15-313: Foundations of Software Engineering

Homework 1: Time Estimation, Scheduling, and Teamwork

In this homework, you will work in teams to implement a group exercise class management system. This project has several challenges, including underspecified requirements, time estimations, and new technology. The homework is designed to focus primarily on gaining practical experience in important non-technical aspects of working on a software development project, such as project time estimation, scheduling, and coordinating within a team. Along with those, you will also get a chance to hone your software development skills.

The key learning goals of this homework are that you are able to:

- Carefully make time estimations, being aware of problems and using mitigation strategies
- Plan and schedule projects in terms of tasks and milestones and re-plan as required
- Perform initial risk assessments and plan mitigation strategies
- Make initial decisions on a process and reflect on experience with the process
- Effectively coordinate among team members and conduct effective team meetings
- Reflect on experience of working in teams
- Collaborate in development projects using Git
- Engage in good software writing practices

Project description

CMU Athletics offers group exercise classes to members of the CMU community (http://athletics.cmu.edu/fitness/groupx). Though students have access to these classes for free, others (such as faculty and staff) have to pay. When people attend a class, they tap their ID on a tablet computer that the instructor has, which runs software that records their attendance and handles the billing. No one has to register to attend a class; people can just drop in on sessions when they want.

Your team's task is to build a new group exercise class tracking system. In particular, it should solve two problems of the current system:

- The management has complained that they have no convenient way of finding out how many students attend each class so they know whether the class is breaking even financially. Each instructor gets paid at their own rate, so each class has a cost that depends on who is teaching it. Attendees either pay nothing or a flat rate per class, depending on whether the attendee is a student.
- Instructors have complained that they have no way of keeping track of who has attended their classes. It would be best if your solution helped instructors keep in touch with their students and encourage students to come back to more sessions.

Note: Assume that mechanisms for tapping/swiping cards and for looking up student data will be provided later. Abstract this behind an interface and provide a sample/dummy implementation for testing and demonstration purposes (e.g., just ask for an ID through a dialog and provide a lookup for some test students).

Tasks

Your team will implement a simple group exercise class tracking system. Do your best effort for the product, considering that this is a three-week homework for a medium-sized team. You will likely not have time to produce a perfect product, but you should be able to provide a reasonable and usable prototype. Plan at least 50% of your time to be spent on non-coding activities. Follow the team policy (especially meeting protocols, see below).

In this homework, you have considerable freedom in your choice of methods and tools. The only explicit requirements are:

1. All development happens in a GitHub repository that we provide. You must use good development practices according to what you learn in Recitation 1.

2. You must use a build automation tool to manage dependencies and build your system

3. If you do not provide an implementation that your TA can *conveniently* run, you must schedule a time to demo your system. Your TA's machine runs Mac OS X 10.11.6. For example, you can provide a virtual machine image or a Docker image, or a Mac binary.

Your software product can be structured as a client-server system in which some client software is installed on a tablet and on the administrator's computer; as a web-based system using a browser as a client; or in any other form that works in this scenario. In line with the course's prerequisites, we assume familiarity with Java (including distributed computation), but your team can decide which programming language and development approach fits your needs bestcs. Additionally, your project must make use of a technology that no one in your team is familiar with. The new technology should be some form of technical artifact used within the final product or used for quality assurance of the developed artifact. This could be a database system (e.g., relational or NoSQL) to persistently store events, a new programming language, a new GUI framework, a GUI testing framework, or some other technology. Do not select a technology that is very similar to one you already know; if in doubt, send an email to your TA.

Process requirements

We have only a few requirements for the development process, and give you freedom otherwise:

- Before you start coding, you should:
    - Plan all project tasks. You should estimate the time needed for each task and document task dependencies and responsibilities.

    - Consider the development process you plan to follow and which tools you will use. For example, GitHub provides a bug tracker; you may consider using FindBugs as part of your quality assurance process; consider using Travis-CI; or you may use some other tool.
    - Think about risks involved in this project and possible documentation strategies.
- Throughout the project, track your time investment, synchronize tasks with your team, and regularly update the plan. Make use of meeting protocols to track how you divide the work.
- The project must include some form of quality assurance (e.g., FindBugs executed as part of the build or unit tests of core functionality).
- All development must happen in the provided GitHub repository. Commits should be reasonable in size, coherent, and include reasonable commit messages. It is not acceptable for one team member to commit all work after synchronizing through other means.
- You should produce documentation within one page (soft limit) on how to set up your system, how to deploy it, and how to use it (for end users). The system should be compiled with a build script, such as make, Apache Ant, Apache Maven, Gradle, or similar. Remember to include dependencies in your committed project if necessary.

