Software Engineering Project

Introduction

The objective of this project is to design and implement a complex software system based on the software process that we have developed in class discussions. Students are expected to work in groups to simulate a real working environment in a small software team to deliver the project by the end of the semester. There will be check points throughout the semester and the final software along with the documentation are expected to be delivered the last week of the semester. Collaboration between all team members is expected using the agile process model, a hybrid model would be accepted too. Each deliverable must be submitted on or before the indicated due date.

The class project should be your own, modeled by the team, designed by the team, and implemented by the team members without external help or use of opensource projects, AI, and libraries unless it is approved by the instructor. If the project uses external code that should be documented and referenced appropriately. The university's regulations and policies for plagiarism should be applied for this project. **NO Machine Learning (ML) or Data Mining projects allowed for this project. The project must have a UI (Web or desktop), save state and data to a database, and must provide user authentication.** Using an IDE to run the project is not considered a UI.

General project requirements:

- The group project size is between three and five students.
- Any Software type will be accepted. The project topic or domain is up to each team to decide, contact the instructor as soon as possible if you need help deciding.
- No specific programming language (or framework) is required as long as all group members agree on the used tools.
- No preference on the IDE or development platform
- Feel free to post a request to join a team on the related Canvas discussion board.

- For project management either Jira or PivotalTracker is recommended, other options are accepted. This is important so each student gets a fair grade for their contribution.
- It's recommended to use gitlab, or Atlassian's free tools for small teams: Jira for project management, Bitbucket for codebase and source control (https://www.atlassian.com/git/tutorials), and Confluence for writing the documentation. This is a free service for students.
- The project history, code commit, and Jira/task board will be part of the project's submission, so make sure to make frequent commits and update the task tracker.
- It's recommended to use PlantUML for modeling, it can be integrated with Bitbucket and Confluence pages. Mermaid is another option.
- The instructor or TA will be asking the team to be added to the project management tool later in the course.
- Project collaboration is required but each team will have to elect a project manager as a point of contact with the instructor and the TA. Students may rotate on leading the discussions and expected to have a review of any pull request to merge to master branch.
- All team members should be included in all communications regarding the project.
- We will be using trunk-based development, read more here (https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development)
- Automation is recommended but not required (https://www.atlassian.com/devops/automation-tutorials)
- All group members should participate equally and help other less experienced team members. Helping other less experienced team members will be captured in each project's survey and counted for extra credit for those who show leadership and more help for the team to succeed.
- Using source control and project management tools will help in solving any team related issues. Make sure to use these tools for all phases of the project.
- **Optional**: If you need resources to run your project on the cloud you can use the UMD cloud or the free tier on AWS, Google cloud, Digital Ocean, etc.

- This is a living document and may change throughout the course, students will be notified of any updates in a timely manner.
- For the class project, there are different checkpoints to be submitted and graded.

At each submission of the project, the delivered document must include:

1- The title page indicating the project title, project description, team member names and roles, and the date.
2- The work executed for the project divided into milestones. A milestone is a defined point in the project timeline that signifies the completion of a specific, measurable goal or deliverable.
3- An estimate number of hours per team member per milestone, as a table.
4- Meeting dates and times between team members since the last document, one entry per week is accepted.
5- Include screenshots of the current project progress in the report. For example, the Software GUI and Jira Board, etc.
6- The team can use Confluence and create a tree structure of documents for each deliverable, export the document and submit it to canvas.

Team info and Project proposal (100 points Due 10/3)

Deliverable: PDF document

- Introduce your team: give your team a unique name and create the group on Canvas (5 pts)
- Introduce each team member: include a biography of each team member, no more than 10 lines. A biography should be professional as if you were introducing yourself to a prospective employer. (15 pts)
- Discuss every aspect of the team agreement in detail and describe all your decisions in the writeup. Your team agreement must include the guidelines for the following (but feel free to add any other items you believe are important): (25 points)

1- methods of communication (email, phone, messenger, text, ...)
2- expected communication response times (email- 24 hours, phone – 4 hours, messenger – 6 hours, text – 6 hours, etc...)

3- meeting attendance (when to meet, whether all meetings are mandatory, ...)
4- running meetings (when, where, face-to-face vs. online, who takes meeting minutes/notes, ...)
5- meeting preparation (whether preparation is needed, what to prepare, ...)
6- version control (what to/not to commit, content of log messages, ...)
7- division of work (how to divide work, who will decide who does what, ...)
8- submitting assignments (when to submit, who will submit, who will review the submission, ...)
9- contingency planning (what if a team member drops out, what if a team member consistently misses meetings, what if a team member is academically dishonest, ...)

