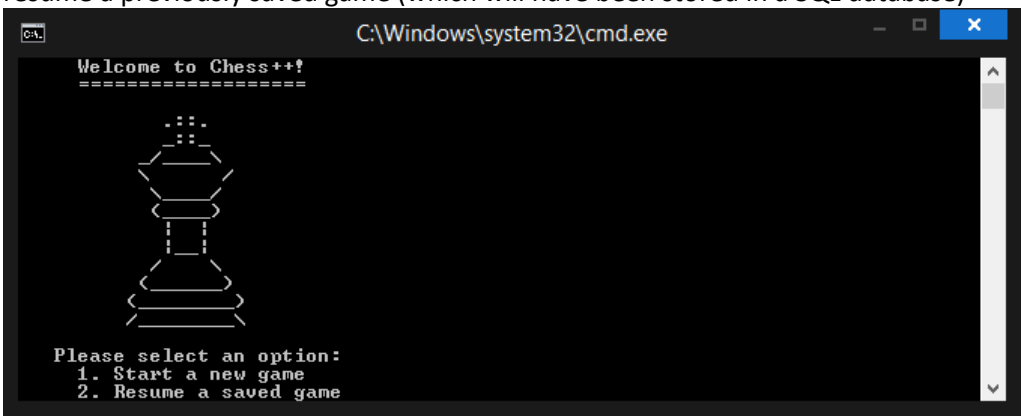
Chess++: Requirements Analysis

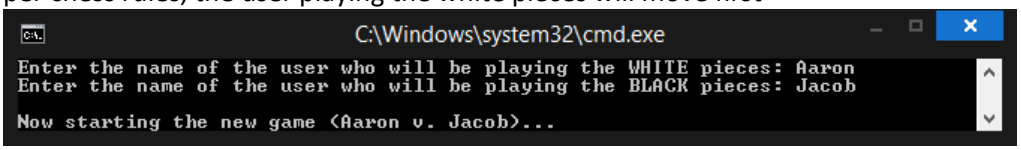
(Aaron Oeder, Jacob Owens, and Trevor Berceau)

End-User Requirements

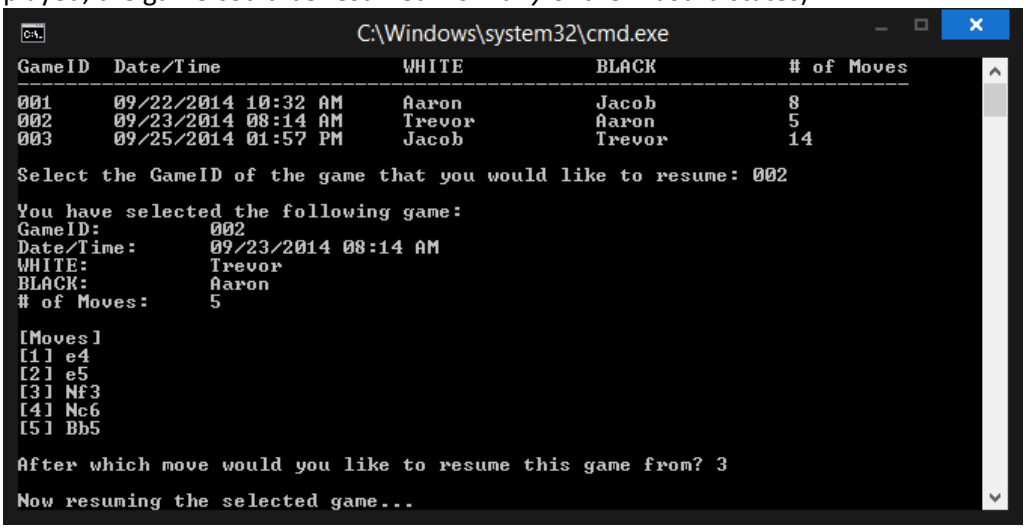
- Will support two-player (non-AI) chess matches
- Upon launching the program, the users will be given the option to either start a new game or to resume a previously saved game (which will have been stored in a SQL database)



