Must Have

Create an account on the system by providing an email and creating a username/nickname and password **Task Finished**

- **Given:** The user has navigated to the login page
- **When**: The user selects the register an account button
- **And**: Has entered all necessary information for creating an account (valid username, valid password, valid email address)
- **Then:** System verifies the information is valid and creates the account and adds them into the database
- **When:** The user selects the register an account button
- **And:** The user has entered an invalid or taken username
- **Then:** The system verifies the information is invalid and a message is displayed to the user reflecting the that the username is not acceptable
- **When:** The user selects the register an account button
- **And:** The user has entered an invalid password
- **Then:** The system verifies that the password is not acceptable, and a message is displayed to the user reflecting that the password is not valid
- **When:** When: The user selects the register an account button
- **And:** The user has entered an invalid or taken email address
- **Then:** The system verifies that the email address is invalid, and a message is displayed to the user reflecting that the email is invalid

Be able to log in once an account is created **Task Finished**

- **Given:** A user has created an account
- **When:** The user enters their correct credentials into the login page
- **Then:** The user will be logged into their account and be able to access the information related to their account.
- **When:** The user enters an incorrect username or email address into the login page
- **Then:** The user is not logged in and a message is displayed reflecting that their credentials were incorrect
- **When:** The user enters a correct username or email address and an incorrect password into the login page:
- **Then:** The user is not logged in and a message is displayed reflecting that their credentials were incorrect

Create a new match **Task Finished**

- **Given:** Player selected opponent by username
- **When:** After the player selected their desired opponents
- **And:** The player clicked the option to send the invites

- **When:** Another opponent has accepted an invite
- **Then:** A new match is created in the database
- **And:** The two players have access to the match

Invite friends/opponents to the match **Task Finished**

- **Given:** The user has logged in
- **When:** The user clicks on the send invite button
- **Then:** A list of possible users is presented to the user to choose from
- **When:** The user selects the users, they want to send the invite to
- **Then:** Invites are sent to the selected users

Be able to play a match according to the rules of Portal Chess **Task Finished**

- **Given:** A game has been started
- **When:** The user enters the game
- **Then:** The user will be able to play a game of Portal Chess against the other user according to the rules of the game.

Player moves their piece to a valid location **Task Finished**

- **Given:** The player is playing a game of Chess
- **When:** The player selects one of their pieces to move, and a valid location to move it to
- **Then:** The system will move their piece, update the state of the board, and switch who's turn it is

Player moves their piece to an invalid location **Task Finished**

- **Given:** The player is playing a game of Chess
- **When:** The player selects one of their pieces to move, but selects to move it to an invalid location
- **Then:** The system will attempt to move their piece, detect this as an invalid move, revert the chessboard to the original state before the move was made, and inform the player the move is invalid

Player attempts to move their opponent's piece **Task Finished**

- **Given:** The player is playing a game of Chess
- **When:** The player selects one of their opponent's chess pieces to move
- **Then:** The system will detect that this piece is not their color, and then inform the player that they can't do this

Player moves their piece onto an opponent's piece **Task Finished**

- **Given:** The player is playing a game of Chess
- **When:** The player attempts to move their piece onto an opponent's piece
- **Then:** After validating that this is a valid move the system will move the piece to the location, remove the opponent's piece, update the board, and switch who's turn it is

Both players place their portal piece on the board **Task Finished**

- **Given:** A game of portal chess has begun
- **When:** Both players must place their portal piece onto the board according to Portal Chess rules
- **Then:** White places their piece first,
- **And:** System validates it is a valid location and places the piece
- **Then:** Black places their piece
- **And:** The system validates this location as well

A player attempts to move their piece onto a portal piece **Task Created**

- **Given:** During a game of portal chess a player is attempting to move their piece onto a portal
- **When:** A player is attempting to move their piece onto a portal
- **Then:** The system will then validate this move
- **And:** Once validated move their piece onto the portal
- **Then:** System will then remember the direction the piece was moving for the next move

A player attempts to promote a pawn to queen **Tasks Finished**

- **Given:** During a game of portal chess a player attempts to promote their pawn to a queen
- **When:** A player moves their pawn to the last row on the opposite side of the board
- **Then:** The pawn will be promoted to a queen.

