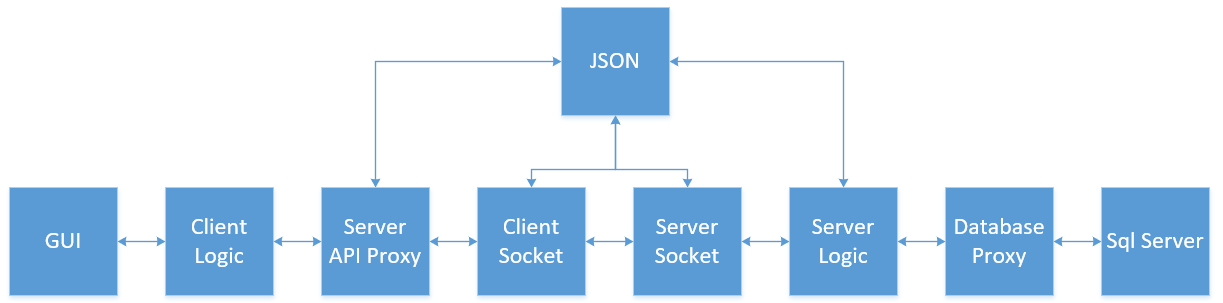
System Architecture (UML)

2D Chess Game

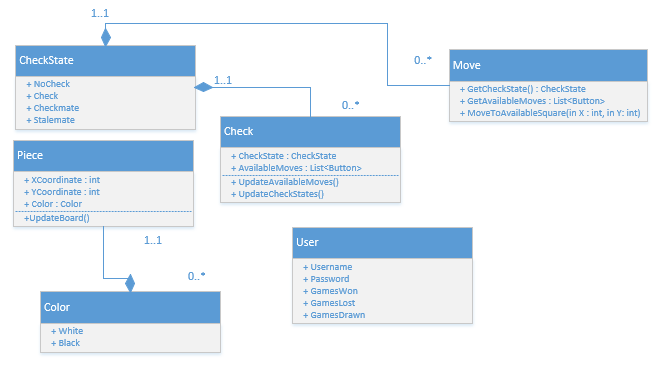
Caleb Gordon and Nicholas Heath

**Module Diagram**



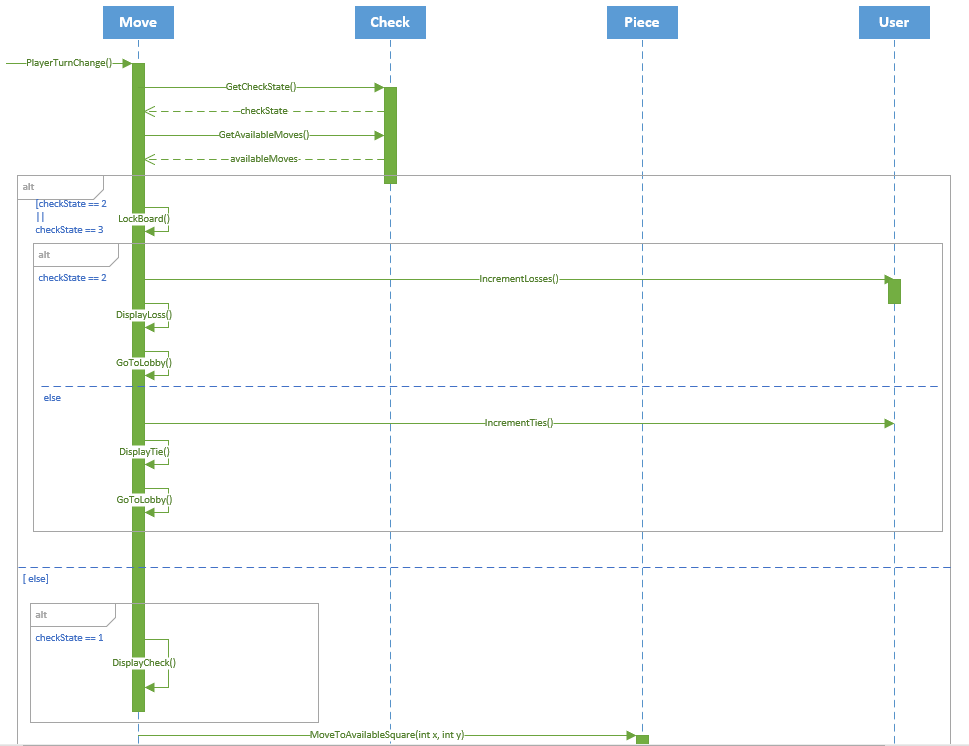
The chess game application will use a client-server architecture. The client will interact with the server via REST APIs and TCP sockets. All data exchanged between the client and server will be in JSON format and will be encoded and decoded accordingly. The server will interact with a SQL Server database.

