Assignment 01: Building a GPU database

Barry Denby

March 4, 2021

1 Assignment Information

Course:	MSCC / MSCBD
Stage / Year:	1
Module:	Cloud Platforms & Applications
Semester:	1
Assignment:	1 of 3
Date of Issue:	2021-03-04
Assignment Deadline:	2021-04-18 @ 23:55 (End of week ??)
Assignment Submission:	Upload to Moodle
Assignment Weighting:	10% of Module

2 Introduction

NOTE: read the whole assignment brief first before implementing it contains very important information

In this assignment you will be tasked with building an interactive database about GPUs (Graphics Processing Units) and the features they support. You will be expected to use a NoSQL type database as storage for this system.

Your application must be capable of the following

- Add in a GPU with a series of features.
- Edit a currently existing GPU and its features.
- Be able to choose a set of supporting features and figure out which GPUs support them.

Your application will be similar to the format found at `http://vulkan.gpuinfo.org/listreports.php`

If you click on any of the GPUs listed here it will bring you to another page. The information we are interested in here is listed under the "features" tab. You are required to map six features of a GPU which are the following: geometryShader, tesselationShader, shaderInt16, sparseBinding, textureCompressionETC2, and vertexPipelineStoresAndAtomics. Each GPU stored should contain the information for these six properties.

When a GPU is added a user should be able to click a series of checkboxes (or similar) to enable or disable features. When an add button is clicked there should be no duplicate names in the database or preexisting objects in the database for that GPU.

The score for your application will depend on:

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

NOTE: This is an individual assignment. It is not a group assignment. You can discuss ideas/algorithms but you cannot share code/documentation

3 Submission and Penalties

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 incur a 10% penalty before grading.
- A PDF containing documentation of your code. **If you do not provide documentation your code will not be marked.** Copying and pasting code into a PDF does not count as documentation.

There are also a few penalties you should be aware of

- Code that fails to compile will incur a 30% penalty before grading. At this stage you have zero excuse to produce non compiling code. I should be able to open your project and be able to compile and run without having to fix syntax errors.
- The use of libraries outside the SDK will incur a 20% penalty before grading. You have all you need in the standard SDK. I shouldn't have to figure out how to install and use an external library to get your app to work
- **An omission of a git repository attached to your email address that is registered for GCD will result in your application and documentation not being graded.**
- The standard late penalties will also apply

You are also required to submit as part of your archive a working Git repository.

- When I unpack your archive there should be a .git directory as part of it.
- This should be a fully working **local** git archive. It should not require access to a remote repository
- You are not permitted to upload your work to Github, Gitlab, or any other publicly visible git repository (assignment will be marked as a zero if it is)
- If you need a remote git repository the only permitted one is the college provided Gitlab which can be found at gitlab.griffith.ie
- There must be a minimum of seven commits in the git repository, one per completed bracket.

Very Important: Take note of the grade brackets listed below. These are meant to be completed in order. If you skip a bracket or do not complete a bracket following brackets will not be considered for marking. You should be well capable of producing strong and generally robust software by now. For example if there are six brackets and you fail the third one, then the fourth, fifth, and sixth brackets will

not be marked. Documentation brackets will be treated separately from Coding brackets.

You should also be aware that I will remove marks for the presence of bugs anywhere in the code and this will incur a deduction of between 1% and 15% depending on the severity. If you have enough of these bugs it is entirely possible that you may not score very many marks overall. I want robust bug free code that also validates all user input to make sure it is sensible in nature.

Also note that the percentage listed after the bracket is the maximum mark you can obtain if you complete that many brackets without error. Everything in all brackets is mandatory.

4 Plagiarism

Be aware that we take plagiarism very seriously here. Plagiarism is where you take someone else's work and submit it as if it was your own work. There are many different ways plagiarism can happen. I will list a few here (this is not exhaustive):

- Finding something similar online (full implementation or tutorial) that does the same job and submit that.
- Finding something similar online (full implementation or tutorial) and transcribing (i.e. copying it out by hand)
- Working together on an individual assignment and sharing code together such that all implementation look the same.
- Getting a copy of someone else's code and submitting/transcribing that
- Doing any of the above and attempting to conceal it by moving functionality around and renaming functions and variables.
- Paying someone to do your assignment
- Logging into someone elses Moodle account, downloading their assignment and uploading it to your own Moodle account.

I've had to deal with many cases of plagiarism over the last six years so I can spot it and diagnose it easily, so don't do it. To prevent plagiarism include but not limited to the following:

- Do all your code by yourself
- Don't share your code with anyone, particularly if anyone looks for a copy of your code for reference.
- Don't post your code publicly online. Remember the use of GitHub, Gitlab, BitBucket etc is prohibited.
- If you need to find information online only query about very specific problems you have don't look for a full assignment or howto.

- Change the default password on your Moodle account. The default password can be determined if someone is connected to you through social media or they get one or two details from you.

Be aware that if you submit your assignment you accept that you understand what plagiarism is and that your assignment is not plagiarised in any way.

5 Coding Brackets (80%)

1. Bracket 1 (10%)
    - Get the shell of the application with login/logout working
    - Generate a model that will store the information of a GPU: name, manufacturer, date issued.
    - Add the six properties listed in the introduction to the model of a GPU.
    - the GPU name should be the key for each GPU in the database.
2. Bracket 2 (20%)
    - Build a UI form that will enable the user to add a GPU and all its information.
    - When the form is submitted the GPU should be added to the database.
3. Bracket 3 (30%)
    - Prevent the overwriting of an object that is already in the database. **Bracket Failure if object overwritten**
    - Display a list of GPUs that are currently in the database by name only.
4. Bracket 4 (40%)
    - Make the GPU name list a set of hyperlinks.
    - When a GPU name is clicked it should go to a seperate page showing the information and features for that GPU **Bracket Failure if not on a seperate page**
5. Bracket 5 (50%)
    - Enable editing of a GPU in the database.
    - The form or editing a GPU should be prepopulated with the existing information of the GPU in the database.
    - Editing must be done on a different page **Bracket failure if not on a seperate page**
6. Bracket 6 (60%)
    - Enable the user to select the features to query by using checkboxes (or similar)
    - enable querying of the database using a boolean combination of user selected features.
7. Bracket 7 (70%)
    - Add in a form that permits the user to choose two GPUs for comparison purposes.
    - When the comparison is triggered a seperate page should be displayed **Bracket Failure if not seperate page**

- The feature by feature comparison of both GPUs should be shown on the seperate page.

8. Bracket 8 (80%)
    - UI design: well thought out UI that is easy and intuitive to use.

6 Documentation Brackets (20%)

NOTE: Documentation should be around 1,700 words in length total

9. Bracket 9 (15%): Document every method in your code from a high level perspective. i.e. give an overview of what the method does. Do not copy and paste code you will be penalised for this.
10. Bracket 10 (20%): Document every datastructure and database design you have used in your code and why you chose them. You do not need to provide an E-R diagram for database designs.