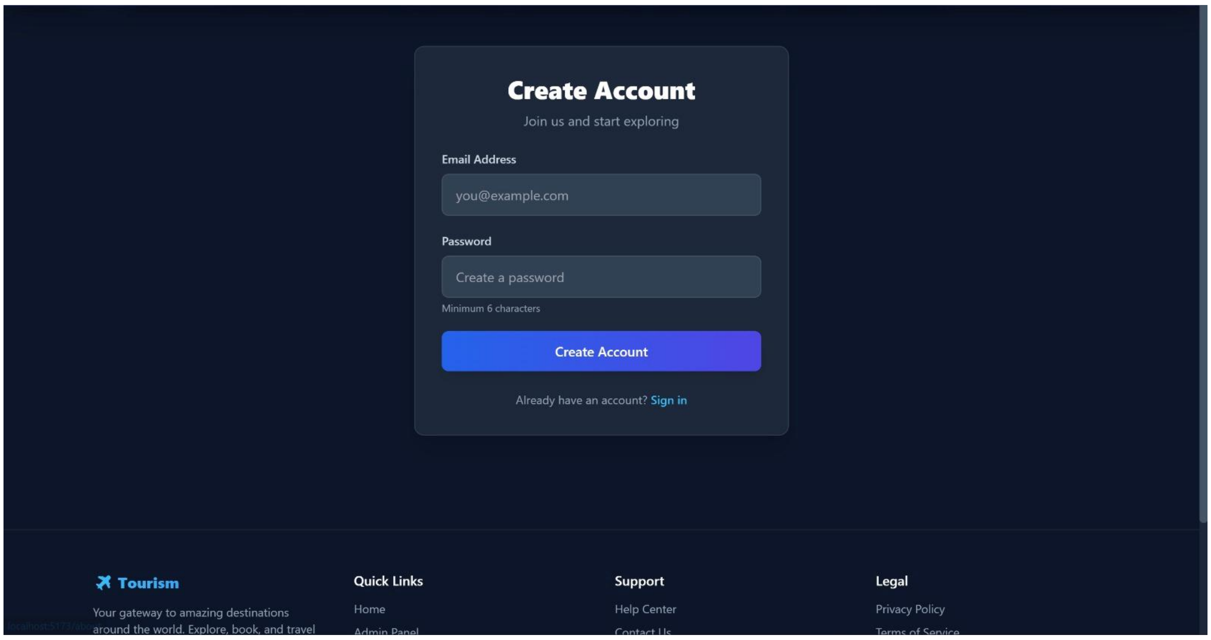
Your gateway to amazing destinations around the world. Explore, book, and travel ence.

