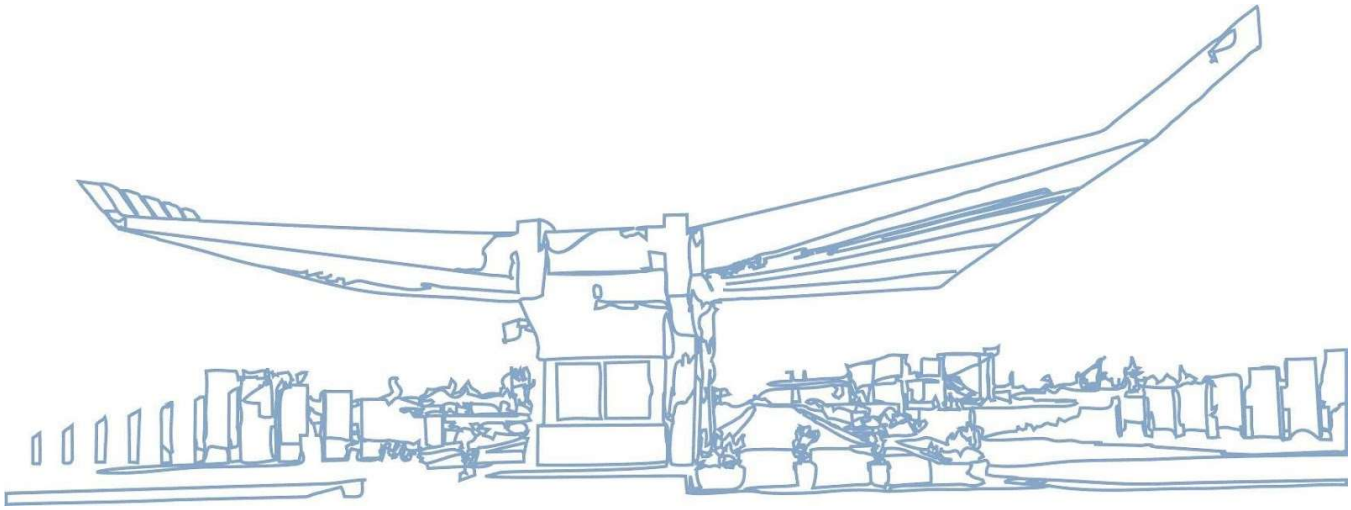
EPOKA UNIVERSITY

INTRODUCTION TO SOFTWARE ENGINEERING

Chess Game



Team Members:

Thanas Papa

Roni Domi

Klejdi Gjyzeli

Chess Game Requirements Specification

EPOKA UNIVERSITY

Table of Contents

1. Executive Summary

1.1 Project Overview

The project is the development and designing of a chess game meant to be played by two players locally. The game incorporates all the moves and features of chess, and allows the user to customize their experience visually but also in gameplay for example by manually choosing the time control of the game.

1.2 Purpose and Scope of this Specification

Purpose:
The purpose of a chess game is to provide an intellectually stimulating and competitive activity for players to test their strategic thinking, problem-solving skills, and tactical abilities.

Scope:
The scope of a chess game encompasses various aspects related to the game itself and the playing experience. This includes:

- Gameplay: The scope of a chess game involves the rules, mechanics, and dynamics of the game. It encompasses the movement and abilities of each chess piece, the objective of capturing the opponent's king, and the strategies and tactics employed by players to outmaneuver their opponents.
- Player Interaction: The game can be played between two players, in person. The scope of the game includes the interaction between players, such as making moves, capturing pieces.
- Decision-Making: Chess requires players to make critical decisions at every turn. The scope of the game involves the evaluation of different moves, considering potential consequences, and selecting the best possible course of action to gain an advantage over the opponent.
- Competitive Element: Chess is inherently competitive, and the scope of the game includes the aspect of competition between players. It involves the quest for victory, the pursuit of outsmarting the opponent, and the satisfaction of winning the game.

In-Scope:

- Game rules and mechanics.
- Valid moves and piece interactions.
- Player decision-making and strategy.
- Capturing the opponent's pieces.
- Checkmate and victory conditions.

Out-Of-Scope:

- Variants and modifications beyond standard chess rules.
- Non-traditional game boards or pieces.
- Non-essential visual and graphical enhancements.
- Platform-specific features not directly related to gameplay.
- Historical or cultural context surrounding chess (unless explicitly incorporated into the game).

2. Product/Service Description

2.1 Product Context

The game of chess is unique allowing it to be independent and stand alone when compared to other games. It has stood the test of time convincingly. The game is intended to be played without third party assistance, external interfaces or connections.

2.2 User Characteristics

Student - Experience with video games, basic computer knowledge, knowledge of chess rules
Staff/Faculty - Casual video gaming experience, knowledge of basic gaming mechanics, basic computer knowledge and knowledge of chess rules
Other - Gaming and computer experience, knowledge of chess rules

2.3 Assumptions

Game is not demanding on the system and can run on most hardware specifications. Basic hardware is required. No software is necessary besides the game itself. Due to unavailability for testing on other operating systems, the game is only known to be functional on Windows systems, however nothing suggests that the game would not be successful on other operating systems.

The user is required to have a basic grasp on computer use and chess rules.

