MONASH University

Assignment 3

FIT2107

Part A: Checklist Preparations (Individual)

Done by:
Sutulova Tatiana 30806151

Code and testing		
Readability and understandability		
#	Description	Y/N
1	Is the code broken down into small chunks, so it is easy to understand the role of every method, class, and function without keeping in mind too many steps?	
2	Are the names of variables descriptive and following naming conventions, which ensures that the role of each variable is understandable?	
3	Are the comments for methods, classes, and functions descriptive enough to understand its purpose without looking at the code? For example, for the function, does it include description of: - parameters passed to the function, - the value that the function returns - functionality - exceptions raised (if any) - time complexity	
4	Are there in-line comments that are explaining the complex processes that may be not obvious for people who did not take part in writing this part of the code?	
5	Are there redundant comments that confuses the reviewer more than helping to understand the code?	
Maintainability		
1	Are there any hardcoded values, which will be difficult to replace if the code is to be changed?	
2	Are there repetitive parts of the code that will lead to difficulties if these parts are to be improved/replaced?	
3	Are there any parts of the code, that do not benefit the program or not used at all?	
Speed and performance:		
1	Are there any cases when the API is called more times than needed, which results in increase of processing time?	
2	Does the code do what it is supposed to do? For example, do all the choices in the menu correspond to the right functionality?	
Complexity		
1	Are the proper python inbuilt functions used in the code, which ensures that the code is made as simple as possible without reimplementing existing python functionality?	
2	Is there any possible way to implement certain functionality with the simpler approach, which is more understandable and has a better performance?	
Reliability:		
1	Is the code failure-tolerant, meaning that it handles various failures not related to the actual code, such as wrong user input, API failures, requests for non-existing data, etc.?	
2	Are the error messages user-friendly, properly explaining the user what the issue is and hinting what the user should do to resolve it (if not obvious)?	
Testing:		
1	Is the same functionality being tested more than once, which leads to numerous repetitive tests?	
2	Is the API being called in any parts of testing, which leads to a high processing time?	
3	Are there any parts of the code, any functionalities or lines of code that are left untested?	