**HABIT TRACKER**

By

**K.Midhun Sai – AP23110010387**

**M.Abhigna – AP23110011051**

**V.Nikitha – AP23110011062**

**K.Sarayu – AP23110010707**

SUBMITTED TO:

**Himanshu Mishra**

****

**SRM UNIVERSITY, AP**

(DECEMBER, 2025)

**Introduction: -**

The **Tourism Management System** is a modern full-stack web application designed to streamline the tourism booking experience by providing an intuitive platform for exploring destinations, booking tours, and managing travel plans. This project addresses the growing demand for digital transformation in the tourism industry, where customers expect instant access to tour information, real-time booking capabilities, and personalized user experiences.

The system serves as a complete solution for tourism businesses, enabling them to showcase tour packages, manage customer bookings, and provide seamless user experiences through a responsive web interface. Built using cutting-edge technologies including React.js, Vite, Tailwind CSS, and JSON Server, the application implements role-based access control with two distinct user types: Administrators with full system management capabilities, and Regular Users who can browse and book tours.

Key features include secure user authentication, dynamic tour catalog with detailed information, real-time booking system with instant confirmation, comprehensive admin dashboard for managing tours and bookings, user profile management with booking history, and a mobile-first responsive design ensuring optimal viewing across all devices.

The project follows a client-server architecture with clear separation of concerns, utilizing React-based frontend, RESTful API endpoints, and JSON-based data storage. This Tourism Management System demonstrates proficiency in modern web development practices, combining contemporary JavaScript frameworks, responsive design principles, and user-centric design to deliver a professional-grade application suitable for real-world deployment.

**Scenario-Based Intro: -**

In today's fast-paced digital world, planning a vacation has become increasingly complex. Imagine a family eager to explore exotic destinations but overwhelmed by scattered information across multiple websites, complicated booking processes, and lack of transparency in pricing and availability. Traditional tourism agencies often operate with outdated systems, requiring customers to visit physical offices, make phone calls, or navigate through clunky websites that offer poor user experiences. Meanwhile, tourism business owners struggle with manual booking management, paper-based records, and limited visibility into their operations, leading to inefficiencies, double bookings, and lost revenue opportunities.

Consider a scenario where a young professional wants to book a weekend getaway to Bali. They spend hours browsing different websites, comparing prices, reading reviews scattered across platforms, and eventually abandoning the booking process due to frustration. On the other side, a tour operator managing multiple packages manually tracks bookings in spreadsheets, leading to errors and missed opportunities. This disconnect between customer expectations and service delivery highlights the urgent need for a comprehensive digital solution.

The **Tourism Management System** addresses these real-world challenges by providing a unified, user-friendly platform that bridges the gap between travelers and tourism service providers. This web application transforms the entire tourism booking ecosystem by offering travelers a seamless experience to discover, compare, and book tours instantly, while empowering administrators with powerful tools to manage their operations efficiently.

Built using modern technologies including React.js, Vite, Tailwind CSS, and JSON Server, the system features secure user authentication, a dynamic tour catalog with comprehensive details, real-time booking capabilities with instant confirmation, and an intuitive admin dashboard for complete business management. The application implements role-based access control, ensuring that regular users can browse and book tours effortlessly, while administrators maintain full control over tour packages, bookings, and customer data.

By eliminating the friction in traditional tourism booking processes and providing a responsive, mobile-first design accessible from any device, this Tourism Management System represents a practical solution to contemporary challenges in the tourism industry, demonstrating how modern web technologies can create meaningful impact in real-world business scenarios.

**Target Audience: -**

**Travel Enthusiasts & Adventure Seekers**

* **Individuals passionate about exploring new destinations, cultures, and experiences.**
* **Travelers looking for curated tour packages with detailed information, ratings, and reviews.**
* **People who prefer hassle-free online booking with instant confirmation and transparent pricing.**

**Families & Group Travelers**

* **Families planning vacations who need comprehensive tour details including duration, pricing, and itineraries.**
* **Groups seeking organized travel experiences with reliable booking systems and clear communication.**
* **Users who value visual tour galleries and detailed destination information before making decisions.**

**Young Professionals & Digital Nomads**

* **Busy working professionals seeking quick, convenient online booking without visiting physical agencies.**
* **Tech-savvy travelers who prefer mobile-responsive platforms accessible from any device.**
* **Individuals who appreciate modern UI/UX with smooth navigation and real-time updates.**