2.4 Constraints

- **parallel operation with an old system**

  If the chess game needs to coexist or interact with an older system, it may need to adhere to specific protocols or data formats to ensure compatibility and smooth operation between the two systems. This constraint requires consideration of integration requirements and potential limitations imposed by the older system.
- **audit functions (audit trail, log files, etc.)**
  To ensure transparency and accountability, a chess game may require audit functions such as an audit trail and log files. These functions track and record important events, moves, and player actions, enabling administrators or players to review and analyze the game's history.
- **criticality of the application**
  While a chess game may not be considered mission-critical like some other applications, it should still prioritize stability, reliability, and consistent performance. Players expect a smooth and uninterrupted gaming experience, and any disruptions or system failures could impact user satisfaction and engagement.
- **system resource constraints**
  If there is insufficient memory whether that is RAM or ROM the game would experience problems or even crash
- **other design constraints**
  Various additional design constraints may apply to a chess game, such as:

  1. Performance: The game should be designed to handle a large number of concurrent players, quick response times for move processing, and efficient resource utilization.
  3. User Interface: The user interface should be intuitive, visually appealing, and responsive, allowing players to easily understand the game state, make moves, and interact with the system.
  4. Platform Compatibility: Depending on the target platforms (e.g., desktop, mobile, web), the chess game may need to conform to specific design guidelines and technical requirements to ensure compatibility and optimal user experience across devices.
  5. Localization: To cater to a global audience, the game may require support for multiple languages and cultural adaptations to make it accessible and enjoyable for players from different regions.

2.5 Dependencies

Platform Dependencies: The requirements of the chess game may be influenced by the platform(s) on which it will be developed and deployed. Different platforms, such as desktop, mobile, or web, may have specific technical considerations, user interface guidelines, and performance expectations that impact the requirements.

Technology Dependencies: The choice of programming languages, frameworks, libraries, and tools can affect the requirements of the chess game.

User Base Dependencies: The target audience and user base of the chess game can impact the requirements. If the game is intended for casual players, the requirements may focus more on simplicity, ease of use, and a visually appealing interface. On the other hand, if the game targets professional or competitive players, the requirements may prioritize advanced features, performance optimizations, and integration with chess databases or analysis tools.

Third-Party Dependencies: If the chess game relies on third-party services, libraries, or components, the requirements may need to consider compatibility, licensing terms, and any limitations or constraints imposed by those dependencies.

3. Requirement

3.1 Functional Requirements

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
FR-01	Chessboard and Piece Representation: The chess video game should accurately represent the chessboard with its 64 squares and display the chess pieces in their correct positions.	The pieces should have proper visual representations and be distinguishable from each other.	1	05/31/23	Approved
FR-02	Legal Move Validation: The game should enforce the rules of chess and validate the legality of moves made by the players. It should prevent players from making illegal moves such as moving a piece to an occupied square, making invalid piece movements, or attempting moves that put their own king in check.		1	05/31/23	Approved

FR-03	Game State Management: The video game should maintain and update the game state, including the positions of the chess pieces, the current player's turn, and the state of the game (check, checkmate, stalemate, draw).	It should accurately reflect the progress and outcome of the game.	1	05/31/23	Approved
FR-04	Local Multiplayer Support: The chess video game should allow for local multiplayer functionality, enabling players to compete against each other on the same device or through a local network. It should provide options for players to take turns on a shared screen or have separate screens with their own view of the chessboard. The game should support multiple input methods, such as using separate controllers, sharing a single controller, or even using touchscreen capabilities for mobile devices.	It should also offer features like player customization, game settings, and the ability to save and resume local multiplayer sessions.	3	05/31/23	Approved
FR-05	Game Modes and Options: The chess video game should offer various game modes and options to enhance the gameplay experience. This may include features such as timed matches, different chess variants (e.g., chess 960 or three-check chess),	These features add depth and variety to the game and cater to different player preferences.	3	05/31/23	Approved

	the ability to save and load games, options for undoing moves, a tutorial or training mode, and multiplayer functionality for online play with other players.				

3.2 Non-Functional Requirements

Product Requirements:

Performance: The game should run smoothly and respond quickly to player actions, with minimal lag or delays.
User Interface: The user interface should be intuitive, visually appealing, and accessible, with clear instructions and visual cues to guide players.
Compatibility: The game should be compatible with various platforms and devices, such as desktop computers, consoles, and mobile devices, to reach a wider audience.

Organizational Requirements:

Development Frameworks and Technologies: The game should be developed using specific frameworks or technologies specified by the organization to ensure compatibility and ease of maintenance.
Development Timeline: The project should adhere to a predefined timeline and milestones, allowing for efficient development, testing, and deployment of the game.
Documentation and Support: The development team should provide comprehensive documentation, including user manuals and technical documentation, to assist users and future development efforts. Additionally, a support system should be in place to address user inquiries and issues.

External Requirements:

Legal and Compliance: The game should comply with applicable laws, regulations, and industry standards, such as intellectual property rights and data privacy.
Localization: The game should support multiple languages and cultural preferences to cater to a global audience.
Accessibility: The game should consider accessibility guidelines, ensuring that players with disabilities can fully enjoy and interact with the game.
Network Connectivity: If the game includes online features, it should be designed to handle different network conditions and provide a smooth online experience.

3.2.1 User Interface Requirements

Main Menu Interface:

Intuitive Design: The main menu should have a clean and intuitive layout, making it easy for users to navigate and access different game modes and options.
Clear Menu Hierarchy: The menu should present a clear hierarchy of options, allowing users to easily understand the available choices and navigate through different levels of the menu.
Visual Feedback: The interface should provide visual feedback, such as highlighting selected options or displaying loading indicators, to provide users with a sense of control and responsiveness.

Gameplay Interface:

Clear Chess Board Visualization: The chessboard should be visually clear, with well-distinguishable squares and distinct pieces. Clear icons or symbols should represent each chess piece.

Move Feedback: The interface should provide immediate visual feedback when a player selects and moves a chess piece. This feedback can include highlighting the valid moves for the selected piece or animating the movement of the piece.
Status Indicators: The interface should display relevant status indicators, such as whose turn it is, check or checkmate alerts, and timers (if applicable) to keep players informed of the game's progress.

Game Options Interface:

Customization Options: The interface should allow players to customize various game settings, such as difficulty levels, time controls, and visual themes. This customization can enhance the gameplay experience and cater to different player preferences.
Clear and Accessible Settings: The options should be presented in a clear and organized manner, making it easy for users to understand and modify the desired settings. The interface should also provide appropriate tooltips or explanations for unfamiliar settings.

