CS473/673 Software Engineering Project – Summer 2016

Project Description

In this course, students will work collaboratively on a big real-world semester-long project: a project management tool, preferably as a web-based application or mobile application. The project management tool should support agile-like iterative process. The tool is intended to be used in the academic setting for group-based class projects or research projects. The tool shall have the following three relatively independent components.

- Project requirement management tool to manage software requirements (such as user stories) and related tasks. (similar software: pivotaltracker)
- Project communication tool to facilitate the communication between members such as a virtual chatting room. (similar software: slack, discussion board)
- Issue management tool to manage the issues or bugs in the project. (similar software: bugzilla, issue tracking in github)

Since currently we have 21 students in the class, we will divide into three groups, working in parallel. To coordinate the group work, each group will have a group leader and a backup leader. Every member of the team is expected to contribute a roughly equal share to the project throughout the whole process. The single biggest determinant of success or failure of a project are the people on your team. Respect your fellow team members and commit to the team work is the key.

The instructor will also act as the customer of the project. Initially the customer will only have a vague project idea. The teams will interact with the customer in order to figure out what the customer want and deliver what the customer wants. The payment will not be dollars, but your grades ☺.

Project Process Description

While several different software development models will be introduced in the lectures, this project will mainly adopt an iterative and incremental development process. We will focus on some agile software development methodologies, but also introduce non-agile approaches and incorporate some into our project as needed. We will have an initial planning phase, and then three iterations. Each iteration will produce a working and quality software, and each iteration is a mini-project, involving planning, requirement analysis, design, coding and testing. A short presentation with demo will be required at the end of each iteration in order to get the feedback from the customer.

We will leave about 40 minutes in each class (8:50-9:30pm) to discuss any issues in project. Besides, each team may also need one or two additional hour(s) for weekly meeting. Meeting minutes are required for each meeting. Students should also communicate with each other through email, phone calls, informal meeting for any questions and issues. Each student should also submit a weekly status report that records the progress in each week, plan for the next week and any issues encounter. Meeting minutes and weekly report are shared through Google drive.

Initial Planning Phase

1. Form teams, designate team leaders, other leader roles and responsibilities for each role. Designate a backup for each lead role if possible. Set up communication plan.
2. Brainstorm the requirements of the project with the customer. Define the project title and a project vision, including purpose of the application, major functionalities and requirements etc. Research related work, describe the challenges, explore feasibility and learn tools. We will use pivotaltracker (http://www.pivotaltracker.com) or Trello (http://trello.com) as the requirement analysis tool.
3. Set up project environment which include management tool, development environment. . We will use GIT as version control tool and github (https://github.com) to host your project. All documents and code should be configuration items and stored in your version control system.
4. Identify risks and risk retirement plan. Estimate the time, make initial coarse granularity schedule of iterations. Define quality metrics and the techniques to assure quality.
5. References: please read book Section 3.3 (P55-59) for team guidance on initial team meeting, communication plan, and test communication plan. Also read hints on team exercises on P61. Please read book Section 7.7 (P162-165) for team guidance on team organization, meeting. Please read book Section 5.9 (P103-117) for the case study of Encounter SQAP. Please read book Section 8.7 (P208-210) for the team guidance on steps to a project management plan, and Section 8.4 (P187-196) for the case study of Encounter SPMP.
6. Submission: presentation, meeting minutes, weekly report. All should be archived in the github repository.

In Each Iteration

1. Planning and Requirement analysis: work with the customer to prioritize the requirements from the backlog and choose the most important ones to implement in the current iteration. It is possible to get new requirements from the customer too. Analyze each requirement; write user stories, the acceptance test for each requirement. Break down into tasks, and estimate the time for each task and/or requirement based experience or the previous

iteration. Do the detailed schedule for the current iteration and coarse granularity schedule for later iterations.

2. Design, implementation and testing: identify design goals, choose the proper architecture. Identify objects/classes and their attributes and operations. Apply design patterns to enhance re-usability if applicable. Refactor code on a regular basis. Apply coding standards across the team, and document source code. You can use tool such as Javadoc to generate API documentation from the documented source code files. Test should be in parallel with implementation. This involves unit testing, integration testing, system testing and acceptance testing. Each iteration should result in working software that passed all tests. Draft SDD (Software Design Document) to include software architecture, subsystem decomposition, design patterns etc.
3. All code and documentation should be peer reviewed at the end of each iteration.
4. Reference: please read book section 12.13 (P309-315) for team guidance on requirement analysis. Also read section 11.10 (P264-268) and section 12.14 (P315-328) for the case study of Encounter SRS. Read book (P329-330) for team exercise to evaluate this process. Please read book section 18.7 (P457-460) for team guidance on software design. Also read section 18.8-9 (P460-466) and section 19.12 (P494-503) for the case study of Encounter SDD. Please read book (P475, 507) for team exercise to evaluate this process. please read book section 22.13 (P565) for team guidance on implementation. Please read section 22.10 (P556-558) for the case study of Encounter implementation. Please also read section 26.5(P652-662), 27.7-27.8 (P683-692) and 28.5 (P714-723) for the case study of Encounter testing. Please read book P665, 693 for team exercise to evaluate this process.
5. Submission: presentation (with demo), final deployment of your software, requirement analysis, updated documents (weekly report, meeting minutes), as well as source code (production and test code) on github. Post iteration self and peer reviews.