CSE 403, Project Phase 2a: Alpha (50 points)
Initial "Alpha" Implementation Release Code

The "Alpha" deliverable includes a first implementation of your product with a partial incomplete feature set. The customers are looking to see a significant effort to implement a testable, usable (yet significantly incomplete) product that implements an adequate amount of functionality to get a feel for the system and how it will work.

Required Functionality:

The alpha version of your code must implement enough functionality to **execute one of your use cases from your SRS in its entirety**, including all steps required to reach the main success scenario as well as all possible **extensions**. If an extension leads to calling another use case, you do not need to completely implement that other use case, but instead can have a "stub" or incomplete screen for that aspect of the behavior.

In addition, your alpha product must execute a **partial second use case from your SRS**. For this second use case, you are only required to implement the code to get through the **main success scenario**. That is, you do not need to handle any of the extensions for this second use case. (You can if you like, but it is not required and will be ignored in grading.)

When the user executes your app or visits your site, they should be able to navigate themselves into an appropriate state such that they can test these use cases successfully. If you like, you can start your app already in the necessary state to test these use cases directly.

Some use cases cannot be implemented until others are already finished. For example, to implement a hypothetical "User purchases an item from the store" case, you would already need to have user accounts working. You may want to choose use cases to implement that have relatively fewer **preconditions**. Or if necessary, you can implement "stub" versions of any precondition functionality, such as a fake user account that can be used during the "purchase an item" scenario.

Our intention is to have you implement a non-trivial amount of functionality in this phase, but still substantially less than the overall end product. Before doing a substantial amount of work on this phase, please have your PM quickly **verify with your customer TA** which use cases you are going to implement, to make sure that you have chosen a suitable amount of work for the allotted time.

If your group feels that your use cases are not well suited to being implemented during this Alpha phase, please speak with your customer TA. If we agree and feel that some different subset of functionality is appropriate, we may grant you permission to complete a different subset. Please follow the above constraints unless you have already received explicit permission from your customer TA to do otherwise.

Documentation and Usage Instructions:

On your group's version control wiki, create a page for your Alpha containing a brief summary of what functionality exists in the Alpha version, including which use cases are implemented, which extensions to the use cases are supported, and (briefly) how the user can download and/or execute your product to experience and use this functionality.

Bugs and Bug Tracking:

Your Alpha product does not need to be 100% free of **bugs.** Any features outside of the functionality described above can be incomplete and/or buggy. It is expected that your product may have some small issues that come up during testing or usage. We do, however, expect a reasonable effort at producing high quality software that does not show excessive or "show-stopping" bugs that stop the customer from being able to perform normal usage and evaluate your work.

Any known bugs or issues in your product should, however, be documented using your group's **bug tracker**. If we encounter significant bugs during our own testing that are not represented in the bug tracker, you may lose points. We expect that several bugs will be present and filed in this system at the time your Alpha code is due, including some bugs that have already been previously resolved/fixed and others that are still open. A user testing the system should not encounter a non-trivial number of bugs that are unlisted in your bug-tracking system.

Coding Standards and Code Reviews:

In the SDS documentation your team is developing concurrently with this Alpha code, you are to describe your group's coding standards and how they will be enforced. We expect that your Alpha code will follow these coding standards.

To earn full credit for your coding standards, your group must show clear evidence that you have performed substantive **code reviews** over all non-trivial code check-ins for the project. The exact structure and organization of your code reviews is up to you, to some degree; it can be somewhere between an informal "walkthrough" and a more structured, formal, organized "code review." But we expect that your reviews will meet the following guidelines:

- A code review must have at least one team member who did not participate in the writing of the code in question.
- The person(s) performing the review should be reasonably familiar with the languages and tools used in the code, so that they will be able to adequately evaluate the code under review.
- Your code review does not need to be in person, though we recommend this. Code reviews can be done by email or online if so desired, so long as the other constraints outlined here are met.
- You must produce some artifact showing evidence that the code review occurred. One way of documenting your code reviews is by making comments on the checkin in your version control repo using built-in tools such as those found in GitHub. Or you can write comments by hand on a checklist such as the form shown in class. You can come up with your own other method to document your code reviews, so long as it contains a suitable amount of information. Even if the code does not require significant changes, we would like to see clear evidence that it was reviewed and that some comments were made about it by the reviewer.
- Your documentation of each code review should include the names of people involved, the source code version or revision number(s) as appropriate, the date at which the review took place, and a description of the code being reviewed (which files? what has changed in them? how many new lines have been written for check-in? etc.). If the version control system captures this information for you, that is acceptable.
- The documentation you produce should make it clear what aspects of the code are being examined in this code review. Are you looking at source code formatting? Comments? The algorithms used? Efficiency/redundancy? Is it more of a security audit? Etc. It's okay if your review covers several of these areas, but explain what is (and possibly what isn't) being examined in each review in your documentation of that review. If you have a standard set of code review guidelines and issues that you are checking, document this somewhere in your repo or wiki.
- The code reviews should be done in a timely manner. For example, waiting until the end of the Beta phase and then reviewing the entire source code base is not acceptable. The review should come either before the code is checked in, just as it is checked in but before it is merged with the main code base, or promptly thereafter. Some projects do not allow any un-reviewed code to be checked in to the project's main source control branch whatsoever; you do not have to follow such a strict guideline, though you can if you like.
- It should be clear what the "result" of the code review was. Is the code ready to be checked in / merged? If not, what must be fixed before it can be checked in?

In the past, some students conducted code reviews where the reviewer felt that nothing was "wrong" with the code in question. The groups were unsure of how to document this; if nothing is wrong, what's to write? Even if the code being reviewed is already high quality, you can still write a description of the review, what code was seen, which aspects of the code were examined, and so on. You can describe why the code is acceptable, what features were done well, what causes it to be good enough to check in right away, etc. Simply saying "the code is fine" will not receive full credit.

As part of your grade, your customer TA will examine your version control for comments made on commits to confirm that code review did occur. In addition, we will **briefly examine selected files** from your Alpha code to verify that they meet appropriate coding standards. The majority of your grade will be from functionality, and we will not rigorously check every file, but we would like to see evidence that your standards are being enforced and followed.

Regardless of the exact coding standards being followed by your group, we expect that each source code file has a clear **comment header** that names its author(s), or that code authorship is present in some clear way in the version control system. We also expect that all members of the team have made significant and visible contributions to the project. In general this means that all members should be checking in code to the repository, but it is also acceptable for some group members to contribute entirely by working on the various non-source-code aspects of the phase such as design documents. If a group member's contribution is not obvious from the repository (such as if a pair worked together), please make this clear, such as by indicating it in the commit messages when checking in that code to the repo.

Submission and Grading:

Submit your Alpha code online by checking it into your version control system on **GitHub** by the due date. Your code review documentation might be on paper, in which case it can be given to your customer or instructor in person. Because you only see the customer/instructor in person at lectures and sections, you may need to slide any paper documents (such as code review reports) under the instructor's office door. This should be done by the due date stated in this document.

Remember that part of your grade comes from having a meaningful in-person **interaction with your customer TA** before the phase is due to show your progress, ask questions, get feedback, and generally make sure you are on the right track.

During grading we will access your work through your version control repository, so this should remain up and reachable by your customers at any time without notice. We expect you to be using your version control system properly, that is, making coherent check-ins of reasonable size, properly labeled with commit comments, to indicate what has changed. If your version control system appears to be an afterthought in your project (for example, if you just do a few very large check-ins at the very end that contain most all of your source code, indicating that you didn't really use the system along the way; or if you perform many trivial checkins that do not represent actual coherent work tasks), you may lose points.