**JAVA PROJECT - CAB BOOKING APPLICATION**

**PROBLEM STATEMENT**

The taxi booking system allows the users to book a taxi for riding to a particular destination. The

customer books a taxi and if the driver is available for the particular time, which is decided by

which the driver isn’t busy riding another ride, the taxi is dispatched to the customer. The customer initiates a ride and reaches the destination using the shortest path. This is decided by a connected network of source, destination and distances between them which is as of now given manually. Payment methods,reviews and insights can be an optional choice which can be added later.

**USE-CASES**

1. Customer registers and logins with an account.
2. Customer requests a taxi for riding to a particular destination by filling the booking details.
3. If the taxi and the driver is available, the taxi is dispatched and the estimated end time and price are automatically calculated.
4. Customer initiates a ride and reaches the destination using the shortest path route.

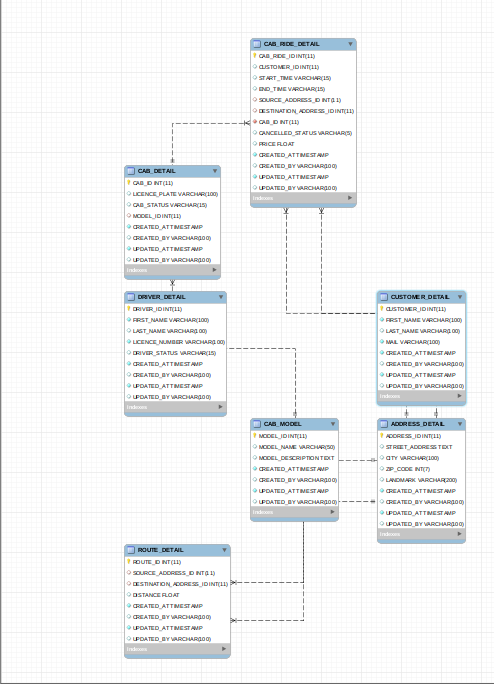
**APPROACH**

Create five packages.

* + com.zilker.taxi.bean
  + com.zilker.taxi.constant
  + com.zilker.taxi.jdbc
  + com.zilker.taxi..dao
  + com.zilker.taxi.util

1. com.zilker.taxi.bean will have pojo classes that have the customer details, invoice details and route information.
2. com.zilker.taxi.constant will have constant class that has all the constants declared and initialised and a query class that contains all the necessary database queries.
3. com.zilker.taxi.jdbc has the main class which contains the business logic necessary for the application to run.
4. com.zilker.taxi.dao will have a DAO class that has the functionality to perform CRUD operations in the database.
5. com.zilker.taxi.util will have a class containing database connectivity essentials and another class to compute the shortest path, estimated end time and price depending upon the source and the destination.

**ENHANCED ENTITY-RELATIONSHIP MODEL**



**FUNCTIONALITY**

1. Customer registers a new account by providing profile details such as first name, last name and mail-ID which is then inserted into the database. Customer can also edit his profile details, if necessary. registerAccount() is used to register an account. updatePersonalDetails() is used to update the profile details.
2. Customer can then book a new ride by filling the booking details such as the start time of journey, the source and the destination. Estimated end time and the fare of the ride is automatically computed. A booking ID is returned which is to be noted by the customer for any future queries. bookRide() is used to book a new ride.
3. The booking details can also be updated provided the time of updation is before the start time of the journey. Customer can also cancel a ride by providing the booking ID which is unique to a ride. updateBookingDetails() and cancelRide() are used to update the booking details and cancel a ride respectively.
4. Customer can view his profile and booking details. displayProfile() and displayBookingDetails() are used to display the profile and display booking details respectively.
5. Customer can also delete the account after which the corresponding history is deleted from the database. DeleteAccount() is used to delete the account from the database.