Assignment: Building a Database that will Store and Display Data about Formula 1

1. Assignment Information

Module: Cloud Platforms & Applications

2. Introduction

NOTE: Read the whole assignment brief first before implementing it. It contains very important information.

In this assignment, you will be tasked with building your first PaaS application, which will be a database that stores information about F1 drivers and teams. You will be required to deal with users who login and logout. This will function a little like Wikipedia in that anyone who is logged in can change anything, and anyone who is logged out can change nothing.

In this application, users should be able to add and update drivers and teams that are in the database. They should also be able to perform comparisons of either teams or drivers. You will also be required to do some filtering. With filtering, however, you will be limited on the types of queries you can perform due to the NoSQL nature of Firestore. It is recommended that you structure your datastore models in such a way to make the required queries easier if possible.

1. How fully featured, complete, and robust your code is, along with how well your UI is thought out (80%)
2. How well documented your code is (20%)

3. Submission

You are required to submit two separate components to the Moodle:

• An archive containing your complete Google App Engine Python project. The accepted archive formats are: zip, rar, 7z, tar.gz, tar.bz2, tar.xz. The use of any other archive format will be rejected
• Python is the required language for the backend. The use of any other backend language (e.g., PHP, JS) will result in an automatic assignment failure.
• A PDF containing documentation of your code. Copying and pasting code into a PDF does not count as documentation.

There are also a few penalties you should be aware of:

• Code without a .git repository will be rejected
• Remote repositories are not permitted. You are not allowed to use Github, Gitlab, BitBucket, or any other remote repository. Your code will not be corrected if you use one of these.
• Code without documentation will be rejected
• A git repository with less than 7 commits will be rejected
• Code that fails to compile will result in a failing assignment.
• The use of libraries outside the SDK is not permitted
• The standard late penalties will also apply.

4. Plagiarism

Be aware that plagiarism is taken very seriously. Plagiarism includes:

• Submitting online resources/tutorials as your own work.
• Transcribing code from online sources or others.
• Paying someone to do your assignment.

5. Coding Tasks (80%)

• Group 1 tasks (20%)

1. Application must have a working login/logout service exactly as provided in firebase-login.js.
2. Firestore collections must represent a driver and a team:
   - Drivers should have: Age, Total Pole Positions, Total Race Wins, Total Points Scored, Total World Titles, Total Fastest Laps, and the team they drive for.
   - Teams should have: Year Founded, Total Pole Positions, Total Race Wins, Total Constructor Titles, and Finishing Position in the previous season.
   - Note: The use of indexes on collections is not permitted.
3. Separate pages for adding a driver and a team to the database.
4. A form to query drivers, allowing the user to select an attribute with comparison options (less than, greater than, or equal).

• Group 2 tasks (40%)

5. A form to query teams with the same comparison functionality as drivers.
6. Query forms must be accessible to logged-out users.

7. Each driver should be shown as a hyperlink leading to a separate page displaying their details.
8. Each team should be shown as a hyperlink leading to a separate page displaying their details.

• Group 3 tasks (60%)

9. Allow logged-in users to edit driver attributes.
10. Allow logged-in users to edit team attributes.
11. Allow logged-in users to delete a driver or team.
12. A separate page for comparing two drivers.

• Group 4 tasks (80%)

13. A separate page for comparing two teams.
14. A two-column table for driver comparison, highlighting better stats in green.
15. A two-column table for team comparison, highlighting better stats in green except for finishing position and year (lower values are better for these fields).
16. UI must be intuitive and user-friendly.

6. Documentation Brackets (20%)

• Documentation should be around 1,500 words.
• Every method should be documented with a high-level explanation