**Tourism Business Owners & Operators**

* **Tour operators and travel agencies needing efficient digital platforms to showcase their packages.**
* **Business owners requiring comprehensive admin dashboards to manage tours, bookings, and customers.**
* **Entrepreneurs looking for scalable solutions to streamline operations and reduce manual processes.**

**Students & Budget Travelers**

* **Students and young adults searching for affordable tour packages with clear pricing information.**
* **Budget-conscious travelers who want to compare multiple options and make informed decisions.**
* **Users who benefit from organized tour catalogs with filtering and search capabilities.**

**Project Goals and Objectives: -**

**Primary Goal**

**To develop a comprehensive, user-friendly Tourism Management System that digitizes and streamlines the entire tour booking process, providing seamless experiences for travelers while empowering tourism businesses with efficient management tools.**

**Specific Objectives**

**1. User Experience & Accessibility**

* **Create an intuitive, responsive web interface accessible across all devices (desktop, tablet, mobile).**
* **Implement smooth navigation with React Router for seamless page transitions without reloads.**
* **Design a visually appealing UI using Tailwind CSS with modern aesthetics and animations.**
* **Ensure fast loading times and optimal performance through Vite build optimization.**

**2. Authentication & Security**

* **Develop a secure user registration and login system with password protection.**
* **Implement role-based access control (RBAC) distinguishing between Admin and User privileges.**
* **Maintain user session management for persistent authentication across pages.**
* **Protect sensitive routes and ensure only authorized users can access admin features.**

**3. Tour Management & Discovery**

* **Build a comprehensive tour catalog showcasing packages with detailed information (pricing, duration, location, ratings, reviews).**
* **Enable dynamic tour browsing with search and filtering capabilities.**
* **Provide detailed tour pages with high-quality images and complete itinerary information.**
* **Create an interactive gallery for visual exploration of destinations.**

**4. Booking System**

* **Implement real-time booking functionality with instant confirmation.**
* **Develop a booking history system for users to track their reservations.**
* **Enable booking status management (Confirmed, Pending, Cancelled).**
* **Provide seamless booking flow from tour selection to confirmation.**

**5. Administrative Control**

* **Design a comprehensive admin dashboard with system overview and analytics.**
* **Implement full CRUD operations for tour package management (Create, Read, Update, Delete).**
* **Enable booking management with status updates and customer information.**
* **Provide user management capabilities for monitoring registered users.**

**6. Technical Excellence**

* **Utilize modern web technologies (React.js, Vite, Tailwind CSS) following industry best practices.**
* **Implement RESTful API architecture using JSON Server for scalable backend operations.**
* **Ensure clean, maintainable, and well-documented code structure.**
* **Apply component-based architecture for reusability and modularity.**

**7. Data Management**

* **Establish reliable data persistence using JSON-based storage.**
* **Implement efficient API communication using Axios for HTTP requests.**
* **Ensure data consistency and integrity across all operations.**
* **Enable real-time data updates without page refreshes.**

**8. User Engagement & Feedback**

* **Integrate notification system using React Toastify for user feedback on actions.**
* **Implement smooth animations and transitions using Framer Motion for enhanced UX.**
* **Provide clear visual indicators for loading states, errors, and success messages.**
* **Create an engaging user interface that encourages exploration and interaction.**

**Expected Outcomes**

* **A fully functional, production-ready tourism management web application.**
* **Improved efficiency in tour booking processes for both customers and businesses.**
* **Scalable architecture ready for future enhancements and feature additions.**
* **Demonstration of proficiency in modern full-stack web development practices.**
* **A practical solution addressing real-world challenges in the tourism industry.**

**Key Features:-**

**1. User Authentication & Authorization**

* **Secure Registration System: New users can create accounts with username, email, password, phone, and address information.**
* **Login/Logout Functionality: Secure authentication with session management and persistent login states.**
* **Role-Based Access Control (RBAC): Two distinct user roles - Admin and User - with different access privileges.**
* **Protected Routes: Restricted access to admin features and user-specific pages based on authentication status.**

**2. Tour Catalog & Discovery**

* **Comprehensive Tour Listings: Display of tour packages with title, description, pricing, duration, and location.**
* **Tour Details Page: Detailed view of each tour including high-quality images, ratings, reviews, and complete information.**
* **Search & Filter Capabilities: Easy discovery of tours based on preferences and requirements.**
* **Rating & Review System: Display of user ratings and review counts for informed decision-making.**
* **Dynamic Tour Cards: Interactive tour cards with hover effects and quick access to details.**

