**Battleship**

**Team #1**

**Skyler Thompson (Lead), Alex Waitkus, Garrett Ellison, Anish Gilani, Taylor Cannon**

1. **Problem Statement**

The proposed software system will allow two people to play the board game Battleship against each other over the internet. The game will be able to authenticate users and assign them to a random game or allow them to play with someone else using a unique shareable link. The server will keep track of the game state so that no player can cheat; it will also determine what moves are valid. When a player makes a move, the other player’s board will be updated instantaneously so both players are always up to date on the state of the game. The server will determine when someone has won the game and will alert both players.

1. **Functional Requirements**

* The player must log in with a valid username/password (Login, LoginError)
* The player creates a new account with their information (CreateAccount, CreateAccountError)
* The player can create a new game (CreateGame)
* The player can join another game(JoinGame)
* The game starts and both players are presented with a gameboard, ships, and a log of coordinates that have been played (BeginGame)
* Each player gets to choose the position of his/her ships (PositionShips)
* Each player chooses a coordinate to attack and the attack is either an hit or a miss (Attack, InvalidAttack)
* Both players’ boards are updated (UpdateBoard)
* A ship has all of its coordinates hit successfully (SinkShip).
* When one player has all of his/her ships sunk, the game ends (GameOver).
* The player can log out at any time (LogOut)
* The player can forfeit at any time (Forfeit)
* The player can can view the other player’s profile at any time (ViewProfile)

1. **Use Case Descriptions**

Use Case Name: Login

Actor: Player

1. Player accesses login screen.
2. Player enters username/password.
3. Username/password verified by system.
4. Player logs into the system and is presented with the “Create Game” menu.

Entry Condition:

None (First use case encountered)

Exit Condition:

“CreateGame” menu is presented to player

Use Case: LoginError (extends Login)

Actor: Player

1. The system determines that the username and/or the password are incorrect.
2. The player is presented with an error message.
3. The player is taken back to the login page to try again.

Entry Condition:

Login has been attempted and failed

Exit Condition:

“Login” menu is shown to the user again

Use Case: CreateAccount

Actor: Player

1. The player chooses the option to create an account at the login screen.
2. The player enters all of their information in (including a username/password).
3. The system verifies that this information is valid.
4. The player is taken back to the login page.

Entry Condition:

The program has been started

Exit Condition:

The player has an account successfully created

Use Case: CreateAccountError (extends CreateAccount)

Actor: Player

1. The player enters information in an invalid information, existing information, or leaves blank fields in the “Create Account” page.
2. The system rejects the information.
3. The player is sent back to the “Create Account” page.

Entry Condition:

The player enters invalid information into the “Create Account” page

Exit Condition:

The player is taken back to the “Create Account” page.

Use Case: CreateGame

Actor: Player

1. One player choose to start a new game
2. The player is taken to a waiting screen where they wait on another player to join

Entry Condition:

The player has logged into the system

Exit Condition:

The player is waiting on another player to join

Use Case: JoinGame

Actor: Player

1. The player decides to join an existing game and is taken to a waiting screen to wait for another player to create one or joins a game that exists.
2. The system registers the new player into the existing game.

Entry Condition:

The player has logged into the system

Exit Condition:

The game is ready to begin

Use Case: BeginGame

Actor: Player

1. Two players are waiting for a game to begin
2. The system creates a new game and both players are presented with a game board, ships, a log of coordinates already entered, and a way to submit coordinates.

Entry Condition:

Both players are logged into the system and are ready for the game to begin.

Exit Condition:

A new game begins

Use Case: PositionShips

Actor: Player

1. The player chooses the initial starting location of his/her ships.
2. The player submits his/her decisions to the server (the player is only presented with the button to present to the server if all ships are placed)
3. The system registers the ship locations and when both players’ ships have been entered, the attacking phase begins.

Entry Condition:

Both Players are logged into the system and a new game has begun

Exit Condition:

The attacking phase of the game begins

Use Case: Attack

Actor: Player

1. The system decides which player’s turn it is.
2. The player who’s turn it is (shown on the UI) enters coordinates for an attack.
3. The system validates these coordinates (to see if they’ve been chosen or not, or if they’re valid coordinates).
4. The system registers this attack and determines if it is a hit or a miss depending on if the attack lands on the other player’s ships.

Entry Condition:

Both players are logged into the system, a new game has begun, and ships have been positioned.