3.2.2 Performance

Static Numerical Requirements:

Maximum Board Size: The system should support the standard chessboard size of 8x8 squares, limiting the maximum number of squares on the board to 64.
Piece Quantity: The system should handle a maximum of 32 chess pieces on the board at any given time (16 for each player).

Time Controls: If the game includes timed matches, there may be static numerical requirements for the minimum and maximum time limits, such as a minimum of 3 minutes and a maximum of 15 minutes per player.

Dynamic Numerical Requirements:

Maximum Moves per Turn: The system should allow for a maximum number of moves per turn, such as one move per turn for each player in traditional chess.

Move Validation Time: The system should provide timely move validation, ensuring that move validation time remains within an acceptable range, such as validating moves within 1 second.

3.2.3 Usability

Intuitive User Interface: The game should have a user interface that is easy to navigate and understand, with clear labels, icons, and visual cues. It should minimize the need for user assistance or extensive learning before being able to play.

Responsive Controls: The game should have responsive controls that accurately reflect player input and provide immediate feedback. The movements of the chess pieces should feel natural and smooth, without any noticeable delays or inconsistencies.

Clear Game Feedback: The game should provide clear and informative feedback to the players. This includes visual indicators of valid moves, highlighting of selected pieces, and notifications of check, checkmate, or other game states.

Error Prevention and Handling: The game should proactively prevent or minimize user errors by validating moves and preventing illegal actions. In cases where errors occur, the system should provide clear error messages or prompts to help users understand and rectify the issue.

Customization Options: The game should offer customization options to allow players to tailor their experience. This may include adjustable difficulty levels, visual themes, sound settings, and other personalization options.

Help and Documentation: The game should provide easily accessible help resources, such as in-game tutorials, tooltips, or a comprehensive user manual. This information should be readily available to assist users in understanding game mechanics, rules, and features.

Consistency: The game should follow consistent design patterns and conventions, both within the game itself and in alignment with common user interface practices. This promotes familiarity and reduces the learning curve for players.

Accessibility: The game should strive to be accessible to a wide range of users, including those with disabilities. This may involve providing options for adjustable font sizes, color schemes, audio cues, and keyboard-only controls.

3.2.4 Manageability/Maintainability

3.2.1.1 Monitoring

Health Monitoring:

System Performance Monitoring: The game should continuously monitor its performance metrics, such as CPU usage, memory consumption, and network latency, to detect potential bottlenecks or issues that may impact the gameplay experience.

Boundary and Input Validation: The system should validate user input, such as move commands, to detect and handle erroneous or invalid data.

Error Handling: The game should have robust error handling mechanisms to gracefully handle unexpected errors, exceptions, or exceptional conditions, providing informative error messages to users when necessary.

Logging and Reporting:

Logging: The system should generate log files to capture important events, errors, and exceptions that occur during gameplay or server operations. This log information can assist in diagnosing issues and provide insights for debugging and improvement.

Reporting: The game should allow users to report any bugs, issues, or feedback through a dedicated reporting mechanism, such as a feedback form or a support email address. This enables users to communicate problems they encounter for prompt investigation and resolution.

Correction and Bug Fixing:

Bug Fixing Process: The development team should have a defined process for identifying, prioritizing, and addressing bugs or issues reported by users or identified through monitoring and testing.

Timely Updates: The game should be regularly updated with bug fixes and patches to address identified issues and improve the overall stability and performance of the game.

3.2.1.2 Maintenance

Modularity: The system should be designed with a modular structure, where different components and functionalities are organized into separate and cohesive modules. This allows for easier maintenance and updates as changes can be made to specific modules without affecting the entire system.

Clear Code Organization and Documentation: The codebase should be well-organized and follow coding best practices, making it easier for developers to understand and modify. Additionally, comprehensive documentation, including comments and API references, should be provided to assist in maintenance tasks and facilitate future updates.

Separation of Concerns: The system should separate different concerns, such as game logic, user interface, and data management, into distinct components. This separation allows for targeted modifications and improvements without impacting other parts of the system.

Version Control: The use of version control systems, such as Git, enables easy tracking of changes, branching, and merging. It ensures that developers can work on different features or bug fixes concurrently while maintaining a clear history of changes, simplifying maintenance tasks.

Error Logging and Monitoring: The system should have robust error logging mechanisms in place to capture and log runtime errors, exceptions, and other relevant information. These logs can help identify recurring issues and guide maintenance efforts to address them effectively.

Configurability and Customizability: The system should provide configuration options or settings that allow for easy customization and adaptation to different environments or user preferences. This reduces the need for extensive code modifications and enables flexibility in maintaining the system.

Testability and Automated Testing: The system should be designed with testability in mind, allowing for the creation of automated tests to validate functionality and detect regressions. Automated testing helps catch issues early and simplifies maintenance by ensuring changes don't introduce new problems.

Clear and Well-Defined Interfaces: The system's interfaces, both internal (between system components) and external (user interfaces, APIs), should have clear definitions and documentation. Well-defined interfaces make it easier to understand and modify specific components without disrupting the entire system's functionality.

3.2.1.3 Operations

Normal Operations:

- Launching the chess video game application
- Navigating the game menu to select play mode (e.g., single-player, multiplayer)
- Starting a new game
- Interacting with the game board to make moves using mouse or keyboard inputs
- Displaying game progress, such as capturing opponent's pieces and checkmate situations
- Offering in-game options, such accessing a help menu
- Ending the game and returning to the main menu or exiting the application

Special Operations:

- Periods of Interactive Operations: This refers to the time when the user actively engages with the game by making moves, analyzing the board, and deciding the next move. It includes all the actions performed during gameplay, such as selecting and moving chess pieces.
- Periods of Unattended Operations: This pertains to the time when the user is not actively interacting with the game. It includes scenarios such as leaving the game idle, allowing the user to take breaks, or the game being minimized while the user attends to other tasks.