**3. Booking Management System**

* **Real-Time Booking: Instant tour booking with immediate confirmation.**
* **Booking History: Users can view all their past and current bookings in one place.**
* **Booking Status Tracking: Clear status indicators (Confirmed, Pending, Cancelled) for each booking.**
* **Booking Details: Complete information including tour title, date, price, and booking ID.**

**4. Admin Dashboard**

* **Centralized Control Panel: Comprehensive dashboard for system overview and management.**
* **Tour Management (CRUD Operations):**
  + **Create new tour packages with all details**
  + **View all existing tours**
  + **Update/Edit tour information**
  + **Delete tours from the system**
* **Booking Management: View and manage all customer bookings with status updates.**
* **User Management: Monitor registered users and their information.**
* **Analytics Overview: Quick insights into total tours, bookings, and users.**

**5. User Profile Management**

* **Personal Profile Page: View and manage user account information.**
* **Profile Editing: Update personal details including username, email, phone, and address.**
* **Booking History Access: Quick access to all user bookings from the profile.**
* **Account Settings: Manage account preferences and information.**

**6. Interactive Gallery**

* **Visual Showcase: Beautiful gallery displaying tourism destinations and experiences.**
* **Image Grid Layout: Responsive grid showcasing high-quality destination images.**
* **Engaging Visuals: Attractive presentation to inspire travel decisions.**

**7. Responsive Design & UI/UX**

* **Mobile-First Approach: Fully responsive design optimized for all screen sizes.**
* **Modern Interface: Clean, professional UI built with Tailwind CSS.**
* **Smooth Animations: Engaging transitions and effects using Framer Motion.**
* **Intuitive Navigation: Easy-to-use navigation bar with clear menu structure.**
* **Interactive Elements: Hover effects, loading states, and visual feedback.**

**8. Notification System**

* **Real-Time Feedback: Toast notifications for all user actions (success, error, info).**
* **Action Confirmations: Immediate feedback for login, registration, booking, and CRUD operations.**
* **Error Handling: Clear error messages for failed operations or validation issues.**

**9. Home Page Features**

* **Hero Section: Eye-catching landing section with call-to-action buttons.**
* **Featured Tours: Showcase of popular and recommended tour packages.**
* **Quick Access: Direct links to tours, gallery, and booking sections.**
* **Engaging Content: Compelling visuals and descriptions to attract users.**

**10. Technical Features**

* **RESTful API Integration: Seamless communication with backend using Axios.**
* **State Management: Efficient state handling using React Context API.**
* **Component-Based Architecture: Modular, reusable components for maintainability.**
* **Fast Performance: Optimized build and loading times with Vite.**
* **Clean Code Structure: Well-organized file structure and coding standards.**
* **Error Boundaries: Graceful error handling throughout the application.**

**11. Data Persistence**

* **JSON Server Backend: RESTful API endpoints for data operations.**
* **Real-Time Updates: Automatic data synchronization without page refreshes.**
* **Data Validation: Input validation for forms and user data.**
* **Consistent Data Structure: Standardized data models for users, tours, and bookings.**

**Technical Requirements:**

**Software & Tools:**

1. **Node.js (v14 or higher) - JavaScript runtime environment for running the application**
2. **npm (v6 or higher) or yarn - Package manager for installing project dependencies**
3. **Modern Web Browser - Chrome, Firefox, Safari, or Edge (latest versions recommended)**
4. **Text Editor/IDE - VS Code, Sublime Text, or any preferred code editor**
5. **Git (Optional) - Version control system for code management**
6. **JSON Server - Installed via npm for backend API simulation**

**System Requirements:**

* **Operating System: Windows 10/11, macOS, or Linux**
* **RAM: Minimum 4GB (8GB recommended)**
* **Storage: At least 500MB free disk space**
* **Internet Connection: Required for downloading dependencies and accessing external resources**

**Knowledge Prerequisites:**

**1. JavaScript ES6+ Fundamentals:**

* **Arrow functions and function expressions**
* **Destructuring (objects and arrays)**
* **Spread and rest operators**
* **Template literals**
* **Async/await and Promises**
* **Array methods (map, filter, find, reduce)**
* **Modules (import/export)**

**2. React.js Basics:**

* **Component structure (functional components)**
* **Props and state management**
* **React Hooks (useState, useEffect, useContext)**
* **Event handling in React**
* **Conditional rendering**
* **List rendering and keys**
* **Component lifecycle concepts**

