Checkmate: Project Design

By: Esai Jimenez, Autumn Hale, & Trentin Barnhart

Table of Contents

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 completing a puzzle, users will be returned to the puzzle selection 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 logout button

Design Description

Our web application will be a chess puzzle program that is hosted using Google Firebase Hosting. Appendix 1 highlights our block diagram. A block diagram is a visual representation of a system or process using simple shapes to represent its different components, and arrows to indicate the relationships between those components. In our block diagram, it illustrates that the server Firebase provides will communicate directly with our Checkmate program inside of Firebase Storage. The user interface and underlying functionality lives inside of the program. This program is also responsible for pulling information from the Google Firebase Realtime Database. The database plays a vital role within our system to store around 100,000 different chess position entries. In every entry, we are storing the “FEN”, “Moves”, “PuzzleId”, “Rating”, and the “Themes. An FEN is the abbreviation for "Forsyth-Edwards Notation" and it is the standard notation for describing a specific chess position on a board. For example, the FEN for the starting position on a chess board is “rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1”. A “Moves” entry is the set of moves needed to complete the puzzle, the first move listed in a “Moves” entry is the move performed automatically once a new puzzle is generated to initialize the puzzle. Every set of moves is stored in UCI (Universal Chess Interface) format. For example, “e5-f6 h7-g8 f8-g8 e1-e8” is an example of a set of moves in UCI format. The first move performed is “e5-f6”, meaning that the piece that is sitting on e5 must move to f6. A standard 8x8 chess board is ranked from “a” to “h” horizontally, and then “1” to “8” vertically. Next, we are storing the “PuzzleId”. Every puzzle is given a number depending on what position they are in the database. Therefore, the 37,000th puzzle in the database will be given a “PuzzleId” of 37,000. Next, we are storing the rating for each puzzle. The rating is the way we will gauge the difficulty level for each puzzle entry. The range starts at 545 and ends at

2841. For scale, with a general understanding of chess, anything between 545 to 1200 is fairly easy, then 1200 to 1700, may be mildly challenging, then anything above 1700 would be challenging. Finally, the last piece of information we are storing is the theme of the puzzle. Each puzzle is either a checkmate in one, two, three, four, or five. Another important factor is user authentication. Google Firebase provides authentication services that will allow users to set up their own chess puzzle account to track their progress and create their own puzzles. The last service we are utilizing from Google Firebase is the storage. There are large files that we need to store in our project for functionality and documentation purposes. The first file that we are storing is our "checkmates.json" file which was used to set up the initial database. The Realtime Database has the ability to import a json file and create a database based on the information inside of the json file. The second file that is stored in the Firebase Storage is the "lichess_db_puzzle.csv". This file was provided by Lichess. Lichess is a free and open-source Internet chess server that allows the free use of this CSV file that contains over 3,000,000 different chess puzzles. However, this many puzzles is not necessary. Therefore, we used the Pandas Library inside of Python to filter out all the puzzles that were not needed. At the end of this process, we ended up with 99,392 unique chess positions and stored them in our "checkmates.json" file that was used to create the database.

Additionally, in our Appendix 2, it covers our component diagram. A component diagram is a type of UML diagram that shows the structural organization of components and their relationships to each other. As shown in Appendix 2, we will start with the MainMenuUI component. This component will create the individual links for each component and also have buttons that correspond to the links. When the user clicks on the "Help" button, it will link the user to the "/help-tutorial". These types of links are created for each component and will be used

by the Checkmate class that acts as a router. In this context, the router's role is to display the component based on what button the user clicked. If the user is in the "/help-tutorial" path, then the Checkmate router displays the "HelpTutorialMenuUI". This functionality is the same throughout the rest of the components. Each component has varying dependencies on other components. An example of this would be the ChessUI implements the SettingsUI that the user set up. In addition to this, the CustomPuzzlesUI needs to inherit some of the functionality of the ChessUI. Once the functionality for each component is completed, the index acts as the interface that renders the Checkmate router and produces the application.

Appendix 3 consists of a user interface (UI) storyboard, which is a visual representation of the user experience for a product. This product is usually a software application or website. It is used to communicate the design ideas, user flows, and interactions to stakeholders and the development team. This storyboard is not a representation of the final product, but rather a mode of communication to ensure unanimous understanding pre-implementation. It is based on requirements agreed upon. All figures are read from top left and follow the arrows. Some arrows split from A and B paths. The titles of pages are above their respective pictures. Notes are written as needed.

In Appendix 3 figure 1, the MainMenuUI is the first page in the Play diagram flow. The MainMenuUI includes the title, short description, and 6 menu buttons corresponding to page redirects. When a user clicks Play, it leads to the Game Mode UI. This page presents the user with the options of Classical Mate or Bullet Mate. When the user clicks the Classical Mate's button option, they are redirected to the Classical Mate UI. After playing a game, the user will either go to path A or B depending on the game's outcome, labeled accordingly. Likewise, when

the Bullet Mate button option is chosen on the Game Mode UI screen, the path is nearly identical (see Appendix 3 figure 2).

Appendix 3 figure 3 depicts a redirect to the Help/Tutorial UI when the Help button is selected from the Main Menu UI. The Help/Tutorial UI content consists of information noted in the requirements (see page 5). Appendix 3 figure 4 depicts a redirect to the Leaderboard UI when the Leaderboard button is selected, where the top Classical Mate and Bullet Mate scores are shown. Appendix 3 figure 5 depicts a redirect to the Settings UI when the Settings button is selected, where the user's saved scores are displayed and any options are available.

Appendix 3 figure 6 depicts the create custom puzzles flow. This shows a redirect to the Custom Puzzles UI when the Custom Puzzles button is selected from the Main Menu UI. From there, when a user selects the corresponding button under Create, they are led to the Creator Page UI. After creating their puzzle, users will see a screen that is similar to the Board State Confirmed UI. From there, they will be redirected to the Solution Entry UI. After completing a custom solution, users will see the Upload UI page.

