A03: Building a basic implementation of Twitter

February 17, 2022

1 Assignment Information

Course:	MSCC/MSCBD
Stage / Year:	1
Module:	CC
Semester:	2
Assignment:	3 of 3
Date of Issue:	2022-02-17
Assignment Deadline:	2022-05-08 @ 23:55 (End of week 12)
Assignment Submission:	Upload to Moodle
Assignment Weighting:	24% 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 10 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: 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 groups listed below. These are meant to be completed in order. Groups must be completed in full before the next group will be evaluated. Completed will mean that all tasks in the groups are visible and testable. If a single one is not visible and testable further groups will not be considered. e.g. if there are four tasks in Group 1 and task 3 is skipped or not visible or testable then Groups 2, 3 and 4 will be ignored. Documentation

will be treated separately irrespective of how many Groups you have completed.

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. Please be aware of the major bugs section. If any of these bugs are present in your application you will lose 20% for each one up to a maximum of 60%

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
- 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%)

- Group 1 (20%)

  1. Write the shell of an application that has a working login/logout service.
  2. 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.
  3. If this is a first time log in ask the user to select a username and set this in their model.
  4. Enable the editing of a user's information the username should not be editable. A user should have a name ands a short profile (280 characters max)

- Group 2 (40%)

  5. Enable the addition of a tweet and restrict to 280 characters.
  6. Enable the ability to search for usernames.
  7. Enable the ability to search for content in tweets.
  8. 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.

- Group 3 (60%)

  9. Enable the ability for a user to start following another user.
  10. Enable the ability for a user to stop following another user.
  11. 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).
  12. Enable the ability to edit a tweet.

- Group 4 (80%)

  13. Enable the ability to delete a tweet.
  14. Enable the ability to upload images to blobstorage (restrict to jpeg/png only) and link it to a tweet.
  15. UI Design: well thought out UI that is easy and intuitive to use.

- Major bugs (20% reduction for the presence of one of these bugs upto a maximum of 60%)

  – none

6 Documentation Brackets (20%)

NOTE: Documentation should be around 1,500 words in length total

1. (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.
2. (10 to 20%): Document every datastructure and model you have used in your code and why you chose them.