**Uber Clone Web Application**

**Table of Contents**

1. [Project Overview](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#project-overview)
2. [Features](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#features)
3. [Architecture & Tech Stack](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#architecture--tech-stack)
4. [Folder Structure](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#folder-structure)
5. [Getting Started](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#getting-started)
6. [Environment Variables](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#environment-variables)
7. [Running the Application](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#running-the-application)
8. [API Endpoints](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#api-endpoints)
9. [Future Enhancements](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#future-enhancements)
10. [Contributing](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#contributing)
11. [License](https://copilot.microsoft.com/chats/2DsNZg2PbrpzYhPmGE59G#license)

**Project Overview**

Uber Clone is a full-stack web application that mimics the core functionality of modern ride-hailing platforms. Riders can register, request rides, view driver locations in real time, and complete payments. Drivers can accept ride requests, navigate to pickup/drop-off points, and manage ride statuses.

**Features**

* User authentication using JWT (sign up, sign in, protected routes)
* Role-based dashboards for riders and drivers
* Interactive map integration for selecting pickup and drop-off locations
* Real-time location updates via Socket.IO
* Complete ride lifecycle: request → accept → start → complete
* Fare estimation based on distance and duration
* Post-ride rating and feedback system

**Architecture & Tech Stack**

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| Client | React, TypeScript & Tailwind CSS | UI rendering and responsive styling |
| Mapping | Mapbox GL JS | Map display, geocoding, and routing input |
| HTTP API | Express.js, Axios | RESTful endpoints for data operations |
| Real-Time | Socket.IO | Bidirectional events for live updates |
| Database | MongoDB, Mongoose | Data persistence for users, rides, feedback |
| Authentication | JWT, bcrypt | Secure login, token issuance, password hashing |

**Folder Structure**

/

├── project/ # Front-end React application

│ ├── public/ # Static assets and index.html

│ ├── src/ # Components, pages, hooks, services

│ ├── tailwind.config.js

│ └── package.json

├── server/ # Back-end Node.js application

│ ├── controllers/ # Route handlers

│ ├── models/ # Mongoose schemas

│ ├── routes/ # Express routers

│ ├── utils/ # Helper functions (e.g., geolocation)

│ ├── app.js # Express app initialization

│ └── package.json

├── .gitattributes

└── README.md # Project documentation

**Getting Started**

**Prerequisites**

* Node.js v14+ and npm
* MongoDB (local instance or Atlas cluster)
* Mapbox Access Token (or Google Maps API key)

**Installation Steps**

1. **Clone the repository**
2. git clone https://github.com/akshaylavan/Uber\_Clone.git
3. cd Uber\_Clone
4. **Install dependencies**
5. cd project
6. npm install
7. cd ../server
8. npm install

**Environment Variables**

Create a .env file in the server/ directory with:

PORT=5000

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret

MAPBOX\_TOKEN=your\_mapbox\_access\_token

If using Google Maps, replace MAPBOX\_TOKEN with your Google API key and update front-end settings accordingly.

**Running the Application**

1. **Start the back-end server**
2. cd server
3. npm run dev
4. **Start the front-end**
5. cd ../project
6. npm start
7. **Access in browser**   
   Navigate to http://localhost:3000 to interact with the app.

**API Endpoints**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | Register a new rider or driver |
| /api/auth/login | POST | Authenticate user and receive JWT |
| /api/rides/request | POST | Rider submits a ride request |
| /api/rides/driver/:driverId | GET | Retrieve pending ride requests for driver |
| /api/rides/:rideId/accept | POST | Driver accepts a specific ride request |
| /api/rides/:rideId/start | POST | Mark ride as started |
| /api/rides/:rideId/complete | POST | Complete ride and calculate fare |
| /api/rides/:rideId/rate | POST | Submit rating and review after ride |

**Future Enhancements**

* Integrate real payment gateways (Stripe, PayPal)
* Push notifications via Firebase Cloud Messaging
* Dynamic surge-pricing algorithm
* Enhanced mobile-responsive design
* Driver earnings dashboard and analytics
* Administrative panel for managing users and rides

**Contributing**

1. Fork this repository
2. Create a feature branch:
3. git checkout -b feature/YourFeatureName
4. Commit your changes:
5. git commit -m "Add awesome feature"
6. Push to your branch:
7. git push origin feature/YourFeatureName
8. Open a Pull Request with a clear description of your changes

Please adhere to existing code conventions, include tests for new functionality, and ensure all linters pass.

**License**

This project is released under the MIT License. See the [LICENSE](https://copilot.microsoft.com/chats/LICENSE) file for full details.