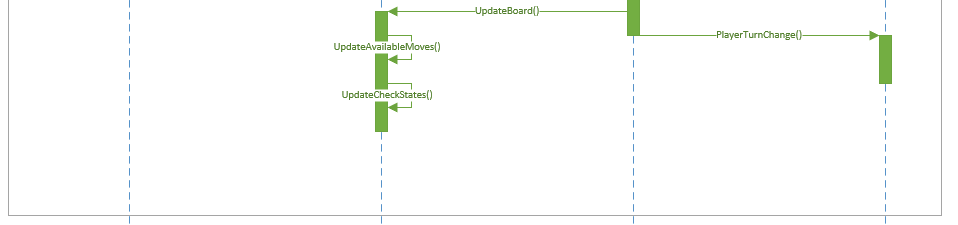
**Client-Side UML Class Diagram**



This UML class diagram explains the relationship between objects during an on-going chess game. CheckState and Color are enum objects which are used to identify the state of certain objects. Piece holds the information about a chess piece’s coordinates and which player it belongs to (black or white). Move is used for getting all available moves and helps a piece move to a specific square on the chessboard. Check is used for determining whether the king is currently in check, checkmate, or stalemate and helps determine what moves are available for the king to move. User holds information about a user and is used in the built-in Windows form object.

**Client-Side UML Sequence Diagram**





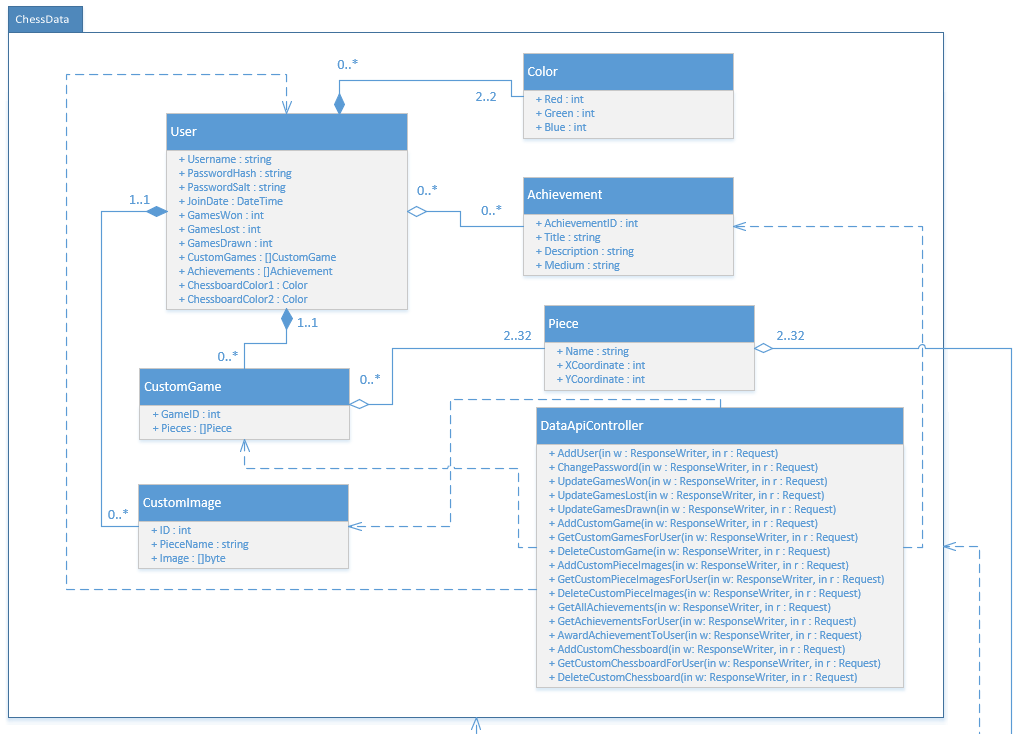
When a change of turn signal is received by the client, the Move class will receive it, and request the check state, and list of available moves from the Check class. The check state will be returned as NoCheck, Check, Checkmate, and Stalemate as 0, 1, 2, and 3, respectively. Depending on the state of check, one of the following situations will occur.