Data Processing Support Functions:

- Validating and verifying moves made by the user to ensure adherence to chess rules
- Determining legal moves for each chess piece based on the current game state

- Checking for checkmate or stalemate conditions to determine the end of the game
- Calculating and updating game statistics, such as player ratings or win-loss records
- Managing and updating the game state, including board positions, captured pieces, and time controls

Backup and Recovery Operations:

- Periodically creating backup copies of game saves and settings to prevent data loss

Safety Considerations and Requirements:

- Ensuring the game adheres to age appropriateness and content guidelines
- Implementing measures to prevent cheating or unfair advantages, such as anti-cheat mechanisms
- Providing appropriate warnings or disclaimers for potential health risks associated with prolonged gaming sessions

Disaster Recovery and Business Resumption:

- Implementing backup systems and redundant infrastructure to ensure availability in case of system failures or disasters
- Regularly testing disaster recovery plans to ensure the ability to recover game data and services
- Developing business resumption strategies to minimize downtime and restore operations in the event of a major disruption or loss

3.2.5 Security

3.2.5.1 Protection

- **User Awareness:** Educate players about secure gaming practices, including avoiding downloading or executing files from untrusted sources, not sharing sensitive information within the game, and being cautious of social engineering attempts or phishing attacks.
- **Regular Updates and Security Patches:** Maintain the game software by providing regular updates and security patches. This helps address any vulnerabilities or weaknesses discovered over time, reducing the risk of exploitation.

3.2.5.2 Authorization and Authentication

The game does not include profile creation therefore it does not make use of authorization and authentication.

3.2.6 Standards Compliance

Age Restrictions: Comply with age restrictions to ensure that the game is not accessible to underage players.

Intellectual Property Rights: Respect and comply with intellectual property laws to avoid copyright infringement related to chess piece designs, music, or any other copyrighted material used in the game.

Accessibility Standards: Follow accessibility standards to make the game accessible to players with disabilities, ensuring that they can fully engage in the gameplay experience.

3.2.7 Other Non-Functional Requirements

- The game should have smooth and responsive gameplay, with minimal latency or delay in moves and interactions.
- The game should be designed to handle varying levels of complexity and game states without impacting performance or responsiveness.
- The user interface should be intuitive, user-friendly, and easy to navigate, allowing players to understand and interact with the game without confusion.
- The game should be compatible with a wide range of devices and operating systems, ensuring that players can enjoy the game on their preferred platforms.
- The game should be stable and reliable, minimizing crashes, freezes, or unexpected termination during gameplay.

3.3 Domain Requirements

4. Game mechanics:
   - 4.1 Movement and Rules: The game should accurately implement the movement rules for each chess piece, including their individual abilities such as castling, en passant, and pawn promotion.
   - 4.2 Checkmate and Stalemate: The game should correctly identify and display checkmate and stalemate conditions, ending the game when necessary.
   - 4.3 Special Moves: Implement additional features like undoing moves, offering draws, or requesting hints to enhance the gameplay experience.
5. Graphics and user interface:
   - 5.1 Visual Design: Create visually appealing and realistic 2D or 3D chessboard graphics with high-quality textures and lighting effects.
   - 5.2 Piece Differentiation: Ensure that each chess piece has a distinct design, making it easy for players to identify and differentiate between them.
   - 5.3 Intuitive Controls: Design a user-friendly interface that allows players to easily initiate and restart games, control piece movement, and access relevant game information.
   - 5.4 Notifications and Feedback: Provide clear visual cues and messages to inform players about check, checkmate, stalemate, and other important game events.
6. Input and output:
   - 6.1 Keyboard Inputs: Enable players to input moves using the keyboard, following standard chess notation or a user-friendly input format.
   - 6.2 Output Display: Render the chessboard, move history, current score, and relevant game messages clearly on the screen to provide players with essential information.
7. Performance:
   - 7.1 Smooth Gameplay: Ensure the game maintains smooth performance, responding quickly to player inputs and rendering changes to the chessboard without noticeable delays.
   - 7.2 Optimize Resource Usage: Implement efficient algorithms to handle complex board positions, multiple moves, and calculations, ensuring the game remains responsive even in demanding scenarios.
8. Compatibility:
   - 8.1 Cross-Platform Compatibility: Develop the game to be compatible with various operating systems, such as Windows, macOS, and Linux, allowing users to enjoy the game on their preferred platform.
   - 8.2 Responsive Design: Create a responsive user interface that adapts to different screen sizes and resolutions, providing an optimal experience across a range of devices.
9. Code structure and organization:
   - 9.1 Modular Design: Structure the game's code into logical and reusable modules, promoting code reusability and maintainability.
   - 9.2 Clear Documentation: Include comprehensive comments and documentation within the code to facilitate understanding and future modifications by other developers.
10. Testing and debugging:
   - 10.1 Comprehensive Testing: Conduct thorough testing to ensure the game functions correctly, covering various scenarios, edge cases, and corner cases to validate its

adherence to the rules of chess.

10.2 Bug Fixing: Address and resolve any identified bugs or errors promptly, ensuring a reliable and bug-free gaming experience.