Deadlines and deliverables

This homework has three (3) deadlines and four (4) deliverables. The first deadline (Thursday, September 8, 23:59) is for a **planning document**. The second deadline (Tuesday, September 20, 23:59) is for the **technical artifacts** of the system developed. The third deadline (Thursday, September 22, 23:59) is for **two (2) reflection documents**. We intentionally separated these deadlines to ensure upfront documentation of the planning process and to give you time to reflect on the experience without the pressure of perfecting the code for submission at the same time. Of course, you are encouraged to start developing before the second deadline and to start reflecting before the third.

(0) Setup

You will need to create a GitHub repository for the team project. First, you'll need to agree on a **team name.** Your team name should be unique, pronounceable, short (e.g. one word), and something you would be proud to shout in your team cheer on the streets of Pittsburgh with children present. Then, you should each go here to set up your team: **https://classroom.github.com/group-assignment-invitations/620c7f589847cf90b13372a7c06db341**

If it makes you feel any better, we are not actually planning to make you write a team cheer.

(1) Planning documents (due Thursday, September 8, 23:59) – 50 points (25%)

- ***An initial time plan****.* You may chose any format as long as it is clear (e.g., network diagram, Gantt diagram, plain text, export from a project management tool). The time plan should include at least:
    - individual tasks and milestones,
    - deliverables for all milestones,
    - estimated effort for each task,
    - dependencies among tasks, and
    - tasks assignments for team members (as far as known).

  You are encouraged to include supporting evidence for your time estimations (i.e., an explanation for how you arrived at that value). Tasks must be presented at a reasonable granularity. One guideline is that each task should take no more than 1-2 days; if a task takes longer, it should be broken into subtasks.

- ***A simple risk assessment****.* Within at most one page (soft limit), identify and briefly describe the key risks in this project (at least two) and discuss mitigation strategies you are planning to use.

- ***An initial process plan****.* Within at most one page (soft limit), briefly describe the process you are planning to follow. This should include at least what quality assurance process you are planning to use and how you are planning to divide and integrate work. Example questions you should consider answering: Who is doing what? What policies will you have for commits in your repository? Will you do code reviews? Do you have a convenient way of running unit tests? How often will you meet as a group? Will you coordinate through other mechanisms, e.g. Slack?

We do not expect that your first plan will work out. In fact, we would be surprised if it does. It is important to learn from failures. We will not grade accuracy of your prediction or how well you stuck with your initial plan, instead we will focus on how well you analyzed and reflected on your experience (see below).

The result should be submitted as a single PDF file by committing to your group repository on GitHub. This document should contain explicit subsections for the time plan, the risk assessment and the process plan. Your plan should go in planning/ in your team repository.

2) Code artifacts (due Tuesday, September 20, 23:59) – 30 points (15 %)

We will take a snapshot of the implementation from your team's GitHub repository at the deadline. Your repository should include the implementation of the system and a short documentation of how to build/start it (for system administrators) and how to use it (for end users) in (or linked from) the README.md file. Indicate which new technology you used in the README.md.

Adhere to good coding practices: your code should have a clear structure, be reasonably modularized, use appropriate variable names, and be documented where appropriate. The repository must demonstrate some form of quality assurance (e.g., FindBugs integration in build process or test cases). Use good practices for cohesive commits with meaningful commit messages.

In addition to the source code, you should provide executable images. Docker is likely to be a good solution for the server-side components, if you have any. Client-side components may be platform-specific; you can provide a VM image[1], or a Docker image, or a binary. Again, if your TA cannot run your binary, you will need to schedule a meeting to demo your system. The link to the image or other binary and the login and password used in the virtual machine (if applicable) should also be included in the README.md file.

Your code artifacts must go in src/ in your team repository.

(3) Reflection documents – Team (due Thursday, September 22, 23:59) – 60 points (30%)

After coding is complete, reflect on your experience. Do this together as a team. Again, we look for honest reflection, which will likely include reflection on failures. We will not grade whether you predicted the effort correctly, but rather what you learned from the process. The reflection document will have 5 parts:

- **Actual Schedule:** Document the *actual* schedule, including the tasks you *actually* performed and the *actual* amount of time each took. Ideally, you will have kept your schedule up-to-date throughout the project, simplifying this task.
- **Schedule Deviations:** Reflect on the differences between the initial and the final schedule. Which milestones were predicted correctly? What was re-planned? Were there any activities you didn't plan for initially or that you had to drop in the end? What were the reasons for changes? Could they have been foreseen with better planning?
- **Process:** Reflect on the process you initially planned to follow, and the process you actually followed. Was the process adequate? Did you skip steps or adopt additional techniques during the project? What challenges did you face? How could the process be improved if you had to do another, similar project? How might you have to change the process to adapt to a different type of project?
- **Team Experience:** Reflect on your experience working as a team. What worked well? Were team meetings efficient? How well, and through which processes, did you communicate? What needs improvement? Were there any teamwork challenges you resolved as the project progressed?
- **Meeting Protocols:** Attach all meeting protocols kept throughout the project, which should include information about topics discussed, decisions made, and explicit work assignments. Recording results of meetings can be onerous; you might want to rotate this responsibility among the members of your team.