Exit Condition:

The system registers the attack as a hit or miss

Use Case: InvalidAttack (extends Attack)

Actor: Player

1. The system rejects a pair of coordinates as being invalid (for any reason)
2. The player is presented with an error message
3. The player is allowed to try a new pair of coordinates

Entry Condition:

The player chooses an invalid set of coordinates for an attack

Exit Condition:

The player is allowed to choose another set of coordinates

Use Case: UpdateBoard

Actor: Player

1. The player’s attack has been registered by the system as a hit or miss
2. The system updates its board to reflect this recent attack
3. Both client UIs are simulateously updated to reflect the new board
4. Both client UIS show the attack coordinates in the log
5. The system lets the players know who’s turn it is
6. The system waits for the player who’s turn it is to make an attack

Entry Condition:

An attack has been made

Exit Condition:

The system awaits the player who’s turn it is to make an attack

Use Case: SinkShip

Actor: Player

1. After an attack is made and is a hit, if the ship that was hit has had all of it’s coordinates “hit” then a ship is sunk.
2. This ship is now taken out of play and both players are presented with a message of which ship has now been sunk.
3. The system awaits the player who’s turn it is to attack

Entry Condition:

An attack has been made, it was a hit, and was the last hit on a ship before all coordinates have been hit

Exit Condition:

A ship has been sunk and the system awaits another attack

Use Case: GameOver

Actor: Player

1. After all of one player’s ships have been sunk, a player forfeits, or a player logs out and the game successfully ends.
2. Both users are alerted of which player has won and that the game is now over.
3. Both players are given the option to exit the game or return back to the login screen.

Entry Condition:

All of one player’s ships have been sunk, a player logs out, or a player forfeits

Exit Condition:

The game is over

Use Case: Forfeit

Actor: Player

1. One player chooses to forfeit the game for any reason.
2. The system registers the forfeit and registers a game over

Entry Condition:

One player chooses the forfeit option.

Exit Condition:

The system registers a game over

Use Case: LogOut

Actor: Player

1. One player chooses to log out before the game is over
2. The system registers the log out and updates the remaining player that the opposing player has logged out
3. The system registers a game over

Entry Condition:

One player chooses to log out before the game is over

Exit Condition:

The system registers a game over

Use Case: ViewProfile

Actor: Player

1. At any point in the game, one player decides to view his/her own or the opponent’s profile.
2. The system gathers all of the player information
3. The profile screen is shown as a pop-up window with all player information shown.
4. The player chooses to escape this window and the game returns to the playing state.

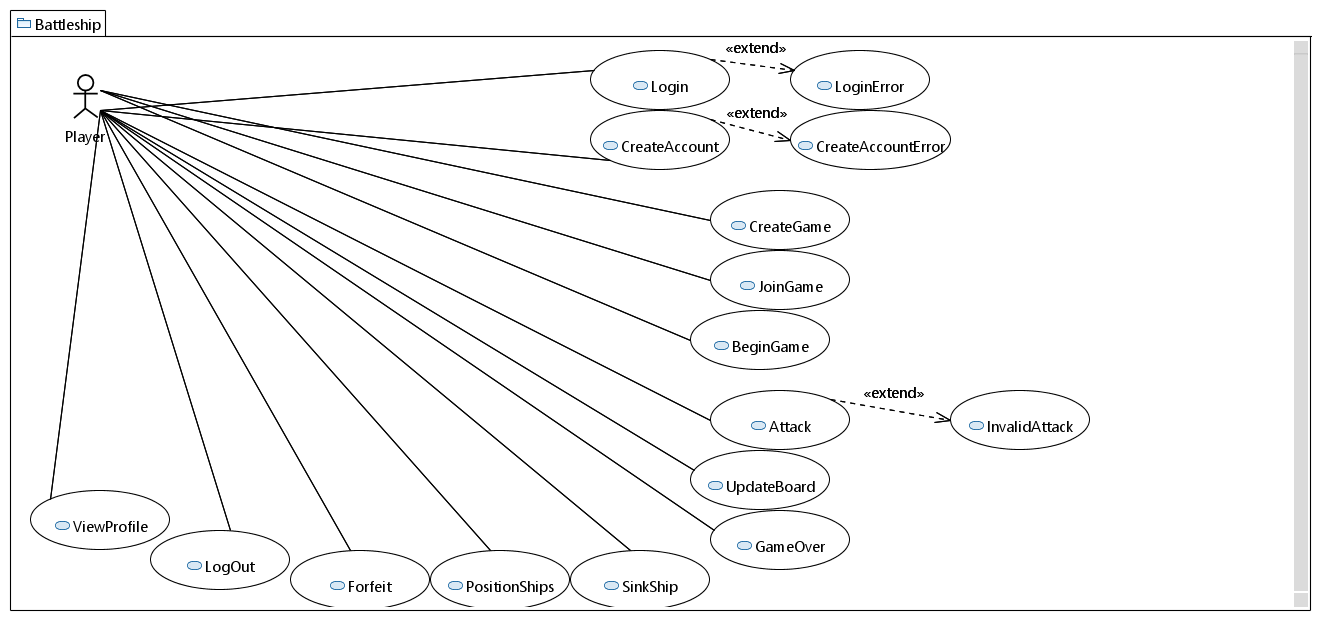
Entry Condition:

The player chooses to view a profile

Exit Conditon:

The game is returned to the playing state

1. **Use-Case Diagram**

****