Similarly, Appendix 3 figure 7 depicts the play custom puzzles flow. This shows a redirect to the Custom Puzzles UI when the Custom Puzzles button is selected from the Main Menu UI. From there, when a user selects the corresponding button under Play, they are led to the Uploads UI page. Here, users see a list of top user created puzzles. From here, users will either go through path A or B, which depends on if they are logged in or not. The figure shows what each corresponding page and flow looks like.

The final button on the Main Menu UI is the Login. When selected, users will be redirected to the Login UI, where they can either login or create an account. In the figure, path A shows a

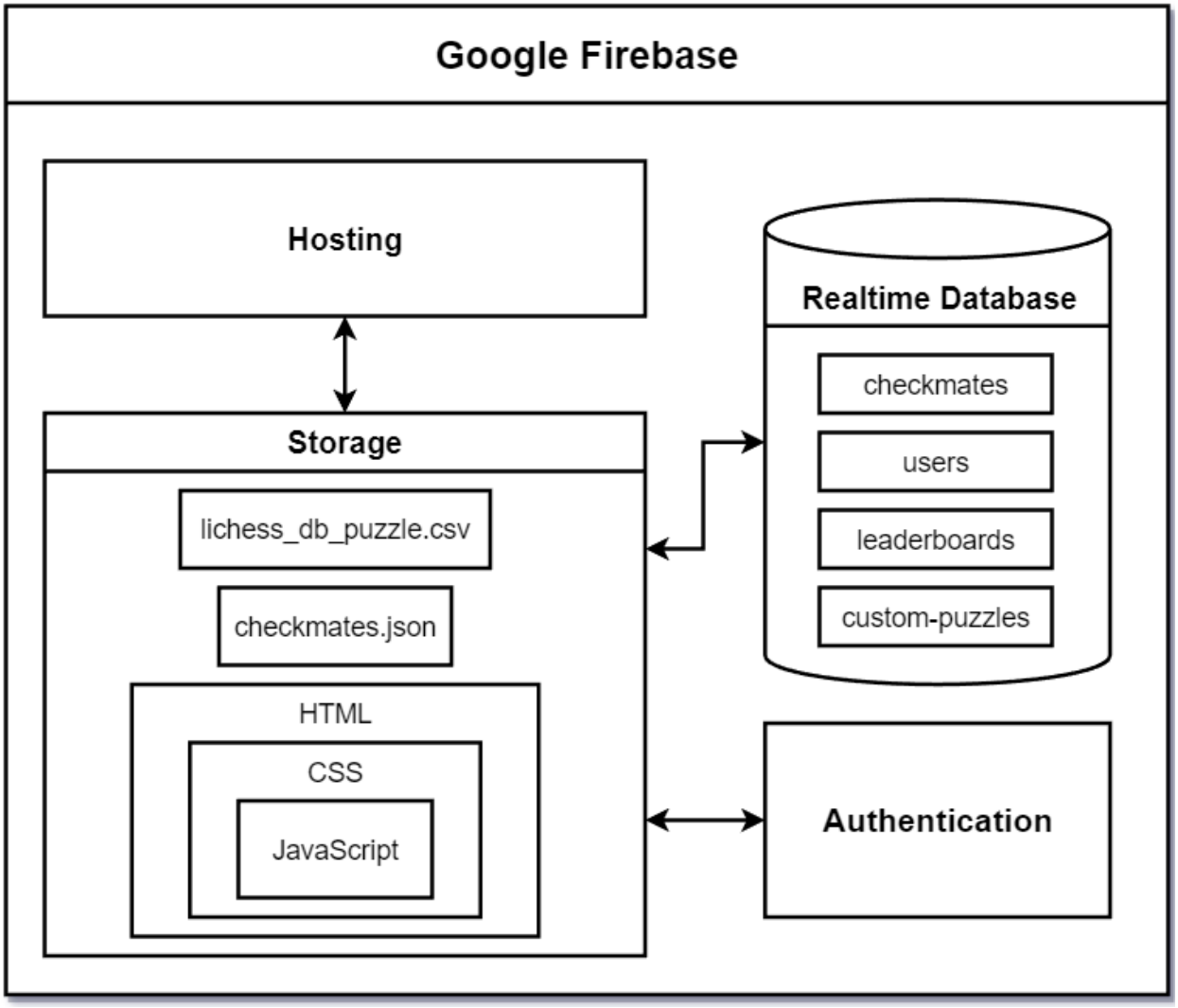
login page, which may lead to the Create Account UI page. Path B shows a login failure. We will be using the FirebaseUI, and therefore the UI is subject to change in any sense.

The Message documentation is a record of the communication between different components of the software. While logins will be handled almost entirely by Firebase, requiring relatively little work beyond what is supplied, there are still many messages that will still need to be handled by the system itself. Appendix 4 will be referenced here to demonstrate a comprehensive breakdown of the messages that will need to be implemented. For the board data a message will need to be sent from the database to the client containing the board state information(see appendix 4.1). The leaderboards will require three different types of messages. For the untimed leaderboard a message containing the current top ten leaderboard holders and their associated scores will be sent to the client. The bullet mate leaderboard will have a near identical leaderboard message but the message will additionally include the time taken. Should the user get a score warranting being sent to the leaderboard the client side will send a message to the database containing the score, username, and time if applicable (see appendices 4.2 & 4.3). The custom puzzles will have three associated messages of their own. There will be a message sending the puzzles for submission (appendix 4.4), a message getting puzzle information to load (appendix 4.1), and a message request that gets a list of custom puzzles to allow the user to select one from a list (appendix 4.5).