11. Design thinking methodologies

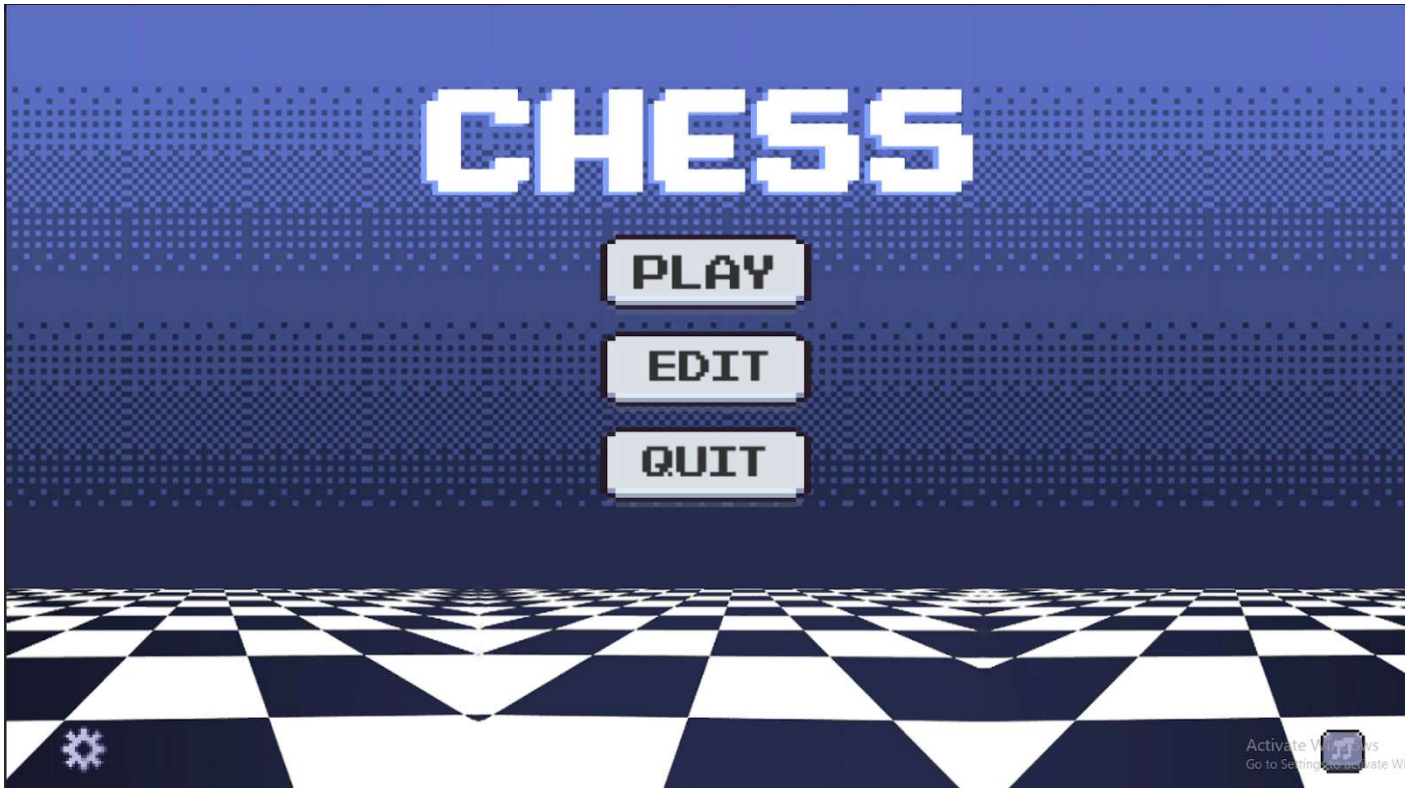
Provide all the Used Design Muscles in Software product

11.1 In the context of designing a chess game, the empathy stage involves gaining a deep understanding of the players and their needs. Game designers engage in activities such as player research, interviews, observation, and immersion to develop empathy and gain insights into players' experiences and challenges. By conducting research, interviewing players, observing matches, and immersing themselves in the game, designers can empathize with the players, understand their preferences, strategies, and pain points, and use this knowledge to create a chess game that meets the players' needs and provides an enjoyable and satisfying experience.

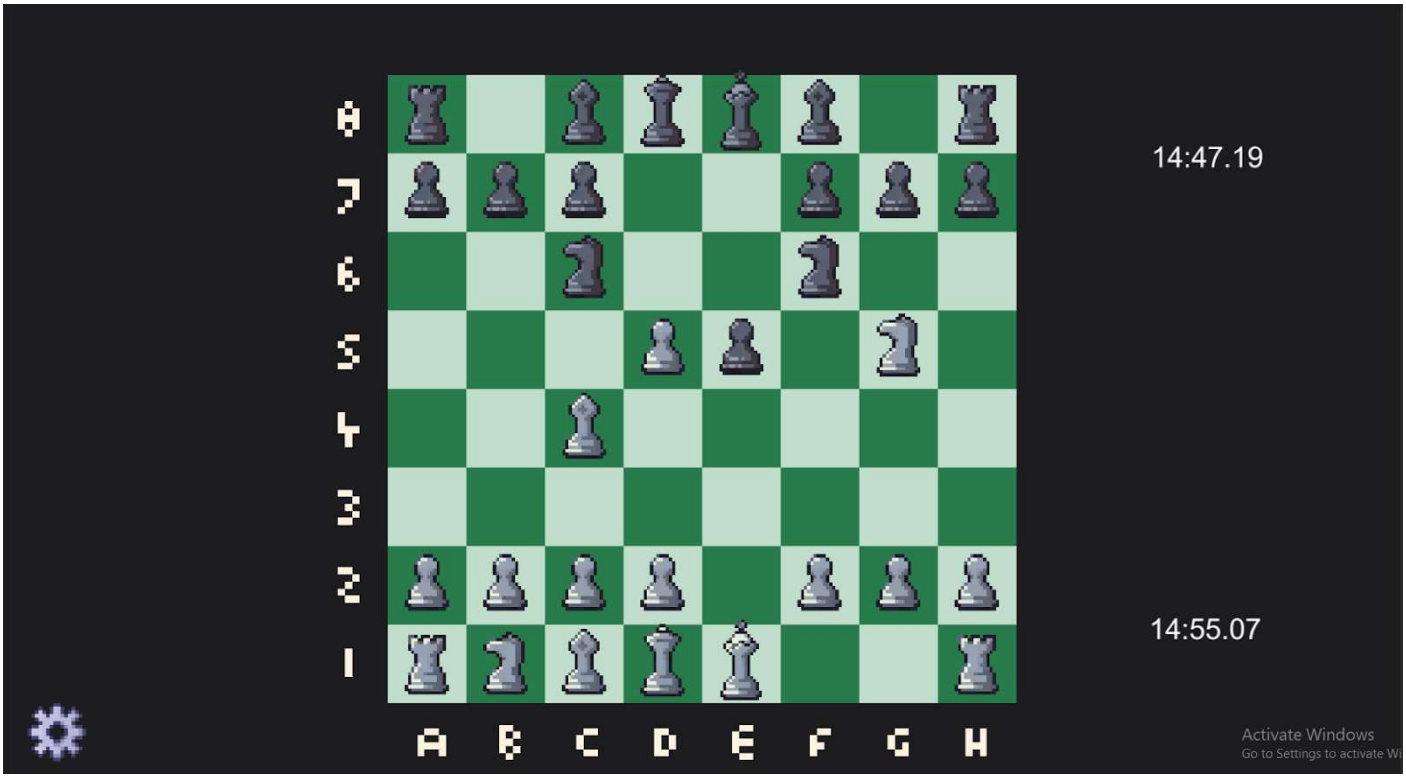
11.2 Define:In the define stage for a chess game, designers analyze the gathered information to identify the core challenges and opportunities. By understanding the pain points, motivations, and goals of chess players, designers form a clear problem statement or design challenge that guides subsequent stages. This ensures that the design solutions address specific player needs, improve gameplay experience, and cater to different skill levels. The define stage sets the direction for creating an engaging and enjoyable chess game.