A player attempts to perform castling **Tasks Finished**

- **Given:** During a game of portal chess a player attempts to perform the castling move
- **When:** A player attempts to perform castling
- **Then:** The system will validate the castling move
- **And:** once all requirements are validated
- **Then:** the system will allow the castling move

A player puts their opponent into check **Tasks Finished**

- **Given:** During a game of portal chess
- **When:** A player attempts to move a piece that will put their opponent in check
- **Then:** The system will detect that this has happened
- **And:** Informs both players who is in check
- **Then:** The player in check must make a move to get out of check or abandon

A player puts their opponent into checkmate **Tasks Finished**

- **Given:** During a game of portal chess
- **When:** A player attempts to move a piece that will put their opponent in checkmate
- **Then:** The system will detect that this has happened
- **And:** Inform the players who the winner/loser is
- **Then:** End the match and add it to match history

Should Have

Accept invite to game sent by another player **Task Finished**

- **Given:** That an invite has been sent from one player to another
- **When:** The player receiving the invitation clicks a button to accept the invitation
- **Then:** The invitation will be accepted, and a game will be created

Reject invitations - user who sent request would be notified **Task Finished**

- **Given:** The user has received an invite to a match from another player
- **When:** The user selects to reject this request
- **Then:** The system will then notify the inviter that this request has been denied

Would like to be able to close the match and come back to it later **Task Finished**

- **Given:** A match has been started between 2 players
- **When:** The user leaves the game
- **Then:** The game state is saved
- **And:** An option to return to the game is presented to the user

Know when a game is over and who has won/lost or if it has been abandoned **Task Finished**

- **Given:** Two players are in an active match.
- **When:** A player moves a piece that can capture a king.
- **And:** The king cannot move out of check.
- **And:** The attacking piece cannot be captured.
- **Then:** The game will announce that the attacking side won the match.
- **And:** The match will be removed from the active games.
- **And:** The match is stored in finished games list with winner and loser stored.
- **And:** Both users are returned to the main menu.

Could Have

Be able to log out **Task Finished**

- **Given:** The user is logged in
- **When:** The user clicks on the sign out button
- **Then:** The state of any open games is saved
- **And:** The user is taken to the sign in screen

Unregister the account **Task Finished**

- **Given:** The user has decided to unregister their account
- **When**: They click the unregister account button
- **Then**: The system removes their account from the database
- **And**: Returns the user to the login page

Switch between multiple games at once **Task Finished**

- **Given:** The user is playing a match
- **When:** The user clicks the switch match button
- **Then:** The state of the game is saved
- **And:** The user is presented with a list of their current games
- **When:** The user clicks on one of the game options
- **Then:** The selected game is resumed

Abandon any game at any time **Task Finished**

- **Given:** The user has an active game available
- **When:** The user clicks on the abandon game button either in the game interface or the active games list
- **Then:** A notice that the opponent has won is sent to the opponent
- **And:** The match history is recorded in the profiles of both players
- **And**: Game state is deleted
- **And:** The game is removed from both players active games list

Would Have

View my/another player's game history (players, start and end date/times, winner/loser of the match, whether a game was abandoned) on my/their profile **Task Finished**

- **Given:** A user is logged into their account
- **When:** A user selects their match history or another player's history
- **Then:** The user will see the game history including start/end times, winner/loser of each match, or whether a game was abandoned.

Be able to tell whose turn it is **Task Finished**

- **Given:** The match is active
- **When:** Constantly displayed as part of the UI
- **Then:** The active player's name will be displayed
- **When:** The active player finishes their turn
- **Then:** The displayed name will change to the other player, who is now the active player

Optional, if there is time

Play against a bot

- **Given:** A match is not yet active
- **When:** The player selects the AI as their opponent
- **And:** The player sends the invite to the AI
- **Then:** A new match will be created in the system
- **And:** That match will have one player and one AI-driven opponent
- **When:** The AI-driven opponent is the active player of the match
- **Then:** The AI-driven opponent will perform their turn
- **And:** The AI-driven opponent will make their turn based on a set of optimal moves calculated given the state of the board

Be able to organize a tournament.

- **Given:** There are at least 3 users in the user database
- **And:** The user has logged in
- **When:** The user selects the organize tournament button
- **Then:** The user is presented with a list of potential users to send an invite to
- **When:** The user selects at least 2 users to send invites to
- **Then:** Invites are sent out to the selected players
- **When:** All the users have responded
- **And:** At least 2 users have accepted
- **Then:** A bracket is set up
- **And:** Matches are created between users that are scheduled to play

Play in a tournament.

- **Given:** a player is in an active tournament.
- **When:** A match between a pair of competitors ends.
- **Then:** the winner moves up to the next bracket in the tournament.
- **And:** the loser does not receive a new match invite but is not removed from viewing the tournament.
- **When:** Two players complete their matches and move up to the next bracket.
- **Then:** New invites are generated for those players.
- **When:** The final match ends.
- **Then:** The tournament stores the places of all the players that participated.
- **And:** The tournament ends and is removed from the active tournaments.

Chat with other users.

- **Given:** A user would like to send a message to another user
- **When:** The user selects send message to a specific other user
- **Then:** A window pops up allowing them to type the message
- **When:** The user clicks send
- **Then:** The message is sent to the other user
- **And:** The recipient can read and reply to this message in their inbox

Get a badge after winning or placing second in a tournament

- **Given:** A tournament has completed
- **When:** A user competed in the tournament
- **And:** The user won the tournament
- **Then:** The user will be awarded a gold badge
- **When:** A user competed in the tournament
- **And:** The user placed second in the tournament
- **Then:** The user will be awarded a silver badge