Practise Activity:

Description

Your task is to design and write code for a start-up company who offers services like the UBER taxi service.

UBER provides its customers with a private taxi service as well as delivering food and packages. UBER also offer other services such as freight transportation and the option to rent Lime electric bicycles and scooters. However, UBER originally started with a taxi service in just a few cities. In this exercise, you will build a simple model of the backend for a taxi service.

Objectives

In this exercise, you will practise designing a modularise application and write code for those modules. You will create unit tests and commit your code and any design document to a local git repository and finally, publish it on GitHub.

You are not required to provide a user interface for taxi drivers and customers, but you can write one if you have spare time.

Keep in mind, this exercise is more about design and the way you produce and test your app but not about producing lots of code.

Tasks performed by your application

Your application should perform the following tasks:

1. **Register drivers and customers**
    a. Credit card details are validated and kept on your system for payment purpose during registration.

    b. Drivers are rated by customers and an overall record (0-5) is kept by the system.

    c. Drivers also specify the type of car they drive and the number of passengers they can carry.

    d. Information such as these will be stored in a database, but the developers have not had time to create the database.

2. **A history of any completed journeys is kept by your system.**
    This log includes the driver and passenger and other details about the journey such as distance, cost and start and end positions.

    a. Customers and drivers could ask for the details of journey/s they have made (by date/driver). This is also important for auditing

purpose and resolving any dispute.

b. You will need to think about how to unit testing any code which needs access to databases or files.

3. **Booking a fare**
    a. The customer taking a journey, enters a post code for the start and end positions.

    b. You will use a web service to find the distance between the two points and use this to calculate an estimate of the fare. You may consider a fixed rate for each mile. The web service is not yet ready, but you will still need to test your code.

    c. The customer will also specify the number of passengers. This will be used to choose the type of car that is required and to increase the fare. You may assume a rate for each added traveller in the same taxi.

4. **Notification of customers and drivers**
    The system uses mobile phone data to connect to the customers and drivers. It can locate the position of the registered drivers and customers using their mobile phone. You will obviously not emulate their mobile phone but just write code for the backend.

    a. The system is notified of the driver's location whenever the driver's position changes. Driver's location is used when assigning drivers.

    b. When a customer opens your app, their starting and end postcode is sent to your system.

    a. All the drivers in the customer's vicinity are sent a message of the request.
        i. The first driver who accepts the fare can hold the far and the request is then taken off the air.
        ii. The customer is then informed how long it would take for the driver to arrive and an estimate of arrival time to destination (ATE).
        iii. The system uses a web service for calculating distances and time, but this service is not yet operational. Think about how you could unit test your code.

5. **Payment**
    a. When the journey is completed, the system will notify the customer of the cost and take payment from their account.

b. UBER collects 1/3 of the fare but your app will take 1/5 to attract new drivers.

c. A record of the payment is kept by your system in a database. The database is not yet produced but you still need to unit test your code.

6. **Rating the driver**
   a. At the end of any journey, customers are sent a notification, asking them to rate the drivers. Customers choose a rate 0-5. This rate is added to all other rates given for the driver.

More details about your work

- Design the structure of your program. Decide what modules you wish to use and how they are layered (UI, Business and Data layers)

- Write basic code in either Java, C#, Python or JavaScript to perform the above tasks. Java or C# are recommended

- Produce unit tests for any code that you write. You will consider the absence of a database and the "location finder" web service. Why not use this opportunity to develop your applications in a TDD manner?

- Apply the *SOLID principle* to any class and method which you create

- Think of applying a design pattern.
  Tip: Think about how you would use a fake database or web service for your unit tests (mock or stub).

- Use GIT to store your code and publish it on GitHub. Committing your design document or class diagram etc to git is highly advisable

Submission of your work

Before starting to work on this activity, submit a simple plan explaining what you are going to do. This will ensure that the piece of work selected meets all the criteria required at this stage.

Be sure to submit your plan within three days. Here are some guidelines:

The submission should describe:

- The project and context description
- The support available to you (e.g. from your line manager or wider team)
- The planned outputs

Submission format

To submit your reflection, please follow these steps.

Instructions and format

Your submission should be in written format.

There is not a set template for this task, so you can structure your submission in any way you want. However, we would suggest something that includes the following:

Introduction (one paragraph):

- Explain the project's context (why is it important?)
- Write in the first person (use 'I')
- Explain your personal input

Conclusion (one paragraph):

Use reflective questions such as:

- What have I learnt?
- What went well?
- What did not go well?
- What will I do the same next time?
- What will I do differently next time?

Evidence:

You should include supporting evidence / material such as:

- Diagrams
- Photographic Evidence (e.g. hand-drawn sketches)
- Prototypes
- Screenshots (e.g. designs, code, or the working app)
- Video Evidence (e.g. screen recording of working app, if needed)
- Written documentation

You can also include appendix items or screen prints using word processing software.

How to submit your work?

- Go to Bud and locate your activity window
- Click on the 'Submissions and Messages' button

- Click on the 'Submit Work/Message'
- Write a message with your answers (you may submit a written document too. However, we would like to encourage you to keep this submission as simple as possible)
- Click on the 'Submit' button

We will be in touch with you shortly. In the meantime, if you need any support, please reach out to us at any time.