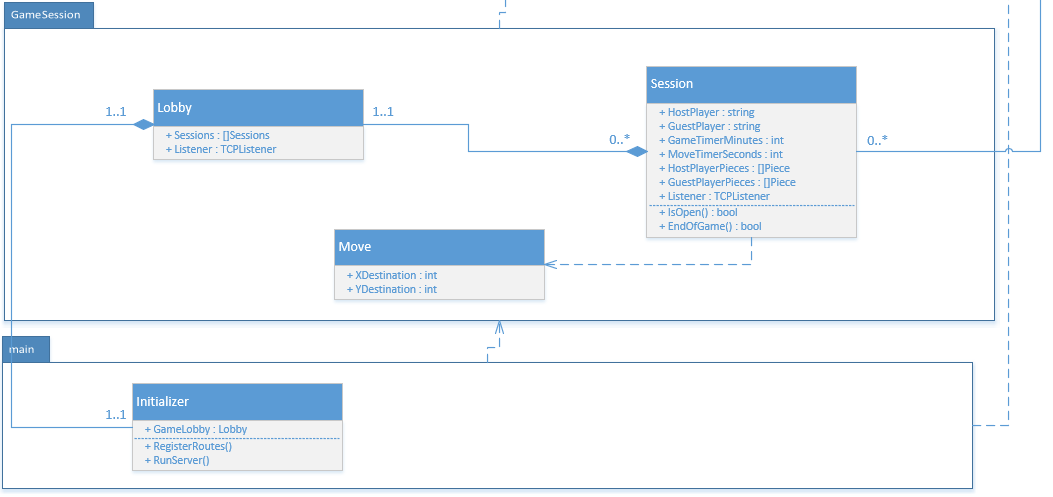
Should the check state be returned as Checkmate (2), this will result in a loss for the player currently active, since the checkmate was caused by the opponent. If this happens, the board will be locked, and the Move class will signal the User class to increment the number of games lost. The move class will also signal the opponent client to increment the number of games won.

Should the check state be returned as Stalemate (3), the same logic as above will occur, except the Move class will signal both clients to increment the number of games drawn.

Should the check state be returned as NoCheck (0) or Check (1), the code will wait for the user to click on a piece with an available move, and move that piece. Once that piece is moved, the Move class will update the Piece’s coordinates (Piece class). This will update the board on both clients, and update the available moves and check state of the enemy’s king. After this is complete, the client will send a signal to the server to switch to the opponent’s turn. The server will signal back to the same code to repeat the process for the other player. This will repeat until the game is over.

**Server-Side UML Class Diagram**

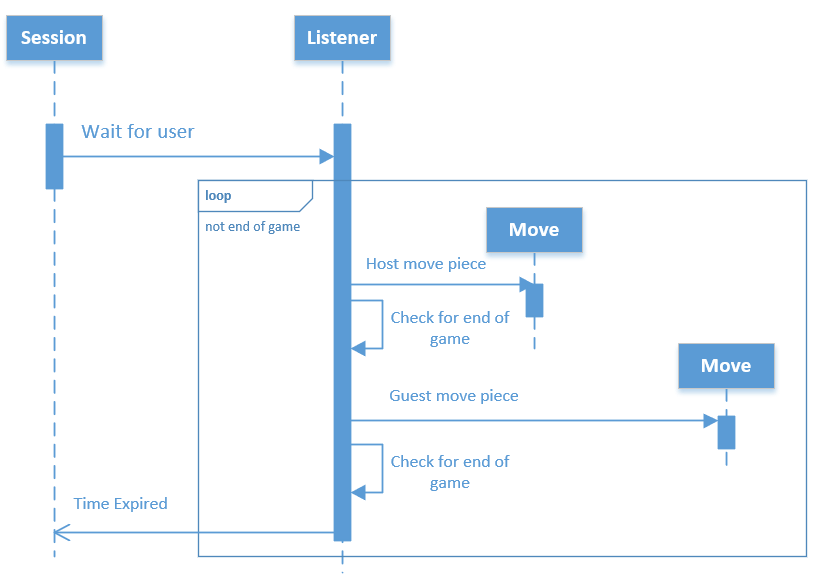




All of the data handling and storage is done in the ChessData package. The DataApiController handles all incoming REST API calls from the client. The functions handle the API calls and add, edit or delete data. New users can be added to the database, user information can be changed/updated, and user data such as custom games, custom piece images, achievements, and custom chessboard color settings. A user can have any number of custom images, any number of achievements, and any number of custom games, but they can only have two colors for the chessboard. Any number of achievements can be awarded to any number of users. Both a CustomGame and session can have between 2 to 32 pieces.

All session logic is done in the GameSession package, and the main package is used for initializing and running the server. The Lobby can have any number of sessions, and there is only one lobby for a session to belong to.

**Server-Side UML Class Diagram**



This sequence diagram explains how a session works on the server. When a session is created by the Lobby, the session waits for the user via the TCP Listener. After the user enters the room, the game starts and enters a loop until the game ends. The server receives a signal from the client to move the piece from the player and checks for an end of game (checkmate or stalemate). If there isn’t, it switches over to the other player’s turn and performs the same procedure. Alternatively, the game can end if either user’s move or game timer expires, which is signaled by the client.