RhombiChess

Completed Functional Requirements

Based on our Software Requirements Specification, we were able to satisfy the following functional requirements outlined by our supervisor and team.

P0 Functional Requirements: Completed with a High Degree of Success (100%)

- Users can register for an account
- Users can use their existing credentials to log-in
- Users can start a new game
- Allow users to click on a piece and make a legal move
- The player must confirm or retract move before the board state changes
- Display a message if a move is not valid – we instead simply prevent invalid moves
- Display message if a check/checkmate occurs
- Prevent further gameplay if a checkmate occurs
- Allow users to choose their opponent (themselves or online opponent)
- Display board (190 rhombuses) with the pieces on each side (16 unique, 47 total on each side with their designated design)
- Allow users to set timer duration for real-time play
- Countdown individual timer based on active player
- Declare loss for player if timer ends
- Allow for untimed games (default option)
- Implement game rules and logic for validating moves and detecting check and checkmate
- Manage user authentication and registration
- Store game state
- Handle online matchmaking
- Implement API for the frontend to communicate with the backend
- Save timer state for each turn and player
- All the required backend endpoints (and a few more)

P1 Functional Requirements: Completed with a High Degree of Success (100%)

- Provide an option for choosing their side – black or white. This is random by default.
- Highlight selected pieces and rhombuses that are legal moves for that piece.
- Display all captured pieces for each player off the board.
- Allow users to send messages to each other during the game.

P2 Functional Requirements: Completed with a High Degree of Success (100%)

- Create a responsive UI that adapts to screen size
- Flip board to active player’s side

Unfinished Functional Requirements

Storing game to DB (P3) [0%]

As part of our performance requirements, specifically the capacity requirements we had mentioned that the backend should be capable of storing game histories, player sessions, game states and information on players. We did not have the time to finish this up and we also deemed it unnecessary as the game is intended to be played in a single session.

Re-play or Retrieve Old Games (P3) [0%]

This was a P3 functional requirement where we wanted to allow users to save, recall and replay a game which ultimately was not a high priority for our project and we were unable to finish it within the time allotted. We wanted to make sure the actual gameplay was our primary focus and perfected as best as we could, so this requirement was not at the top of our list.

End-to-end Encryption for Messages (P3) [0%]

We had listed as one of our privacy requirements that the game chat must be end-to-end encrypted but this was once again low on our list of priorities as we wanted to make sure the actual gameplay itself was completed within the expectations of our supervisor and the encryption of the game chat is not necessarily a focus for chess players.

Sound and Translation (P3) [0%]

These were P3 front-end functional requirements where we wanted to play sound based on events that occur during the game (i.e. check, chat ping etc.) and have game text in various languages for supporting different player groups. This was also an additional nice-to-have functionality that we deemed less important than the actual gameplay which is what we spent most of our time on. The sound creation would have required effort on its own as we would have to carefully pick sounds that helped enhance the game and not irritate or annoy the player. We might have also had to have the option available for users to mute the sound. Translating the text accurately would have required extensive research as well.

Save User Preferences (P2) [0%]

This was a P2 back-end functional requirement that did not end up being implemented in the game because we did not end up collecting preference data as it did not fit the scope of the game. Therefore, there was no need to store this data.

AI Opponent and Move Suggestions / Statistics (P3) [0%]

These were P3 back-end functional requirements where we wanted to implement an AI opponent to allow single-player mode and add features such as move suggestions and statistics. These were unrealistic and ambitious within our timelines and were low priority considering we wanted to get the P0 requirements as close to the requirements laid out by our supervisor as possible. This would require a deep understanding of chess strategies and as this is a new variant it would require hours of gameplay to create strategies. It could also potentially affect performance.

Non-Functional Requirements [100%]

We were also able to satisfy all of the outlined non-functional requirements from our initial SRS. This included appearance, style, performance, maintainability, security [1], and legal requirements. Please refer to our SRS if you'd like a complete breakdown of these requirements – they are not included here as they were not high priority and were flexible.

Implementing these non-functional requirements allowed us to write better code, create a clean and beautiful user interface, and ultimately complete the project with a desirable product.

[1] Except "End-to-end Encryption for Messages" as mentioned in the previous section.