**Uber Application**

This Java application is a clone of the popular ride-sharing service Uber. It aims to replicate the core functionality of Uber, allowing users to request rides, drivers to accept requests, and facilitating the entire ride process.

**Features**

* **User Authentication:** Users can create accounts and log in securely
* **Ride Request:** Users can request rides by specifying their pickup and drop-off locations
* **Driver Assignment:** Drivers receive ride requests and can choose to accept or reject them.
* **Real-Time Tracking:** Users can track the location of their assigned driver in real-time
* **Wallet Integration:** We are using the wallet feature for payment.

**Technology Used**

* **Java (JDK – 20):** The core programming language used for backend development
* **Spring Boot:** Framework for building Java-based applications
* **Hibernate:** ORM (Object-Relational Mapping) tool for database interaction.
* **Lombok:** Library that reduces boilerplate code by automatically generating common methods like getters, setters, and constructors during compile time through annotation
* **Jackson data format XML:** Jackson is a library used for parsing and generating XML data formats, providing seamless integration with XML-based applications
* **Tomcat:** Tomcat is a servlet container that implements the Java Servlet and Java Server Pages specifications, providing a platform for deploying and running Java web applications.
* **Spring boot security:** Spring Boot Security is a module that provides authentication and authorization support for Spring Boot applications, enabling secure access control and user management.
* **Jason web token:** JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties, often used for stateless authentication and authorization in web applications.
* **Java Persistence API:** Java Persistence API (JPA) is a specification that provides a framework for managing relational data in Java applications through object-relational mapping (ORM)
* **Google distance API:** The Google Distance Matrix API is a service that provides travel distance and time estimates based on various transportation modes between multiple origins and destinations.

**Tools**

* **Post Man:** Postman is a popular API client tool used for testing and debugging HTTP APIs by sending requests and inspecting responses
* **Maven:** Maven is a build automation tool that manages dependencies, compiles code, and facilitates project management using XML configuration files

**Design Patterns**

* **Builder Design pattern:** Builder Design Pattern separates the construction of complex objects from their representation, allowing the creation of different object configurations through a builder class.
* **Factory Design Pattern:** Factory Design Pattern provides an interface for creating objects in a superclass, while allowing subclasses to alter the type of objects that will be created.
* **Singleton Design Pattern:** Singleton Design Pattern ensures that a class has only one instance and provides a global point of access to that instance.
* **Chain of responsibility:** Chain of responsibility is a behavioral design pattern where a request is passed through a chain of handlers, each capable of processing or passing it to the next handler in the chain.

**Login Instructions**

* **User Name: hetul@gmail.com**
* **Password: Hetul@123**

**For login you have to use this ID and Password**

**Setup Instructions**

1. **Registration: (POST METHOD)**

[**http://localhost:8080/passenger/signup/create**](http://localhost:8080/passenger/signup/create)

**for registration you have to copy this link and then pass the JSON body. As a result you will get response at the created or error.**

**{**

**"name": "John",**

**"gender": "m",**

**"phoneNumber": "1214567890",**

**"email": "john@gmail.com",**

**"password": "John@123",**

**"on\_ride": "false",**

**"isTeenAccount": "false"**

**}**

1. **Login: (POST METHOD)**

[**http://localhost:8080/auth/passenger/login**](http://localhost:8080/auth/passenger/login)

**Then you have to use this link for login and then you have to pass the request inside the JSON body as a response you will get JWT token which will be useful in all the steps after this.**

**{**

**"email": "seemant@gmail.com",**

**"password": "Seemant@123"**

**}**

1. **Display Account Information: (GET METHOD)**

[**http://localhost:8080/passenger/account/**](http://localhost:8080/passenger/account/)

**For account visiblity you have to use this link and when you request with this link for authorization you have to pass JWT token.**

1. **Book Ride: (POST METHOD)**

[**http://localhost:8080/passenger/account/bookride**](http://localhost:8080/passenger/account/bookride)

**This link will be used for book the ride after then you have to use the JWT Token after that we have to pass JSON query inside the body and it will ask for pick-up and drop-off location for destination, and we have to insert the locations then request will be passed to accept the ride.**

**{**

**"pickUpLocation": "1280, St. Marc, Montreal, Ca",**

**"destinationLocation" : "750 cote de la place D'arms, Montreal, Ca"**

**}**

1. **Accept the ride: (POST METHOD)**

[**http://localhost:8080/passenger/account/bookride/accept**](http://localhost:8080/passenger/account/bookride/accept)

**for accept the ride you have to use this link and then pass the JWT Token after this you have to pass the JSON query inside the body.**

**{**

**"acceptTheRide": "true"**

**}**

1. **Update Account. (PUT METHOD)**

[**http://localhost:8080/passenger/account/update**](http://localhost:8080/passenger/account/update)

**For Update account you have to use this link and pass the JWT Token and then you need to pass JSON query inside the body to update the account.**

**{**

**"passenger\_id": 1,**

**"name": "Hetull",**

**"gender": "m",**

**"phoneNumber": 12345678900,**

**"email": "hetul@gmail.com",**

**"password": "$2a$10$3SQTuczuPLGf.xIJ9n1UZ.123CcPROR1IWzUGX5Wmln8N7KRW4h76",**

**"on\_ride": false,**

**"role": "ROLE\_USER",**

**"wallet": {**

**"wallet\_id": 1,**

**"money": 200.0**

**},**

**"bookRide": {**

**"bookRideId": 1,**

**"pickUpLocation": null,**

**"destinationLocation": null,**

**"requestDriverU": null,**

**"fareIn$": 0.0,**

**"distanceInKm": 0.0,**

**"duration": null,**

**"acceptTheRide": false**

**},**

**"teenAccount": false**

**}**

1. **Delete Account.**

[**http://localhost:8080/passenger/account/delete**](http://localhost:8080/passenger/account/delete)

**For delete account use this link and then pass the JWT Token.**

1. **Adding amount in wallet.**

[**http://localhost:8080/passenger/wallet/add**](http://localhost:8080/passenger/wallet/add)

**In this step you have to add the amount in you wallet by using this link and then you need to pass the JWT Token to complete this step. After this you need to pass JSON query inside the body to see the result as amount is added in wallet**

**{**

**"money" : "50"**

**}**

1. **Logout.**

**For this step we have to get the request and then we have to pass JWT Token to fulfil this step.**

**Future Improvement**

* **Multi-platform Support: Develop mobile apps for Android and iOS platforms.**
* **Enhanced Security: Implement additional security measures to protect user data and transactions.**
* **Advanced Features: Introduce features like ride scheduling, multi-stop trips, and fare estimation.**