Code review report

Assignment 3 - Part C

Done by:

Elaf Abdullah Saleh Alhaddad 31063977

Sutulova Tatiana 30806151

Content

1. Introduction

Code review is an activity conducted by a group of people that review the program code with the presence of the author of the code. In this activity the reviewers identify defects that can be eliminated and the improvements that can be made to our code. Code review is a beneficial practice for both reviewers and developers, since they are able to share their own knowledge about programming techniques and learn new skills from others. After conducting a review on each other's code and discussing our weaknesses, we needed to hear from the other group, who have completely different coding styles in order to build an entire understanding of issues that we had in our Calendar and the tests that we created for it.

2. Meeting details

Date	11/02/2020 - 3:00 PM	**Duration**	2 Hours
Members	• Elaf Abdullah Saleh Alhaddad - Author • Tatiana Sutulova - Author • Tah Wen Zhong - Reviewer • Ethan Hor Sheng Jian - Reviewer		

3. Process of peer review

The meeting was conducted through the discord due to the global pandemic and all the members of both teams attended it. The reviewers were responsible for pointing out defects in our code and functionalities, whereas our role as developers was to explain the code, understand and note down the defects that were pointed out by the reviewers.

Before the start of the meeting, the reviewers have already looked at our code and tested it by comparing it to their checklist requirements. Hence, they have already noted down the failures and errors in our code that they discussed with us during the

meeting. We started the meeting by running the program and presenting the functionalities that we implemented. Most of the issues raised by the reviewers were about the functionalities that are missing from the program rather than the source code itself. After running the code, the reviewers went through the source code to inspect it and pointed out lines that were not following good programming practices. This includes documentation, naming conventions and code duplication. When pointing out errors, the reviewers made sure to include a possible improvement that we could implement to fix our code.

4. Defects summary

Defect Id	Reviewer	Defect Category	Defect found	Name of the defect in the issue tracker	Description	Possible improvement
#1	Ethan Hor	Method	**get_event_by_keyword(..)**	(Calendar.py) Searching events with date and time as the keyword does not work.	The issue is that the keys used to search for date and time in the database are wrong and therefore the KeyError is always raised independently on what kind of input the user gives.	The possible way to improve that is to remove the date and time as keywords. It may be considered redundant because there is a functionality search by timeline which is responsible for that feature.
#2	Wen Zhong	Method	**delete_event_reminder(..)**	(Calendar.py) Program can not delete specific reminders but delete all reminders	When there are multiple reminders that belong to an event, the user does not have the option to delete only one reminder. Instead they delete all the reminders for the event.	Add the ability for the user to delete only one specific reminder for an event when an event has multiple reminders. This can be implemented by looping through all the reminders and giving them unique ids (number). After that the user will choose the id of the reminder

						they want to delete.
#3	Ethan Hor	Handling Errors	**get_event_by_timeline(..)/ delete_event_reminder(..)**	(Calendar.py) The functions do not properly handle errors.	When the user inputs a value that would raise an error the code will crash. This is because the code will raise an error but this error will not be caught.	Add code that will catch the raised errors and allow the user to re-input the value instead of terminating the application.
#4	Wen Zhong	Method	**get_event_by_keyword(..)**	(Calendar.py) Keywords must completely match whatever is being searched when searching for events	The keyword should not be the entire complete name of the event, the keyword is a certain part of the event name, location or the organizer email. As for our code, when the user is searching by keyword they must type the entire word as it is in the calendar API.	This may be improved by making a code that will prompt the user for the keyword. Afterwards, it will check if the inputted keyword corresponds to any part of the name/organizer email/location, meaning that the keyword may be just a small part of the information of the event (e.g part of the title/location/organizer email)
#5	Ethan Hor	Commenting and documentation	**get_event_by_keyword(..)/ represent_events_reminders(..)/ get_event_by_timeline(..)**	(Calendar.py and CalendarTest.py) Insufficient Commenting	The functions do not contain any in-line comments that explain the algorithm of the functionality making it hard for others to understand the code at first glance.	Adding more in-line comments that will explain the algorithm of the codes.