**3. HTML & CSS:**

* **Semantic HTML5 elements**
* **CSS fundamentals (selectors, box model, flexbox, grid)**
* **Responsive design principles**
* **Basic understanding of CSS frameworks (Tailwind CSS)**

**4. REST API Concepts:**

* **Understanding of HTTP methods (GET, POST, PUT, DELETE)**
* **Request and response structure**
* **Status codes (200, 404, 500, etc.)**
* **JSON data format**
* **API endpoints and routing**

**5. React Router:**

* **Client-side routing concepts**
* **Navigation between pages**
* **Route parameters and dynamic routes**
* **Protected routes and authentication flows**

**6. Command Line Basics:**

* **Navigating directories (cd, ls/dir)**
* **Running npm/yarn commands**
* **Installing packages**
* **Starting and stopping development servers**

**Project structure:**

**tourism-management-system/**

**│**

**├── public/ # Static assets**

**│ └── (public files)**

**│**

**├── src/ # Source code directory**

**│ ├── assets/ # Images, icons, and media files**

**│ │**

**│ ├── components/ # Reusable React components**

**│ │ ├── Layout/ # Layout components**

**│ │ │ ├── Navbar.jsx # Navigation bar component**

**│ │ │ ├── Footer.jsx # Footer component**

**│ │ │ └── Logo.jsx # Logo component**

**│ │ │**

**│ │ └── Tours/ # Tour-related components**

**│ │ ├── TourCard.jsx # Individual tour card component**

**│ │ └── TourForm.jsx # Tour creation/editing form**

**│ │**

**│ ├── context/ # React Context for state management**

**│ │ ├── AuthContext.jsx # Authentication context provider**

**│ │ └── (other context files)**

**│ │**

**│ ├── pages/ # Page components (routes)**

**│ │ ├── Home.jsx # Homepage**

**│ │ ├── Login.jsx # Login page**

**│ │ ├── Register.jsx # Registration page**

**│ │ ├── Tours.jsx # Tours listing page**

**│ │ ├── TourDetails.jsx # Individual tour details page**

**│ │ ├── Dashboard.jsx # Admin dashboard**

**│ │ ├── Profile.jsx # User profile page**

**│ │ └── Gallery.jsx # Gallery page**

**│ │**

**│ ├── services/ # API service files**

**│ │ └── api.js # Axios API configuration and calls**

**│ │**

**│ ├── App.jsx # Main application component**

**│ ├── main.jsx # Application entry point**

**│ └── index.css # Global styles and Tailwind imports**

**│**

**├── db.json # JSON Server database**

**│**

**├── node\_modules/ # Project dependencies (auto-generated)**

**│**

**├── .gitignore # Git ignore file**

**├── ACCESS\_CONTROL\_GUIDE.md # Access control documentation**

**├── README.md # Project documentation**

**├── package.json # Project metadata and dependencies**

**├── package-lock.json # Dependency lock file**

**├── vite.config.js # Vite configuration**

**├── tailwind.config.js # Tailwind CSS configuration**

**├── postcss.config.cjs # PostCSS configuration**

**├── eslint.config.js # ESLint configuration**

**└── index.html # HTML entry point**

**THE PROJECT CONTAINS:**

**• Frontend (React.js)**

Contains all UI screens, components, and client-side logic.

**• Components:** Navbar, Footer, Logo, TourCard, TourForm, Toast Notifications, Protected Routes, Loading Indicators.

**• Pages:** Home, Tours, TourDetails, Login, Register, Dashboard, Profile, Gallery.

**• State Management:** Handles global and shared states such as user authentication, tour lists, booking data, and session management using React Context API.

**• Styles:** Global CSS, Tailwind CSS utility classes, Framer Motion animations, component-level styling.

**• Utils:** API helper methods (Axios), authentication utilities, data formatting functions, error handling, form validation.

**• Backend (JSON Server)**

**• db.json stores all tourism data** Contains fields such as:

* **Users:** id, username, email, password, role, phone, address
* **Tours:** id, title, description, price, duration, location, image, rating, reviews
* **Bookings:** id, userId, tourId, tourTitle, date, status, price

**• JSON Server handles all CRUD operations** No MySQL or PHP required.

**Supports:**

* Create users, tours, and bookings
* Read all records or by ID
* Update tour details, user profiles, and booking statuses
* Delete tours and bookings
* Filter and search data

**• API (Fetch / Axios)**

Used for communicating with JSON Server.

