Chess Game Project: Phase 3 Submission Instructions

Objective

In this final phase, you will combine the console-based backend logic from Phase 1 with the GUI developed in Phase 2, creating a fully functional, interactive GUI-based chess game. You'll also document your code structure and prepare for an in-class presentation.

Requirements

1. Integration of Backend Logic and GUI

- Combine the console-based chess game logic from Phase 1 with the GUI developed in Phase 2.
- Implement all game rules within the GUI, including:
  - **Move Validation**: Ensure moves are valid according to chess rules.
  - **Capture and Check**: Refresh the GUI to reflect captured pieces and check scenarios.
  - **Checkmate Detection**: Recognize when a checkmate occurs and end the game.
- Refer to the provided **TicTacToe** game source code on the GitHub repository to see an example of integrating GUI with backend game logic.

2. Class Diagram

- Create a Class Diagram using one of the UML diagram tools introduced in class.
- The diagram should illustrate the architecture of your project, including:
  - **Class Details**: Name, attributes, and operations for each class.
  - **Access Specifiers**: Clearly indicate access levels (e.g., public, private).
  - **Relationships and Multiplicity**: Show class relationships (e.g., association, inheritance) and multiplicity numbers where applicable.

3. Code Quality and GitHub Repository Management

- Ensure that your code is well-commented, with proper explanations for complex sections.
- Push the code to your GitHub repository, ensuring there are multiple commits with clear and descriptive commit messages.

4. README File

-Create a README file on your GitHub repository, including:

- Team name.
- Team members' names.
- Semester, course number, and section number.
- One or more preview pictures to showcase the GUI Chess game interface.
- Display the Class Diagram.
- Instructions on how to compile, start, and run the chess game.
- A checklist of achieved features within the GUI chess game.

-Note: The Phase 1 and Phase 2 code can be placed in other branches or separate folders. Detailed README documentations for Phase 1 and Phase 2 are not required.

-Note: Please look at the provided README.md template file to prepare your project README.md. You can edit, add, delete any information from the template file.

5. In-Class Presentation

- Prepare a **5** minutes presentation to demonstrate your project.
- Demo: Show a live walkthrough of your chess game, demonstrating:
  - Valid moves and captures.
  - Handling check and checkmate.

  - Other extra credit features that you implemented
- Be ready to answer questions regarding your source code and design choices.

- The final presentation will be held in class on **Dec 2nd and 4th**. Each group should have at least one member present for the presentation, though I encourage all members to participate. All teams should complete all requirements on **Dec 1st**, and be prepared to present on **Dec 2nd**. I will post the presentation order by midnight on Dec 1st, and it will be randomly generated. You may switch the presentation order with another team as long as both teams agree. Teams that do not have enough time to present on Dec 2nd will continue on Dec 4th.

To set up for the presentation, you should bring a laptop that can run your project without issues. Your laptop should also have internet access and Zoom installed. During the presentation, the instructor will start a Zoom meeting, and each group will share their laptop screen to present.

Grading Criteria

- Integration and Functionality: 50% - Successfully integrating game rules with the GUI and achieving a fully functional chess game.
- Class Diagram and Documentation: 35% - Clarity and detail in the UML diagram, README file, and code comments.
- Code Quality and GitHub Management: 5% - Proper commit messages, code organization, and repository management.
- Presentation: 10% - Demonstrating the game's functionality and effectively explaining the project's structure.

Extra Credits

- For the teams who want to go beyond the basic requirements and earn extra credit, you can implement any of the following advanced features for **additional 10% credit each**. These credits can be accumulated if you implement multiple extra features. Each extra feature should be functional and fully integrated into your chess game. Please indicate what you have achieved in the GitHub repository README and demonstrate it in the presentation clearly.

Extra Credit 1: AI Chess Player

- Implement an AI chess player that can automatically move chess pieces on the board in response to the human player's moves. The AI should:

  -Analyze the board and make legal moves as an opponent. The AI should make moves that adhere to chess rules and provide a reasonable challenge to the human player.

  -Use a basic chess algorithm or integrate an AI model (such as a chess engine or deep learning model) to decide its moves. The AI may be based on any chess algorithm (e.g., Minimax, Alpha-Beta pruning) or a deep learning model if you choose.

Extra Credit 2: Online Multiplayer Mode

- Enable two players to play the chess game remotely over the internet. Players should be able to:

  - Install the chess game on their computers and connect via IP address.

  - Take turns to move pieces, with both computers displaying the current game board and game status in real-time.

You can also continue implementing one or more extra credit features described in the Phase 2 submission instruction if you didn't do them in the phase 2.

Useful resources:

https://git.txstate.edu/x-l30/CS3354-Fall2024/tree/main/ProjectMaterial/Phase%203