A03: Building a basic implementation of Twitter

January 7, 2019

1 Assignment Information

Course:	MSCC/MSCBD
Stage / Year:	1
Module:	CC
Semester:	2
Assignment:	Assignment 1
Date of Issue:	2018-02-04
Assignment Deadline:	2018-05-05 @ 23:55 (End of week 12)
Assignment Submission:	Upload to Moodle
Assignment Weighting:	20% 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 a basic replica of Twitter on top of Google App Engine. You will need to build the basic facilities for creating and searching through tweets and users.

Users should be able to follow other users on the application. They should also be able to see which users are following them. Finally they should see a timeline of the most recent 50 tweets of who they are following (including their own tweets) in reverse chronological order.

It should also be possible for users to edit and delete their tweets and to create tweets that have an image attached to them in either JPEG or PNG format.

NOTE: that you are also required to show me a working example of your brackets in labs. I will only consider marking those brackets that I've seen in labs. If you have only shown me the first four brackets in a lab I will not consider the rest.

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 with at least two commits 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

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 Coding Brackets (70%)

1. Bracket 1 (10%)
    - Write the shell of an application that has a working login/logout service.
2. Bracket 2 (20%)
    - Write a model of a user that contains a list of tweets and has a list of people who they are following and people who are following them.
    - If this is a first time log in ask the user to select a username and set this in their model.
3. Bracket 3 (30%)
    - Enable the editing of a user's information the username should not be editable.
    - Enable the addition of a tweet and restrict to 280 characters.
4. Bracket 4 (40%)
    - Enable the ability to search for usernames.
    - Enable the ability to search for content in tweets.
5. Bracket 5 (50%)
    - Given any user name show a profile page for that user showing their basic information, their last 50 tweets, and a button that permits the user to follow or unfollow that user.
    - Enable the ability for a user to start following another user.
    - Enable the ability for a user to stop following another user.
6. Bracket 6 (60%)
    - Generate a timeline for the user that will contain and display the last 50 tweets from their following list in reverse chronological order (this must include the current user's own tweets).
    - Enable the ability to edit a tweet.
    - Enable the ability to delete a tweet.
7. Bracket 7 (70%)
    - Enable the ability to upload images to blobstorage (restrict to jpeg/png only) and link it to a tweet.

5 Documentation Brackets (30%)

8. Bracket 8 (0 to 10%): 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.

9. Bracket 9 (10 to 20%): Give a high level description of all models and datastructures used. Explain your choices and reasoning for each model and datastructure used.

10. Bracket 10 (20% to 30%): 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'm looking for would be the following

    - Labels and entry boxes have a small horizontal distance to indicate they belong together.
    - The colour scheme was chosen to avoid the main form of colour blindness and produce high contrast for the visually impaired.
    - **NOTE Do not include instructions for how to use the UI. You will lose marks if you include such** instructions.