Checkmate: Requirements Specification

By: Esai Jimenez, Autumn Hale, & Trentin Barnhart

Table of Contents

Project Name **3**

Team Member Names **3**

Abstract **3**

Tools & Technologies **4**

Requirements List **5**

1. Main Menu UI 5
2. Game Mode UI 5
3. Chess UI 5
4. Help/Tutorial UI 7
5. Leaderboard UI 7
6. Settings UI 8
7. Custom Puzzles UI 8
8. Login UI 9

Updated Timeline **10**

Project Name

With the growing popularity in chess, we decided to create **Checkmate**, a way for users to practice and develop their chess skills in an interactive way through chess puzzles.

Team Member Names

Esai Jimenez, Autumn Hale, Trentin Barnhart

Abstract

The software project is a website that provides users with a platform to attempt various chess puzzles in a gamified way. Users are presented with a virtual, standard 8x8 chessboard in which they are prompted to solve a particular chess position starting from relatively easy to progressively becoming more difficult as each puzzle is solved. For each puzzle completed correctly, one point will be added to their counter. The user must find the best move on the prompted board to move on to the next puzzle. After three incorrect attempts are made, the user's progress resets.

The user will be initially told which color's move it is. If the puzzle says "Black to move" or "White to move" then that is the color that the user must move with. In addition to this, in order to initialize the position, the first move is automatically executed to show the user what was the last move played by the opposite color. The puzzles will require the user to "solve" the chess game by getting their color to win by checkmate in the allowed number of moves. Puzzles will be given randomly and progressively increase in difficulty. The user's scores will be saved.

Tools & Technologies

Location in System	Technical Requirement	External Dependency	Purpose
Backend	Python	Pandas Library	Pandas is a Python Data Analysis library that allows us to extract key information from a CSV file.
Backend	Google Firebase - Realtime Database	N/A	Firebase Realtime Database is a cloud-hosted NoSQL database offered by Google's Firebase platform.
Backend	Google Firebase - Storage	N/A	Firebase Storage is a cloud-based object storage solution that is designed to handle large amounts of data and is optimized for fast and reliable access.
Frontend	JavaScript	React	React is a JavaScript library for building user interfaces.
Frontend	Google Firebase - Hosting	N/A	Firebase Hosting is a fast, secure, and scalable web hosting service that makes it easy to deploy and host web apps.
Frontend	HTML	N/A	HTML is the standard markup language used to create web pages.
Frontend	CSS	N/A	CSS is a styling language that will be used to style the presentation of the HTML
Frontend	JavaScript	Chessboardjs	Chessboard.js is a JavaScript library for creating chessboards on web pages.

Requirements List

1. Main Menu UI
    1.1. The main menu will display the title and include the following:
        1.1.1. The phrase “Find the checkmate” is displayed to the user
        1.1.2. A play button that will redirect the user to the Game Mode UI (requirement 2)
        1.1.3. A help button that will redirect the user to the Help/Tutorial UI (requirement 4)
        1.1.4. A leaderboard button that will redirect the user to the Leaderboard UI (requirement 5)
        1.1.5. A settings button that will redirect the user to the Settings UI (requirement 6)
        1.1.6. A custom puzzles button that will redirect users to the Custom Puzzles UI (requirement 7)
        1.1.7. A button that will redirect users to the Login UI (requirement 8)

2. Game Mode UI
    2.1. The user is presented with two options, “Classical Mate” and “Bullet Mate”
    2.2. When the user selects Classical Mate, it takes the user to the Chess UI with the functionality of Classical Mate.
    2.3. When the user selects Bullet Mate, it takes the user to the Chess UI with the functionality of Bullet Mate.

3. Chess UI
    3.1. Classical Mate mode
        3.1.1. A message will appear when the board populates telling the user what side they are controlling, and therefore what color moves next.
        3.1.2. Board GUI will display a board state with the following features:
            3.1.2.1. A standard 8x8 2D chess board display.
                3.1.2.1.1. The pieces will be automatically placed in their puzzle positions.
                    3.1.2.1.1.1. The positions will be generated from a database.
            3.1.2.2. The user will be able to move pieces
                3.1.2.2.1. The user selects a piece by clicking on it and then selects where it moves by then clicking the corresponding space on the board.
                    3.1.2.2.1.1. The selected piece’s tile is highlighted when clicked on.

3.1.2.2.2. If the “Legal Move Dots” setting is enabled then legal moves for the selected piece will be displayed as dots on legal squares.

3.1.2.2.3. Clicking a piece that has already been selected will unselect that piece.

3.1.2.2.4. Once a piece captures another, the captured piece will disappear, and the capturing piece will be moved to its new location.