- Describe the tools to be used for project development and implementation (5 pts)
- Project domain and proposal:
  - Intended use of the system: who and how the system will be used. (10 pts)
  - Its overall functionality: what will the system do, how will the system help its users accomplish their tasks. (15 pts)
  - Main components of the system: break down the system into logical or architectural components and provide the rationale for this breakdown. (25 pts)

System Requirements (100 points, due 10/17)

Deliverable: PDF document

- General overview of the system's functional requirements. Provide one big use case diagram illustrating the overall functionality of the system. Describe each use case in an easy-to-understand language. (20 pts)
- Provide a separate list of any relevant **nonfunctional** requirements for the system. (10 pts)
- Using the use case diagram as a starting point, convert each use case into a user story. Organize your user stories as a numbered list. (20 pts)
  - A typical format of the user story is as follows: As a **user role**, I want to be able to **goal** so that **reason**:
    - The **user role** represents an actor or a developer who benefits from this story

    - The **goal** of the story is a feature or function in the system, a tool, or a part of a production pipeline.
    - The **reason** describes the benefit to the customer or user when this feature or function is used.
- For each user story, provide a set of pre- and post-conditions (refer to each corresponding user story by their numbers). List pre- and post-conditions under the corresponding user story. There should be a single list of user stories in your document. (20 pts)
- Are there any user stories that are too big or complex? Can some of them be decomposed into smaller user stories? For each user story, mention whether it may need to be broken down (this will be done later). (10 pt)
- Include a glossary that defines all relevant terms that may have a special meaning in the context of the system. (20 pts)

Product backlog (100 pts, due 10/24)

Deliverable: PDF document

- Refine your user stories considering the instructor's feedback. Break down previously identified large user stories. (10 pts)
- Estimate the size of your user stories. Use Fibonacci numbers to represent a relative size of each user story. Note the cumulative size of all user stories in your product backlog. (20 pts)
- Provide an updated numbered list of all user stories; indicate pre- and post-conditions. (10 pt)
- Considering the pre- and post-conditions, identify a subset of user stories to be implemented during the first iteration (there will be a total of three iterations). Be sure that the cumulative size of the selected user stories is about 1/3 of the size of the full backlog. Describe the functionality that your partially implemented system will have at the end of this iteration. (30 pts)
- Design key features of the user interface; provide sketches of your designs. (20 pts)
- Provide screenshots of the Jira board and Github/Bitbucket commit history so far. (10 Points)

Checkpoints (3x100 pts Due 11/11, 11/21, 12/8):

Deliverable: PDF document

- What functionality does the system have at the end of this iteration? (20 pts)
- List the user stories that you have implemented at this iteration. (10 pt)
- Did you end up making any changes to any of these user stories? Did you break down further any the user stories? Did you identify any new user stories during this iteration and, if so, did you add them to the product backlog or decide to implement them right away? Explain. (20 pts)
- What are the "lessons learned" at the end of this iteration? What would you do differently next time? Explain. (20 pts)
- Provide an updated numbered list of all user stories yet to be implemented; indicate pre- and post-conditions. (10 pt)
- (20 pts)
    - If this is not the last iteration:
    - Given the current functionality of the system and considering the pre- and post-conditions, identify a subset of user stories to be implemented during the next iteration. Be sure that the cumulative size of the selected user stories is about 1/3 of the size of the full backlog. Describe the functionality that your (partially implemented) system will have at the end of this iteration.
    - If this is the last iteration: Are there any user stories left unimplemented in the backlog? Are there any new user stories that you would consider adding to the backlog. List these user stories and explain them.

Project demonstration (100 pts, due 12/12)

Deliverable: PDF document and Video Recording

All software developed for this project must be successfully demonstrated by submitting a recorded presentation and demo of the project's functionality. The recording can be between 20 and 45 minutes maximum. This is like presenting in front of the class showing your project highlights. For example, programming languages, tools, frameworks, challenges faced, cool facts about the domain problems and how you solved them. After that comes the

more fun part, ending the presentation with a demo of the running application, showing how your application can be used by stakeholders, such as users, admins, etc. The grading will be based on a slide presentation (25 points) and a walkthrough of your running application (45 points) All students in the group should participate. You can go in details if you would like but make sure the whole presentation and demo are not going over the time limit, that is a total of 45 minutes.

Project documentation and compressed codebase: (30 Points)

- Detail all necessary steps needed to deploy/install your system. Provide all necessary technical specifications. Instructions and scripts. (10 points)
- Explain the main features of the system to a potential user who may not be familiar with it. (10 points)
- Provide a walkthrough for the main scenario of using your system; include screenshots as necessary. (5 points)
- Provide walkthroughs for at least two additional scenarios with additional/alternative functionality; include screenshots as necessary. (5 points)

Grading and final survey

Canvas Survey.

All deliverables will be graded based on the work of the entire team. However, individual students may receive different grades based on the degree and quality of their involvement in the project. To facilitate the objectivity in grading, each student will be **required** to complete a confidential survey about the involvement of other members of his or her team in the project. **These surveys will be strictly confidential. Students who fail to complete this survey will receive a grade of 0 for the entire course project.**