**Performs:**

* GET all tours, users, and bookings
* GET single record by ID
* POST new users, tours, and bookings
* PUT/PATCH updates for tours, profiles, and bookings
* DELETE tours and bookings

**• Assets (Media Files)**

Includes tour images, icons (React Icons library), UI graphics, backgrounds, and logo used across the app.

**• Configuration Files**

**• package.json:** Project dependencies and scripts.

**• vite.config.js:** Vite development and production configuration.

**• tailwind.config.js:** Tailwind CSS customization settings.

**• postcss.config.cjs:** PostCSS processing configuration.

**• eslint.config.js:** Code quality and linting rules.

**• db.json:** Local database file for users, tours, and bookings.

**• .gitignore:** Specifies ignored files (node\_modules, dist, logs).

**• index.html:** Root HTML file where the React app is mounted.

**PROJECT FLOW: -**

**1. User Opens the Application**

**The user lands on the Home Page, which displays a hero section with featured tours and call-to-action buttons.**

**2. Home Page Loads All Tours**

* **React sends a GET request to JSON Server (**

**/tours**

**).**

* **Tour cards are displayed with:**
  + **Tour image**
  + **Tour title**
  + **Location**
  + **Price**
  + **Duration**
  + **Rating and reviews**
  + **"View Details" button**

**3. User Browses Tours**

* **User navigates to the Tours Page to view all available tour packages.**
* **Tours are displayed in a responsive grid layout.**
* **User can click on any tour card to view detailed information.**

**4. User Registers an Account**

* **User clicks "Register" in the navigation bar.**
* **Navigated to Register Page.**
* **User fills the registration form → username, email, password, phone, address.**
* **Form validation runs.**
* **On submit:**
  + **React sends a POST request to create a new user.**
  + **A success toast message appears.**
  + **User is redirected to Login Page.**

**5. User Logs In**

* **User enters email and password on Login Page.**
* **React sends a GET request to verify credentials.**
* **If credentials are valid:**
  + **User data is stored in AuthContext.**
  + **Success toast is shown.**
  + **User is redirected to Home Page.**
  + **Navbar updates to show "Profile" and "Logout" options.**
* **If credentials are invalid:**
  + **Error toast is displayed.**

**6. User Views Tour Details**

* **User clicks "View Details" on a tour card.**
* **Navigated to TourDetails Page.**
* **Page displays:**
  + **Full tour description**
  + **High-quality tour image**
  + **Price and duration**
  + **Location information**
  + **Rating and number of reviews**
  + **"Book Now" button (for logged-in users)**

**7. User Books a Tour**

* **User clicks "Book Now" on the tour details page.**
* **App checks if user is logged in.**
* **If logged in:**
  + **Creates a booking object with userId, tourId, date, status, and price.**
  + **Sends a POST request to create the booking.**
  + **Success toast is shown.**
  + **Booking is added to user's booking history.**
* **If not logged in:**
  + **User is redirected to Login Page.**

**8. User Views Profile and Booking History**

* **User clicks "Profile" in the navigation bar.**
* **Navigated to Profile Page.**
* **Page displays:**
  + **User information (username, email, phone, address)**
  + **"Edit Profile" option**
  + **Complete booking history with tour details and status**

**9. User Edits Profile**

* **User clicks "Edit Profile" button.**
* **Form fields become editable.**
* **User updates information.**
* **On submit:**
  + **React sends a PUT request to update user data.**
  + **Success toast is shown.**
  + **Profile page refreshes with updated information.**

**10. Admin Accesses Dashboard**

* **Admin user logs in with admin credentials.**
* **Clicks "Dashboard" in the navigation bar.**
* **Navigated to Dashboard Page.**
* **Dashboard displays:**
  + **Total tours count**
  + **Total bookings count**
  + **Total users count**
  + **Tabs for managing Tours, Bookings, and Users**

**11. Admin Creates a New Tour**

* **Admin navigates to "Tours" tab in Dashboard.**
* **Clicks "Add New Tour" button.**
* **TourForm component appears.**
* **Admin fills the form → title, description, price, duration, location, image URL.**
* **Form validation runs.**
* **On submit:**
  + **React sends a POST request to create the tour.**
  + **Success toast is shown.**
  + **New tour appears in the tours list.**

**12. Admin Edits an Existing Tour**

* **Admin clicks "Edit" on a tour in the Dashboard.**
* **TourForm pre-fills with existing tour data.**
* **Admin updates fields.**
* **On submit:**
  + **React sends a PUT request to update tour data.**
  + **Tour updates in**