Quick Links: Home, Admin Panel

Support: Help Center, Contact Us

Legal: Privacy Policy, Terms of Service

localhost:5173/budget-planner



Create Account

Join us and start exploring

Email Address: you@example.com

Password: Create a password

Minimum 6 characters

Create Account

Already have an account? Sign in

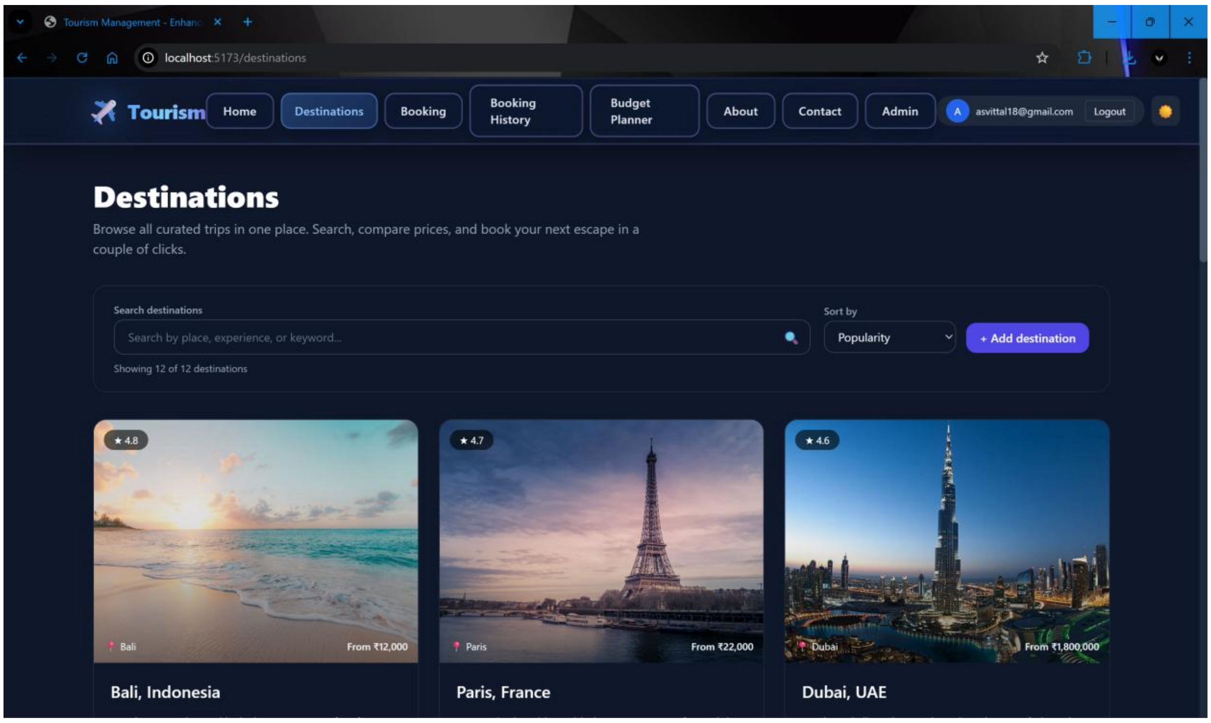
Tourism

Your gateway to amazing destinations around the world. Explore, book, and travel

Quick Links	Support	Legal
Home	Help Center	Privacy Policy
Admin Panel	Contact Us	Terms of Service

Destinations Page

Lists all available travel places using dynamic cards.



Tourism Management - Enhanc

localhost:5173/destinations

Tourism | Home | Destinations | Booking | Booking History | Budget Planner | About | Contact | Admin | asvittal18@gmail.com | Logout

Destinations

Browse all curated trips in one place. Search, compare prices, and book your next escape in a couple of clicks.

Search destinations: Search by place, experience, or keyword...

Sort by: Popularity

+ Add destination

Showing 12 of 12 destinations

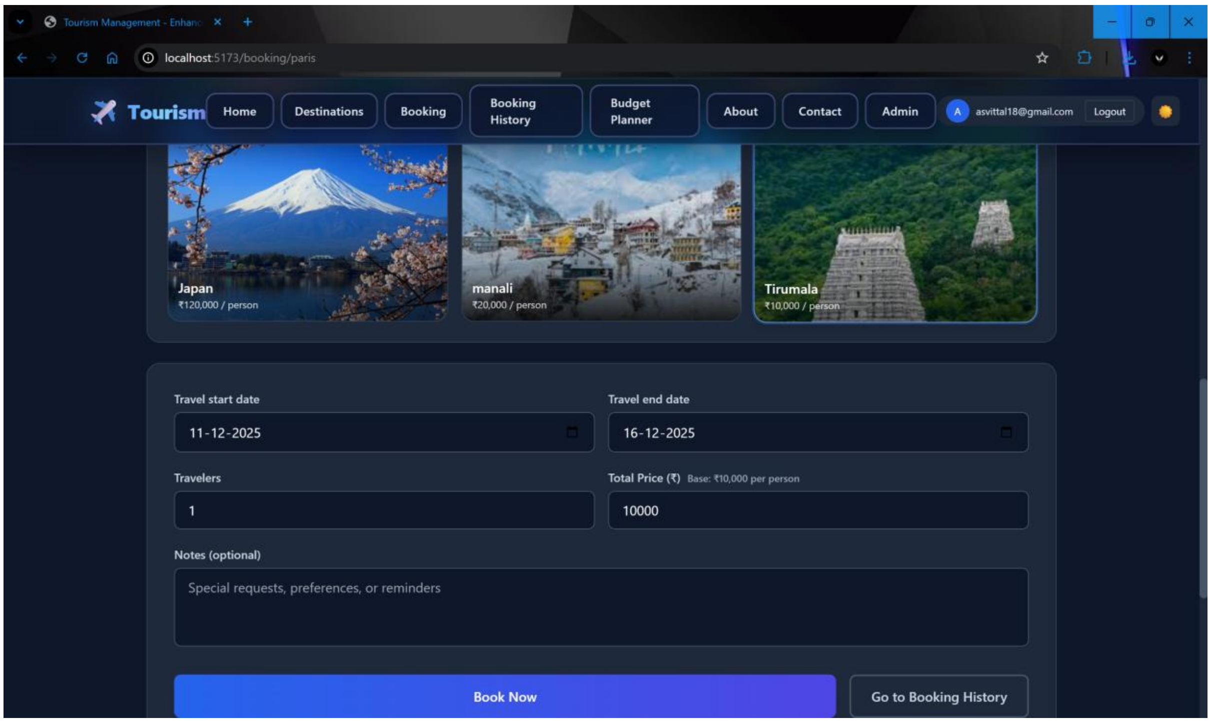
★ 4.8 Bali From ₹12,000 — Bali, Indonesia

