FIT2107-S2-2020 Assignment 2

Automated Unit Testing, Whitebox testing and Continuous Integration

This assignment will be completed in pairs

Marks:	20% of your final marks.
Due Date:	Week 9, 15/10/2020, 11:55 pm AEST (9:55 pm Malaysia Time).
Submission:	READ THE INSTRUCTIONS CAREFULLY. 1. Incremental commits on GIT on the repository provided. We will use the last commit for marking your assignment. 2. As per the faculty policy a copy of your final commit on GIT for the assignment MUST be uploaded on Moodle. 3. Download your final commit for the assignment on the GIT, convert it to a ZIP or RAR file (if not done automatically), and upload it on Moodle under the Assignment 2 submission link. The zip file should be named using the repository name (or team name) i.e. REPOSITORYNAME.zip.
Lateness:	Late penalty of 10% per day after the due date, including the weekends.
Student Statement:	An electronic student statement MUST be accepted as per the faculty policy while submitting your assignment. Once you press the submit button, the electronic student statement will appear, accept it and press continue. If the assignment is left in a DRAFT mode, it will not be accepted for grading. All team members MUST accept the student statement.

Goal

In this assignment, you will create a small program in Python 3 (or another language, if you really insist), and test it using unit testing and mocking frameworks. You will also set up continuous integration, so that testing occurs as you update your source code repository. In addition, you will apply Whitebox testing techniques to measure the coverage of the code.

Typically, developers are expected to write unit tests for their own code, hence the need to write your own test suite. However, you are not going to be marked on the quality of your implementation - you just have to have one to test against; However, your implementation

should be complete. Comprehensive unit tests and fixing bugs that you find as you commit changes should result in a module that is as close as possible to functionally correct.

Git Infotech

We will be using Monash FIT's private Gitlab repository at Monash Infotech, accessible at:

https://git.infotech.monash.edu.

All changes must be regularly committed to this system. There are instructions for using Git and GitLab on Moodle.

Your username on this Gitlab server will be your Monash authcate login. To commit to the repository, you must create an "access token" and use that instead of a password (just type it in where it asks for your password – normally with git clients). Git via ssh does not work on this server, unfortunately.

Groups & Repositories

This assignment will be done in groups of two (or, if there are uneven numbers and there is no alternative, groups of three). The groups are same as created for assignment 1 so you do not need to create groups again. If you want to change a group, contact your workshop facilitator. If there will be a genuine reason, we will make a change.

We will create a repository on a Git Infotech Server for you. Please do all project work in the repository we have created for you. **DO NOT** create your own repository, or any other git hosting service, for this assignment. **DO** make sure you commit to the repository frequently. You will be assessed on whether you commit to the repository appropriately. Optionally, you can add TAGS to the commit that you want us to mark for assignment 2. We will only mark the MASTER branch.

The repositories will be set up with a "skeleton" module and tests that you will need to extend, and with a functioning but rudimentary Git Infotech CI setup. The current setup runs all the tests in `CalendarTest.py`. If you want to add other steps to your CI (for instance, run a coverage tool) you will have to edit `gitlab-ci.yml`.

The Calendar Module

When you clone your git repository, you will note that it contains two files - `Calendar.py` and `CalendarTest.py`. These files are skeletons that you must complete for the assignment!

To access the Google Calendar API, you must have to register an API key. The details are provided here:

https://developers.google.com/calendar

You must use Google Calendar Python API to implement and test the functionalities below. You will lose marks for not using the correct API.

The module should have tests and functionalities for following features.

1. As a user, I can see the events, reminders, and notifications – notifications are notices such as out of office notice - for at least 5 years in past from the today's date.
2. As a user, I can see events, notifications, and reminders for at least next two years (in future).
3. As a user, I can navigate through different days, months, and years in the calendar so that I can view the details of events. For example, if the year 2019 is selected, all events, reminders, and notifications will be shown. On selecting the specific event, reminder, and notification, it displays the detailed information.
4. As a user, I can send invitations to attendees on student.monash.edu email address. The system does not support sending invitations to attendees with private emails such as Gmail, Outlook etc.
5. As a user, I can search events, reminders, and notifications using different key words.
6. As a user, I can delete events, reminders, and notifications.

You do require an authentication with Google Calendar API first time to access it. There is no need to write test cases for authentication.

Extra Functionalities & Tests for the Groups of Three Students ONLY

In addition to the features mentioned in the Calendar Module section, the groups of three students must write unit tests for the following additional features.