11.3 In the ideate stage for a chess game, designers engage in brainstorming to generate a variety of ideas and solutions for the defined challenges. They foster creativity, open-mindedness, and divergent thinking to explore different possibilities. Techniques such as mind mapping, sketching, and rapid prototyping are utilized to facilitate idea generation and exploration of innovative features, gameplay mechanics, and user interface enhancements for the chess game.

Test: In the testing stage for a chess game, designers create functional prototypes and organize playtesting sessions with chess enthusiasts or players of different skill levels. They observe the players' interactions, taking note of how they move the pieces, make decisions, and navigate the user interface. Designers actively collect feedback through surveys, interviews, or usability testing, asking specific questions about game mechanics, clarity of rules, intuitiveness of controls, and overall satisfaction. They pay attention to specific pain points, such as difficulty understanding certain rules or challenges in making strategic moves. This feedback is carefully analyzed and used to make iterative improvements, such as refining the layout of the game board, enhancing visual cues for valid moves, simplifying rule explanations, or adjusting difficulty levels. By incorporating this concrete feedback, designers ensure that the chess game provides a smooth and engaging experience for players, enabling them to fully enjoy the strategic challenges of the game.


CHESS
PLAY
EDIT
QUIT


PICK A TIME CONTROL
15 MINUTES
10 MINUTES
5|2
5 MINUTES
3/1
3 MINUTES
Activate Windows
Go to Settings to activate W


8
7
6
5
4
3
2
1
A B C D E F G H
14:47.19
14:55.07
Activate Windows
Go to Settings to activate Wi


PAUSED
RESUME
MENU
QUIT
BLACK TIME
14:55.83
WHITE TIME
14:58.42
Activate Windows
Go to Settings to activate Wi