★ 4.7 Paris From ₹22,000 — Paris, France

★ 4.6 Dubai From ₹1,800,000 — Dubai, UAE

Destination Details Page

A detailed view of a selected destination with images, descriptions, and pricing.



Travel start date: 11-12-2025

Travel end date: 16-12-2025

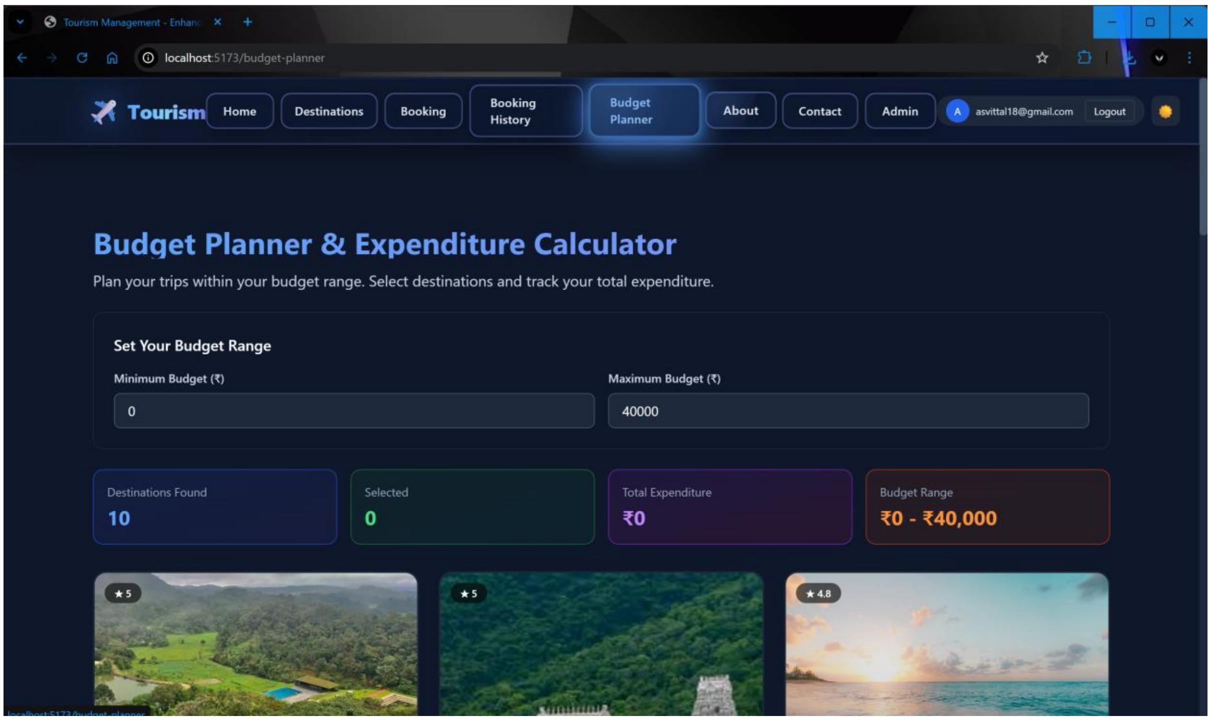
Travelers: 1

Total Price (₹) Base: ₹10,000 per person: 10000

Notes (optional): Special requests, preferences, or reminders

Packages Page

Shows available travel packages with key highlights.



Budget Planner & Expenditure Calculator

Plan your trips within your budget range. Select destinations and track your total expenditure.