1. As a user, I can edit events, reminders, and notifications.
2. As a user, I can cancel events, reminders, and notifications.

Using Another Language

If you really want to try using another language for this assignment, we will not stop you, however:

- You will need to find a Google Calendar API for the language of your choice. It is NOT feasible to write your own functionality yourself without using the API.
- You will have to get Gitlab CI for your language of choice working ON YOUR OWN. We strongly recommend, under the circumstances, that you use Python.

Discuss with your facilitator before making a final decision.

Test Strategy

You only need to test the functional correctness of the code that you write the module and run the tests using the `unittest` module, performing continuous integration using Git Infotech.

The strategy you use to select your test cases is up to you. You should explicitly document your test case selection strategy in a Markdown file called "`test-strategy.md`" and add it to the repository.

We will be assessing you both on the reasonableness of your test strategy, and whether you have actually applied it. We expect you to try to use the strategies we have discussed, possibly including the black-box strategies.

You may assist your marker by adding a section to your test strategy document explaining how you got your test cases from your strategy. While this is not compulsory, remember that if we cannot see how your strategy got you your test cases, we will mark you down for it!

Coverage

To measure statement and branch coverage as part of your test strategy, there is a standard Python tool called `coverage.py` that can help you achieve this. The documentation and usage instructions are accessible at:

https://coverage.readthedocs.io/en/coverage-5.2.1/.

If you choose to use it, if you can, you should integrate it into your CI scripts so that it runs along with the tests and generates the report.

Mocking

The point of this exercise is to test your code, you should "mock out" calls to the Google Calendar API in your tests. You should use the Python mock module for this where possible.

Mocking can be quite complicated, so use the simplest possible mocks at first, and if and when you have time you can make your mocks more featureful to test your code more fully.

Your mocks should return sensible dummy data to allow your tests to test your code effectively. You may also consider whether you should use mock's facilities to check the way the API is called.

Unit Tests

Based on your test strategy, devise a test suite for the module. It is recommended you do this before or as you develop your module (TDD), not afterwards. You should use your unit tests to fix bugs in your module.

Continuous Integration

You must set up and use GitLab's Continuous integration to run your tests. Every time you commit to the master branch, the (appropriately mocked) tests should be run on Git Infotech.

You will have to figure out how to ensure that the Google Calendar module, and any other submodules is installed on the Gitlab CI docker instance when you run your tests. There are two options - using setuptools or invoking pip as part of your test script!

Demonstrations

You will be asked to demonstrate your working program and unit tests in your week 10 tutorial. This will be a brief process and is purely to verify that your program and unit tests run on your machine.

Your unit tests are still expected to run on the CI server, and your demonstrator may need to run your program again on their own machine. This will be assessed by the marker later.

Working as a Team

You are expected to work as a pair and contribute close to equal amounts on the group section of this project.

The primary way we will assess this is by examining your repository and assessing the number and nature of commits by each member of the group. Therefore, it is important to commit your work - do not get your partner to commit on your behalf.

If you genuinely work on something together (at the same time), please include the text (PAIR WORK) in the comments for the commits where this occurs. ONLY do this if both of you worked on the material at the same time (preferably in the same place).

Plagiarism

Please do not copy code from other teams. You can use code you find on the Internet, provided that you are not violating copyright and you acknowledge where you got it from, AND it's not a solution to a large part of the assignment. We will use a code similarity tool to find code similarities if needed.

If you wish to use a downloadable Python module other than Google Calendar or any of their dependencies, please ask teaching staff first.

If you find some code on the internet that IS a solution to a large fraction of the assignment, please contact teaching staff.

Special Consideration

If a student faces exceptional circumstances (serious illness or injury, family emergency etc.) that prevent them completing the assignment, they may apply for special consideration according to the policy and procedure on the Faculty website:

https://www.monash.edu/exams/changes/special-consideration.

Assignment Forum

You can ask questions on the discussion forum, which the Lecturer will monitor and respond to daily (and at least once on weekends). All students should monitor this forum - I may clarify aspects of the assignment on the forum and any such clarifications are considered "official". You may of course also email the lecturer or arrange to see him virtually if you prefer.

Marking Criteria

- Completion of module according to specifications.
- Appropriateness of test case selection strategy.
- Application of strategy in actual selection of test cases.
- Quality of test cases (including quality of test case code).
- Effectiveness of mock usage.
- Appropriate usage of CI system.
- Level of individual contribution (should be close to equal between partners)
- Quality of test analysis.

- Readability of test analysis.

A detailed rubric is available on the Moodle submission link.