SE 3XA3: Development Plan

Team #16, Lines Per Minute (lpm)
Jay Mody - modyj - 400195508
Jessica Lim - limj31 - 400173669
Maanav Dalal - dalalm1 - 400178115

April 3, 2021

Table 1: Revision History

Date	**Developer(s)**	**Change**
February 3, 2021	Jay/Jessica/Maanav	Initial document write-up.
February 5, 2021	Jay/Jessica/Maanav	Gantt Chart, Document completed.
March 25, 2021	Jay/Jessica/Maanav	Project Review Added. R1 Document Complete
April 3rd, 2021	Jay/Maanav	Added next steps.

1 Team Meeting Plan

Meeting Details

Team meetings will take place during Lab sections after our assigned work has been completed. As a buffer, we have also added an hour before our Thursday tutorial, in case Tutorial assignments take up the entirety of the 2 hours. Weekend meetings may be scheduled as needed. Meetings will occur within our designated group teams chat (Lab2Groups3xa3 → Group16). All members must be punctual to group meetings.

- Tuesdays 2:30pm to 4:30pm
- Thursdays: 6:00pm to 9:00pm

Secondary meetings will be held on the following days, whenever required:

- Thursdays: 3:30pm to 5:30pm
- Sunday: 1:00pm to 3:00pm

Agenda / Roles

An informal agenda for the meeting will be discussed prior to the meeting, and officially drafted by group member **Maanav Dalal**, to ensure group meetings are productive and succinct. For a given meeting, the agenda will follow **Scrum** guidelines to ensure focused work. He will also update the GANTT chart as needed. This includes covering blockers, what we completed between our last meeting and the current one, goals for the next sprint, and a member-by-member recount of their current progress and confidence in meeting upcoming sprint goals.

2 Team Communication Plan

Primary Communication Platform (for team members): Facebook Messenger Chat

Secondary Communication platform & TA contact: A Teams chat (Lab2Groups3xa3 → Group16) will be used for TA communication & questions.

GitLab will be used for committing code, reviewing code, creating tasks via submitting issues, and resolving tasks via submitting pull requests. (See Section 4 for more details)

3 Team Member Roles

Jay Mody: Technology expert (Python / pip / package creation), Typist, Code Tester
Maanav Dalal: LaTeXpert, CLI design expert, Typist, Code Tester
Jessica Lim: Git expert, Documentation expert, Typist, Code Tester

4 Git Workflow Plan

We will be using the Gitlab platform to organize tasks.

- All major features and goals for our project will be considered **Milestones**. Milestones will be documented on Gitlab as **Epics**.
- Milestones will be broken down into tasks. These will documented on our Gitlab board as **Stories**.
- Any **Story** that requires coding or documentation work will have one associated **Issue** in the repository.
  - Only one individual will be assigned to each issue (other members may help, but the assignee is the primary individual that will be working on the issue)
  - Every single Issue must have an associated **Branch** and **Pull Request**

  - These issues may include several sub tasks that will be documented as a checklist.

- The **Main** branch will only be used for merging pull requests into
  - All code changes will be made on a separate branch, titled based on the issue the branch will address (e.g. fix_symbol_bug)
  - Pull Requests must have the approval of at least one other member before they can be completed.

- **Pull requests** will be created for each **Issue**
  - A **Pull request** should only have one associated **Issue** and **Story**
  - Completed Pull Requests will result in the associated Story being removed from the board
  - Completed Pull Requests will result in the associated Issue to be completed

5 Proof of Concept Demonstration Plan

First, we will demonstrate the ability to install the package via (given a valid python and pip installation):

```
$ pip install lpm
```

From there, we will launch the application CLI via:

```
$ lpm
```

As a minimal POC demo, a single code snippet will be outputed when the CLI tool is started. The user will have the opportunity to type the code snippet, which will be regurgated by the program once they finish. For now, the user's input will not be compared with the code snippet. We just want to demonstrate the ability of the program to fetch the user's input.

6 Technology

Programming Languages: Python (project must work with ~~Python2, Python3, and PyPy~~ Python3)
Makefile: For defining commands for devops (run linting, run formatting, run tests, etc ...)
Package Manager: pip
Package Distributor: PyPI
Version Control: Git
Repository Hosting: Gitlab

7 Coding Style

Coding Style: pep8
Documentation Style: numpydoc
Code Formatting: black code formatter, which conforms to pep8.
Coding Philosophy: The Zen of Python (pep20)
Code Linting: pylint
Versioning: Semantic Versioning 2.0

8 Project Schedule

Our Gantt Chart can be found here. All tasks must be completed in line with the milestone deadlines shown on the Gantt chart. The chart will be updated as further tasks and project requirements are added.

9 Project Review

~~This section will be completed after revision 1~~

Overall, the implementation of the lpm program was a success. The team was able to implement a complete version of lpm. This new program followed the overall structure and functionality of wpm, but included additional features, testing frameworks, and better module implementations.

This re-implementation taught the team every architectural decision is critical to the overall process of implementation. When structuring the lpm codebase, a focus was made to decouple the different modules, to ensure modularization and module hiding between the different modules. Doing so required us to make some changes to the wpm codebase, which was less modular. This slight restructuring essentially required us to rewrite the entire codebase (from scratch). However, this was worth the additional effort as it improved the project's property of **separation of concerns**.

One of the biggest challenges of this project was ensuring consistency between the different documents and the codebase. Extensive documentation was done throughout the project, and traceability matrices were included between requirements, tests, and modules. However, the traceability between the different documents were oftentimes not done up to standard. Since our team was knowledgeable about the system, codebase and documents, we were able to navigate between the requirements, tests and codebase. For R1, more of an effort was made to make clear links between requirements, tests, code modules, and test results.

Given the opportunity to continue development on this project, the next steps would be to:

- Clean up the way data is stored in the user's directory.
- Allow user to add their own code snippets
- Improve the implementation of resize
- Get lpm working on Python2 and older versions of Python3

Overall, our team is happy with the overall lpm product and is proud that lpm is now available on PyPI for anyone to install via pip.