CS100 Final Project

Spring 2021

Description

You will be forming a group of **THREE** students and working on an interesting project. A list of proposed project ideas that have been successful in previous quarters is listed at the end of this document. You can select an idea from the list, start thinking about the features you will implement, what design patterns can help you implement them, and why. If you want to propose your own original idea, you will have to contact an instructor to discuss the project and obtain written permission **before you submit your project proposal**. Your project needs to implement two design patterns iteratively.The project work should be divided almost equally among team members and each member is expected to work on at least one design pattern (more than one partner may work on a pattern) and some of its test cases. You can of course help each other, but it needs to be clear who will be responsible for which patterns and for which features.

Expectations

- Incorporate at least TWO distinct design patterns
    - You need to include at least one design pattern that we will teach this session (Composite, Strategy, Abstract Factory, Visitor, or Decorator)
- All design patterns need to be linked together (it can't be two distinct projects) ● Your project should be implemented in C++. If you wish to choose another programming language (e.g. Java, Python), please discuss with your lab TA to obtain permission. ● You can incorporate additional technologies/tools, but they must be approved (in writing) by an instructor or the TA.
- Each member of the group must be committing code regularly and should make sure their code is correctly attributed to them. We will be checking the attributions to determine if there was equal contribution to the project.
- All project phases are to be submitted to GitHub. You should modify the README file in your project GitHub repository to reflect the different phases of the project. In addition, you should regularly hold sprint meetings with your group. You will need to hold one check-in meeting with your lab TA in addition to the final demo. Details of these meetings are presented below.

Phase I - Proposal (Due Wednesday 4/21/2021 10:00 pm)

You will be completing Phase I prompts in the `README.md` file in your project GitHub repository. The `README.md` file will need to have:

- Title of the project
- Group member names, and links to their GitHub pages
- A Project description
  - Why is it important or interesting to you?
  - What languages/tools/technologies do you plan to use?
  - What will be the input/output of your project?
  - What are the two design patterns you will be using? For each design pattern you must explain in 4-5 sentences:
    - Why you picked this pattern and what feature you will implement with it
    - What problem you anticipate encountering when implementing your project that you will solve using the design pattern
    - Why the chosen design pattern will lead to a good solution to that problem

The description should be in enough detail that the TA/instructor can determine the complexity of the project and if it is sufficient for the team members to complete in the time allotted.

Points: 50

Phase II - Design documents (Due Friday 5/7/2021 10:00 pm)

Update the README.md file by adding the following:

- A class diagram of your project (in OMT)
- A description of the class diagram

Additionally, you will need to:

- Set up your GitHub project board as a Kanban board for the project. It should have (at least) columns that map to Backlog, TODO, In progress, In testing, and Done. ● Create an “Epic” (note) for each feature and each design pattern (in the Backlog column) and assign to the appropriate team member
- Complete your first ***sprint planning*** meeting to plan out the next 7 days of work. ○ Break down the “Epics” into smaller user stories (i.e. smaller development tasks), (in the TODO column). Create issues for them, and assign them to the appropriate team member.
  - These cards should represent roughly 7 days worth of development time for your team, taking you until your sprint meeting with the TA.

Points: 100

Phase III – Development, Testing and Scrum Meeting (Due week 8 at lab time)

You will need to schedule a check-in with the TA (during lab or office hours). Your entire team must be present. Before the meeting you should perform a sprint plan like you did in Phase II. In the meeting with your TA you will discuss:

- How effective your last sprint was (each member should talk about what they did)
- Any tasks that did not get completed last sprint, and how you took them into consideration for this sprint
- Any bugs you have identified and created issues for during the sprint. Do you plan on fixing them in the next sprint or are they lower priority?
- What tasks you are planning for this next sprint.

You may notice that this is effectively a "scrum" meeting although we suggest that as a team you have daily (or every other day) scrum meetings.

Make sure your README file is up-to-date reflecting the current status of your project (e.g. any changes that you have made during the project). Previous versions should still be visible through your commit history.

Points: 50

Phase IV - Final Deliverable (Due week 10 at lab time)

All group members will give a demo to the TA during lab time. The TA will check the demo and the project GitHub repository and ask a few questions to all the team members.

Before the demo, you should update the README.md file by adding the following:

- Screenshot of input/output after running your application
- Installation/usage instructions for the application (including dependencies)
- How the project was tested/validated

Note that your README file should reflect any recent changes you made in your project. Previous versions should still be visible through your commit history.

In addition to this, you need to plan one more sprint (that you will not necessarily complete before the end of the quarter). Your In-progress and In-testing columns should be empty (you are not doing more work currently) but your TODO column should have a full sprint plan in it as you have done before. This should include any known bugs (there should be some) or new features you would like to add. These should appear as issues and cards on your Kanban board.

Points: 100

List of Project Ideas

- Task Scheduler: In this application, a user can create tasks including a title, description, classification (e.g. personal, work, study) priority, duration and due date. Some of these features can also be optional. Users can also create task lists where each list includes multiple tasks. Task lists can represent larger tasks that have subtasks within them. Users can display, edit, and delete tasks and task lists. Users should also be able to undo these operations.

- Quiz Maker: This application allows instructors to create quizzes by uploading text files representing question pools. The uploaded file can contain a list of different types of questions (e.g. multiple choice, true/false, free response) as well as the correct answers (answers can be optional for free response questions). The instructor can then automatically create a quiz based on the uploaded question pool by specifying the required questions types and number of questions. A quiz can also have questions consisting of other sub-questions of different types. Instructors can allow students to take a quiz once or multiple times. The quiz maker can also support multiple file formats for uploading question pools (e.g. plain text, JSON).

- A Text-Based Role-Playing Game (RPG): In this game users can choose among different characters to play with. Characters are associated with different types of weapons and armors. At each step of the game, a narrative is presented and the user can select between different options (e.g. attack, run, access inventory, use a heal item, etc).

- Movie Recommender: In this application movies are organized into categories and subcategories to represent their genres and subgenres respectively. The application allows displaying movies under different categories/subcategories. It also allows adding and removing movies, categories and sub-categories. A user can provide a name of a movie that he/she likes and the application can recommend a list of movies that the user might like. Different recommendation algorithms can be implemented (e.g. based on the movie genre, director, actors) and the resulting recommendations can also be sorted based on their rating, release date, etc.

- Chess Game: This is a traditional two-player chess game, with added features such as saving and loading of incomplete games, allowing players to undo moves, and/or playing with a computer player.

- Library System: A library system allows users to sign up for accounts. After logging in, users can borrow books, show their account debt, or display available books. Books are organized under different genres/sub-genres. The system allows displaying all books or books under a selected genre. Administrative accounts are allowed to add/edit/remove books and reorganize book categories. The library may also provide book recommendations to users based on their previous reading history.