17-313: Foundations of Software Engineering

Homework 2: Teamwork

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

- Plan and schedule projects in terms of tasks, milestones, and time estimations, and re-plan as required
- Make initial decisions on a process, and reflect on experience with the process
- Effectively coordinate among team members and conduct effective team meetings
- Meaningfully reflect on the experience of working in teams
- Collaborate in development projects using Git and engage in good software writing practices
- Practice getting to know a pre-existing code base and developing new features for it using previously unfamiliar technology

Project context

CMU has over 7,000 graduate students. The university uses a software system to manage the graduate student admissions process, including collecting relevant information from applicants, accepting recommendation letters from third-parties, showing this information to the admissions committee and accepting ratings and commentary from the admissions committee to support decisions, and notifying applicants of the decision. This system is old, clunky, and universally disliked.

You and your team are developing a new system. Requirements for this new system remain a bit vague, and in fact you hope to be able to use an initial prototype to help clarify those requirements with stakeholders (in a future assignment!). In the meantime, your point of contact insists primarily that the new system needs to be "less garbage" than the current system. When pressed for details, she elaborates that faculty complain that the current system makes it too difficult to evaluate the applicants; students (applicants) and their letter writers complain that the system is hard to use. Everyone finds configuration and management difficult.

You can't (re-)implement the entire system at once! You instead have to familiarize yourself with the existing software; set up your development environment, workflow, and collaboration practices; and plan carefully to decide which features to develop, in what order, and then start writing code effectively as a team. These are the tasks you will engage in in this assignment.

Tasks

You and your team are beginning to sketch out and implement a replacement admissions system. You start with a simplified workflow that you hope to build from later: assume a single academic program, and that applicants upload single resumes to apply. Resumes are assessed by 3--4 reviewers along several dimensions (like "skills", "experience", "GPA", etc), along with additional notes/commentary. This information is used to inform admissions committee discussion and decisions.

You have already decided to use the Mayan EDMS open source software as a launching point: beyond being a pretty good document management platform, it provides complex features that you expect will be useful, like tagging, workflows, and fine-grained permissions and ACLs.

But, it's missing a lot, even for the simplified workflow you're starting with. In your brainstorm session, you come up with a few features you could start with:

1. A dashboard or pane for reviewer assignment and management.
2. A form for reviewers to enter and score candidates along various (custom/admissions-specific) directions, to be saved and possibly aggregated across multiple reviewers.
3. A dashboard or pane for aggregating and displaying statistics, like average review score per candidate; average review score per reviewer; or other aggregate statistics for the applicant pool.

Your first goal is to decide which feature of these three you should/will implement. Then, you will do so, over the course of a 1-1.5 week sprint, while establishing and following good software engineering process, including appropriate git flow and testing of the eventual feature. Specifically, this entails:

1. Setup your initial development repository, environment, and process:
    a. Create a new team repository (instructions below) that forks the Mayan EDMS.
    b. Merge your individual changes from Homework 1 into your clean repository using proper git flow.
    c. Set up CI testing and metrics collection.
    d. Make other development process and toolset decisions, like communication strategies, bug and issue tracking, etc.
2. Do an initial sprint planning meeting. Assess your three feature options (listed above) and choose which one you will implement first, for this assignment. Construct your *backlog* (decompose the task, roughly estimate the time required for the sub-tasks, identify dependencies, and plan the implementation accordingly).
3. Perform a 1-1.5-week sprint to implement and test the feature in question. You will likely not have time to produce a perfect feature, but you should be able to produce a reasonable prototype.
4. Throughout the project, track your time investment, synchronize tasks with your team, and regularly update the plan. Take meeting minutes and track how you divide up the work.
5. Document your code/feature, testing approach, and process and decision-making. It should be easy for the TA to run and test your feature within Mayan EDMS.

We encourage you to read and follow the Team Policy, which has suggestions for effective groupwork in a course setting (you may have to adapt some of these suggestions for collaborating on a software project, but you may still find the structure helpful). It is separately linked from the assignment in Canvas.

Team and GitHub repository.

You will need to create a GitHub repository for the team project. First, collectively choose a **team name.** Your team name should be unique, pronounceable, short, and something you would be proud to shout in your team cheer on the streets of Pittsburgh in the presence of small, impressionable, multi-lingual children.

WARNING: After you join a team, you cannot change teams! Make sure that only one of you creates the team, and make sure that the remaining teammates join the right team.

After you have read the above warning, go here to set up your team:

https://classroom.github.com/g/JXl9c4pR

Use the team repository for all your development, and be sure to use good development practices, including keeping your commits cohesive and your commit messages informative. It is not acceptable for one person to commit all work after synchronizing through other means. This will factor into your grade.

Your team repository will come with starter code that contains a version of Mayan EDMS that is configured to share resources (Python files, some CSS and HTML) from your computer into the container. This allows you to make modifications without having to rebuild or restart the container if you operate on just the shared files. See the instructions at the top of the README.md file on how to run this container on your computer.

Note: While all of the Python code is shared into the container, only the base.css file is shared in the static/appearance directory; if you need to work on other CSS files outside of this (which, you should be able to make most of your changes inside of that single CSS for the purposes of this assignment) please contact one of the instructors.

The **code, testing, and documentation deliverables** below will be taken by snapshotting your repository at the deadline.

Deadlines and deliverables

This homework has three (3) deadlines and four (4) deliverables. The first deadline (Tuesday, September 22, 23:59) is for an **initial planning document and backlog**. The second deadline (Tuesday, September 29, 23:59) is for the **technical artifacts**. The third deadline (Thursday, October 1, 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. Of course, you are still encouraged to start developing before the second deadline and reflecting before the third.