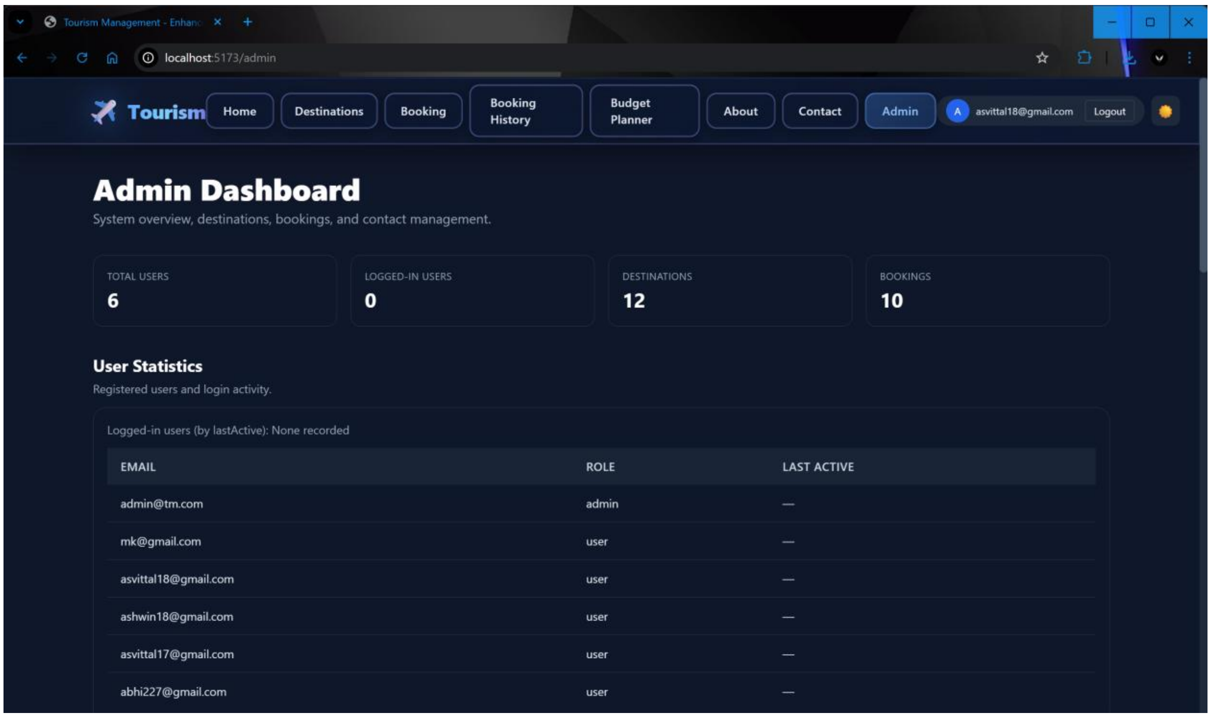
Set Your Budget Range

Minimum Budget (₹): 0

Maximum Budget (₹): 40000

Destinations Found	Selected	Total Expenditure	Budget Range
10	0	₹0	₹0 - ₹40,000

Admin Page :



Admin Dashboard

System overview, destinations, bookings, and contact management.

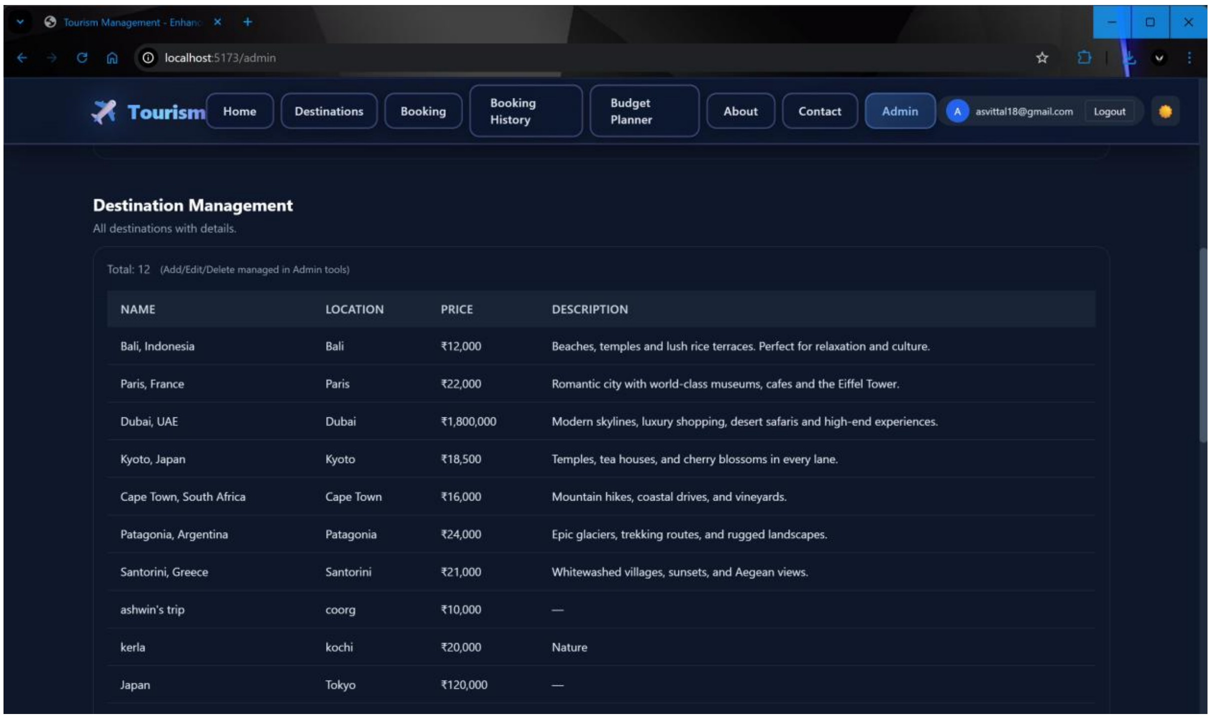
TOTAL USERS	LOGGED-IN USERS	DESTINATIONS	BOOKINGS
6	0	12	10

