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: 09/26/2021

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).

Owen	● Ramp-up score is calculated slightly differently. It is now a weighted average of ○ existence of a readme file ○ number of code files ○ number of non-empty lines in the code files This method provides far more useful metrics than the previous method. ● We have adjusted our timing requirement to be less than 3 minutes on average per URL. Unfortunately, our previous metric was too optimistic. API requests (especially cloning the repository) take too long, and we are unable to perform computations in the time limit we originally set.
Ryan	● Refined CLIHandler to better address edge cases and bad URL's ○ Bad URL's will print a warning to the command line interface, but will not stop calculations on the rest of the URL's ○ Bad URL's will be assigned a score of -1 on all metrics ● run.py now properly handles "install" and "test" command line arguments ● License score can now be calculated and returns as expected ○ Acceptable licenses have been modified slightly from Milestone 1. See Appendix 1.0
Mohammed	● Modified the EQN2 calculation for bus factor score ○ Reason: The Pygithub library did not provide us with the number of contributors per release that we needed for our calculations ○ Old equation: **1- 1/# of contributors for most recent release** ○ New equation: **# of authors in last 10% or commits/Total # of authors**

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

Owen	● Completed the ramp-up score metric: ○ Added method to clone a repository locally, parse each code file and return the number of lines that aren't blank ○ Added method to check for the existence of a README file ○ Added method to return the number of code files present in the repository ● Added error handling functionalities: ○ Each API call is surrounded in a try block to handle any exceptions.

	○ Score calculations can properly handle invalid and empty repositories ● Added helper functions to
Ryan	● Updated acceptable licenses ○ If new licenses are supported, they simply need to be added to a list in Repo.py ● Can now give score based on licenses ● Output now outputs in correct format ● Working on PyInstaller to get efficient executable
Mohammed	● Refined run.py to install dependencies directly and return the number of library installations that were successful. ○ Screenshot of this below ● Refined run.py to also directly call a function with our test cases and return the coverage of each python file we are implementing and also the missing lines that still need coverage ● Completed 2/3 equations for calculating our Bus factor score. The 3rd equation was refined as explained above so will be implementing that in week 4. The 2 equations that were implemented have been tested and work as expected. ● Completed 2/3 equations for calculating our Maintenance score. The 3rd equation will be implemented in week 4

   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?

Goal	**What was accomplished**	**Owner**	**Expected completion time (hrs)**	**Actual completion time (hrs)**
Implement ramp-up factor metric	Ramp-up factor can be calculated successfully as a function of the existence of a readme, the number of code files and the number of lines of code total	Owen	5	4
Clone repository system and parse	Repository can be cloned, parsed and deleted once it is no longer needed	Owen	10	3

npmjs link handling	We can now retrieve the link to git repositories through links to npmjs sites	Owen	1	1
Implement bus factor metric	We can now calculate our Equation 1 and Equation 3 for the bus factor. The average of these two scores + our equation 2 score will give us the overall bus factor score	Mohammed	5	5
Implement license score metric	Successfully reads license from Github and compares to compatible licenses	Ryan	6	6
Implement Maintenance score	We can now calculate our Equation 1 and Equation 3 for the Maintenance score. The average of these two scores + our equation 2 score will give us the overall Maintenance score	Mohammed	4	3

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.

Very few changes to the timeline. We are slightly ahead of schedule, having some work done on the correctness features already. There was a change in requirements with error handling, but this requires little to no deviation from our current design

```
(base) PS D:\ECE 46100\Github\project-1-project-1-4> python run.py install

Installing...

12 Dependencies Installed...

(base) PS D:\ECE 46100\Github\project-1-project-1-4>
```

```
(base) PS D:\ECE 46100\Github\project-1-project-1-4> python run.py test
Testing...
Total:  20
Passed:  0
0/20 test cases passed

Coverage Report:

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
CalcHandler.py         18      9    50%   1-18, 25, 29
CalcHandlerGit.py     175     88    50%   2-31, 53, 57-63, 66-70, 78, 91, 107, 112, 117-121, 130-133, 137, 146-147, 152, 156, 167-168, 176-192, 196,
 200-201, 214, 223-266, 269-271
LogWrapper.py          57     30    47%   1-16, 20-25, 49, 52-55, 65-70, 83, 94-102, 110-118
Repo.py               101     59    42%   1-26, 40-76, 85-94, 98-99, 117, 126-130, 141-144, 149, 160, 169
test_coverage.py      101     61    40%   1-19, 33, 56, 63, 84, 88, 91, 94, 97, 100, 103, 106, 109, 112, 115, 118, 121, 124, 127, 130, 133, 136, 139
, 142, 146-181
-------------------------------------------------
TOTAL                 452    247    45%


(base) PS D:\ECE 46100\Github\project-1-project-1-4>
```

```
(base) PS D:\ECE 46100\Github\project-1-project-1-4> python run.py "D:\ECE 46100\Github\a test\test_url_list.txt"
ERROR!
https://www.indeed.com/jobs?q=software%20engineer&l=47906&vjk=80b2e4e2e820a5c5
IS NOT AN ACCEPTABLE INPUT. ONLY GITHUB.COM AND NPMJS.COM URL'S ARE CURRENTLY SUPPORTED. THE SCORE FOR THIS URL WILL NOT BE CALCULATED.

URL NET_SCORE RAMP_UP_SCORE CORRECTNESS_SCORE BUS_FACTOR_SCORE RESPONSIVE_MAINTAINER_SCORE LICENSE_SCORE
https://github.com/pandas-dev/pandas 0.68 0.12 1.00 0.61 0.67 1.00
https://github.com/numpy/numpy 0.68 0.12 1.00 0.60 0.67 1.00
https://github.com/jonschlinkert/even 0.45 0.56 1.00 0.00 0.37 1.00
https://www.npmjs.com/package/even 0.45 0.56 1.00 0.00 0.37 1.00
https://github.com/spdx/license-list-XML 0.00 0.56 1.00 0.56 0.67 0.00
https://github.com/comparison-sorting/odd-even-merge-sort 0.00 0.56 1.00 0.43 0.50 0.00
https://www.indeed.com/jobs?q=software%20engineer&l=47906&vjk=80b2e4e2e820a5c5 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
(base) PS D:\ECE 46100\Github\project-1-project-1-4>
```

Appendix 1.0

Present on Milestone 1 Document but NOT ON LIST AT spdx.org/licenses/

GNU All-Permissive License (#GNUAllPermissive)
Modified BSD license (#ModifiedBSD)
Cryptix General License (#CryptixGeneralLicense)
Expat License (#Expat)
License of Netscape JavaScript (#NetscapeJavaScript)
License of Perl 5 and below (#PerlLicense)
Public Domain (#PublicDomain) -> This one has multiple
License of WebM (#WebM)
WxWidgets Library License (#Wx)

Additions based on piazza post and https://dwheeler.com/essays/floss-license-slide.html

Apache 2.0

MIT