Group, Term Project - CS 524 Principles of Software Engineering - Fall 2023

INSTRUCTOR Prof. George Dimitoglou *E-mail:* dimitoglou@hood.edu
Tel: 301.696.3980

INTRODUCTION

Developing a well-designed, tested, functional, and reliable software product is the culmination of multiple software engineering activities. Today, producing and maintaining software is a task that is almost exclusively performed by teams of people. Not surprisingly, a significant portion of success in software projects does not depend on just coding abilities but on a number of other factors such as proper software design, planning, estimation, people communication, activity coordination, project scope management, accurate software measurement, complexity estimation, etc. Scheduling constraints along with the size and complexity of most tasks in the software process are too big to be completed by a single person. It is essential that in every team, tasks are allocated to leverage team strengths and weaknesses. As such, in any team activity, there are occasions for individuals to make significant contributions as well as times in which individuals perform thankless tasks. It is also necessary for each person to contribute and then accurately assess their own performance to meet the team's demands.
Embarking on the production of a sizeable software project requires much more than technical skills. This project will provide you the opportunity to experience and deal with almost every possible facet of the software engineering process.
To succeed, work as a team, "dig in" and "get into it". Good Luck!

2023 SOFTWARE ENGINEERING PROJECT

This semester's project (Development Codename: *ALEX*) will produce an online, multiplayer *CS Jeopardy-style Game* web application. The application will enable users to play the game alone or with others.

Software Requirements. Following is the client description of the requirements. As you can imagine, it is incomplete, so more requirements will emerge, further explanations may be required for some items or clarification may be required for others. It is your responsibility as a software engineering team to fully understand what needs to be accomplished and deliver it.

The application should be interactive, have a user-friendly, intuitive, and appealing web-based GUI, and fully comply with the rules of the official Jeopardy Quiz Show. This means having different categories of questions of varying difficulty, score-keeping, question and game time-keeping, etc. As both a single and multiplayer game, authentication is important and should be provided by a traditional mechanism (e.g., username and password) that includes CAPTCHA functionality and face recognition-based authentication. The application should provide detailed logging about every player's activity during the game. The logging capability should be so advanced that users can retrieve and see a replay of the game. Any user can create a new game and invite other users to play using an invitation board. Administrators should be able to start, and stop game sessions and have access to a game management interface that allows them to visualize the state of the users and gameplay; they should also be able to manually, add, edit and remove, categories and questions and batch upload questions and answers via a CSV file.

TEAMS & LIAISON

Teams. There will be a number of teams each working on a separate product development. It is required that every team member actively participates and contributes to all aspects of the projects. ***Everyone writes documentation, everyone codes***.

Each team must appoint a person as the liaison and point-of-contact (POC) of the team with the instructor. All remote team communication (ex. questions, assignment submissions, software release notifications) about project matters should be channeled through the POC. Failure of the liaison to submit deliverables on time, constitutes a zero grade for the entire team. The role of the POC is abandoned during face-to-face meetings and project presenta-

tions (all team members can, should, may and will have to participate). The duties performed by the POC, ***do not relieve the member in any way from software development or documentation duties***.

Instructor. The instructor is the Principal Investigator (PI), the Client and the End User for this project. In business-speak, the PI is your boss, the person you work for, the person who is paying to develop the project, the person who knows what needs to be done, the person who has a schedule to be met and is the ultimate judge of the process, artifacts, project and results.

The PI assesses your progress, evaluates your intermediate artifacts and grades your results.

The Client and End User roles are obvious here, they are the user of the final product and, for better or for worse, your customer, who ..."*is always right*".

Process & Schedule

Here are some points and directions about the overall process and work protocols.

1. Each team is expected to operate as a single unit.
2. Each team must maintain **a constantly up to date**, publicly accessible version control repository. Every team member is expected to interact when it comes to artifacts through the repository. A team that has any member that operates outside the repository will be subject to a -10% (the team) and -20% (the member) penalty for the entire project.
3. The entire team gets credit for good results and project success, the entire team fails when the results are not there or the project fails.
4. Each team is expected to interact with the PI on a regular basis for consultation, questions, feedback, reviews etc. It is paramount that team questions are aggregated, vetted by the entire team before sent; no random, piecemeal questions.
5. Different teams are welcome to collaborate on technical issues. However, each team is accountable to the PI separately. As teams work on their projects, the direction and feedback may be different, depending on each team's situation, progress and approach. When information is required to be propagated to all teams the PI will do so. Teams are not supposed to make assumptions that have not been agreed upon with the PI. In other words, what the one team is told to do or to avoid is not necessarily applicable to another team.
6. Teams will cross-test releases during the testing phase. Your software release will be tested by another team and they will report on their findings.

Lessons Learned

We are all entitled to make our very own mistakes but here are some hard lessons from previous software engineering projects:

- Being a team player is required for both the individual student and the team to succeed.
- The team matters. Take it seriously. Look for teammates, not buddies. Assign roles and responsibilities.
- Have a leader that coordinates the work. Rotate leadership. Strive for consensus.
- From the time you start development, try to have a working product at all times.
- Unless you become interested and invested in the project, the end product will be sub-par and you will not enjoy working on it.
- Making a wrong turn is part of the process. Try things out sooner rather than later.
- Deal with "unfamiliar" or seemingly difficult tasks earlier in the project or attempt a similar task on a smaller scale during prototyping. For example, if you are developing a multi-threaded web application, concurrency should be the challenging task, not HTML development. "Play" with concurrency first to see the challenges. It is the only way to find out how hard a task may be.

- Dream "grand", promise "doable", deliver on-time.
- I have heard too many times to count the "*I am a seasoned software engineer*" self-description, followed, later, by different variations of:
  - "*I can do the entire project by myself. I don't need a team.*"
  - "*The deliverable is not correct (or, perfect) unless I do it myself.*"
  - "*I have to re-do the whole project to make it better.*"
  - "*The requirements were not well described.*" [no kidding, genius!]
  - "*I don't understand the code, I have to re-write it.*" or, the more ...*sic*, "*I need to refactor the code"*
  - "*I can finish my part the night before is due.*"
  - "*If the code is not optimized, what is the point?*"
  - "I blame *requirements/team member(s)/client needs/technology/existing code-base/<you name it>...* for the
  - "*I don't write code/develop tests/write documentation, I am a software engineer, I delegate.*"

  ...and many other similar statements.

  If anyone in your team makes any of these statements, then it is a huge **red flag**. Chances are this person will seriously jeopardize your project unless you deal with their approach immediately.

LAST WORDS Work hard, work smart, enjoy the project and good luck!