Realtime Trip Planner - Full Project Documentation

# 1. Overview

The Realtime Trip Planner (RTP) is a full-stack MERN-style application with a React (Vite) frontend, Node.js + Express backend, MongoDB database, and Socket.IO for realtime updates. It allows users to register/login, plan trips, and collaborate with others.

# 2. Tech Stack

- Frontend: React, React Router, TailwindCSS, Zustand (state management), Axios, Leaflet (maps).  
- Backend: Node.js, Express.js, MongoDB (Mongoose), JWT Authentication, Socket.IO.  
- Tooling: Vite (bundler), dotenv (env vars), bcryptjs (password hashing).

# 3. Frontend Components

App.jsx: Main entry point with routing (Home, About, Planner, Auth pages).

Navbar.jsx: Top navigation with conditional auth links (Login/Register or Logout).

Auth/Login.jsx & Auth/Register.jsx: User authentication forms with Zustand for global state.

Planner.jsx: Displays user trips in grid layout, supports Add/Edit/Delete with modal forms.

Home.jsx: Landing page with hero section.

About.jsx: Informational page about the app.

# 4. Backend Components

server.js: Main entry, sets up Express app, connects MongoDB, integrates Socket.IO.

app.js: Configures Express middlewares and routes.

routes/auth.js: Handles /auth endpoints (register, login, me, logout).

controllers/authController.js: Business logic for authentication using JWT.

routes/trips.js: Handles CRUD endpoints for trips (create, read, update, delete).

controllers/tripController.js: Implements trip management logic.

models/User.js: User schema with email, passwordHash, etc.

models/Trip.js: Trip schema with title, destination, dates, owner, collaborators.

# 5. Authentication Flow

1. User registers with email & password -> password hashed -> stored in MongoDB.  
2. On login, password is compared -> JWT is generated -> returned to client.  
3. Frontend stores token in Zustand -> attaches it in axios headers for requests.  
4. Protected routes (e.g., Planner) require token validation.

# 6. Trip Management Flow

1. User opens Planner -> trips are fetched via /api/trips.  
2. Add New Trip button opens Tailwind modal form.  
3. Submitting trip calls POST /api/trips -> saved in MongoDB.  
4. Trips can be edited (PUT /api/trips/:id) or deleted (DELETE /api/trips/:id).  
5. Realtime updates can be pushed via Socket.IO (future expansion).