# Trip Planner Application Documentation

## 1. Overview

Trip Planner is a full-stack web application for browsing, booking, and managing trips. It supports two roles:

- **User**: browse trips, enroll (book) seats, pay, view & cancel their bookings.
- **Admin**: manage users, trips, and all enrollments (approve/reject/cancel).

---

## 2. Architecture

### 2.1 Frontend

- **Framework**: Angular standalone components + Angular Material
- **State & Services**:
  - `AuthService`: handles login, logout, token storage
  - `TripService`: CRUD trips, enrollments, cancellations
  - `UserService`: CRUD users (admin only)
  - `LoadingService` + interceptor: global loading bar & error notifications
- **Routing & Guards**:
  - **AuthGuard**: protects authenticated routes
  - **AdminGuard**: restricts `/admin/*` to admin users

### 2.2 Backend

- **Framework**: Express.js (ES modules)
- **Database**: MongoDB + Mongoose
- **Auth**: JWT-based, `authenticate` + `requireAdmin` middleware
- **Controllers**:
  - `authController` (register/login)
  - `tripController` (trips, enroll, list user’s enrollments)
  - `enrollmentController` (list/update all enrollments)
  - `userController` (profile + admin user CRUD)

### 2.3 Data Models

- **User**: `{ name, email, passwordHash, role, enrolledTrips[] }`
- **Trip**: `{ title, startPoint, vehicle, date, attractions[], cost, totalSeats, seatsLeft, usersEnrolled[] }`
- **Enrollment**: `{ user, trip, seats, customerEmail, bookingDate, status: Pending|Approved|Rejected|Canceled, canceledAt?, cancellationCharge? }`

---

## 3. User Roles

| Role | Capabilities |
|---|---|
| **User** | Register, login, view trips, view trip details, enroll, pay, view/cancel own bookings |
| **Admin** | Everything a user can do, plus: manage users, create/edit/delete trips, approve/reject/cancel *any* booking |

---

## 4. Features

### 4.1 Authentication & Authorization

- **Register**: email + password, hashed on save
- **Login**: JWT returned, stored in `localStorage`, appended to API calls
- **Route Guards**: prevent unauthorized access

### 4.2 Dashboard

- **Public**: none
- **Authenticated User**: shows two grid sections
  - **Available Trips**: a grid of cards (title, cost, “View”)
  - **My Enrollments**: a grid of cards summarizing each booking and status

### 4.3 Trip Browsing

- **Trip List**: paginated/grid view with image placeholder, cost, seats left
- **Trip Detail**: full description, date, vehicle, cost/seat, seats left, form to choose seats

### 4.4 Enrollment (Booking)

- **Booking Form**: select # seats (max = `seatsLeft`), click “Book & Pay” → navigates to payment

### 4.5 Payment

- **Razorpay Integration**: test key used
- **After Payment**:
    1. Creates `Enrollment` (status=Pending)
    2. Decrements `trip.seatsLeft`
    3. Adds enrollment to `user.enrolledTrips`

### 4.6 Viewing Enrollments & Cancellation

- **User's Enrollments**: list or grid detail view of all bookings
- **Cancellation**:
    - Allowed when status = **Approved**
    - Calls `/enrollments/:id/cancel` → sets `status=Canceled`, stamps `canceledAt = now`, computes a `cancellationCharge` (e.g. 10% of seats×cost or flat fee)
    - Refund logic can be added server-side

### 4.7 Admin Management

#### 4.7.1 User Management

- List all users (excluding passwords), create/edit roles, delete

#### 4.7.2 Trip Management

- Table CRUD interface: date, title, start point, vehicle, cost, total seats, seats left
- Form with datepicker, validation, "Create" / "Update"

#### 4.7.3 Enrollment Management

- Table of **all** enrollments (populated user & trip)
- Approve / Reject buttons for Pending bookings
- Optionally Cancel (charging logic same as user)

---

## 5. Cancellation & Fine Charge

- **When**: only after booking is **Approved**
- **Charge**: configurable (e.g. 10% per seat )
- **Stored**: `Enrollment.cancellationCharge` & `canceledAt` timestamp
- **Displayed**: in both user's and admin's enrollment views

---

## 6. Data Flow & API Endpoints

| Method | Endpoint | Auth | Description |
|---|---|---|---|
| POST | `/api/auth/register` | Public | Register new user |
| POST | `/api/auth/login` | Public | Login, returns JWT & user data |
| GET | `/api/trips` | Public | List all trips |
| GET | `/api/trips/enrolled` | User | List this user’s enrollments |
| GET | `/api/trips/:id` | Public | Trip details |
| POST | `/api/trips/:id/enroll` | User | Enroll in trip (creates booking) |
| POST | `/api/enrollments/:id/cancel` | User/Admin | Cancel booking |
| GET | `/api/enrollments` | Admin | List all enrollments |
| PUT | `/api/enrollments/:id` | Admin | Approve/Reject |
| GET | `/api/users` | Admin | List users |
| POST | `/api/users` | Admin | Create user |
| PUT | `/api/users/:id` | Admin | Edit user |
| DELETE | `/api/users/:id` | Admin | Delete user |

---

## 7. UI/UX & Styling

- **Angular Material** theme (primary, accent, warn colors)
- **Responsive Grids**: `mat-grid-list` for trip cards & enrollment cards
- **Tables**: `mat-table` with paginator/striped rows
- **Forms**: `ReactiveFormsModule`, validation feedback
- **Global Loader**: top progress bar via `LoadingService` + interceptor
- **Snackbars**: errors & success toasts

---

## 8. Future Enhancements

- **Dynamic cancellation-fee** based on booking date vs. trip date
- **Email notifications** on booking, approval, cancellation
- **Reviews & ratings** per trip
- **Internationalization** (i18n) for multiple locales

---