User Statistics

Registered users and login activity.

Logged-in users (by lastActive): None recorded

EMAIL	ROLE	LAST ACTIVE
admin@tm.com	admin	—
mk@gmail.com	user	—
asvittal18@gmail.com	user	—
ashwin18@gmail.com	user	—
asvittal17@gmail.com	user	—
abhi227@gmail.com	user	—

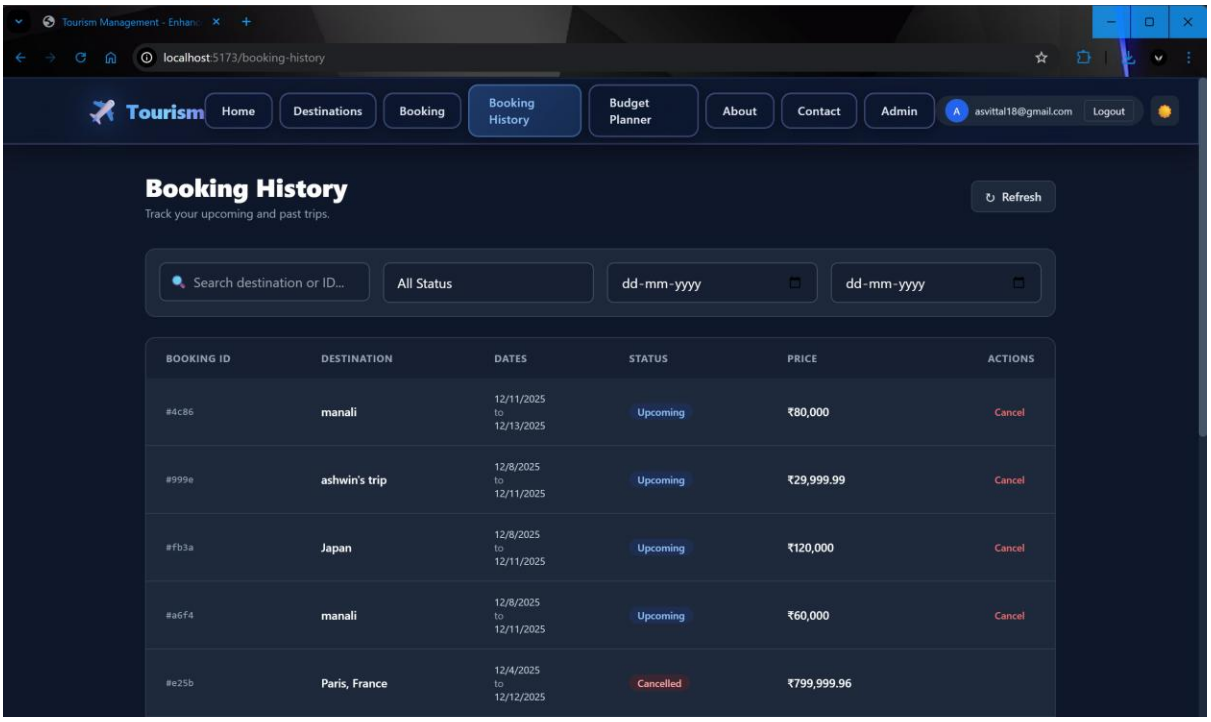


Destination Management

All destinations with details.

