Project Documentation format

1. Introduction

Project Title: Cab Booking Application

Team Members:

- Jaswanth: Project Manager
- Mothish: Frontend Developer
- Rishi: Backend Developer
- Prasad: Database Administrator

2. Project Overview

Purpose:
The Cab Booking Application aims to provide a seamless and user-friendly platform for booking cabs. It allows users to book rides, view ride history, and rate drivers, while admins can manage cab and driver details.

Features:

- User and Admin login with role-based access.
- Cab booking and ride history for users.
- Admin dashboard for managing cabs and drivers.
- Real-time ride status updates.
- Rating and feedback system for users.

3. Architecture

Frontend:
The frontend is built using React, providing a responsive and interactive user interface. Key components include the login page, user dashboard, booking interface, and admin management pages.

Backend:
The backend is powered by Node.js and Express.js, handling API requests, authentication, and business logic. It serves as the intermediary between the frontend and the database.

Database:
MongoDB is used for storing user data, cab details, ride history, and driver information. The database schema is designed to efficiently manage relationships and queries.

4. Setup Instructions

Prerequisites:

- Node.js
- MongoDB

Installation:

Clone the repository:

git clone https://github.com/your-repo/cab-booking-app.git

Navigate to the project directory:

cd cab-booking-app

Install dependencies for both frontend and backend:

cd client

npm install

cd ../server

npm install

*Set up environment variables in .env files for both client and server.

5. Folder Structure

Client:

- **/src**
    - **components**: Contains React components.

    - **pages**: Contains different page components like Login, Dashboard, Booking, Admin.
    - **services**: Contains service files for API calls.
    - **styles**: Contains CSS files.

Server:

- **/src**
    - **controllers**: Handles request logic.
    - **models**: Contains Mongoose schemas.
    - **routes**: Defines API routes.
    - **middlewares**: Contains middleware functions for authentication, error handling, etc.
    - **utils**: Utility functions and constants.

6. Running the Application

Commands to start the servers:

- **Frontend:**

```
cd client
npm start
```

- **Backend:**

```
cd server
npm start
```

7. API Documentation

Endpoints:

- **User Authentication:**
    - **POST /api/auth/login:** Authenticate user/admin.
    - **POST /api/auth/register:** Register a new user.

- **Cab Management (Admin):**
    - **GET /api/cabs:** Retrieve all cabs.
    - **POST /api/cabs:** Add a new cab.
    - **PUT /api/cabs/:id:** Update cab details.
    - **DELETE /api/cabs/:id:** Delete a cab.
- **Driver Management (Admin):**
    - **GET /api/drivers:** Retrieve all drivers.
    - **POST /api/drivers:** Add a new driver.
    - **PUT /api/drivers/:id:** Update driver details.
    - **DELETE /api/drivers/:id:** Delete a driver.
- **Booking (User):**
    - **POST /api/bookings:** Create a new booking.
    - **GET /api/bookings/:userId:** Get bookings for a user.
    - **PUT /api/bookings/:id:** Update booking status.

8. Authentication

Authentication and Authorization:

- JWT tokens are used for authentication.
- Middleware checks token validity and user roles.
- Sessions are managed using cookies and local storage.

9. Known Issues

- Occasionally, booking status updates may lag.
- Admin dashboard filtering options need improvement.

10. Future Enhancements

- Implement real-time cab tracking using GPS.
- Enhance user interface with more interactive elements.
- Add support for multiple payment options.

- Improve the rating and feedback system with detailed analytics.