CSE 403, Winter 2012
PHASE 5 (60 points): Initial Code Implementation (Beta)
due Sat Feb 25, 11:30pm

For the "beta" product release, you must produce a working initial version of your project, reflecting several (but not all) of the features listed in your SRS and SDS. Submit or provide the following three (3) items:

1. Binary Distribution

The binary distribution contains the resources necessary to run and use your system. If your system is web-based, its binary distribution consists of the site being up and running by the due date. If it is a desktop or mobile application, your web site as created in the ZFR phase should exist and should contain links to download and run the product.

This item will be graded on whether it reflects substantial work and effort on the part of your team, has a solid and polished user experience, and successfully implements the usage cases you have described in previous phases.

You should have implemented a reasonable number of features in your product, though you certainly do not need to have a complete product ready nor finish every feature you listed in your SRS documents previously. The customers are looking to see a significant effort to implement a testable, usable product that implements an adequate amount of functionality to get a feel for the system and how it will work. The features you implement in this phase should target core functionality such as that described in your use cases and sequence diagrams from previous phases.

Your beta product does not need to be 100% free of bugs; it is expected that it may have some small issues that would 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 evaluation of your work. The customer must be able to successfully test and use your system to perform non-trivial tasks.

Any known bugs in your system should be documented in your bug-tracking system. We expect that several bugs will be present and filed in this system at the time you submit your beta, including some bugs that have already been 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. Your system should be robust so that errors occur as gracefully as possible.

Part of your grade will come from the usability of your UI, based on feedback and changes since your UI prototype.

Since your work will change between the time the beta is due/submitted and the time it is graded (especially if it is a web application), place your beta resources at a "beta URL" location that will remain at the "beta" version indefinitely. When the primary customer TA goes to this beta location, the binary found there should be in beta form and not in the form of the latest version of the code that you have completed since the beta. This assists us in grading your work.

2. Source Distribution

The source distribution contains all source code and other resources that were created by the development team. These resources should be bundled into one or more compressed archives. Assume that this item is being prepared for one or more developers who would pick up development where you left off.

We will also access your work through your version control system (SVN repository), so this should be set up and reachable by your customers at any time without notice. We expect you to be using your version control system properly, making small and numerous check-ins properly labeled to indicate what has changed. If your SVN 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), you may not get full credit.

Your source code will **not** be examined in detail in this phase. We will look at it at a high level to satisfy ourselves that a sufficient number of lines of code have been written. We will also verify the following:

- that it reflects non-trivial work and effort
- that each file has a clear comment that names its author(s) (or that code authorship is present in some way)
- that several members of the team have made significant and visible contributions to the source code
- that your project is using a suitable set of object-oriented languages, frameworks, and tools
- that your code and design properly follow the Model-View-Controller (MVC) pattern as appropriate

Otherwise your code will **not** yet be graded on style and design, commenting, elegance, redundancy, and so on. It should bear some general resemblance to the design you produced for your SDS, but designs evolve during the development process, so it does not have to match your SDS exactly. In later phases we will evaluate your code and design more strictly. For example, we will check that you follow style conventions and use design patterns as appropriate. We encourage you to use good design and style even during the Beta.

Along with your binary please supply a short document **`readme.txt`** that explains at a high level what features are / are not implemented in your beta, as well as any brief necessary instructions about how to run/use the beta binary.

3. Code Reviews

To earn full credit on your Beta turnin, your group must show clear evidence that you have performed at least **six (6)** substantive code reviews over 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 highly 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. We strongly recommend using the "checklist" form shown in section (or creating a similar form/process of your own). You can come up with your own other method to document your code reviews, so long as it contains a suitable amount of information.
- 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.).
- 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.
- The code review should cover a significant, non-trivial change to the project's code. A code review that covers too few lines, or one that tries to cover more than 3-400 lines, is less effective and may not receive full credit.
- 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, or promptly thereafter. Some projects do not allow any un-reviewed code to be checked in to the project's main source control branch; you do not have to follow such a strict guideline, though you can if you like.
- Each code review should include a list of things that were discussed/examined, decisions made, bugs or problems to be corrected, etc. It should be clear what the "result" of the code review was. Is the code ready to be checked in? 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 "code was fine" will not receive full credit.

Submission and Grading:

Submit your code online through the course web site. 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 standard due date stated in this document.

Part of your grade comes from having a meaningful in-person **interaction with your customer** before the phase is due to show progress, ask questions, and get feedback.