Full Stack Travel Planner Project Documentation

Project Overview

This is a full stack travel planner application built with React (frontend), Express + Node.js (backend), and MongoDB (database). It allows users to register, login, and manage their trips (CRUD operations). Authentication is JWT-based with secure routes.

Tech Stack

- **Frontend**: React 18, TailwindCSS for styling, Zustand for state management, React Router for routing.
- **Backend**: Express.js, Node.js, JWT-based authentication, Mongoose for MongoDB ORM.
- **Database**: MongoDB storing users and trips.

Frontend Components

- **Login.jsx / Register.jsx**: Provide authentication forms styled with Tailwind. Use Axios to call backend `/auth` routes.
- **Planner.jsx**: Main dashboard for managing trips. Uses `getTrips`, `createTrip`, `updateTrip`, `deleteTrip` API calls.
- **AddTripModal.jsx**: Tailwind + HeadlessUI modal for adding new trips.
- **EditTripModal.jsx**: Modal for editing trips with pre-filled values.
- **Home.jsx**: Landing page, styled with Tailwind layouts and typography.
- **API folder**: Contains axios wrappers for `/api/auth` and `/api/trips` endpoints.

Backend Components

- **server/models/User.js**: Defines user schema with fields like name, email, password.
- **server/models/Trip.js**: Defines trip schema (title, destination, startDate, endDate, user reference).
- **server/controllers/authController.js**: Handles register, login, and user info retrieval.
- **server/controllers/tripController.js**: Handles CRUD operations for trips.
- **server/routes/auth.js**: Routes for authentication (/register, /login, /me, /logout).
- **server/routes/trip.js**: Protected trip routes with JWT authentication.

Authentication Flow

1. User registers via `/auth/register`. Password is hashed.
2. User logs in via `/auth/login`. Server responds with JWT cookie.
3. All `/api/trips` routes use `requireAuth` middleware to validate JWT.
4. User can fetch trips, create new trips, update trips, or delete trips. Only trips belonging to the logged-in user are accessible.

Trip Management Flow

1. Planner.jsx loads trips using `getTrips()` from backend.
2. User clicks 'Add New Trip' → opens AddTripModal → sends POST `/api/trips`.
3. Trips are displayed in Tailwind-styled cards with Edit/Delete buttons.

4. Edit opens EditTripModal → PUT `/api/trips/:id`.
5. Delete sends DELETE `/api/trips/:id` and updates UI.
6. All changes persist in MongoDB tied to the user.

Code Flow

Frontend → Axios → Express routes → Controller → Mongoose → MongoDB.

Example: Create Trip → React AddTripModal calls createTrip() → axios POST `/api/trips` → Express trip route → tripController.createTrip() → Trip model → MongoDB → Returns saved trip → Updates UI.