**db.json.**

* + **Success toast is shown.**
  + **Tours list refreshes with updated information.**

**13. Admin Deletes a Tour**

* **Admin clicks "Delete" on a tour in the Dashboard.**
* **Confirmation alert appears.**
* **If confirmed:**
  + **Sends a DELETE request.**
  + **Removes the tour from the database.**
  + **Tours list updates automatically.**
  + **Success toast is displayed.**

**14. Admin Manages Bookings**

* **Admin navigates to "Bookings" tab in Dashboard.**
* **Views all customer bookings with details.**
* **Can update booking status (Confirmed/Pending/Cancelled).**
* **Sends PUT request to update status.**

**15. User Views Gallery**

* **User clicks "Gallery" in navigation.**
* **Navigated to Gallery Page.**
* **Page displays a responsive grid of destination images.**
* **Provides visual inspiration for travel planning.**

**16. User Logs Out**

* **User clicks "Logout" in navigation.**
* **AuthContext clears user session data.**
* **Success toast is shown.**
* **User is redirected to Home Page.**
* **Navbar updates to show "Login" and "Register" options.**

**17. UI Re-Renders with Updated State**

**After every action (Login, Register, Book, Add Tour, Edit Tour, Delete Tour):**

* **The updated data is fetched again OR the local state is updated via Context API.**
* **React re-renders the updated components without reloading the page.**

**18. Data Storage**

* **All users, tours, and bookings are stored in**

**db.json using JSON Server.**

* **The app works like a fully functional backend system with RESTful API endpoints.**

**19. Smooth UX Throughout**

* **Toast notifications provide instant feedback for all actions.**
* **Responsive Tailwind UI adjusts seamlessly across devices.**
* **Navigation is handled by React Router for smooth page transitions.**
* **Framer Motion animations enhance user experience.**
* **Protected routes ensure secure access to authenticated pages.**

**Code Description: -**

A React-based Tourism Management System with user authentication, tour booking, admin dashboard, and JSON Server backend. Users can browse tours, make bookings, manage profiles, while admins can perform full CRUD operations on tours, bookings, and users via a modern responsive UI.

**Tech Stack**

**Frontend:** React + Vite, Tailwind CSS, React Router DOM, Axios, React Toastify, React Icons, Framer Motion  
**Backend:** JSON Server with db.json as the data source  
**Architecture:** Component-based UI + Context API for state management + centralized API service

**App Structure**

**App.jsx**

* Wraps the app with BrowserRouter and AuthProvider
* Loads global Navbar, Footer, and ToastContainer
* Defines routes:
  + / → HomePage
  + /tours → ToursPage
  + /tours/:id → TourDetailsPage
  + /login → LoginPage
  + /register → RegisterPage
  + /dashboard → DashboardPage (Admin only)
  + /profile → ProfilePage (Protected)
  + /gallery → GalleryPage

**Key Components**

**Navbar.jsx**

* Top navigation bar with logo and menu links
* Dynamic menu based on authentication status
* Responsive design with mobile menu support

**TourCard.jsx**

* Displays tour info (image, title, location, price, duration, rating)
* "View Details" button for navigation
* Responsive card layout with hover effects

**TourForm.jsx**

* Reusable form for both adding and editing tours (Admin only)
* Fields: title, description, price, duration, location, image URL

**AuthContext.jsx**

* Global authentication state management
* Provides login, logout, and user session functions
* Role-based access control (admin/user)

**Pages**

**HomePage.jsx**

* Hero section with featured tours and call-to-action buttons
* Displays popular tour packages
* Responsive layout with animations

**ToursPage.jsx**

* Fetches all tours on mount
* Displays tours in a responsive grid
* Shows loading spinner and empty state

**TourDetailsPage.jsx**

* Shows full tour information with high-quality images
* Rating and reviews display
* "Book Now" button for authenticated users

**LoginPage.jsx**

* User authentication form with email and password validation
* Redirects to home after successful login
* Toast notifications for feedback

**RegisterPage.jsx**

* New user registration form
* Fields: username, email, password, phone, address
* Creates new user via API

**DashboardPage.jsx** (Admin Only)

* Admin control panel with tabs for tours, bookings, and users
* Full CRUD operations for tours
* Booking status management

**ProfilePage.jsx** (Protected)

* Displays user information and edit profile functionality
* Booking history with tour details
* Update user data via API

**GalleryPage.jsx**

* Visual showcase of destinations in responsive image grid

**API Service (api.js)**

Centralized Axios wrapper for all REST communication.