3.1.2.2.5. If an incorrect move is made, then one life will be lost and a new puzzle will be populated.

3.1.2.2.6. If the user checkmates, then their score will increase by one and a new puzzle will be loaded in.

3.1.2.2.7. If the user makes an incorrect move then their lives will be reduced by one and a new puzzle will be loaded in.

3.1.3. The score counter will begin at zero and increment by one each time the user successfully completes a puzzle.

3.1.3.1. Upon a game over the score counter will reset to zero.

3.1.4. The lives counter begins at 3 and will decrement by one each time the user makes an incorrect move.

3.1.4.1. When the lives counter reaches zero and if the user utilized the “View Solution” button at any point during the round, then a game over message will be displayed and the user will be prompted to either play again, or return to the title screen.

3.1.4.2. When the lives counter reaches zero and if the user did not utilize the “View Solution” button, then their run will be documented in the top 10 leaderboard if they outperformed the others on the leaderboard.

3.1.5. The difficulty rating will display the difficulty of the current puzzle.

3.1.5.1. This rating will be taken from the database alongside the other information required for the puzzle.

3.1.6. The user may choose to view the current puzzle’s solution at any point.

3.1.6.1. This will disable leaderboard submissions for that session.

3.1.6.2. The solution will be shown to the user visually on the board GUI. The pieces will move themselves one at a time.

3.1.6.3. Viewing the solution will credit the user zero points.

3.1.6.4. Viewing the solution will give the user a new puzzle after they click a button to move on.

3.2. Bullet Mate mode

3.2.1. All functionality is the same as Classical Mate except there is a timer.

3.2.2. The timer will begin at a preset value when a user starts a session.

3.2.3. The timer will continuously deplete while the user is playing.

3.2.4. Upon completing a puzzle, a fixed value will be added to the timer's total, granting the user more time to continue with.

3.2.5. If the user's timer goes to zero or their lives run out, then the game over message will be displayed and the user will be prompted to either play again, or return to the title screen.

3.2.6. Viewing a solution will pause the timer while the solution plays out.

4. Help/Tutorial UI

4.1. The Help/Tutorial menu will display information deemed significant to Checkmate and its usage.

4.1.1. About

4.1.1.1. A section with general information about the website.

4.1.1.1.1. Description regarding Checkmate, its intended use, creators, and date made.

4.1.2. Piece Movement and Capture

4.1.2.1. Each piece's movement and capture will be explained in a text section.

4.1.2.2. Piece movement and capture sections will be accompanied by a graphic showing what the piece looks like.

4.1.3. How to Check and Checkmate

4.1.3.1. Text section(s) explaining what a check and a checkmate are and how they are achieved.

4.1.3.2. The Check and Checkmate section(s) will be accompanied by examples in graphic form.

4.1.4. Other Rules

4.1.4.1. Castling and en passant will be explained in a text section.

4.1.4.2. The castling and en passant text sections will also be accompanied by examples in graphic form.

4.2. The menu will contain a button to return to the title page.

5. Leaderboard UI

5.1. The Leaderboard page will display the Bullet Mate mode and Classical Mate mode top 10 scores simultaneously.

5.1.1. Bullet Mate mode

5.1.1.1. Ranking will include display name, time, and score.

5.1.2. Classical Mate mode

5.1.2.1. Ranking will include display name and score.

6. Settings UI

6.1. The user will be able to view their profile

- 6.1.1. The user will be able to view their previous scores and the date the score was received.

6.2. Legal Move Dots Indicator Toggle

- 6.2.1. If toggled on, pieces will have their legal moves displayed by dots when each piece is selected.
- 6.2.2. If toggled off, pieces will not have their legal moves displayed by dots when each piece is selected.

7. Custom Puzzles UI

7.1. The custom puzzles page will have two buttons. One button will lead to a Custom Puzzle Creator page, and the other will lead to a Custom Puzzles page.

7.2. Custom Puzzle Creator will open with a 8x8 blank chessboard and a series of pieces that the user will be able to populate the board with.