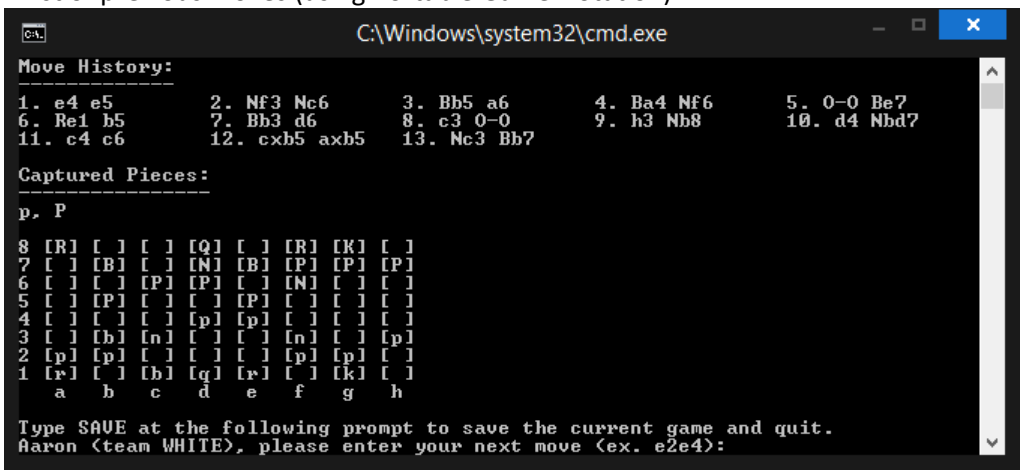
  - If starting a new game, the program will prompt the users for the names of the user playing the white pieces and the user playing the black pieces prior to starting the game; per chess rules, the user playing the white pieces will move first



  - If resuming a previously saved game, the users will get to choose which board state to resume the selected game in (i.e., if a previously saved game had a total of n moves played, the game could be resumed from *any* of the n board states)



- Moves will be entered by the users in the form of $C_iR_iC_fR_f$ [where C_i is the column of the initial coordinate, R_i is the row of the initial coordinate, C_f is the column of the final coordinate, and R_f is the row of the final coordinate] (ex. e2e4)
    - The program will accept all moves present in standard chess games, including special moves such as castling, pawn promotion, and en passant captures
    - If a move is illegal, an error message will be displayed on the console and the user will be once again prompted to enter a legal move
- The program will alternate in prompting for moves from the user playing the white pieces and the user playing the black pieces
    - If a user instead types SAVE at such a prompt, the current game will be saved to the SQL database and the program will exit
- The following will be displayed to the user (via console output) after each turn:
    - A visual representation of the game board state
        - White pieces are abbreviated using lowercase letters (p, n, b, r, q, k) while black pieces are abbreviated using uppercase letters (P, N, B, R, Q, K)
            - p/P represents a pawn, n/N represents a knight, b/B represents a bishop, r/R represents a rook, q/Q represents a queen, and k/K represents a king
    - A list of captured pieces (using the piece abbreviations)
    - A list of previous moves (using Portable Game Notation)



- If and when a user moves a pawn to its final rank, the user will be prompted to enter the piece abbreviation that the pawn should be promoted to
    - For example, if the user playing the white pieces moves a pawn from h7 to h8, he/she would type ‘q’ at the prompt to promote said pawn to a queen
- A message will be displayed on the console when a player is in check and when the game is over (either by checkmate or by stalemate); the program will exit after acknowledgement of this message

Technical Specifications

The program will be developed on Windows-based machines using Microsoft Visual Studio 2012/2013 and the C++11 programming language. Chess++ will be compatible with Windows and Mac OS as well as other Unix-based operating systems. A SQL database will be kept alongside the executable program (for the purpose of saving/loading previous chess matches) and will require that a database management system (such as SQLite) be installed. Minimal disk space will be required for the program and the aforementioned database.

Acceptance Criteria

Input Data Items

- When the program starts, the users must select whether they want to start a new game (denoted by a '1') or resume a saved game (denoted by a '2')
    - If starting a new game, the users must enter the names of the user who will be playing the white pieces and the user who will be playing the black pieces
    - Else if resuming a saved game, the users must select the move after which they would like to resume the game from; if a game has n moves, the user must enter a number between 1 and n
- Users will input moves in the form of $C_iR_iC_fR_f$ [where C_i is the column of the initial coordinate, R_i is the row of the initial coordinate, C_f is the column of the final coordinate, and R_f is the row of the final coordinate] (ex. e2e4)
- At any point during the game, the users can type SAVE at the prompt that asks for their next move, at which point the current game will be saved to the SQL database and the program will exit