**Endpoints:**

* GET /users → fetch all users
* GET /users/:id → single user
* POST /users → register user
* PUT /users/:id → update profile
* GET /tours → fetch all tours
* GET /tours/:id → single tour
* POST /tours → create tour (admin)
* PUT /tours/:id → update tour (admin)
* DELETE /tours/:id → remove tour (admin)
* GET /bookings → fetch all bookings
* POST /bookings → create booking
* PUT /bookings/:id → update status
* DELETE /bookings/:id → cancel booking

**Data Structures**

**User:** { id, username, email, password, role, phone, address }

**Tour:** { id, title, description, price, duration, location, image, rating, reviews }

**Booking:** { id, userId, tourId, tourTitle, date, status, price }

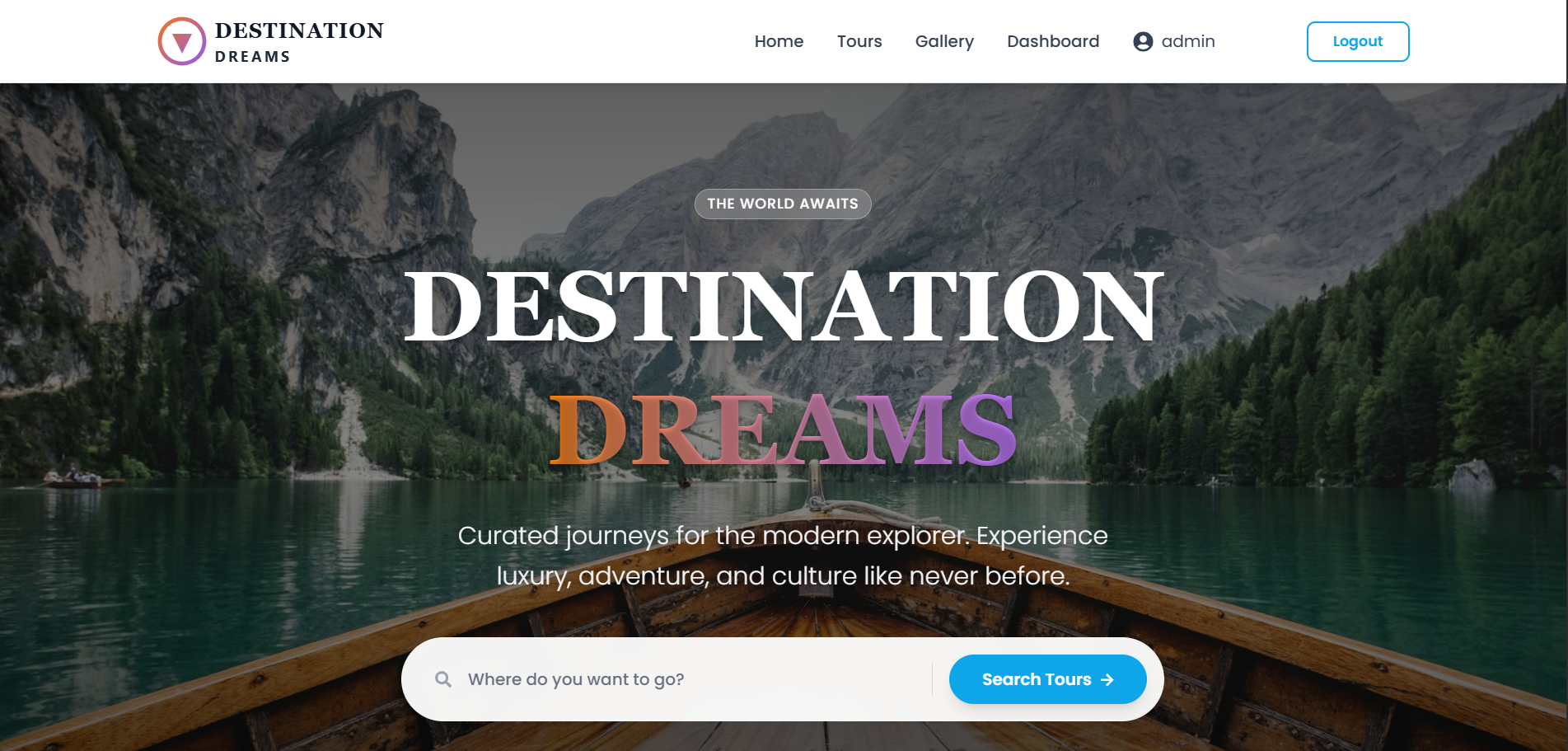