- 7.2.1. If the user is not logged in, they will be redirected to the login page instead.
- 7.2.2. Populating the board will be done by clicking a piece and then clicking the spot on the board to place the piece.
  - 7.2.2.1. Placed pieces can be moved by clicking them and then on another space on the board.
  - 7.2.2.2. This section will not check for legal moves since no actual game is being played.
  - 7.2.2.3. Clicking off of the board while a piece is selected will remove the piece from the board.
  - 7.2.2.4. There can be a maximum of 8 pawns, 1 Queen, 1 King, and 2 Knights, Bishops, and Rooks for each side.
  - 7.2.2.5. A button to confirm the board state will proceed to the solution portion of puzzle creation.
- 7.2.3. The user will provide a solution after confirming a board state
  - 7.2.3.1. The user will input moves for white and black as if a real game is being played.
    - 7.2.3.1.1. This section will check for legal moves and not allow illegal moves.
    - 7.2.3.1.2. A counter will keep track of how many moves white makes. This will be saved with the puzzle information.
    - 7.2.3.1.3. Once checkmate is reached, the user will be prompted to enter the puzzle name and confirm the puzzle.
    - 7.2.3.1.4. Once confirmed the puzzle will be uploaded and saved.

7.3. Custom Puzzles will display a sorted list of uploaded puzzles.

7.3.1. The puzzles will be sorted by rankings.

7.3.2. Clicking on one of the puzzles will load that puzzle and allow the user to attempt to solve it.

7.3.3. A rating system will allow users to rate puzzles on a scale from 1-5.

7.3.3.1. Ratings will only work if a user is logged in.

7.3.3.2. Each user will only be able to rate a puzzle once.

7.3.3.2.1. Attempting to rate a puzzle that has already been rated will override the old rating.

7.3.4. Upon succeeding at a puzzle, users will be returned to the puzzle play menu.

7.3.5. Failing at a puzzle will simply reset it to its base state.

7.3.6. A return button will allow the user to exit a puzzle at any time.

8. Login UI

8.1. The login page will allow the users to either login or register a new account.

8.1.1. Login

8.1.1.1. The user will be able to login through Google.

8.1.2. New Account

8.1.2.1. The user will be able to sign up through Google and create their own username.

8.2. After logging in or signing up the user will be returned to the Main Menu UI.

8.3. If a user is already logged in, the login button will turn into a log out button.

Updated Timeline

Week of:	Esai	Trentin	Autumn
January 22nd - January 28th	Learn about Google Firebase and figure out how to use the Lichess puzzle file.	Become familiar with technology being used and begin next step early.	Get an understanding for technologies that will be used and how they will work together.
January 29nd - February 4th	Implement what I learned about Firebase to host our website and utilize the information in the Lichess puzzle file.	Work on requirements list and planning.	Create template mockup of the website and collaborate to ensure that it meets MVP.
February 5th - February 11th	Help Trentin and/or Autumn with any issues or facilitate their coding.	Work towards a functional visual display for the chessboard.	Code the bare front end of the website according to mockup.
February 12th - February 18th	Collaborate with Autumn to finish the 2D, 8x8 chessboard interface and have it be hosted on Firebase.	Work towards a functional visual display for the chessboard.	Chess board interface put on the website. Collaborate on hosting the website.
February 19th - February 25th	Collaborate with Trentin to finish the functionality of the chess pieces.	Work on control functionality for the chessboard.	Chess board piece interpretation from data to basic GUI onto the board. Graphical fixes and other frontend work.
February 26th - March 4th	Collaborate with Trentin on generating a random puzzle position.	Ensure chessboard UI details are in place. (timer if enabled, legal move indicators, etc.)	Chess board piece interpretation from data to basic GUI onto the board. Graphical fixes and other frontend work.

March 5th - March 11th	Spring Break March 4-12.	Spring Break March 4-12.	Spring Break March 4-12.
March 12th - March 18th	Collaborate with the others to connect our backend and frontend.	Write the help/tutorial section.	Collaborate with Esai and Trentin to connect the backend and frontend. Ensure the mechanics of the pieces work with the interface.
March 19th - March 25th	Work with Trentin to get the functionality for the tutorial.	Make sure leaderboard submissions are only sent under the right criteria.	Ensure the mechanics of the pieces work with the interface.
March 26th - April 1st	Try to add a login feature and leaderboard if time allows.	Cleaning up bugs, doing some basic polish, and filling any gaps we left so that we have an MVP ready.	Ensure the mechanics of the pieces work with the interface. Graphical fixes and other frontend work.
April 2nd - April 8th	Collaborate with Autumn to ensure the mechanics of the pieces work with the interface.	Cleaning up bugs, doing some basic polish, and filling any gaps we left so that we have an MVP ready.	Graphical fixes and other frontend work. Clean up and bug fixes.
April 9th - April 15th	Work with both group members to ensure that we have a minimum viable product for the upcoming presentation.	Work with the group to ensure our minimum viable product for the upcoming presentation.	Work with both group members to ensure that we have a minimum viable product for the upcoming presentation. Clean up and bug fixes.