Output Data Items

- A welcome screen is displayed upon launching the program that gives users the option to start a new game or resume a previously saved game
    - If the users would like to start a new game, a prompt is displayed to the console asking the users to enter the name of the user who will be playing the white pieces and the user who will be playing the black pieces
        - The program will begin by prompting the user playing the white pieces for his/her first move
    - Else if the users would like to resume a saved game, a table listing the previously saved games is displayed
        - After specifying which game the users would like to resume, the program displays this game's information on screen, including all of the moves that were made in the game
- The program will alternate in requesting moves from the user playing the white pieces and the user playing the black pieces
    - If any of these moves are illegal, an error message will be displayed on the console and the user will have to enter a different move
- After each move, a visual representation of the game board state as well as lists of both captured pieces and previous moves will be displayed on the console
- If and when a user moves a pawn to its final rank, a prompt will be displayed on the console asking the user for the piece abbreviation that the pawn will be promoted to
- When a player is in check, a message will appear directly above the prompt asking them for their next move
- When either player is in checkmate/stalemate, a message will appear on the console

Testing/Debugging Plan

- Verify that only moves legal according to standard chess rules are allowed to be played

- This will be accomplished by entering moves that should be legal and seeing that they are allowed to be played as well as making moves that should be illegal and seeing that they are not allowed to be played
- The table below summarizes the necessary conditions for a legal move:

Move in Question	Conditions for Legality
ANY MOVE	The destination square cannot contain a "friendly" piece (one that is the same color as the piece that is moving)
	Making the move cannot place one's own king in check
PAWN moving one square forward	The destination square must be unoccupied
PAWN moving two squares forward	The pawn must be on its starting rank
	The square between the initial square and the destination square must be unoccupied
	The destination square must be unoccupied
PAWN moving one square diagonally	The destination square must contain an enemy piece unless it is an en passant capture (in which case the destination square must be unoccupied)
BISHOP moving **ROOK** moving **QUEEN** moving	The squares between the initial square and the destination square must be unoccupied
KNIGHT moving **KING** moving one square in any direction	No additional restrictions
For the following castling moves, the coordinates of the initial square will be denoted by (r, c) where r is the row and c is the column	
KING moving two squares to the left	The squares with coordinates (r, c - 1), (r, c - 2), and (r, c - 3) must be unoccupied
	Neither the king nor the queenside rook can have moved in the game so far
	The king cannot currently be in check
	The opponent cannot be attacking the coordinate that the king will pass through (r, c - 1)
KING moving two squares to the right	The coordinates (r, c + 1) and (r, c + 2) must be unoccupied
	Neither the king nor the kingside rook can have moved in the game so far
	The king cannot currently be in check
	The opponent cannot be attacking the coordinate that the king will pass through (r, c + 1)

- Verify that an error message is displayed after entering an illegal move and that the user is prompted to enter a different move
- Verify that a game can be successfully saved to the database
- Verify that a game can be successfully resumed (after being retrieved from the database) in *any* board state
- Verify that the move history is being properly displayed in PGN notation
    - This will be done by playing through various types of moves and ensuring that the moves are being recorded in proper PGN notation for each
        - A few sample test cases:

- A pawn moving from e2 to e4 should be recorded as “e4”
- A knight moving from g1 to f3 should be recorded as “Nf3”
- A pawn on e4 that captures a pawn on d5 should be recorded as “exd5”
- A queenside castling move should be recorded as “O-O-O”
- A white pawn on the h-file promoting itself to a queen should be recorded as “h8=Q”
- A bishop moving from f1 to b5 that delivers check to the opponent should be recorded as “Bb5+”
- A rook moving from h1 to h8 that delivers checkmate to the opponent should be recorded as “Rh8#”

- Verify that the list of captured pieces is accurate
  - This will be done by playing several games and making sure that:
    - (pieces remaining on the board) + (pieces in the list of captured pieces) = pieces that were present when the game began
- Verify that all notifications of check, checkmate, and stalemate appear only when they should

Feasibility Report

The program will require a database management system (such as SQLite) to be installed in order to save/load previous chess games. If such software is not installed (or unable to be installed due to system incompatibilities) on a client’s machine, errors will occur when trying to save/load chess matches. The program also poses several restrictions with regards to usage of the program. Users of the program will be expected to have a complete understanding of the rules of chess, the syntax of move entry, and the abbreviations of the chess pieces.