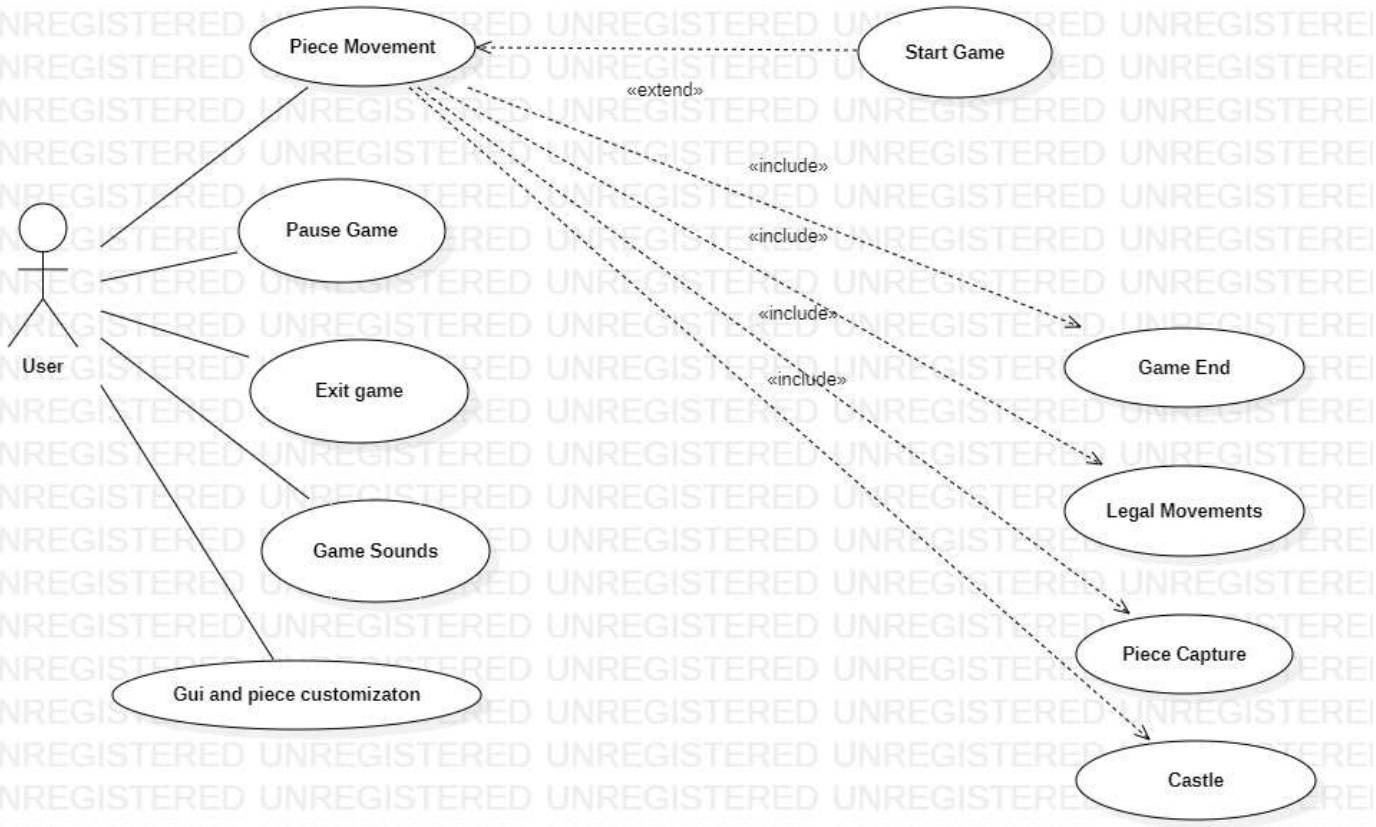

PAUSED./
WARNING./
BLACK
TE TIME
15:0
00.00
YOU ARE ABOUT TO LEAVE TO THE MAIN MENU. ALL PROGRESS WILL BE LOST./
CONTINUE
BACK
Activate Windows
Go to Settings to activate W


WHITE WINS./
PLAY AGAIN
MAIN MENU
QUIT
Activate Windows
Go to Settings to activate W

12. Software Design

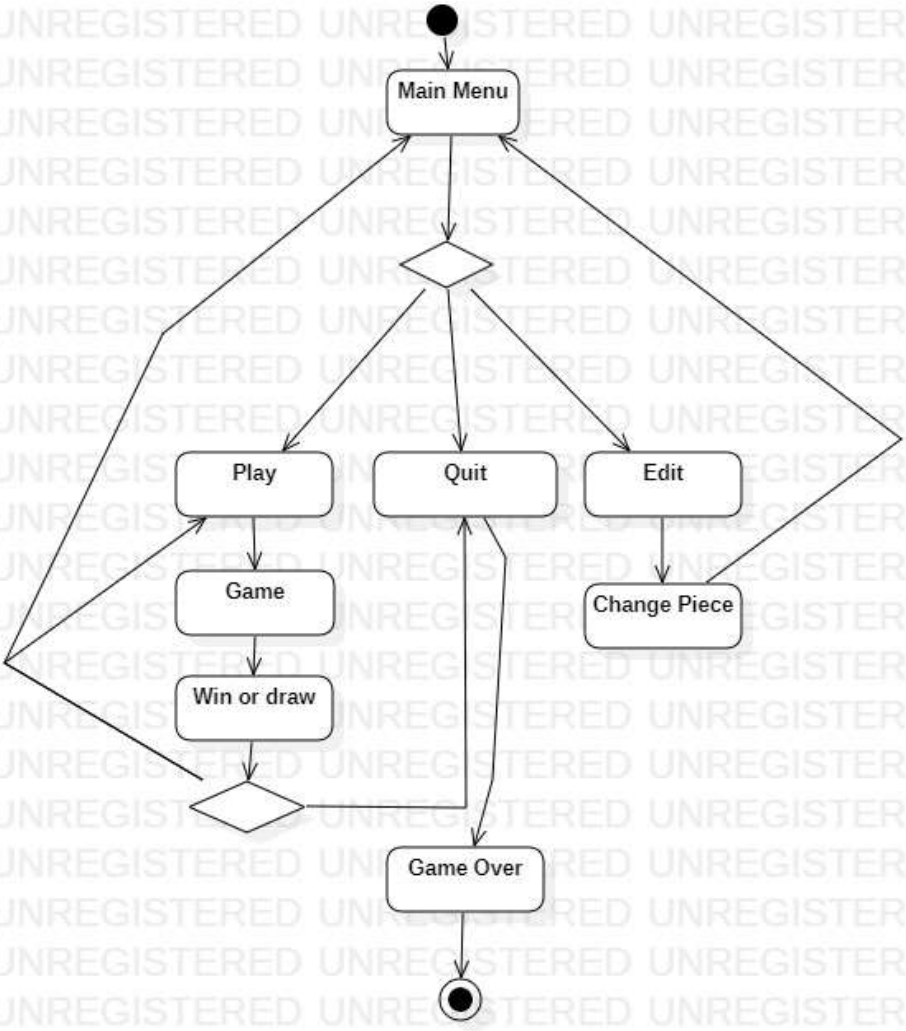
12.1 Use Case

- Has a functional and easy to navigate interface
- The chess game works as intended
- Allows for the customization of the chess pieces
- Has time control
- Has audio for all the moves in the game