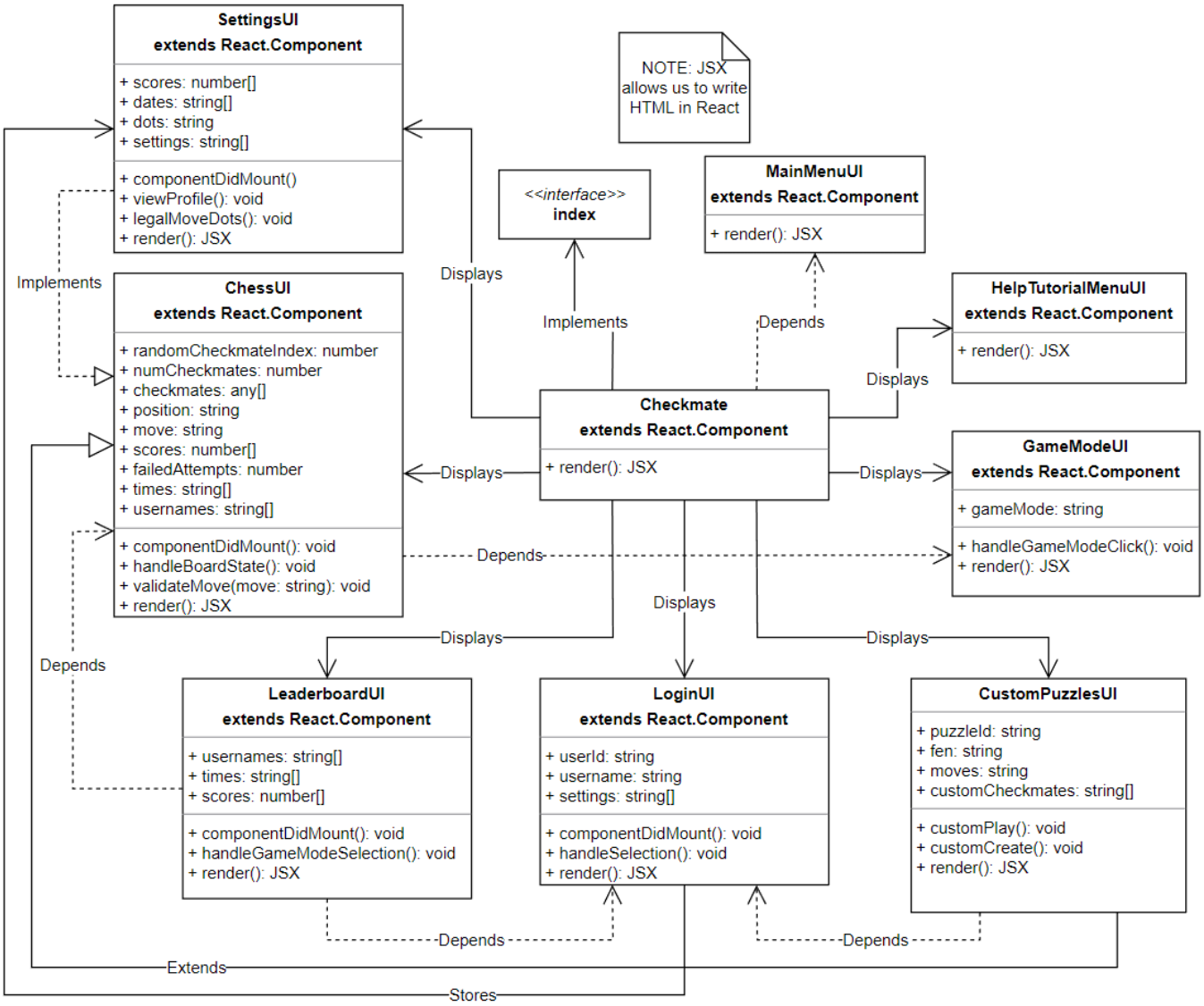
Lastly, our Appendix 5 contains our ER diagram. The ER diagram describes how data is stored in our system. Every database entry will be stored in JSON (JavaScript Object Notation) format. In our database, there will be the “checkmates”, “users”, “leaderboards”, and “custom-puzzles” pathways. The first pathway, “checkmates”, will contain the “FEN”, “Moves”, “PuzzleId”, “Rating”, and “Themes”. The second pathway, “users” will contain the “userId”, “username”,

and “userSettings”. The userId will simply be a number starting from 1, and increase by 1 as each user signs up for an account. The user will be able to create their own username that they will be identified by. Lastly, the user’s settings will be stored as well. The third pathway is the leaderboards. Users will have the opportunity to take a spot on the leaderboards. There will be two leaderboards to fight for, bullet and classical. The bullet leaderboard will store the “username”, “score”, and “time”. The classical leaderboard will store only the “username” and “score”. If the user performs well, they have a chance at having a seat on the leaderboards. The last and final pathway will be custom-puzzles. The custom-puzzles path will store the “puzzleId”, “username”, “fen”, and “moves”. Users will be able to create their own puzzles and other users will be able to play their puzzles.