Total: 12 (Add/Edit/Delete managed in Admin tools)

NAME	LOCATION	PRICE	DESCRIPTION
Bali, Indonesia	Bali	₹12,000	Beaches, temples and lush rice terraces. Perfect for relaxation and culture.
Paris, France	Paris	₹22,000	Romantic city with world-class museums, cafes and the Eiffel Tower.
Dubai, UAE	Dubai	₹1,800,000	Modern skylines, luxury shopping, desert safaris and high-end experiences.
Kyoto, Japan	Kyoto	₹18,500	Temples, tea houses, and cherry blossoms in every lane.
Cape Town, South Africa	Cape Town	₹16,000	Mountain hikes, coastal drives, and vineyards.
Patagonia, Argentina	Patagonia	₹24,000	Epic glaciers, trekking routes, and rugged landscapes.
Santorini, Greece	Santorini	₹21,000	Whitewashed villages, sunsets, and Aegean views.
ashwin's trip	coorg	₹10,000	—
kerla	kochi	₹20,000	Nature
Japan	Tokyo	₹120,000	—



Booking History

Track your upcoming and past trips.

BOOKING ID	DESTINATION	DATES	STATUS	PRICE	ACTIONS
#4c86	manali	12/11/2025 to 12/13/2025	Upcoming	₹80,000	Cancel
#999e	ashwin's trip	12/8/2025 to 12/11/2025	Upcoming	₹29,999.99	Cancel
#fb3a	Japan	12/8/2025 to 12/11/2025	Upcoming	₹120,000	Cancel
#a6f4	manali	12/8/2025 to 12/11/2025	Upcoming	₹60,000	Cancel
#e25b	Paris, France	12/4/2025 to 12/12/2025	Cancelled	₹799,999.96	

Future Enhancements

To improve the scalability, usability, and overall user experience of the Travel Booking Web Application, the following future enhancements can be implemented:

1. Online Payment Integration

Add payment gateways such as Razorpay, Stripe, or PayPal to allow users to complete bookings with secure online payments.

2. Advanced Search and Filtering

Enable users to filter destinations and packages by price, location, duration, season, ratings, etc., for a more customized browsing experience.

3. User Profile & Booking History

Allow users to manage their profiles, update personal details, and view their previous booking history.

4. Wishlist / Save for Later

Users can save destinations or packages they are interested in and view them later.

5. Reviews and Ratings

Enable users to provide feedback, ratings, and comments for destinations and packages, improving transparency and helping new customers decide.

6. Admin Analytics Dashboard

Provide admins with visual analytics such as charts for:

- Most booked destinations
- Monthly booking reports
- User growth statistics

7. Email & SMS Notifications

Send automated booking confirmation, cancellation, or reminder messages to users.

8. Multi-language Support

Include different language options so users from various regions can use the application easily.

9. Responsive Mobile App Version

Develop a cross-platform mobile application using React Native to extend accessibility beyond the web browser.

10. AI-based Recommendation System

Recommend destinations or packages based on user preferences, browsing history, and trending travel packages.

11. Chatbot Integration

Integrate a chatbot for answering user queries in real-time, improving customer support.

12. Multi-role User System Add more roles such as:

- Travel Agent
- Manager
- Customer Support

Each with different permissions.

Extended Key Features

- Responsive design ensures compatibility across laptops, tablets, and mobile phones
- Dynamic routing for seamless navigation
- Centralized component-based architecture
- Secure authentication with protected admin routes
- Detailed views for destinations and packages
- Integrated CRUD operations for booking management
- Clean visual design using Tailwind CSS spacing, colors, and layouts
- Organized folder structure for easy scalability

Conclusion

The Travel Booking Web Application provides a streamlined and engaging solution for exploring travel destinations and managing bookings.
It eliminates complexities associated with traditional travel booking by offering a fully digital, responsive, and intuitive interface.

The project demonstrates strong frontend development skills, including routing, state handling, UI design, and secure API integration.
The architecture is scalable and adaptable for future enhancements, such as:

- Adding advanced search & filtering
- Integrating a payment gateway
- Allowing user profiles and booking history
- Adding reviews and ratings
- Admin analytics (charts, dashboards)

The project fulfills its objectives effectively and serves as a practical example of building a real-world web application using React.js.