#6	Wen Zhong	Method and handling errors	**get_events_reminders_in_the_future(..)** / **get_past_events(..)**	(Calendar.py) Functions fail to handle events without titles	When there is an event saved in the calendar API that does not have a title the code will crash because the code does not handle events that have no title name.	Adding an if statement that will check if the event contains a title. If it doesn't the event can be ignored and not displayed in the Calendar.
#7	Ethan Hor	Application User Interface	**main(..)** / **get_events_by_keyword(..)**	(Calendar.py) UI is not detailed enough	The prompts for the user input are not descriptive enough. For example, when the user is asked to input the keyword, the app does not explain what format it should be.	The issue may be solved by adding more descriptive messages at the user interface, providing an explanation of what format is requested for all the input fields.
#8	Ethan Hor	Code duplication	All tests in the test file CalenarTest.py	(CalendarTest.py) Test functions contain redundant data.	In the hardcoded events that are used to mock the api, there are certain properties inside the dictionary that are not used by the functions. Having these properties is redundant.	The resource representations of the events should be reduced to only contain the required properties to be tested.
#9	Wen Zhong	Wrong testing method	**test_keyword(..)**	(CalendarTest.py) Tests depend on imports with inconsistent values used for testing	The test for the searching of an event using the keyword feature contains a variable whose value is inconsistent as it varies	The value, which is dependent on an external factor, should be replaced with the hardcoded one.

					based on external factors that are out of the program's control, which in this case, is time.	
#10	Wen Zhong	Missing Tests	**test_keyword(..)**	(CalendarTest.py) Missing tests for the searching an event using a keyword feature	The testing strategy was based on the assumption that all the keywords will be treated the same way. It was mainly aiming to test the various external options.	The testing strategy must include tests for various types of inputs.
#11	Ethan Hor	Code duplication	All tests that test **get_event_by_timeline(..)** in CalendarTest.py	(CalendarTest.py) Certain functions can be merged as they have similar test case scenarios.	The tests that are used to test **get_event_by_timeline(..)** are split into multiple methods. This is redundant because the lines of code are identical with only one or two lines that are different.	All the methods may be combined into a single one, which will have two arrays with hardcoded inputs and expected outputs and a for loop that will iterate through them simultaneously. The loop will be testing the new possible combination every iteration.
#12	Wen Zhong	Naming conventions	Found repeatedly around both Calendar.py and CalendarTest.py	(Calendar.py and CalendarTest.py) Inconsistent naming convention	Variables named through different delimiting conventions including Snakecase and Camelcase	Have a fixed naming convention. For Python it is usually lowercase with words separated by underscores.

5. Review outcome

When we conducted our own code review, we were able to identify defects in our source code that were strictly based on good coding practices, since that was what our review checklist was based on. This allowed us to identify insufficient comments, code duplication, wrong naming conventions and other kinds of coding standards violations.

However, when our peers conducted the review on our code they were mostly focused on identifying the defects in functionalities that are identifiable by the blackbox testing, not by reading the source code. After they properly blackbox tested the code, they were relating the problems to our source code by pointing out where the problem might be and suggesting possible improvements. For instance, our functionality for searching for an event by using date and time in **get_event_by_keyword(..)** was not giving the expected output. We did not identify this error as the source code did not violate any programming practices, but our peers' approach allowed us to find more significant issues in our functionalities rather than our coding methods.

The code review activity was an important experience, since it allowed us to understand what problems we had in our coding style, which were resulting in significant issues in the actual code and, therefore, functionality. Reviewers possessed knowledge about programming practices and the code base that we did not know before due to lack of experience and it helped us to eliminate the structural errors in our Calendar code as well as in our own coding habits.