Appendix 1 - Block Diagram



Appendix 2 - Component Diagram



Appendix 3 - User Interface Storyboard

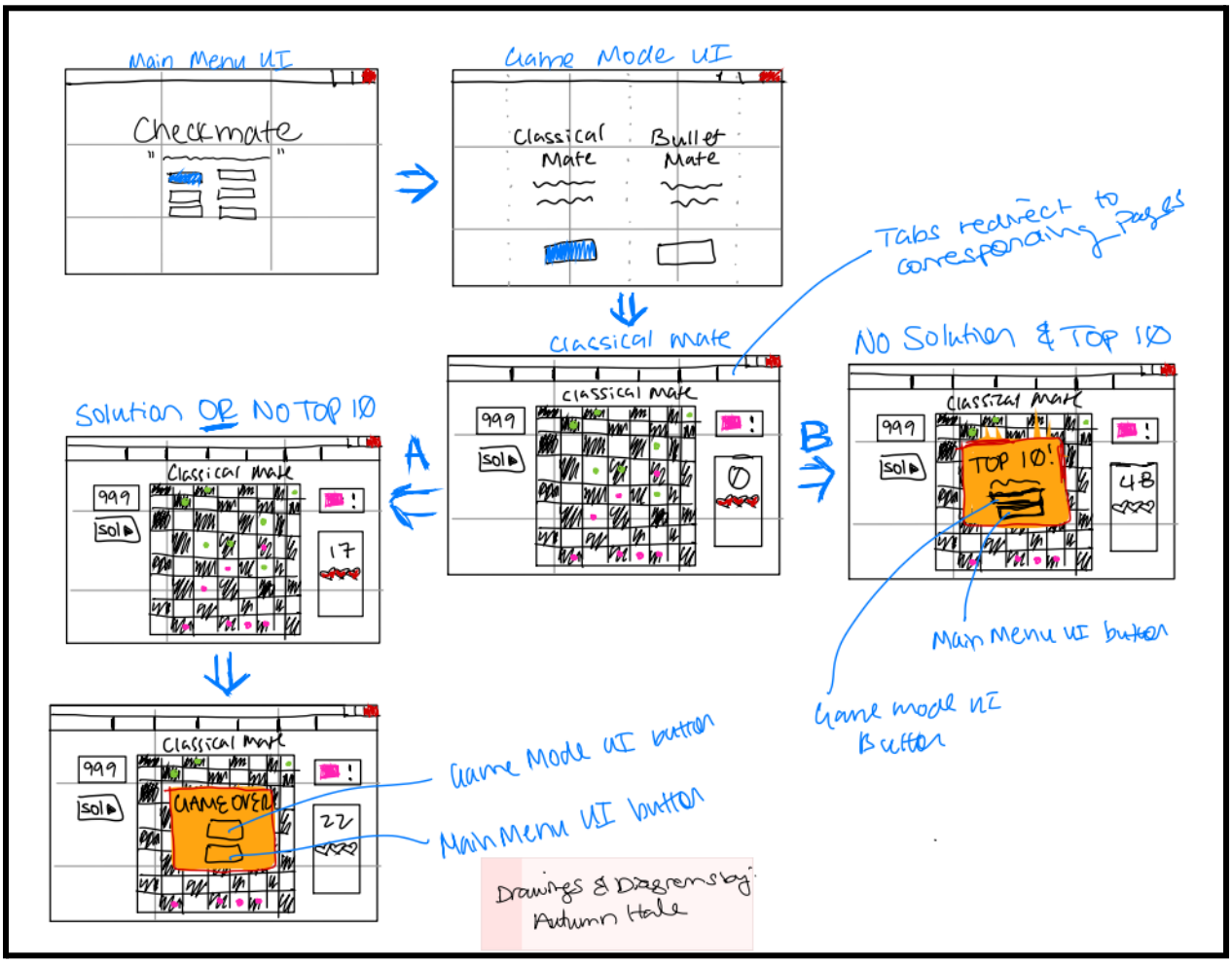


Fig. 1

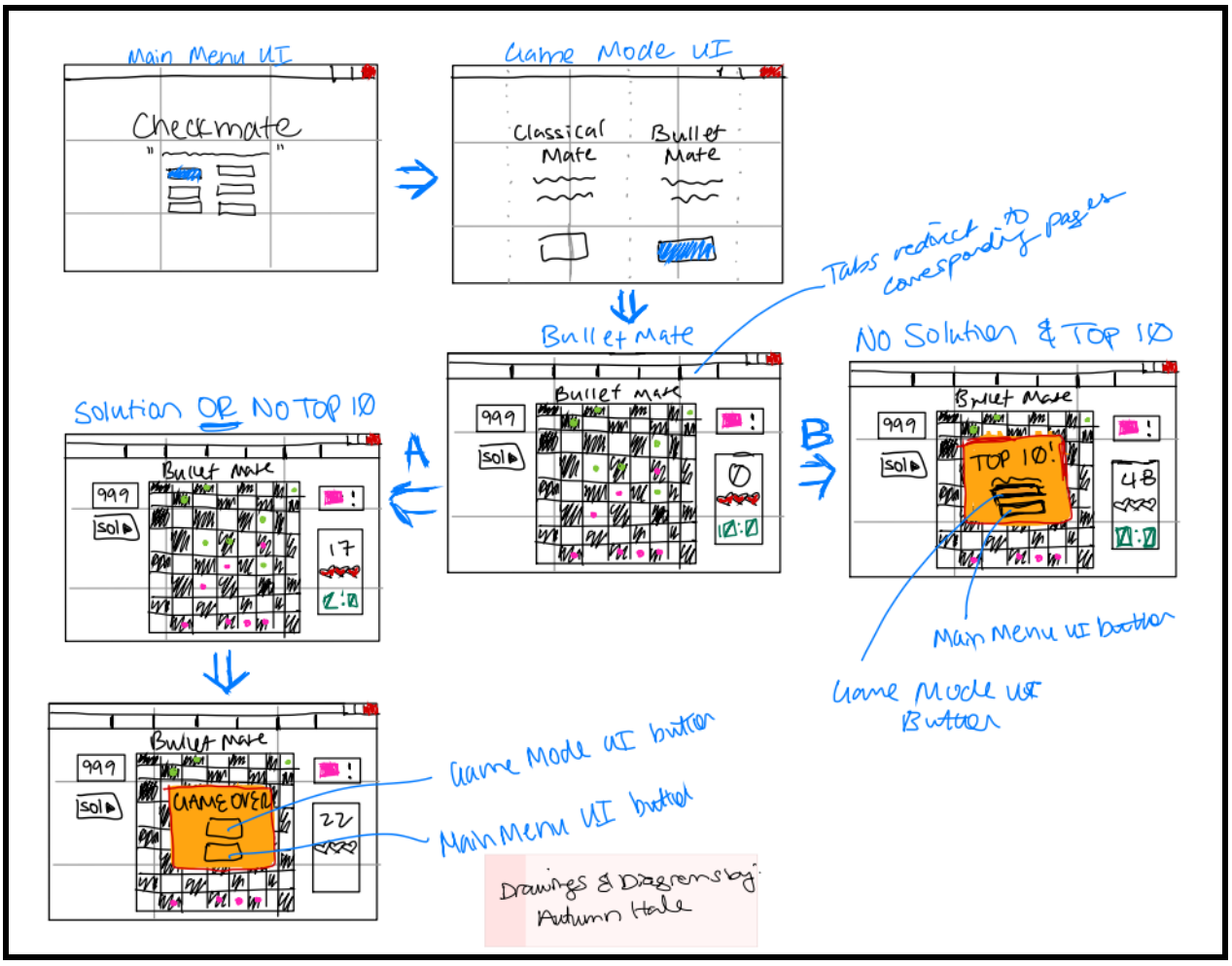

Main Menu UI
Checkmate
Game Mode UI
Classical Mate
Bullet Mate
Bullet Mate
Tabs redirect to corresponding pages
Solution OR NO TOP 10
A
