CPSC:480 Software Engineering Project 3

Background

In project 2, you planned and designed a software product. At the end of that process, you identified what would need to be done to make a minimal prototype of the software. In this project, you will perform additional work toward constructing this prototype. It is not expected that you will produce a viable product or even meet all the requirements you identified for a minimal prototype, but your team will need to produce *something* that can be built, run, and tested. You may revise any of your planning and design work, repository issues and tasks, or original requirements and specification as needed during this project.

Part 1: Code

As a team, select a code style guideline to follow for each language that you will use. Industry-standard guidelines can be found on codingart.readthedocs.io or google.github.io/styleguide/. The Keyfactor standards are available for C# projects; contact the instructor if you would like to try to apply them to another language. Alternative coding standards may also be used with instructor approval. All code in a given language must use the same standard.

Using your repository from project 2, each team member should complete one requirement or task toward a minimal prototype. If an item is too large to practically complete in less than 10 hours, break it down or reevaluate what needs to be done. Conversely, if it takes much less than 100 lines of code, complete more of the same requirement; ideally, the most closely related. Develop this code in your own branch (each developer should create a branch named after the team member's UANet ID, that only that team member will write to). While you develop the code, track completed items in the GitHub issue. Also track the amount of time it takes you to complete the task and record it in the GitHub issue as well. If you encounter any bugs or known issues that you will not address at this time, create a GitHub issue with a bug report.

Part 2: Tests

For each function that you write in part 1, write at least one unit test case that covers the function. If a function has several distinct code paths, create separate test cases to provide code coverage. Some portions of a program, like HTML or SQL, cannot be effectively unit tested. If you are not sure whether (or how) to test a piece of code, consult your teammates, and contact the instructor with any unresolved questions. You should be able to run each test you create and have the code pass. Each developer should also create at least one scenario, integration, or end-to-end test applicable to the task being developed. You will be graded on code coverage, suitability of test cases, and ability to run and verify results of the tests. These tests should also go into your branch. Track time spent for testing in the GitHub issue as well.

Part 3: Code Reviews

When your code and test cases are complete, issue a pull request to the repository. For each author's pull request, every other team member should then complete a code review for that pull request. Enter comments for each item that may require revision or further discussion, and when all your comments have been addressed, approve the pull request. Pull requests should not be merged into main until at least three other team members approve it and there are no outstanding comments. After merging the code, you should ensure that it builds and *all* tests pass. It is not necessary for everyone to make a comment on every pull request, but every team member should make at least *one* comment on *one* pull request. You should review the code in the pull request for accuracy, quality, style, and test coverage, and deficient code approved without adequate review will result in a penalty for the whole team.

Part 4: Build and Deployment

Develop and document a process to build and run the software. A user should be able to clone the GitHub repository, follow the directions to build the software, and be able to have it run. Then, document the process of running all of the tests for the software, including any configuration or data input needed to make the tests pass. You do not need to wait for branches to be integrated to make the product build, but you will of course have to wait to run the tests. You may need to write additional tests to achieve code coverage.

Part 5: Report

Write a report on the project, including the following components.

- Features and capabilities as of the due date.
- The tasks completed to reach the current point.
- A link to the coding standards you used.
- The build/run/test documentation from part 4.
- Analysis of how each of the estimates you produced in project 2 compare to the amount of actual time taken to complete code and test cases for each task. For any tasks that took significantly longer than estimated, identify the cause of the underestimation.
- Description of any bugs you found while developing the code.
- Identify three cross-cutting concerns that apply to your product and describe how you address (or would address) these concerns.

Submission

Complete all code reviews, approvals, and merges by **Sunday, Nov 6, 11:59 PM**.

Create a release in your GitHub repository with the source code, pdf version of the report, and the compiled binary, if applicable. Create the release by **Sunday, Nov 13, 11:59 PM**. 1% extra credit if submitted by Saturday, Nov 12, 11:59 PM, and 2% extra credit if submitted by Friday, Nov 11, 11:59 PM. You will again submit a Team Participation Survey, due **Monday, Nov 14, 11:59 PM;** the same terms apply as on project 2 (it will count as a quiz, and may result in bonuses or penalties to some team members so not everyone may receive the same grade).

To ensure meeting all deadlines, it is recommended to complete initial development by Wednesday, Oct 26, complete all test cases and have code reviews out by Sunday, Oct 30, have all code review issues resolved by Friday, Nov 4. This leaves one week to complete any remaining tests, implementation to reach the desired program state, the report, and the release.

Grading

Grades will be 35% individual and 65% shared, as follows:

- 15% Task completion (individual)
- 15% Test cases (individual)
- 05% Work item and time tracking (individual)
- 10% Code review processes and comments (2% each)
- 15% Average of team task completion and test case grades (3% each)
- 20% Calculated overall code coverage * test pass rate
- 10% Ability to build, run, and test according to documentation
- 10% Report and directions