Smart Travel Planner App - Fullstack Documentation

Overview

The Smart Travel Planner App is a fullstack web application that helps users plan, organize, and manage their trips. It offers features such as trip creation, destination management, budgeting, stopovers, allergy tracking, and admin analytics.

- Frontend: Vue.js SPA (Single Page Application)
- Backend: Node.js/Express REST API
- Database: MySQL Server Version 9.3.0

Features

- User Authentication: Register, login, and manage user profiles.
- Trip Planning: Create, view, update, and delete trips.
- Destinations: Add and manage destinations for trips.
- Stops: Add multiple stops (waypoints) to a trip.
- Budgeting: Set and view trip budgets.
- Allergy Tracking: Track and manage user allergies.
- Alerts: Receive notifications for important actions.
- Admin Dashboard: View user and trip statistics, manage data.
- Export: Export data to Excel for reporting.

Architecture

Frontend

- Vue 3
- Vuex (State Management)
- Vue Router
- Axios (HTTP requests)

Backend

- Node.js + Express.js
- MySQL Server Version 9.3.0
- JWT for Authentication
- dotenv for environment configuration
- CORS for cross-origin requests

APIs and External Services

- Google Places API: For place search and autocomplete during trip and stop creation.
- Google Maps JavaScript API: For displaying maps and directions.
- Axios: Used for frontend HTTP requests to the backend.

Frontend

Project Structure

graphql
CopyEdit

```
Smart-Travel-Planner-App-Frontend/
├── public/
│   └── index.html
├── src/
│   ├── api/                # API calls to backend
│   ├── assets/             # Images and static assets
│   ├── components/         # Vue components (TripPlanner,
StopSelector, etc.)
│   ├── utils/              # Utility functions
```

```
|   ├── views/              # Page-level components (TripsPage,
HomePage, etc.)
|   ├── App.vue
|   ├── main.js
|   ├── router.js
|   ├── store.js
|   └── config.js           # API keys and config
├── package.json
└── README.md
```

Key Components

- `TripPlanner.vue`: Main trip creation wizard; integrates destination, stops, and budget inputs.
- `StopSelector.vue`: UI for adding and removing stops using Google Places API.
- `TripsPage.vue`: Displays the list of trips with all associated data.
- `TripDirectionsPage.vue`: Shows trip routes on a map using Google Maps API.
- `AdminDashboard.vue`: Provides analytics and administrative functionality.

API Integration

- Axios is used for all API requests.
- All protected endpoints require an `Authorization: Bearer <token>` header.
- Base URL is set in `src/api/BackendApi.js`.

Backend

Project Structure

pgsql
CopyEdit

```
Smart-Travel-Planner-App-Backend/
├── routes/
|   ├── adminRoutes.js
|   ├── alertRoutes.js
```

```
|   ├── allergyRoutes.js
|   ├── budgetRoutes.js
|   ├── destinationRoutes.js
|   ├── exportRoutes.js
|   ├── profileRoutes.js
|   ├── stopRoutes.js
|   ├── tripRoutes.js
|   └── userRoutes.js
├── server.js
├── package.json
└── README.md
```

API Endpoints

User

- POST `/api/users/register`
- POST `/api/users/login`

Profile

- GET `/api/profile/:userId`

Trips

- GET `/api/trips`
- GET `/api/trips/user/:userId`
- GET `/api/trips/:tripId`
- POST, PUT, DELETE supported as needed

Destinations

- GET `/api/destinations`
- GET `/api/destinations/user/:userId`

Stops

- GET `/api/stops`
- GET `/api/stops/trip/:tripId`

Budgets

- GET `/api/budgets`
- GET `/api/budgets/:tripId`

Allergies

- GET `/api/allergies`
- GET `/api/allergies/:userId`

Alerts

- GET `/api/alerts`

Admin

- GET `/api/admin/stats`
- GET `/api/admin/users`
- GET `/api/admin/trips`

Export

- GET `/api/export/:table`
- GET `/api/preview/:table`

All endpoints are RESTful and return JSON responses.

Database Schema and ERD

Tables

- `user` (user_id, user_role, username, email, password, created_at)
- `trip` (trip_id, user_id, destination_id, title, start_date, end_date, description, starting_point, number_of_people)
- `destination` (destination_id, user_id, location, address, rating, photo_url)
- `stop` (stop_id, trip_id, name, address, location, photo_url, order_index)
- `budget` (budget_id, trip_id, min_amount, max_amount)
- `allergy` (allergy_id, user_id, allergy_type, severity, notes)
- `alert` (alert_id, user_id, type, message, created_at, seen)
- `admin` (id, number_of_users_registered, number_of_users_deleted)

Entity Relationship Diagram

- `user` 1---n `trip`
- `trip` n---1 `destination`
- `trip` 1---n `stop`
- `trip` 1---1 `budget`
- `user` 1---n `allergy`
- `user` 1---n `alert`
- `admin` is a stats table (not directly related to others)

Legend:

- 1---n = one-to-many
- n---1 = many-to-one
- 1---1 = one-to-one

Relationship Summary

- A user can have many trips, allergies, and alerts.
- A trip belongs to one user and one destination and may have many stops and one budget.
- A destination can be used by many trips.
- A stop is linked to one trip.
- A budget is linked to one trip.
- Allergy and alert data are specific to each user.
- Admin table stores aggregate statistics only.