**Core Features**

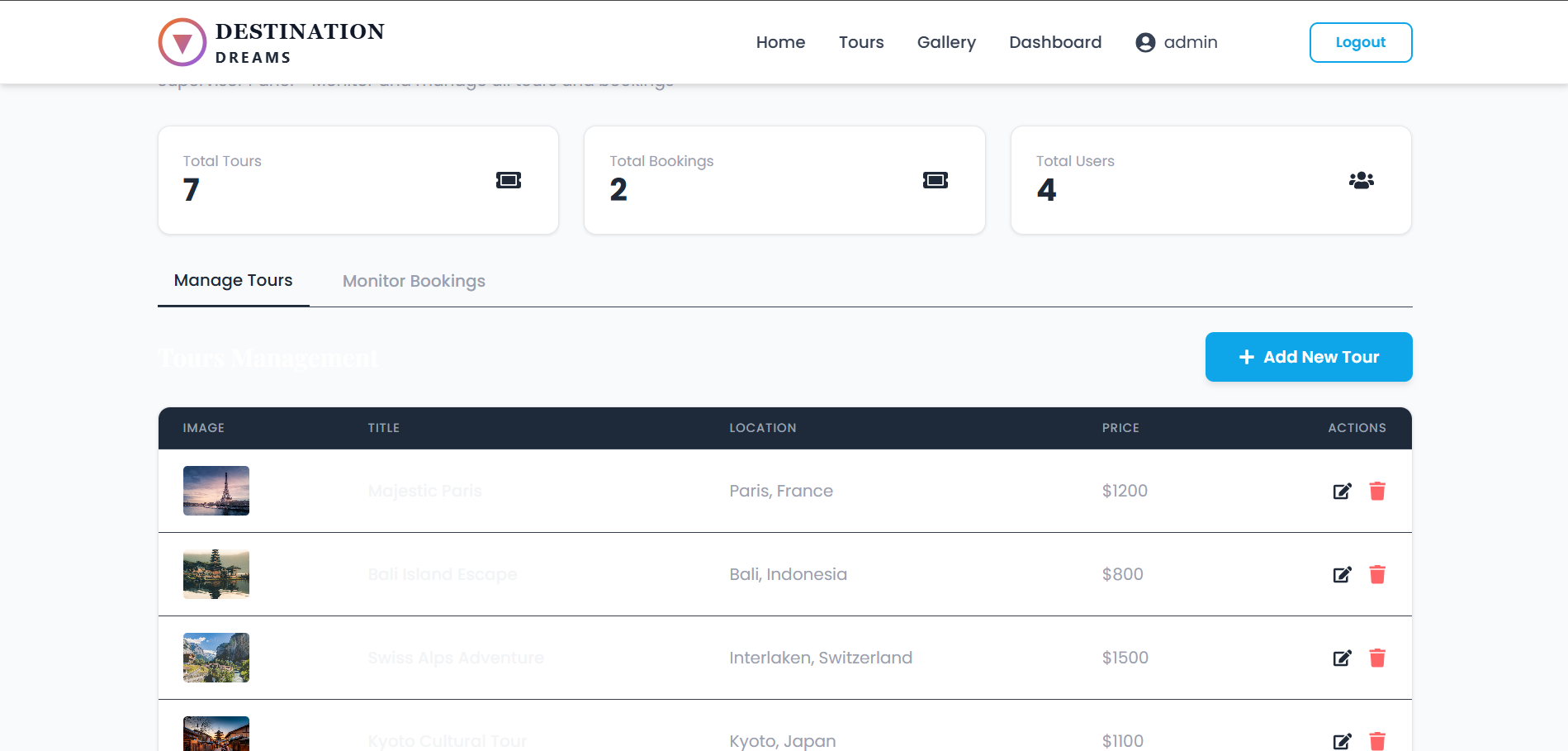
* User authentication and authorization
* Role-based access control (Admin/User)
* CRUD tour management (Admin)
* Real-time booking system
* Booking history tracking
* Profile management
* Toast notifications for all actions
* Responsive Tailwind UI with Framer Motion animations
* Protected routes for authenticated users
* Smooth data flow from UI → API → Backend

**Development**

* Frontend: npm run dev (Vite dev server on port 5173)
* Backend: npm run server (JSON Server on port 3000)
* Build: npm run build
* Lint: npm run lint

**Output**

Admin login   




User

