Assignments

Viewing assignment...

Settings for "Implementation (2014 Team Assignment 5)"

Created by	Gail E. Kaiser
Date created	Oct 19, 2014 9:02 pm
Open	Oct 21, 2014 12:00 am
Due	Nov 10, 2014 5:00 pm
Accept Until	Nov 10, 2014 5:00 pm
Modified by instructor	Oct 24, 2014 9:34 pm
Student Submissions	Single Uploaded File only
Number of resubmissions allowed	Unlimited
Accept Resubmission Until	Nov 10, 2014 5:00 pm
Grade	Points (max 20.0)
Alert:	Yes
Honor pledge:	Yes

Assignment Instructions

Your team implementation document should consist of a single file uploaded to courseworks plus the specified material entered into JIRA at http://ase.cs.columbia.edu/jira. The actual implementation code (including concrete test cases, scripts, sample data, etc.) should be maintained using git in STASH at http://ase.cs.columbia.edu/stash. Only one member of your team should submit the courseworks file (but all team members can edit jira and stash). You may submit the courseworks part as often as you want up to the deadline (and continuing changing in jira and stash thereafter).

Every team should schedule a meeting with their TA to discuss their design as soon as possible after the design assignment has been graded. Note this is a separate meeting than the preliminary demo below, and should occur long before that point.

Schedule a preliminary demo with your TA any time between Wednesday November 5 and Monday November 10 (inclusive), but before this assignment is due. Note this assignment is due on Monday, not Tuesday. Also schedule an in-class demo on one of the demo dates, November 11, 13, 18 and 20. You will be sent a scheduling form to do the signup, do not send email asking for a date.

The first page of your document should indicate your team name and list the full names and uni's of every team member.

The second page should give a short synopsis (overview) of your project. This can be copied verbatim from a previous document if nothing has changed.

The third and remaining pages should present the requirements and design you really implemented, in light of how you handled the risks assessed in the previous assignment, followed by a brief discussion of how you applied (or didn't apply, as the case may be) TDD.

Starting on the third page labeled "Requirements and Acceptance Testing", and continuing for one or more pages, present the use cases you really implemented along with corresponding acceptance test cases (pseudo-code or readable/commented real code). Anything that has not changed can be copied verbatim from your previous Requirements document. Your use cases and corresponding acceptance test "issues" should be updated in JIRA to match. The code, scripts, sample data, etc. for the actual acceptance testing should be uploaded to STASH.

Starting on a new page labeled "Design and Unit Testing", and continuing for one or more pages, present the design you really implemented and tested. Describe the major classes (and/or interfaces), including any relationships between them, and also the main interactions among classes that collectively implement all your

use cases. Sketch class diagrams and sequence diagrams. For each class, present unit tests (pseudo-code or readable/commented real code) that exercise all the public methods of that class and collectively cover all the sequence diagrams. Discuss how the test cases consider the equivalence partitions for each method parameter, making sure the test cases cover the "middle" of the partition as well as "at", "just above" and "just below" each equivalence partition boundary, for both valid and invalid inputs. Discuss any design patterns adopted or adapted and how they affect your classes and their interactions. Anything that has not changed can be copied verbatim from your previous Design document. Update the corresponding class diagram, sequence diagram and unit test issues in JIRA. The code, scripts, sample data, etc. for the actual unit testing should be uploaded to STASH, along with the entirety of the actual code implementing your system.

Starting on a new page labeled "Testing Process", and continuing for one or more pages, discuss the process you followed in writing the actual test cases (in code) *before* writing the corresponding code, and then actually writing the corresponding code, at both the acceptance and unit testing levels. This is intended to be open-ended. Tell us what you really did, and what you found worked or didn't work.

Optionally, for extra credit, start again on a new page labeled "Extra Credit" and develop one or more alternative designs for your system based on design patterns and combinations of patterns (such as MVC). Include corresponding class diagrams. Compare and contrast these to your actual design; they might be better or worse than the design you implemented. Its ok to include "bad ideas" here, as long as they are plausible designs, but the main intent is to consider what you would do differently starting over if you knew then what you know now. This might include changing your requirements (use cases) in some way; if so make sure the new vs. old requirements are clearly identified here.

Additional resources for assignment

Assignment5.pdf (294 KB; Oct 20, 2014 12:41 am)

Assignment5-InitialImpAndRevPlan.pdf (549 KB; Oct 20, 2014 12:41 am)