B
NO Solution & TOP 10
Bullet Mate
999
sol
17
22
48
TOP 10!
GAME OVER!
Main Menu UI button
Game Mode UI button
Main Menu UI button
Drawings & Diagrams by: Autumn Hale

Fig. 2

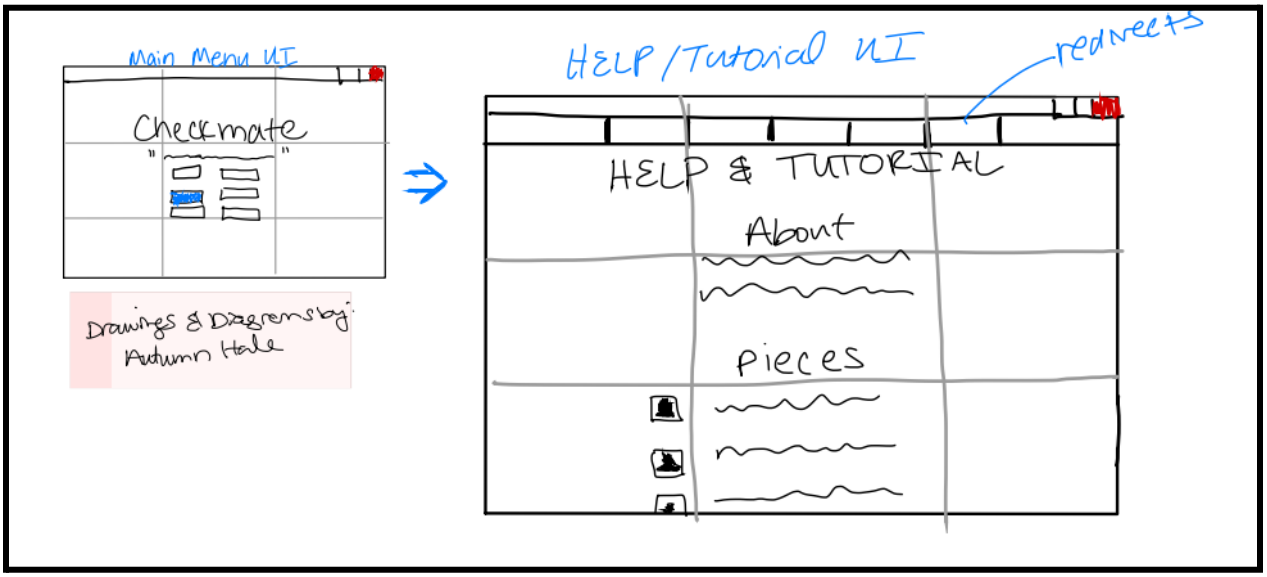

Main Menu UI
Checkmate
HELP/Tutorial UI
redirects
HELP & TUTORIAL
About
Pieces
Drawings & Diagrams by: Autumn Hale