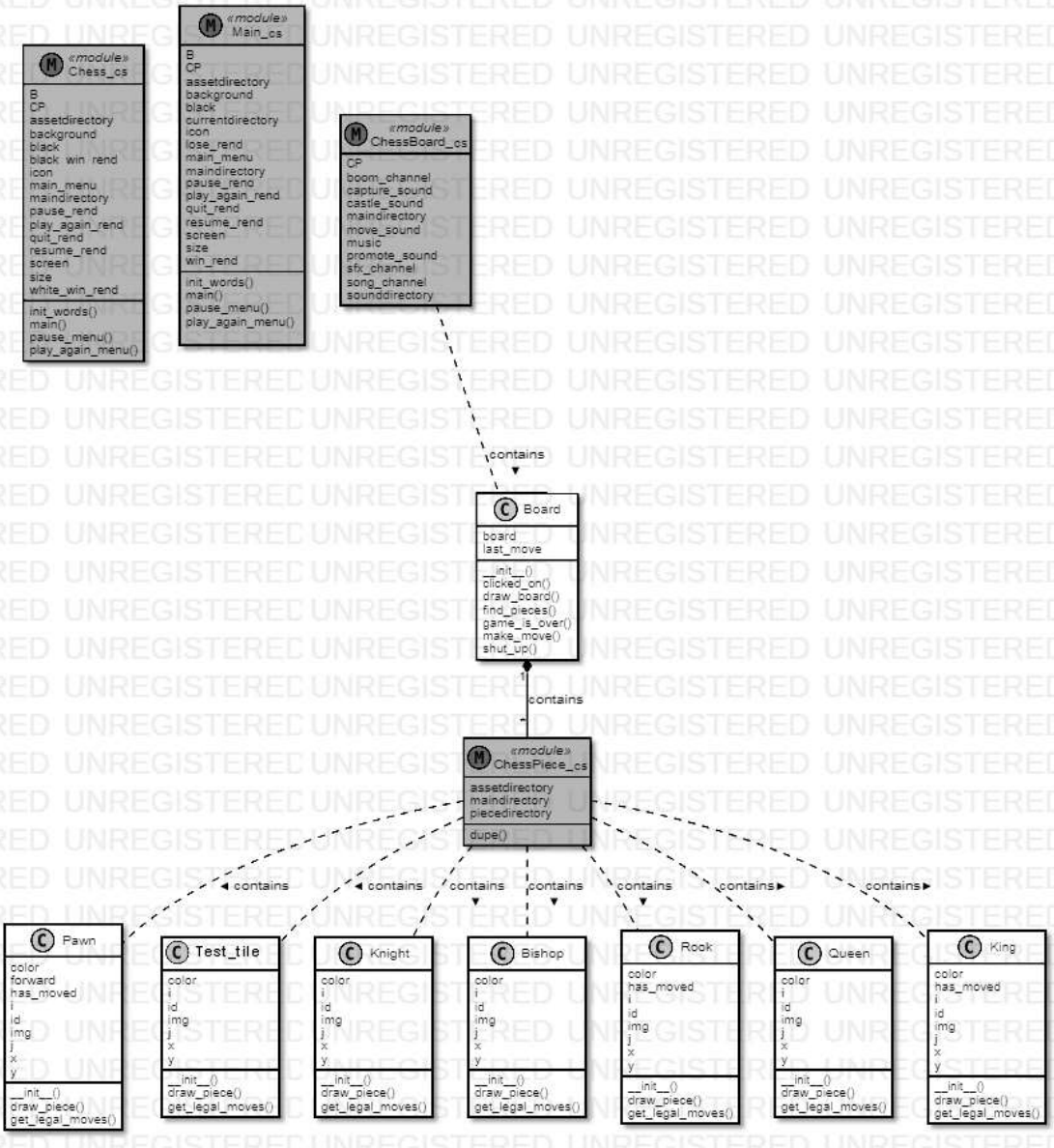


12.2 State Diagram

State diagram shows the behavior of classes in response to external stimuli. Specifically a state diagram describes the behavior of a single object in response to a series of events in a system.



12.3 Class Diagram



APPENDIX

The appendixes are not always considered part of the actual Requirements Specification and are not always necessary. They may include

- Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;
- Supporting or background information that can help the readers of the Requirements Specification;
- A description of the problems to be solved by the system;
- Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

When appendixes are included, the Requirements Specification should explicitly state whether or not the appendixes are to be considered part of the requirements.

Appendix A. Definitions, Acronyms, and Abbreviations

CPU - CPU stands for Central Processing Unit. It is the primary component of a computer that performs most of the processing inside the computer system. The CPU is often referred to as the "brain" of the computer, as it executes instructions, performs calculations, and coordinates the activities of all the hardware and software components. It carries out tasks such as data manipulation, logical operations, and control of the computer's operations. The CPU is responsible for executing program instructions, managing data storage, and handling input/output operations. It consists of various components, including arithmetic logic units (ALUs), control units, and registers, working together to process and execute instructions.

RAM- RAM stands for Random Access Memory. It is a type of computer memory that is used for temporary storage and quick access to data that the computer is actively using. Unlike permanent storage devices such as hard drives or solid-state drives, RAM is volatile memory, which means that its contents are lost when the computer is powered off or restarted.

ROM - ROM stands for Read-Only Memory. It is a type of computer memory that stores data and instructions that are permanently written during manufacturing and cannot be easily modified or erased by normal computer operations.

Appendix C. Organizing the Requirements

1. Game Modes:
    - Single-Player: Allow players to play chess against themselves, providing an opportunity for practice, experimentation, and self-improvement.
    - Local Multiplayer: Enable players to play against friends or family on the same device, supporting pass-and-play or hot-seat gameplay.

2. Game Editor:

- Develop a user-friendly game editor that allows players to set up custom chess positions or scenarios.
- Provide options to place specific pieces on the board, set initial game states, and customize rules if desired.
- Allow players to save and load custom setups for later use.

3. User Interface:

- Design a clean and intuitive user interface that displays the chessboard, pieces, and relevant game information.
- Ensure clear visibility of the board and pieces, allowing players to easily identify positions and moves.
- Include indicators for check and checkmate to notify players of important game states.
- Provide a minimalist design option for players who prefer a distraction-free experience.

4. Sound and Visual Effects:

- Incorporate optional sound effects to enhance the gaming experience, such as piece movements, captures, and checkmate alerts.
- Offer visual effects, such as animations or transitions, to provide feedback on moves and actions, adding polish to the overall presentation.

5. Customization Options:

- Chessboard Themes: Include different visual themes for the chessboard, such as various colors, textures, or materials.
- Piece Styles: Provide multiple sets of chess piece designs to suit different preferences and aesthetics.
- Board Orientation: Allow players to choose their preferred board orientation, whether it's white at the bottom or a flipped perspective.

6. Settings and Options:

- Display Options: Provide customization options for font size, board size, and other visual preferences.
- Control Preferences: Allow players to choose their preferred input method, such as mouse, keyboard, or touch controls, based on their device and personal preference.

7. Offline Availability:

- Ensure the game is fully functional offline, allowing players to enjoy chess even without an internet connection.
- Provide offline access to saved games and custom setups.

8. Additional Features:

- Legal Move Highlighting: Highlight valid moves for selected pieces to assist players in making legal moves and avoid illegal moves.
- Promotion Selection: Enable players to select the desired promotion piece when a pawn reaches the opposite end of the board.