(1) Initial plan (due Tuesday, September 22, 23:59) – 60 points (25%)

- **An initial technical/sprint plan and backlog, with justification.** First, indicate which feature you chose to implement first, and why. Then, present your initial backlog for the sprint (from before you started implementing; you will discuss changes/replanning that happened in a later deliverable). The backlog should include at least:
    - a set of overall tasks necessary to implement the feature (these tasks together should result in a feature, but each task is a small step along the way to the feature),
    - estimated effort for each task,
    - dependencies among tasks,
    - which you plan to tackle for the first sprint to build the core functionality,
    - task assignments for team members for the initial sprint planning.

  You are encouraged to include supporting evidence for your backlog decisions and time estimates (i.e., an explanation for how you arrived at that value).
- **An initial process plan.** Within at most one page (soft limit), briefly describe the *process* you are planning to follow. By process, we mean *how* you are going to develop the system and which

steps you are going to follow, rather than a technical design model of the software. Specifically, we are interested in how you are planning to collaborate (e.g., communication channels, meetings and their frequency, pull requests or dropbox) and in what overall development activities you plan (e.g., how much design, which quality assurance steps, how shall parts be integrated).

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 decomposed the problem and justified your original plan, how you responded to changes in the spring replanning, and how well you analyze and reflect on your experience (see below).

The result should be submitted as a single PDF file uploaded to Gradescope. This document should contain explicit subsections for the backlog/sprint plan and the process plan. Include the names of all team members on the first page of the document.

2) Code artifacts (due Tuesday, September 29, 23:59) – 60 points (25 %)

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, tests for the new feature, and a short documentation of how to use and test your new feature (indicate where this documentation is found from the top-level README.md file).

Adhere to good coding practices. For example, your code should have a clear structure, be reasonably modularized, use appropriate variable names, and be documented. Your feature should be tested. We will look at the commit history; use good practices for cohesive commits with meaningful commit messages.

(3) Reflection documents – Team (due Thursday, October 1, 23:59) – 60 points (25%)

After coding is complete, reflect on your experience 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 Minutes:** Attach all meeting minutes kept throughout the project, which should include information about agenda/topics discussed, decisions made, and work assignments. Recording

results of meetings can be onerous; you must explicitly rotate this responsibility among the members of your team (and indicate who took which minutes).

Summarize your results and submit them as a single PDF file with explicit subsections and all of your names on the first page. Upload your PDF file to Gradescope. The three reflection steps Schedule Deviations, Process, and Team Experience should not exceed a page each single spaced (soft limit); there are no format restrictions on the schedule or meeting minutes.

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. There are many convenient tools for collaborative text editing; the course staff used Google Docs for this homework document.

Extra credit, due with group reflection – 12 pts (5% of the overall assignment grade)

Getting to know your colleagues in a friendly context can often lead to more effective collaboration; healthy teams often get lunch together, for example, in non-pandemic times. Normally, we give extra credit for going out to eat together to build team spirit. Because of covid, we encourage you to do an equivalent activity that you're all comfortable with, to get to know one another better. Consider an online gaming session (drawasaurus?), social "zoom lunch" or a socially distant picnic (i.e., not a "working lunch!"), or some other creative idea you come up with. Include in your reflection document a picture or screenshot documenting the activity for 5% extra credit.

(4) Reflection documents – Individual (due Thursday, October 1, 23:59) – 60 points (25%)

(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 *at least two of the three aspects* mentioned above. You do not need to answer any specific questions, but the following questions may guide possible content for the reflection: 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.

Submit the document as a single PDF file with explicit subsections on Gradescope. Put your name and andrew id on the first page. The entire document should not exceed 1.5 pages single spaced (soft limit).

Grading

You will be graded as a team, with an individual component. This homework is worth 240 points. We will grade you based on the learning goals listed above. The initial planning document constitutes 60 points (25%), the technical artifacts and adherence to process constitutes 60 points (25%), the team reflection document constitutes 60 points (25%), and individual reflection document constitutes 60 points (25%).

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

- A sensible justification for the feature you selected to implement in this sprint.
- A backlog that includes tasks, dependencies, time estimates, and assignments, as described above.
- An outline of the process steps to be adopted in this project.

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

- A running prototype implementation
- Reasonable documentation of the new feature.
- Reasonable code structure and style, including documentation where appropriate
- Coherent commits of reasonable size with meaningful commit messages by all team members
- Reasonable QA/testing for the new feature.

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.
- Inclusion of meeting minutes(s) that adequately demonstrate meeting process

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 on document length can be broken if there is a reason to do so.

The Mayan EDMS is built on top of Django, with most of its core logic written in Python. Learning to use new or unfamiliar frameworks, languages, and technologies is a core skill you will reuse over and over again as a software engineer. So, if you've never built a Django app before, don't worry (and if you've never programmed in Python before, it is a *very* accessible language)! Navigating unfamiliar code is one of the learning goals of this assignment. There are tutorials and templates online for how to extend Mayan EDMS in particular and Django apps in general; we will post supplemental materials as starting points. As a starting point for this system in particular, note that Mayan EDMS features are often integrated as "apps"; if you inspect the source code, you can find the implementation of a number of features as apps. Exploring how these are implemented; how they are deployed/connected to the deployed system; and how/where changes to them manifest as part of the Mayan EDMS may be a good place to start learning how to implement your own new feature. But, as a hint, if no one on your team is familiar with this type of application or development paradigm, you probably want to account for the learning curve in your planning.