Fig. 3

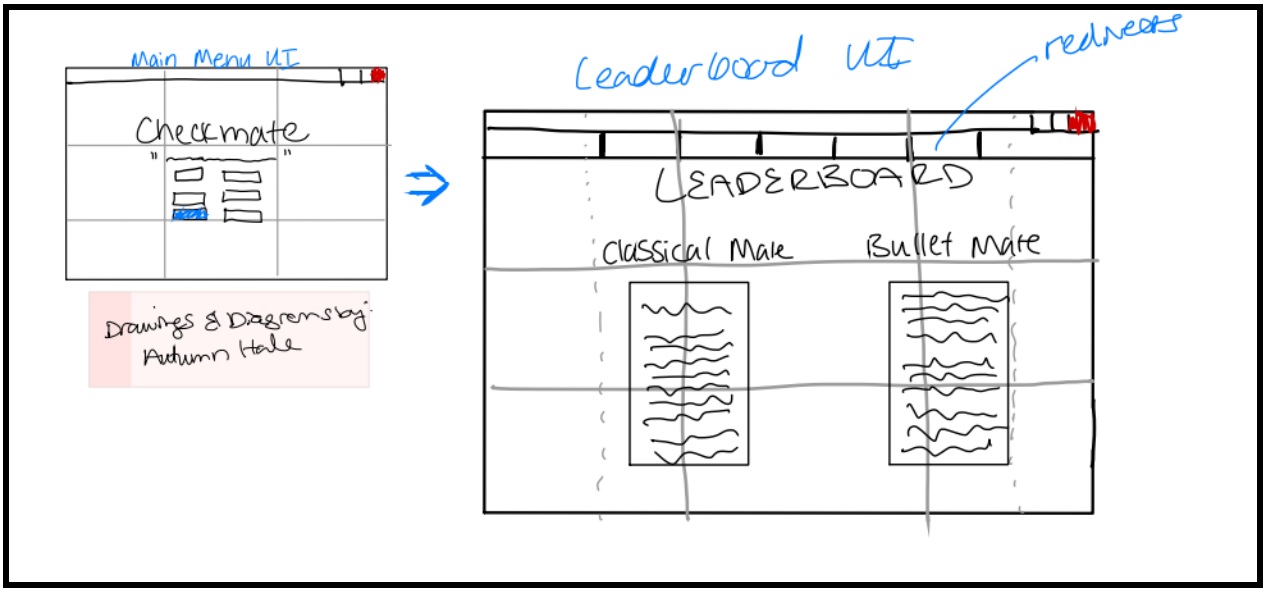

Main Menu UI
Checkmate
Drawings & Diagrams by: Autumn Hale
Leaderboard UI
redirects
LEADERBOARD
Classical Mate
Bullet Mate

Fig. 4

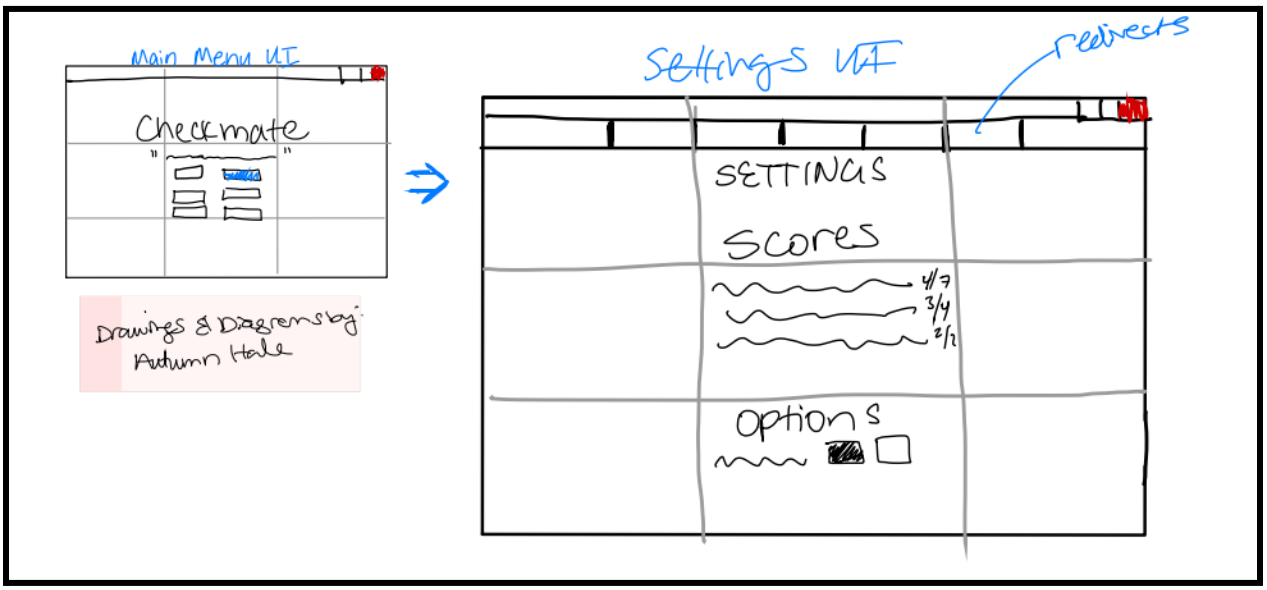

Main Menu UI
Checkmate
Drawings & Diagrams by: Autumn Hale
Settings UI
redirects
SETTINGS
scores
4/7
3/4
2/2
options

