Project Milestone Template

As a Boilermaker pursuing academic excellence, we pledge to be honest and true in all that we do. Accountable together – We are Purdue.

Type or sign your names: Ryan Villarreal, Owen Prince, Mohammed Fashola
Write today's date: 09/20/2021

Assignment Goal

A weekly project update is a normal component of many engineering teams. Such an update has many uses. It helps you understand what you've actually accomplished and whether you are on track.

- It gives you a "paper" record of your accomplishments, to help justify your request for a raise or a bonus at the end of the quarter. It helps you track accomplishments that might not be captured by product-centric metrics (it is easy to measure "lines of code committed", but hard to measure "helped onboard the new intern").
- It informs other teammates or other teams about your areas of expertise, so they know whom to ask when they need help.
- It helps project managers see how things are going, re-prioritize the team's activities, and assign more personnel to shore up difficult areas.

This assignment does not imitate every aspect of such an update, but hopefully it gives you the flavour of the activity. Through the milestone assignments for this course, you have an opportunity to *report* and *reflect* on your team's progress to date.

Relevant Course Outcomes

A student who successfully completes this assignment will have demonstrated the ability to

- Outcome i:
  - Identify and follow an appropriate software engineering process for this context.
- Outcome iii:
  - Experience social aspects of software engineering (communication, teamwork).

Resources

Perhaps relevant are these ones:

- **Postmortems**
  - Postmortems at Google
  - Postmortems at Amazon

Assignment

In your project plan (Milestone 1), you submitted a design as well as a list of weekly milestones. In each intermediate milestone report, you will present:

1. "Substantial" updates to your design (I leave this definition to your engineering judgment).
2. A statement of the tasks your team accomplished by this date.
   a. How did you measure that they are accomplished?
   b. Who did the work? How long did they spend?

3. A comparison of what you accomplished vs. what you planned to accomplish. Are you on track? How were your time estimates?
4. Any changes in your planned timeline as a result of falling behind your initial plan.

If you *deviate substantially* from your timeline, consider attending one of the course staff office hours to discuss the deviation.

You can communicate this information using tables, charts, etc.

If you have working functionality that will add value to ACME Corp., feel free to include a screenshot of a demo. This will build trust with your customer, Sarah, and gives her the opportunity to propose tweaks ("requirements changes").

Grading

These assignments are worth relatively little in the overall Project grade (2.5% each).

This weighting reflects the relative *time* you should spend on the milestone report, but not the relative *importance* of the report. These documents are critical to help you understand whether you are on track to succeed.

Owen:

1. "Substantial" updates to your design (I leave this definition to your engineering judgment).

Owen	The only notable update is the logging framework of the design. We added a logging superclass ("LogWrapper") that each other class inherits from. This allows each class and their methods to print log methods with a single call to a decorator. The framework here is very flexible, reusable and requires little to no effort to extend to other classes.
Ryan	No substantial updates I can think of. We have a framework up and can now correctly take in a file of Github URL's as well as use the "test" and "install" command line items. We can also output in the correct format.
Mohammed	I did not make any substantial updates to my design. We created test skeletons wrapped with code coverage that successfully returns the percent coverage of our program. We can also now successfully install dependencies.

2. A statement of the tasks your team accomplished by this date.

Owen	Set up the inheritance skeleton and class framework, which consists of: ● LogWrapper, to handle logging. Superclass of the other classes. ● CLIHandler, handling all command line input and output. ● Repo, which holds all the metrics for a single repository. ● CalcHandler, a superclass that serves as a template for handling interactions with repository APIs. Create subclasses that override the superclass methods with specifics for handling different APIs ● CalcHandlerGit, a subclass of CalcHandler that overrides CalcHandler with methods that interact specifically with the Git API
Owen	Set up the logging framework via the LogWrapper class, which encapsulates logging functionality. All classes we use will inherit from this class. Contains methods that will set up logging objects as well as get the current values of each class variable and output to the log file. Implements decorators to log class variables before and after methods execute. Any method that needs logging can use the decorator and avoid direct calls to the `logging` library.
Ryan	● Created CLIHandler class and functions ● filled out run.py to correctly parse input and output arguments ● Investigated running our program on the ECN grid ● Ensured PyInstaller would work for our needs ● Updated repo.py to correctly parse input URL's
Mohammed	After gathering all the required libraries needed for our current code implementation. I ran the completed code on my computer and it successfully installed all the dependencies. I then asked my team member, Ryan, to attempt to install on his computer and the results were the same. Lastly, I tested on ECE grid and it was also successful. Although this was a small scale test of just three devices, there were three different types of operating systems (Mac OS, Windows, and Linux) so I made a safe assumption that the code will run on all types of OS.
Mohammed	I successfully created a test skeleton with code coverage code implemented. At the time, we had not completed any metric calculations so I put in random values into my conditions to test the code coverage and it returned as expected with the percentage of the number of lines covered in all the functions and classes that were called by the test file. It also provided the number of missing lines, so we can Identify the lines that still need coverage and create test cases for it. My task with this file was to create a test skeleton. Once we begin our

	metric implementation, this will be updated with standard hard coded values to test the results of our calculations against expected values of each repository.

3. A comparison of what you accomplished vs. what you planned to accomplish. Are you on track? How were your time estimates?

I finished my goals much faster than expected this week, so I worked on implementing logging functionality to distribute workload better.

Goal	**What was accomplished**	**Owner**	**Expected completion time (hrs)**	**Actual completion time (hrs)**
Create class skeletons	Target was met, all classes were built and work as expected	Owen	6	1
Populate repo class functions	Target was met, a framework was set up in the Repo class to perform all necessary calculations	Owen	1	1
Dummy values in calc_handler	Target was met, CalcHandler functions can be called by Repo object to perform calculations	Owen	1	0.5
Add logging functionality	Target was met, every method now produces logging output and functionality can be extended easily	Owen	3	3
CLI Handler	CLIHandler can now take in correct inputs and outputs correctly based on Repo objects	Ryan	8	8
Look into permissions, getting values from API	Some items we do not have permission for because other peoples repositories may not allow it, we can easily get some values by API however, as was discussed during team meeting 9/18/21	Ryan	2	1.5
CLI Installer	We can successfully install all libraries needed for our program	Mohammed	2	1

Create test skeletons	Test skeleton was successfully created and will be publicated with expected hard coded values of each repository when we implement our metric calculations	Mohammed	3	4-5

4. Any changes in your planned timeline as a result of falling behind your initial plan.

No changes to the timeline. The program framework is set up and we are able to demonstrate the program for test, install and normal execution just as planned. The next two weeks can be fully focused on generating metrics and replacing the dummy values that are currently inserted in the program for testing purposes. We are on track as outlined in our initial project plan.