UTD - KJSIEIT Taxi Booking Application (Zwoop)

This document serves as the documentation of the project and the setup required to run the application locally or on a server are listed below.

1. If not available on your system already, install NodeJS and Python.

   To install **Node JS**, visit https://nodejs.org/en/ and download the installer and run it.
   To install **Python**, visit https://www.python.org/downloads/ and download the installer and run.

2. The application uses **MongoDB Atlas**, a cloud database server that provides free cloud storage. To use the cloud database, register yourself here or simply sign up using your Google credentials. Follow the steps given below -

   a. Click on **Create**.
   b. Choose cluster type as **Shared**.
   c. Choose your cloud provider and region.
   d. Click on "**Create cluster**."

   Your MongoDB cluster will start its provisioning and will be available to you in a few minutes.

   Once the cluster is set up, click on **Connect** and choose "Connect using MongoDB Compass". A connection string is provided along with a hyperlink to download MongoDB Compass.

   MongoDB Compass is a GUI tool to access and edit Mongo collections present locally on your system or on cloud.

   Replace the <username> and <password> with your encoded credentials to get the URL for your cloud database.

   Once MongoDB Atlas is installed, paste the URL into the provided text area and click on Connect to access the data within the cloud.

Store this URL for the time being as it will later be used in the application.

3. For the Maps features, **Google Maps API** is used. To create an API key to enable the map features, visit https://console.cloud.google.com/.

   Here, sign in with your desired Google Account to start your Google Cloud Platform.

   Open the side panel and click on Library inside the APIs and Services dropdown.

   Click on the Maps Javascript API and enable the API. Create a new project with a Maps Billing account if not already available.

   Once a project and a maps billing account is created, generate a new key and store it for later use. Also, restrict the key to the Places API, Maps Javascript API, Directions API and the Distance Matrix API since these are the only APIs required for the project and helps in monitoring the usage of the same.

4. For the payment gateway, **Razorpay** is integrated into the system for handling transactions between riders and drivers.

   To use the payment gateway, visit https://razorpay.com/ and create an account. Fill in all details and continue. Inside the setting screen, click on Create a new key option your key will be generated. You will get Key ID and Key Secret. You will be shown the Key Secret only once, so immediately copy it somewhere.

5. Clone the Git repository for the project and open 2 terminals in the directory where the repository was cloned.

   First, open the backend directory and navigate to the file - app.py. Replace the dbUrl on line 12 with the connection string from your MongoDB Atlas cluster obtained in Step 2. Also change the collection name from user_login_system on line 14 to your liking.

Second, open the routes.py file in the same directory. On line 11, change the rzp_id and rzp_secret variables to the Razorpay key ID and key secret obtained in step 4.

Now, open the frontend directory and navigate to the file src/App.js. Here, on line 15, replace the Google Maps API key with the one obtained in step 3.

6. To start the system locally, open a terminal and navigate to the frontend directory.

   Install all the necessary dependencies for the frontend using the command -

   ```
   npm i
   ```

   While all the frontend dependencies are installing, open a second terminal and navigate to the backend directory.

   Install all the necessary dependencies for the backend using the command -

   ```
   pip install -r requirements.txt
   ```

   These two commands will install all packages required for the system to work.

   Once installed, to start the frontend, type the following command and run -

   ```
   npm start
   ```

   And to start the backend, run the following -

   ```
   python app.py
   ```

If the backend and frontend are hosted separately, change the baseurl in frontend/src/apis/url.js to the API URL of the hosted backend before running the system.