Fig. 5

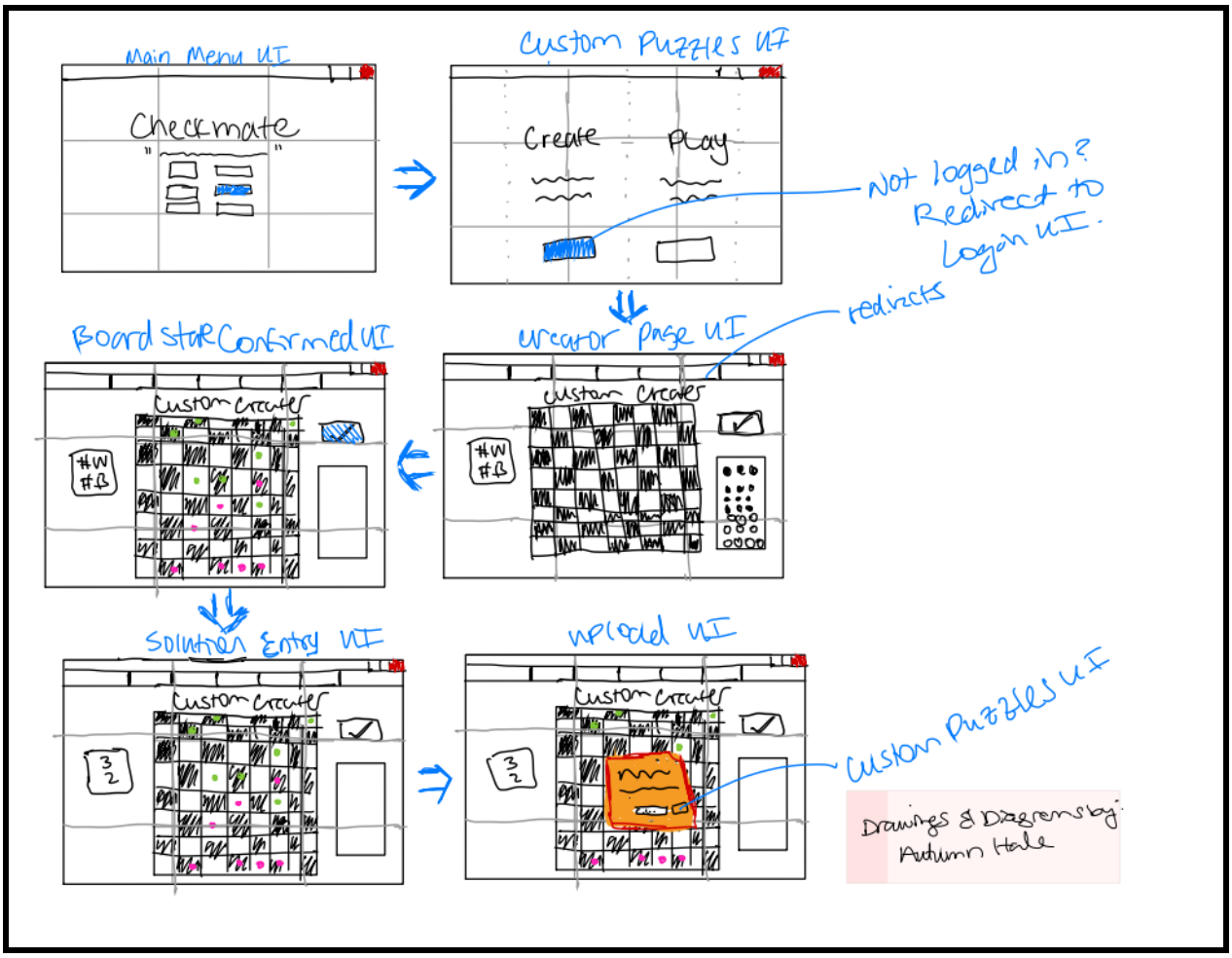

Main Menu UI
Checkmate
Custom Puzzles UI
Create
Play
Not logged in? Redirect to Login UI.
redirects
Board StarConfirmed UI
Creator Page UI
Custom Creator
Solution Entry UI
Uploaded UI
Custom Puzzles UI
Drawings & Diagrams by: Autumn Hale

