Software Engineering

SRS Document

Name: Zain Rehman

Roll no.: BSCE20032

Project Name: Chess Game Development

1. Introduction

1.1 Purpose

- The purpose of this document is to define the requirements for a Chess game software application. This application aims to provide users with an interactive and engaging experience of playing Chess against a computer opponent or other players.

1.2 Scope

- The Chess game software will include features such as game setup, move validation, game logic implementation, user interface, and multiplayer capabilities. The application will be developed for desktop platforms.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- GUI: Graphical User Interface

2. Overall Description

2.1 Product Perspective:

- The Chess game software will be a standalone application, providing users with a virtual Chessboard and pieces. It will implement the rules and logic of Chess, enabling players to make valid moves and compete against each other.

2.2 Product Features

The Chess game will include the following features:

- **Game Setup:** Allow users to set up the board, choose player colors, and configure game options.

- **Move Validation:** Verify the legality of moves based on the rules of Chess, including checks, captures, castling, en passant, and promotion.

- **Game Logic:** Implement the game rules, including turn-based play, checkmate detection, and stalemate conditions.

- **User Interface:** Provide an intuitive and visually appealing GUI for players to interact with the Chessboard and game controls.

- **Single Player Mode:** Allow users to play against a computer opponent with adjustable difficulty levels.

- **Multiplayer Mode:** Enable players to compete against each other over a network or locally on the same machine.

2.3 User Classes and Characteristics

- The Chess game will be designed for users who are familiar with the rules and gameplay of Chess. Players should have basic computer literacy and an understanding of Chess strategies and tactics.

2.4 Operating Environment

- The Chess game will be developed as a desktop application using a suitable programming language such as Java, C++, or Python. It should be compatible with common operating systems like Windows, macOS, and Linux.

3. Functional Requirements

3.1 Game Setup

3.1.1New Game

- Users should be able to start a new game with a clean   Chessboard and default game settings.

3.1.2Customize Game

- The system should allow users to configure game options, such as player colors, time controls, and AI difficulty levels.

3.2 Move Validation

3.2.1Legal Moves

- The system should validate moves according to the rules of Chess, considering piece movement, captures, special moves (en passant, castling), and promotion.

3.2.2 Check Detection

The system should detect checks (when a player's King is under attack) and restrict moves that would leave the King in check.

3.2.3Checkmate and Stalemate Detection

- The system should detect checkmate (when a player's King is in check and there are no legal moves to escape) and declare the winner.

- The system should detect stalemate (when a player has no legal moves but is not in check) and declare a draw.

3.3 User Interface

3.3.1 Chessboard Display

- The system should visually display the Chessboard and pieces with proper colors and arrangement.

- Players should be able to interact with the Chessboard using a mouse or keyboard.

3.3.2 Move Input

- The system should provide a user-friendly interface for players to input their moves, either by dragging and dropping pieces or using algebraic notation.

3.4 Single Player Mode

3.4.1 AI Opponent

- The system should implement an AI algorithm to provide an intelligent computer opponent.

- The AI difficulty level should be adjustable to cater to players of different skill levels.

3.5 Multiplayer Mode

3.5.1Network Multiplayer

- The system should support multiplayer functionality, allowing players to connect over a network and play against each other.

- The application should provide matchmaking capabilities and handle game synchronization.

4. Non-Functional Requirements

4.1 Performance

- The Chess game should respond swiftly to user actions, providing a seamless and immersive gaming experience.

- AI computations should be optimized to ensure fast and efficient move generation.

4.2 User Experience

- The user interface should be visually appealing, with clear and intuitive controls.

- The application should provide feedback on move validity, check notifications, and game outcome.

4.3 Security

- The system should implement appropriate security measures to prevent unauthorized access or tampering with game data.

- User access should be clearly defined, with different roles and permissions for players, administrators, and other stakeholders.

- Storage encryption should be implemented to protect sensitive user data, such as login credentials and game history.

4.4 Reusability (Object-Oriented Approach)

- The software should be designed and developed using an object-oriented approach, promoting modularity and code reuse.

- Classes and components should be organized in a hierarchical manner, with clear separation of concerns and encapsulation of functionality.

- The system should utilize inheritance, polymorphism, and other object-oriented principles to enable easy extension and adaptation of the software for future use.

4.5 Maintainability (Customer Maintainability)

- The software should be designed for ease of maintenance and updates by the customer, with minimal dependency on the development team.

- Code should be well-structured, modular, and well-documented, enabling customers to understand and modify the system as needed.

- The system should provide a comprehensive user manual and technical documentation, including instructions for troubleshooting and common maintenance tasks.

4.6 Reliability (Operational Availability)

- The Chess game should have a clearly defined operational availability, specifying the expected uptime and downtime of the system for maintenance and updates.

- The system should be robust and resilient, with mechanisms in place to handle errors, exceptions, and unexpected events.

- The application should include appropriate error handling and recovery mechanisms to minimize the impact of failures on the user experience.

5. Constraints

- The Chess game should be developed within the specified timeframe and adhere to the project's technical and resource limitations.

- The application should be scalable and able to handle different board sizes, variants, and rule modifications, if desired.