2025 November 5

MARKERS' COMMENTS ON STUDENTS' PYTHON PROJECT SUBMISSIONS

Ref: INF1002 Programming Fundamentals Python Projects, by Module Lead, 2025 Sept.

Students in teams are required to submit for marking three deliverables pertaining to their Python projects: Written report, Video (YouTube link or mp3 file), and GitHub access link. Marking of student submissions was difficult due to substandard composition of a majority of the deliverables. This memo is intended to explain the level of composition expected, so that future deliverables will be of a higher standard.

Comments on Report deliverable

General comments

The purpose of the Report deliverable is to record your team's academic effort in conceiving & constructing a software program that realizes a product or service, i.e., *context problem*, belonging to a third party. "Realizing the context problem" means producing the software that solves the problem, a.k.a., *software solution*. In order for the team to do the software solution, it must a) understand the context problem — this involves research of literature and possibly first-hand observation — as well as b) know well the software development process which includes programming. The team is using Python programming language to effect the realization. Having been schooled in Python, the report must show using explanations and artifacts, that team members have acquired sufficient knowledge and programming skill in Python.

If the perspective and theme of this report still seems too generic, then let the *SDLC* which is the foundation of all SWE underpin the report. That is, every stage of the SDLC is reflected in the sections of the report, though not literally as the titles of the sections.

Last point is that the report must reflect a level of uniqueness consistent with the context problem, your logical reasoning, and solution production, as well as apply previously learned aspects of SWE and adhere to SIT's standards for academic report writing.

Specific comments

1. 'Title page' is mandatory. It contains at least four items for referencing report, namely its: type or project, unique moniker, author(s), and effective date.
2. "Table of contents (ToC)" section is mandatory.

3. “Abstract” section, a.k.a., “Executive Summary”, reflects the report’s purpose and ToC. It briefly identifies what information the report will provide to its reader.

4. “Introduction” section reflects the ‘*Requirements* stage of SDLC’. It is for briefly specifying the context problem (product or service) that your Python program is going to realize. Related works or literature can be ignored if your research turns up blank.

5. “Methods” section reflects the ‘*Analysis & Design* stages of SDLC’. Accordingly, it has a minimum of two subsections: Analysis sub-section and Design sub-section.

5a. In the Analysis stage of SDLC, the developer gathers and studies the business processes, scenarios, algorithms, formulas, tools, libraries, etc. that are relevant to the context problem, and necessary for understanding the breadth and depth of it. In this sub-section, the team lists all of these that it expects are necessary to use for coming up with a design of the software solution.

5b. In the Design stage of SDLC, the developer produces high-level design and low-level design of the software solution. In the report sub-section, the team produces a minimum of three artifacts: a) “System diagram”, a.k.a., ‘*Architecture*’, b) “Algorithm”, a.k.a., ‘*software solution process*’, and c) ”Main tasks”, a.k.a., ‘*work breakdown*’. All are high-level; your team can ignore the low-level design effort.

5b.1. Architecture specifies the intended modular structure of the software solution. Only a diagram form is acceptable. If you have the training, draw it using *UML* standard. If not, draw an artistic illustration of your program, with boxes or icons representing modules, lines representing connection between modules, and clouds representing external programs on the web.

5b.1.1. Modules are either a) small collections of Classes related by related by structure or behaviour, or b) files (one each header and footer encapsulating the whole content). Crucially, all of the modules' names must be nouns not verbs.

5b.2. Software solution process specifies the succession of steps that your program is intended to take to complete its execution from precondition to post condition of the outside/environment for your program. (It is analogous to the ‘scenarios’ you came up with in ‘Use case descriptions’.) Its form is either text (outline or table) or an UML Activity diagram.

5b.2.1. If data pre-processing is to be conducted, it is a step in the software solution process.

5b.3. Work breakdown cross-references all the units of work belonging to the Methods section with all the team members. Its form is either text (outline or table) or a Gantt Chart.

6. The code and dataset in GitHub both reflect the ‘*Coding* stage of SDLC’. In the report, reference GitHub by its link and do not insert screenprints of the code. Also, ensure the link is active for visitors.

7. "Result and insight" section reflects '*Testing* stage of SDLC'. It explains how you *verified & validated* (V&V) the software solution. This includes *white* and *black-box* testing, relating time & space complexity with time taken, accuracy of the outputs (graphs), readability, speed of page rendering, and other performance criteria.

8. Instructions for "Conclusion" and "References" sections were satisfactory.

9. 'Appendices' contain detail that is associated with a part of the main body of the report, yet is suppressed from the main body for the sake of enhancing its readability. It is at author's discretion whether that detail is important and necessary to provide.

9.1. Appendices must be labelled. The label is positioned at the centre top of the first page of each appendix. If there is only one appendix, it is simply labelled "Appendix". If there are multiple appendices, assign a unique capital letter for each, e.g., "Appendix A", "Appendix B", "Appendix C", etc.

9.2. Teams must declare their use of AI. SIT has published instructions on how to cite the use of AI,

https://libguides.singaporetech.edu.sg/citation/aichatbot#s-lg-box-22537884

If AI is used, teams must do a citation, that must be in the report, preferably in an Appendix.

9.3. Again, if GitHub access is provided, there is no reason to put screenshots of code into an Appendix.

****

Please adhere to page count set in the project guide

****

Comments on Video deliverable

The purpose of the Video deliverable is to show that the completed software solution works without any sudden crashing, freezing, or outputting nonsense. The video reflects the 'User Acceptance Testing' (UAT) activity belonging to the '*Testing* stage of SDLC'. To get the right impression in your video, the team must imagine that if the video does not convince their customer who contracted for the software solution that the solution is operable, available, reliable, and meets all of the stated requirements, then your team will lose the contract and their employment!

The customer wants to see the software solution operate in all the behaviours specified in its requirements. So, the video shows a requirement, followed by solution doing that requirement; and so on for the next requirements until the entire suite of requirements have been demo'd. If you can do a split screen, that would be the best. If not, make do with stop-start-stop-start, and repeat.

The customer also doesn't want to see: a) what code structure produces what behaviour in the solution, b) any unit and block testing, and c) any special things that the team may have added to the solution that the customer is not aware of. Why? Because the customer is interested only in finished product, and not how the developer achieved it. (To put it crass terms, the customer pays for the privilege of selective ignorance.) Also, the developer never adds new things into the requirements without full approval of the customer. This includes *load* and *stress* behaviour; if these are not in the requirements, then don't demo them and hope the customer doesn't notice them until way later.

****

Video must be not more than 10 minutes. If you need more, your lab supervisor will decide if you can do so.

****

Comments on GitHub deliverable

The purpose of GitHub is to facilitate the inspection of your code. As stated earlier, the code and data set both reflect the '*Coding* stage of SDLC'. The low-level inspection provides assurance to the teachers that the team has learned the right programming knowledge and skills, e.g., naming convention, dynamic inputting, modularity, maintainability, and extendibility.

Crucially, GitHub must be accessible to visitors. Either the site is made public, or invites-to-join are sent out from the GitHub to the academic markers so that they can have permanent access.

The "Readme" item must briefly specify the project, the team, the organization of the GitHub, and the organization of the files. The Architecture and Algorithm diagrams may also be provided.

Final remarks

Please ensure that you understand all of the content herein, so that you don't repeat the same difficulties in your next project. If you wish clarification, you are invited to speak the lab supervisors.

Kevin Anthony Jones, PhD, Senior Fellow AHE
Associate Faculty, ICT Cluster