🌍 TravelDestiny - Fullstack Travel Site

TravelDestiny is a fullstack travel destination site built using React (Vite) for the frontend, Node.js + Express for the backend, and MongoDB for the database. It allows users to view, search, add, update, and delete travel destinations and their famous sites.

# 🧰 Tech Stack

- Frontend: React + Bootstrap + Axios

- Backend: Node.js + Express.js

- Database: MongoDB (Mongoose)

- Tooling: Vite, Postman, Git, GitHub

# 🧭 Project Workflow

## 1. Backend Setup

- Created Express server (index.js)

- Connected to MongoDB using Mongoose

- Defined Destination model with fields: destinationname, description, image, sites[]

- Built RESTful API routes: GET, POST, PUT, DELETE for /home

## 2. Frontend Setup (React + Vite)

Used Vite to scaffold the frontend React project. Folder structure:

src/  
├── components/  
│ ├── Navbar.jsx  
│ ├── DestinationCard.jsx  
│ ├── SiteCard.jsx  
│ └── AddDestinationForm.jsx  
├── pages/  
│ └── Home.jsx  
├── api.js  
├── App.jsx  
└── main.jsx

## 3. Functionality Implemented

- Homepage shows all destinations with name, image, description, and site names

- Search bar filters by destination name and displays only site images and names

- Add new destination using a '+' card

- Edit destination inline inside the card (no separate form)

- Delete destination from database

# ⚙️ Installation Instructions

## Backend

1. Navigate to backend folder:  
cd backend

2. Install dependencies:  
npm install

3. Start backend:  
npm start

## Frontend

1. Navigate to frontend folder:  
cd frontend

2. Install dependencies:  
npm install

3. Start frontend:  
npm run dev

# 📡 API Structure (frontend/src/api.js)

import axios from "axios";  
const API = axios.create({ baseURL: "http://localhost:3000/home" });  
  
export const getAllDestinations = () => API.get("/");  
export const createDestination = (data) => API.post("/", data);  
export const updateDestination = (id, data) => API.put(`/${id}`, data);  
export const deleteDestination = (id) => API.delete(`/${id}`);

# 🔄 Full Application Workflow

This section explains how each part of the application works together, step by step:

## 1. Start Backend Server

The backend is built using Express.js and connected to MongoDB using Mongoose. When you run `npm start`, the server connects to the MongoDB database and starts listening on port 3000 for incoming HTTP requests.

## 2. Start Frontend Server

The frontend is built using React (with Vite as the bundler). When you run `npm run dev`, Vite starts a development server (typically on port 5173) and provides hot-reloading for React components.

## 3. Fetching Destination Data

On app load, the `Home.jsx` component fetches all destinations using `getAllDestinations()` from `api.js`, which sends a GET request to the backend API. The response is used to render destination cards.

## 4. Adding a New Destination

When the user clicks the '+' card, an add form appears. On submitting the form, `createDestination(data)` is called, which sends a POST request to the backend. The backend creates a new entry in the MongoDB database.

## 5. Updating a Destination

Each card includes an inline editing option. When Save is clicked, the updated data is passed to `updateDestination(id, data)` which sends a PUT request to the backend. The backend updates the document in MongoDB.

## 6. Deleting a Destination

Each destination card includes a delete button. When clicked, it calls `deleteDestination(id)`, which sends a DELETE request to the backend. The backend deletes the corresponding entry from the database.

## 7. Search Functionality

The Navbar includes a search bar. When the user types a destination name, the state updates and filters the existing destination data in memory. If matched, it displays only the sites related to that destination.