# Problem => Taxi booking System

**Taxi Booking System**: Implement an online taxi booking system for a customer where the customer can select a category to book based on his pickup location and drop location.

**Constraints:**

1. Various categories of taxi will be available like mini, sedan, suv, share, outstation
2. Each vehicle is tied to an agency to which it belongs.
3. Assume a method exists getDistanceTime (Point a, Point b) which return distance and time taken to travel from a Point A to point B, where Point refers to a model representing the location in terms of latitude and longitude.
4. Fare is a factor of city specific fare, distance, time and number of active users in the system. Fare increases as demand increases with a factor of 0.1 \* total users requested rides/total inventories within a category, when the demand crosses the supply else the fare remains same.
5. Active users refer to the users who are trying to book a cab at that time. Dormant users are those who have not created any booking in last 1 week.
6. Commission is based on the category selected. Some categories have high commission like suv, luxury, outstation while others can have lower commission

**Implement any of 2 api’s:**

1. Find the estimated fare for a booking from point a to point b based on category selected
2. Find number of vehicles available and occupied based on category provided at any given point in time
3. Total number of bookings based on category given in the date time range supplied
4. No of active and dormant users within a date time range. Active users refer to the users who are trying to book a cab at that time. Dormant users are those who have not created any booking in last 1 week.
5. Agency wise total revenue generated for a given date range
6. Total booking of an agency based on category supplied.

Expectations

1. Code quality should be production ready for merge and deployment.
2. Guidelines have the highest weightage than finishing more api’s.
3. Code should be demo able. Create the sample data yourself in a file, test case or main driver program itself (no external data store). Don’t spend time parsing the inputs.
4. Code should be readable, modular (no monoliths), testable, extensible with proper naming conventions.
5. Code should handle edge cases properly and fail gracefully.

**Guidelines:**

o Define a detailed object model for entities required by system

o Make proper use of Inheritance, Abstraction, interfaces, exception handling

o Have proper commenting in code and should follow best coding practises

o Use design patterns like Builder, Factory, Visitor etc wherever applicable

~~o~~ Justify his/her coding choices i.e. why did he/she choose to define a separate function for a feature or why did he/she not define constructor for initialising class

o Define Enums, Singleton classes where applicable

o Separation of concern is addressed

o Implement unit test cases for key sections of his/her code

o Use Java 8/7 features like functional interfaces, Auto Closable resources etc.