CSE 403, Winter 2012
PHASE 3 (20 points): Zero Feature Release (ZFR)
due Mon Feb 6, 11:30pm

The zero-feature release milestone consists of a skeletal implementation of your product and a document with instructions for accessing some of your development tools. The purpose of this milestone is to work out basic system organization and access bugs, as well as making sure the major tools and subsystems of your project connect properly, such as your version control, bug tracking, deployment process, etc. Unless otherwise specified, no functionality needs to be working other than a "front page" (for web projects) or "initial welcome screen GUI" (for desktop app projects) as described below.

1. Featureless live product web site

Provide us with a URL to reach a front page for your product. If your project is a web app, this URL would be the front page of your actual web app (though it does not need to be functional). If your product has some functionality, such as if you are getting started early on coding, that is fine; but such functionality will be largely ignored for this phase.

If your project is a client-side app or mobile app, this URL would be the address of a web page where we can download the app to install or run it. (The page does not need to be at all fancy, just a link to the file(s) to download is fine.) If you're doing a web app, at least one page you post should have some kind of active content on it, such as Ruby on Rails code that runs on the server and is then displayed in the user's browser. What the code does can be trivial, e.g. add 1+1.

Part of completing your ZFR is creating your database or other data source. Therefore, your ZFR product must in some way connect to this source of data, fetch at least one piece of data from it, and display the data in some way on the screen or web page. If your data source is plain files, your product should open and display some content from at least one file. If your project involves several sources of data, several databases, etc., your ZFR should connect to each of them, though what it does with this connection may be trivial, such as fetching and displaying one row or record of data from each.

2. ZFR instructions document

Your group should also submit a ZFR document describing the items below. Some documents describe processes the user or developer must perform. Part of your grade will be based on the simplicity of these processes, and how accurately your directions match what the user must actually do.

You should also put the resources in place so that the graders can examine and test the processes described in the ZFR document promptly after your submission. You may want to arrange a "dry run" with the customer beforehand. Your ZFR document should address the following items:

a. Source control and build process instructions

Your project's build process is the set of tools and commands to compile and "build" your system. Turn in a set of directions to find your build system, check out its files, and build them. In grading, we will follow these directions.

For full credit, **your team should have a reasonable resolution to the issue of version control**. For example, you could handle this issue by using a CVS or SVN repository in a reachable location, or by hosting the code on a public system such as SourceForge. Your instructions should explain how to access any such system or repository.

The directions should be written in sufficient detail that an intelligent developer can follow them. If the system(s) require login information to access them, you should provide this information.

Part of your grade for this item relates to the number and complexity of commands the developer must use. Ideally your system will have a single command that does a "one-step build," that checks out all source code from your repository, builds all necessary binaries, packages them, and places them in a known location.

Beginning with the night of your ZFR turnin, **the grader will log in to your version control system once a week** to count the number of files and lines present in the system, as well as the number of lines modified, and log your group's development progress. Lack of significant weekly progress may impact your team's grades on later phases.

b. Bug-tracking system instructions

Describe the set of tools used to document existing bugs and missing features in your system's code. Turn in a set of directions to your customers, describing briefly how they find your bug-tracking system, examine the list of current bugs, and file a bug. In grading, we will follow these directions and examine whether the bug tracking system exists and is usable. The directions should be written so that an intelligent developer can follow them.

Your bug tracking system does not need to contain a comprehensive list of bugs or missing features. But for full credit, it should have at least one bug filed for each active developer. These bugs may be requests for features, such as "TODO: implement login behavior." If possible, the bugs should list priority and a timeline to fix them.

Beginning with the night of your ZFR turnin, **the grader will examine your bug-tracking system once a week**. Part of your grade in later phases will reflect whether a significant number of accurate bugs are present that contain proper information such as severity and assignment to specific developers to fix them.

c. Data access instructions

Since your product must have a server-side data component, your ZFR should contain a set of instructions about where this data is stored and how to access it. Turn in a set of directions that tells the grader how to find your data, and how to briefly perform a trivial access of this data. For example, if your data is in a database, inform the grader how to connect to this data and perform one very simple query against it. In grading, we will follow these directions.

Submission and Grading:

Part of your grade comes from your web URL actually working, from the grader being able to connect to it on the first attempt. The URL should remain "live" while we are checking your work.

Your data access instructions should be comprehensive and should list all steps necessary for us to find and access your data. For full credit, these directions must match the actual steps we need to perform to find and retrieve the resources.

Your build process instructions should be complete and correct. We should be able to successfully build and run your app on the first attempt, barring grader error. The build process should not be overly complicated; streamline the build process so that a reasonably intelligent developer can follow it.

Your bug-tracking system should actually exist when we go to examine it, and we should be able to log in and access it successfully on the first attempt. One bug must be listed per developer with suitable details such as to whom the bug is assigned and its priority.

A small part of your grade comes from the looks or aesthetics of your documents. They do not need to be beautiful or excessively formatted, but your customers need to be able to read them and extract information from them. This means they should be clearly written, with proper spelling and grammar, clear wording, and formatted with a enough organization to present your ideas clearly to the reader.

Your ZFR does not need to reflect "customer" interaction, but you can ask your customer to try testing out various aspects of your site before you officially submit your ZFR ("Are you able to connect to the following URL?" Etc.). If you make your request a reasonable amount of time before the due date, the customer will do his/her best to try to accommodate such requests in a timely manner and give you feedback about the results.