Advanced Tic Tac Toe Game

By Dr.Omar Nasr, and ChatGPT
April, 2024

Project Overview:

Develop an advanced version of Tic Tac Toe game that incorporates user authentication, personalized game history, and an intelligent (Artificial Intelligent) AI opponent. The game will support both player-vs-player and player-vs-AI modes, allowing users to log in, track their game history, and analyze past games. The project will employ best practices in software engineering, including secure user management, rigorous testing, and professional version control workflows.

Objectives:

- Create an interactive Tic Tac Toe game using C++, implementing a user-friendly GUI.
- Incorporate user authentication to manage individual player profiles and game histories.
- Develop a challenging AI opponent using strategic algorithms to enhance gameplay.
- Employ unit and integration testing to ensure software quality and reliability.
- Integrate CI/CD practices using GitHub for continuous integration and deployment.

Core Components:

Game Logic Implementation:

- Develop the basic game mechanics allowing two players and a player and AI to take turns marking a 3x3 grid.
- Implement game rules to check for a win or a tie after each move.

AI Opponent:

- Integrate a minimax algorithm with alpha-beta pruning to power the AI opponent, enabling it to make strategic moves based on the game state. You can use other techniques to develop the AI agent

Graphical User Interface (GUI):

- Design a GUI that displays the Tic Tac Toe board and allows players to interact with the game through clicks.
- Include additional GUI elements for managing user sessions, including login forms, registration forms, and game history views.

User Authentication and Management:

- Implement a secure login and registration system where players can create accounts, log in, and manage their profiles.
- Use password hashing for secure storage of credentials and session management for keeping users logged in.

Personalized Game History:

- Allow players to save and view their game histories, with details about each game session stored securely in an efficient format.
- Enable players to review and replay past games from their history.

Testing and Quality Assurance:

- Develop comprehensive unit tests using Google Test to validate each component of the game.
- Implement integration tests to ensure all parts of the system work together seamlessly.

CI/CD Integration:

- Set up GitHub Actions to automate the testing and deployment processes, ensuring that updates are smoothly rolled out and maintain high code quality.

Performance Optimization and Metrics:

- Monitor and optimize the game's performance, focusing on metrics such as response time and system resource utilization.

Implementation Details:

Technologies and Tools:

- Programming Language: C++
- Use Object Oriented Programming techniques
- Testing Framework: Google Test for unit and integration testing.
- Version Control System: Git, with GitHub as the remote repository.
- CI/CD Tools: GitHub Actions for automated testing and deployment.
- Database: SQLite or a similar lightweight database system for managing user data and game histories. You can also using a customized file storage with a format of your choice
- Security Practices: Use of secure hashing algorithms for password storage
- GUI Framework: Qt or similar libraries suitable for C++.

Data Structures:

- Trees for implementing the minimax algorithm for the AI opponent.

- Other data structures like stacks, hashtables and others can be used in different parts of the system

Research and Development Guidelines:

- Team members are encouraged to use resources like ChatGPT and online research to aid in problem-solving but must understand and be able to justify every line of code they write.
- Proper citations are required for all third-party resources used.
- Collaboration and knowledge sharing are encouraged, but all submissions must be original and properly attributed.

List of Deliverables

1. Software Requirements Specification (SRS):
    - Purpose: Document detailing all functional and non-functional requirements, including game rules, system behavior, and performance requirements.
2. Software Design Specification (SDS):
    - Purpose: Contains the software architecture and design details, including UML diagrams such as class diagrams and sequence diagrams, to ensure a well-structured implementation.
3. Codebase:
    - Purpose: Complete source code of the game in C++, following Google C++ Style Guidelines, encompassing game mechanics, AI algorithms, and user management.
4. Testing Documentation:
    - Purpose: Documentation of testing strategies and results, including unit and integration tests using Google Test, with detailed test cases and coverage reports.
5. CI/CD Pipeline Configuration:
    - Purpose: Configuration for Continuous Integration and Continuous Deployment using GitHub Actions, including scripts for building, testing, and deploying the software.
6. Performance Measurement and Optimization:
    - Purpose: Report on performance benchmarks and optimization efforts, measuring metrics like response time, memory usage, and CPU utilization to ensure efficient game performance.
7. Version Control Repository:

- Purpose: A Git repository containing all project components, including code, documentation, and configurations, ensuring effective version control and change management.

Team size and deadline:

- Team size : up to 5 students per project
- Every team member should be aware of all aspects of the project
- Deadline for submission: one week before the first final exam (18 May, 2024)
- This project is worth 15 points

Submission format:

- To be announced soon
- There will be an oral presentation by each team member of the project