Fig. 6

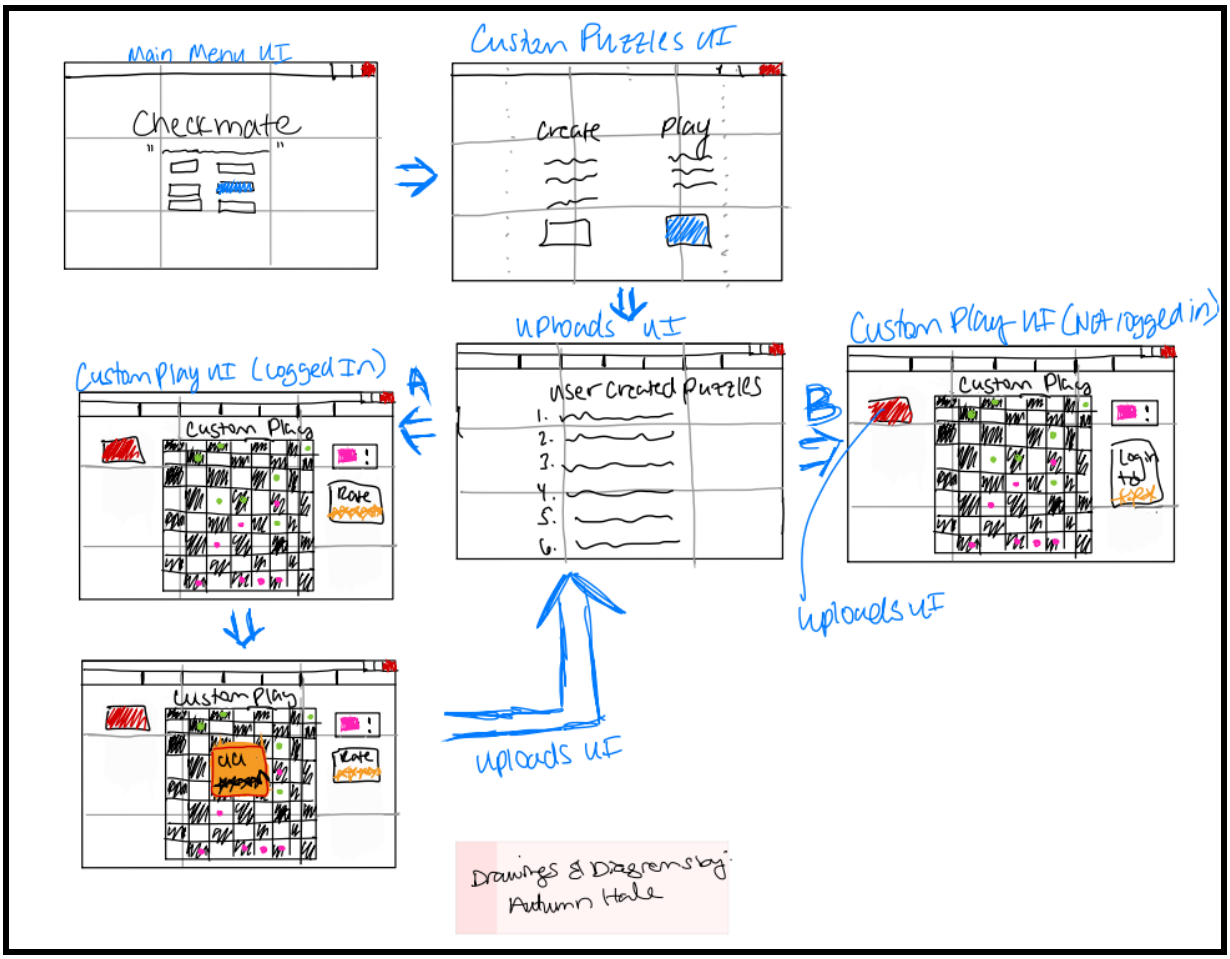

Main Menu UI
Checkmate
Custom Puzzles UI
Create
Play
Uploads UI
User Created Puzzles
1.
2.
3.
4.
5.
6.
Custom Play UI (Logged In)
Custom Play
Rate
Custom Play UI (NOT logged in)
Custom Play
Log in to
Uploads UI
Custom Play
Rate
Uploads UI
A
B
Drawings & Diagrams by:
Autumn Hale

Fig. 7

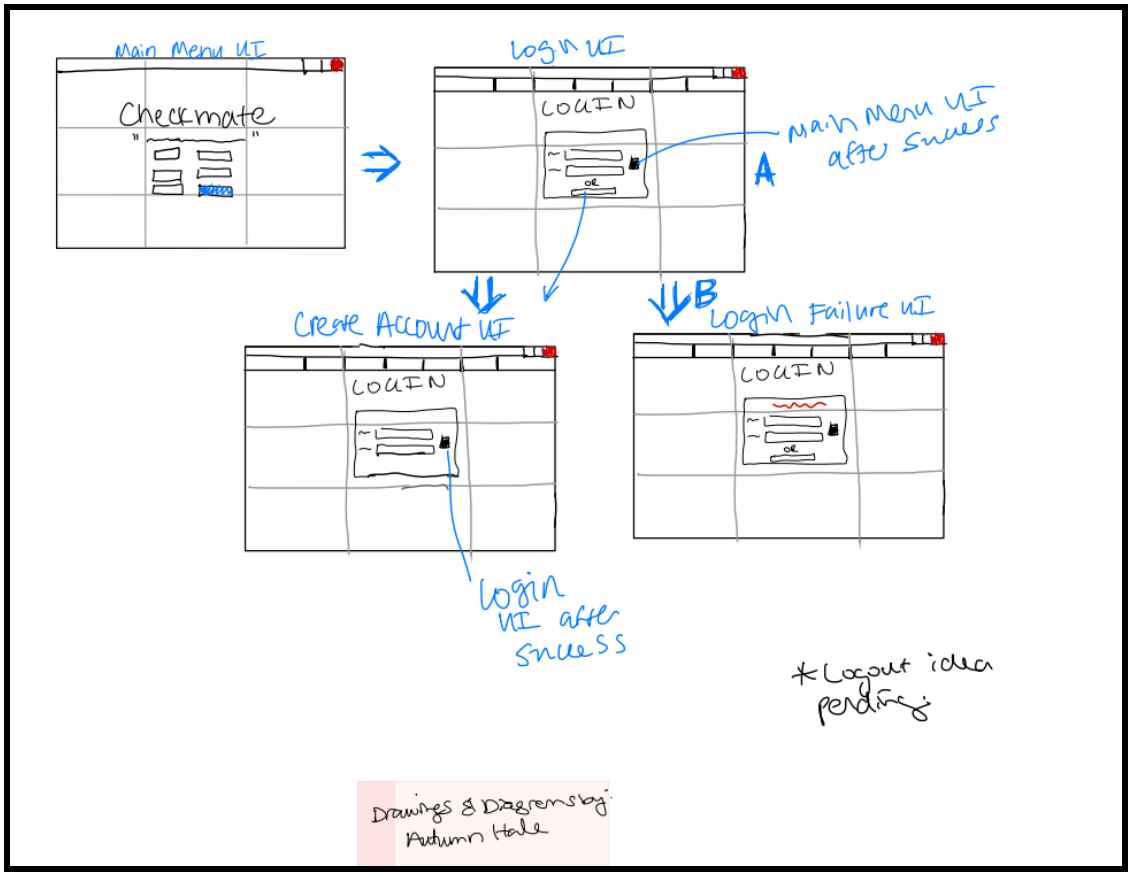

Main Menu UI
Checkmate
Login UI
LOGIN
Main Menu UI after Success
A
B
Create Account UI
LOGIN
Login Failure UI
LOGIN
login UI after success
*Logout idea pending.
Drawings & Diagrams by: Autumn Hale

Fig. 8

Appendix 4 - Message Documentation

1. Retrieve board data
    a. Database to client
    b. FEN, moves, puzzleID, rating, and themes from index
        i. These will be use to populate the board and verify moves for both regular, and custom puzzles
2. Send leaderboard data
    a. Client to database
    b. Sends current score, username, and time if applicable
        i. These are used to send a score to the leaderboard and save a user's past scores for their viewing
3. Retrieve leaderboard information
    a. Database to client
    b. Retrieves Usernames (string), Scores (number), and times (time) if applicable from all entries in the current leaderboards
        i. These are used to display the leaderboard info as well as verify if a score should be submitted to the leaderboard
4. Send custom puzzle
    a. Client to database
    b. Sends a json with a custom puzzle's information to be stored in the database
        i. This will be used for other users to play in the custom puzzles menu. (req. 7)
        ii. Information includes the board state (string), correct moves (string) and puzzleid (string)
5. Retrieve custom puzzle list
    a. Database to client
    b. This will retrieve a list of custom puzzles' names (string), and rankings (number)
        i. This information will be used to populate the custom puzzle selection window
6. Firebase Authentication services will handle all messages relating to handling our logins

Appendix 5 - ER Diagram