[1] Preferably, use CMU Box Storage service to store and share the image link with the instructors.

Summarize your results and submit them as a single PDF file with explicit subsections. Put your PDF file in your team repository in the reflection/ directory. The three reflection steps Schedule Deviations, Process, and Team Experience should not exceed a page each (soft limit); there are no format restrictions on the schedule or meeting protocols.

One of the main purposes of this homework is to encourage an *in-depth analysis* of the reasons for good or bad time estimation, scheduling, and teamwork coordination. Doing poorly in these is not unusual (as numerous reports from real-life projects show). Therefore, we will not evaluate how well (or badly) the project went, but instead *how well you understood the reasons why* the project went as it did, and what lessons you drew from your experience to inform your future work. A good reflection document will include concrete statements about lessons learned, with clear supporting evidence, such as examples, to support the claims. It is a good idea to *reference your meeting protocols* to support your claims and provide examples. For example, "We could have communicated better." is weakly supported. One could strengthen it with examples from the development experience as follows: "One source of [defects/development slowdown/quality problems] was the integration of components A and B, because the API for A was not well-understood by the developer of B… In the future, we might try to use [such-and-such a process] for clearly documenting and communicating such design decisions, rather than [the process we did follow/failed to follow]."

Being able to communicate effectively is an important software engineering skill. As such, your reflection documents should be well-written and easy to read. Be sure to leave time after writing for revision and proofreading.

(4) Reflection documents – Individual (due Thursday, Sept. 22, 23:59) – 60 points (30%)

(Note that this is due at the same time as the team reflection document.)

Separate from the reflection done as a group, each team member should individually reflect on (1) the process, (2) the scheduling, and (3) the teamwork, and *connect this project experience with their previous experience* (preferably experience from a non-academic setting, but experience from class projects is also fine). This reflection piece should examine *all three aspects* of the project mentioned above, and touch upon such questions as: How does this project experience align with your previous experiences? What was similar? What was different? What did you personally learn from this project's development process? Is there something you are planning to do differently in your future projects? Similar to the team reflection task, we will grade the *quality* and *depth* of your reflection. See the notes for the team homework for additional guidelines.

Submit the document as a single PDF file with explicit subsections on Canvas. The entire document should not exceed two pages (soft limit).

Notes

This homework (except for the individual reflection) is to be done in your assigned teams. You are highly encouraged to openly discuss all team issues that may arise in the process of working within the teams. After this homework, we will perform a peer-feedback survey to identify any common issues that we will then address in class. If severe issues occur reach out earlier to the course staff.

Soft limits can be broken if there is a reason to do so.

Grading

You will be graded as a team, with an individual component. This homework is worth 200 points. We will grade you based on the learning goals listed above. The initial planning document constitutes 50 points (25%), the technical artifacts and adherence to process constitutes 30 points (15%), the team reflection document constitutes 60 points (30%), and individual reflection document constitutes 60 points (30%).

To receive full credit for the planning document, we expect:

- A schedule that includes tasks, milestones, dependencies, and so forth, as described above.
- A time estimate for every task
- A list of at least two risks and corresponding mitigation strategies
- An outline of the process steps to be adopted in this project.

To receive full credit for the technical artifacts, we expect:

- A compiling and running prototype implementation
- A pre-configured OS/environment to run your prototype or a meeting scheduled for a demo
- Reasonable documentation for system administrators and end users
- Reasonable code structure and style, including documentation where appropriate
- Use of one new technology (mentioned in the README file)
- Coherent commits of reasonable size with meaningful commit messages by all team members
- Some form of basic quality assurance

To receive full credit on reflection documents, we expect:

- A detailed, well written, and well structured reflection on the issues listed above
- A comparison between the planned and the actual schedule
- An analysis beyond mere descriptions and superficial statements, including supporting evidence for claims, that reflects on the causes of deviations, conflicts, and so forth or on your own experience.
- An attachment of the meeting protocol(s)