Google Project Challenge

Interview Day
Higher Level Apprenticeship in Digital Innovation

Please complete <u>one</u> of the assignment options below before attending the Interview Day onsite at Google.

For your selected assignment you must provide us with all the code and a file with instructions on how to build and install your application. Do this by uploading your code to GitHub in this repository.

You may use any available resources, tutorials, and assistance on these projects. What resources you used, how much you learned, and what you found challenging is of as much interest to us as how well you complete the task.
A few tips: While there are plenty of online resources where you can find code samples to get started, we'd like to see a significant amount of code written by you. You should understand any code in your project that you reuse. Including 3rd-party libraries is fine, and even encouraged rather than rewriting something very complex or very basic… as long as it's done smartly.

Option 1 - Shopping List

Build a simple webapp that provides users with an ability to keep track of a shopping list.

- User must be able to add an item to the shopping list
- User must be able to view their whole shopping list
- User must be able to delete an individual item from the shopping list
- User must be able to delete their entire shopping list, with a single button click (without going and deleting each individual item one by one)
- Each user must be able to login with their Google account and their shopping list must persist between their logins

One way to do this is to use Google App Engine. Here are some resources to help you get started:
https://cloud.google.com/appengine/docs/python/getting-started/creating-guestbook

---

Option 2 – Maps API

Create a webpage that uses the Google Maps JavaScript API to show a map, which is centered on the user's current location.

- There should be a circle drawn on the map, whose center is also the user's current location
- It should be possible to see all the streets inside the circle (in other words, the circle shouldn't be filled in)
- Allow the user to switch between a radius of 500m, 1000m, or 2000m for the circle
- Allow the user to switch traffic information on or off

Option 3 – Banking Web Application

Create a small web application for a retail bank using the framework of your choice. The application should contain the following tables:

- **Profiles**: User ID (key), Password, Full name
- **Accounts**: Account ID (key), Account number, User ID, Current balance
- **Transactions**: Source account ID, Target account ID, Amount, Date and time

Employees of the bank should be able to do the following:

1. Create new clients.
2. Create new accounts for existing clients.

Clients of the bank should be able to do the following:

1. Log onto the application
    - *Hint*: This functionality is more challenging than it looks. If you are having trouble, consider listing clickable profiles on the client welcome page for a start. You can come back to this question after finishing tasks (2), (3) and (4) below.
2. Display the balance of all their accounts
3. List the transactions for each account over the past month / quarter / year
    - *Hint*: You can use a dropdown list to let users select the period for which they want to list transactions.

4. Transfer money between their accounts or to someone else's account
    - *Hint*: Users can only perform a single transaction at a time, but make sure you address error cases.
5. Update their profile including their password (*bonus*)

Create 2-3 test profiles (each with a couple of accounts) to showcase the web application.

Option 4 - Breakout Game

Create a game similar to https://en.wikipedia.org/wiki/Breakout_(video_game)

- Your game should have a single screen with 8 rows of bricks, a paddle, and a ball.
- If the ball touches the bottom of the screen, you lose a 'life'.
- After 5 lives, the game is over.
- Every time the ball touches a brick, the brick disappears and the score increases.
- The ball bounces off the paddle, the bricks, and the top, left and right edges of the screen.

You can watch a video of the game being played here.

You may implement this in PyGame, or using any other technologies of your choice.

Option 5 - Formula 1 game

You are tasked with creating a small web application for a Formula 1 game app using a framework of your choice.
The application allows creating Formula 1 drivers that have different skills. They are racing on different tracks that have unique characteristics.
Their performance depends on their skills, but also on the tracks.
You will have to build a small app that will be able to add new drivers and tracks, calculate their performances and then display some results.

Step 1 - Create driver

Let's first define what skills a driver has:

* Name
* Age
* Concentration [0,100]
* Motivation [0,100]
* Stamina [0,100]
* Aggressiveness [0,100]
* Talent [0,100]
* Overall

The Overall skill is derived using the following formula:

*Overall = (Concentration * 3 + Motivation * 3 + Stamina * 3 + Aggressiveness + Talent) / 2*

The Overall should always be stored as an integer and rounded up.

Create a structure(s) that would provide an easy to create new drivers, modify existing drivers and delete them.

Step 2 - Create tracks

Drivers do races at different tracks. Each track has the following characteristics:

* Name
* Location
* Race distance
* Average speed

Create a structure that will represent a track.

There should also be an option for easy creation, modification deletion of tracks.

Step 3 - Add driver’s favourite track

Every driver has a favourite track. On that track it performs better than on the rest of the tracks. Modify the Driver class to be able to keep information about the driver's favourite track.

Step 4 - Create races

Whenever a driver races at a track, the following information is kept:
* Driver
* Track
* Date
* Race time
* Weather [dry or raining]

Create a structure that will represent a race.

There should also be an option for easy creation, modification & deletion of races.

Step 5 - Calculate race performances

Every driver that participates in a race has a race time. The race time depends on driver's skills, on track's characteristics and on the actual conditions during the race.
For each driver that participated in a race, their race time can be calculated using the following guidelines:
* base time = 1000 / Overall * (race distance / avg speed)
* add random number between 1 and 10 to the base time
* if it's driver's favourite track then the race time will be lowered by 25
* if it's raining add 20 if driver's talent is below 25, 5 if above 80, else 10

Step 6 - Display results

Add a few sample drivers and tracks. Create races and then calculate the race time for every driver in every race.

Then, write functions that will:
* return the winner of a particular race
* list all drivers that have won a race