BSCH - CSP Repeat Assignment 2020

Building an application for managing a collection of images with Google App Engine

Assignment Information:

Course:	BSCH
Stage/Year:	4
Module:	Cloud Services and Platforms
Semester:	2
Assignment:	Repeat Assignment
Date of Issue:	TBC
Assignment Deadline:	TBC
Assignment Submission:	Moodle, two components: Code and Documentation
Assignment Weighting:	50%

Introduction:

Note: read the whole assignment brief first before implementing it. The brief contains very important information.

In this assignment you will be tasked with building an image gallery system. You will be required to keep track of the galleries and images that a user will upload. You will also be required to do some housekeeping tasks on the images and galleries towards the end of the assignment.

Make sure you email me and inform me that you are taking the repeat assignment.

Note that Google App Engine does not permit recursive object relationships i.e. you cannot store a directory as part of another directory. You will need another mechanism to represent directories. The suggestion here would be to use the user id combined with the full absolute path of a directory. You will also be required to implement file upload and download functionality along with some other more advanced functionality.

Submission and Penalties

You are required to submit **two separate components**:

- A copy of **your complete Google App Engine project**. The accepted archive formats are: zip, rar, 7z. 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 will not count as documentation.

There are also **penalties** you should be aware of:

- Code that fails to compile will incur a 30% penalty before grading. At this stage you should be completely capable of producing fully compiling code. I should 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 application to work. If you are not sure if a library is outside the SDK please ask beforehand.
- The standard late penalties will also apply.
- If your code is available on a publicly hosted repository (e.g. GitHub) it will **not be graded at all**. Should you need access to a remote git repository only a the college provided GitLab will be permitted (gitlab.griffith.ie)

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, the subsequent brackets will not be considered for marking. You should be well capable of producing strong and generally robust software by now. For example if you fail the fourth bracket then brackets five, six, and seven will not be marked. Documentation will still be marked independent of code.

Marks will also be removed for the presence of bugs anywhere in the code and this will incur a deduction between 1% and 15% depending on the severity. If you have enough of these bugs it is entirely possible that you may not score many marks overall despite answering all brackets. 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** before the bracket is the **maximum mark** you can obtain if you complete that many brackets without error.

Coding Brackets (70%)

No	%	Coding Brackets
1	10	• Generate an application shell with a working login/logout system • Generate models to represent users, directories, and files
2	20	• When a user logs in for the first time, generate a user object and store it. This should be retrieved when the user logs in subsequently • Enable the user to add galleries through a form. You will fail this bracket if two galleries of the same name can be added.
3	30	• Enable the ability to edit the names of the galleries (bracket failure if same name as another gallery is allowed) • Enable the ability to delete a gallery (bracket failure if a non-empty gallery can be removed)
4	40	• Enable gallery name to be clicked on and transfer the user to another page. • On the gallery page show a form for uploading an image, make sure to restrict this to JPG and PNG formats only • Include a link that will bring the user back to their list of galleries
5	50	• When images have been upload arrange them in a grid format on the gallery page • Include the ability to delete an image
6	60	• Include the ability to detect duplicate images in a single gallery • Include the ability to detect duplicate images in multiple galleries
7	70	• Modify the list of galleries to be arranged in a grid like format and show the first image of each gallery alongside the name of the gallery

Documentation Brackets (30%)

No	%	Documentation Brackets
1	15	Document why you designed the UI the way you did. This should detail your choices in widget layout and position and how they make user interaction easier. Examples of what I am looking for are as follows: • *The login button was placed at the top right as this is where it is placed in many web applications.* • *The colour scheme that was chosen to avoid the main form of colour blindness and produce high contrast for the visually impaired.*
2	30	Give a high-level description of every method in your Python code. You should also document the data structures you have used and why they are used. **Note**: there should be no copying and pasting of code here.