ETH zürich

FS 2023

Software Requirements Specification

for

Battleship

Version 1.0 approved

Prepared by Tejas Gupta, Marina Mancoridis, Nico Müller, Armin Riess, Mike Schmid and Robin Sieber

Ship Happens

17.03.2023

Table of Contents

Revision History

Name	Date	Release Description	Version
Felix Friedrich	03/17/23	Template for Software Engineering Course in ETHZ.	0.3
Ship Happens	03/17/23	Adapted to Battleship project	1.0

Introduction

Purpose

Document Conventions

Intended Audience and Reading Suggestions

Product Scope

References

Overall Description

Product Perspective

Product Functions

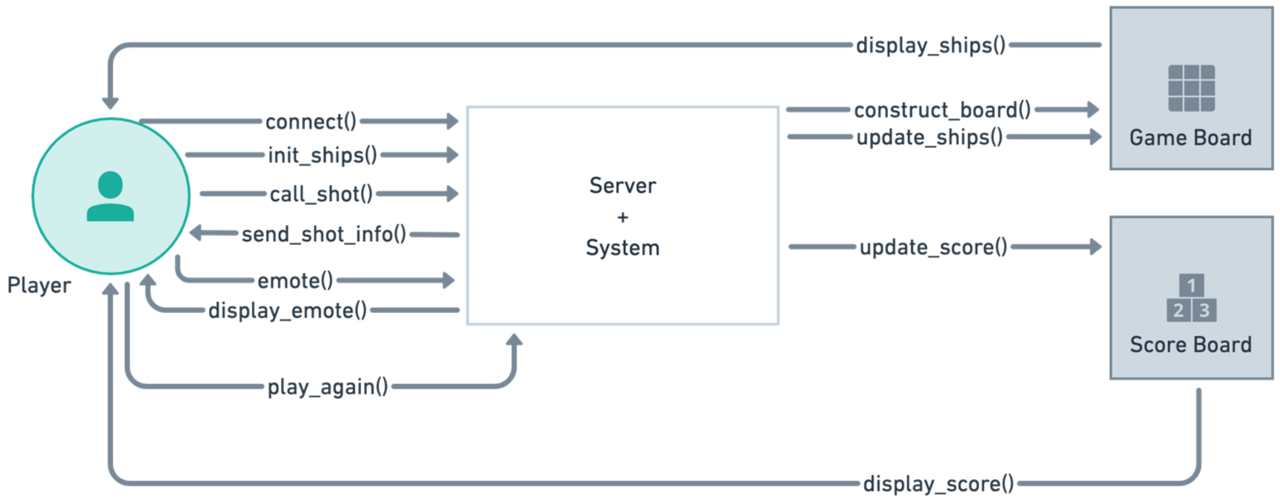


Figure 1: Top level data flow

This product is intended to enable exactly two players to compete online in the classic boardgame Battleship. The list below highlights the product’s major functions:

- **Starting the game server**: Communication between the two players will be enabled when both of their game clients are connected through the game server.
- **Initializating the game**: When the game initializes, each player will have a Gameboard object associated with them. The Gameboard is meant to model a Battleship game unit. Meta-data about the status of the game will be available to both players. This information will include the total number of ships sunk by each player, the cumulative number of hits of each player, and the player names. The game will begin with no ships on the board and and no pegs in play.
- **Playing the game**: Players will communicate through the server and play the game of Battleship. They will take turns guessing ship locations. Throughout the game, the cumulative scores will be displayed to both players and player Gameboards will be updated. The software will enable only valid moves. When the game ends, the winner will be indicated to both clients.
- **Ending the game**: Players will be given the choice to play another round (*return to second bullet point*) or end the game (*terminating the connection*).

User Classes and Characteristics

This implementation of Battleship is intended for two players, ages seven and up. These players have to either already know the rules of the game or be able to interpret the game’s instructions during gameplay. They need to be able to see, read, and click. They do not need any experience using other products, special security or privilege levels, or special education levels. They need to be able to run the game’s executable file and interact with the game’s GUI. No product function will disabled for specific players of the game.

Operating Environment

This game can run on any UNIX-based operating system.

Design and Implementation Constraints

User Documentation

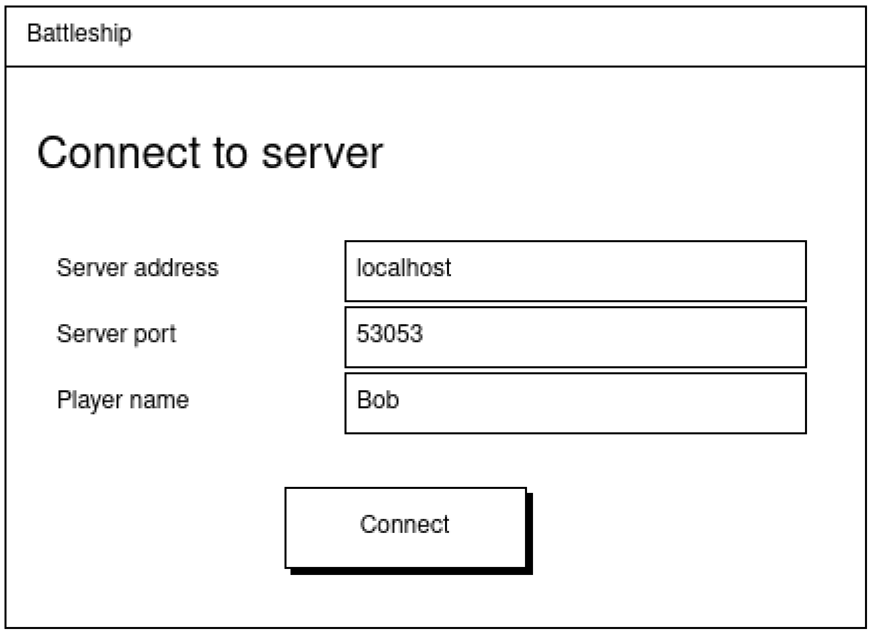
Assumptions and Dependencies

External Interfaces and Requirements

User Interface

The users will interact with the software via GUI. This GUI includes three main views:

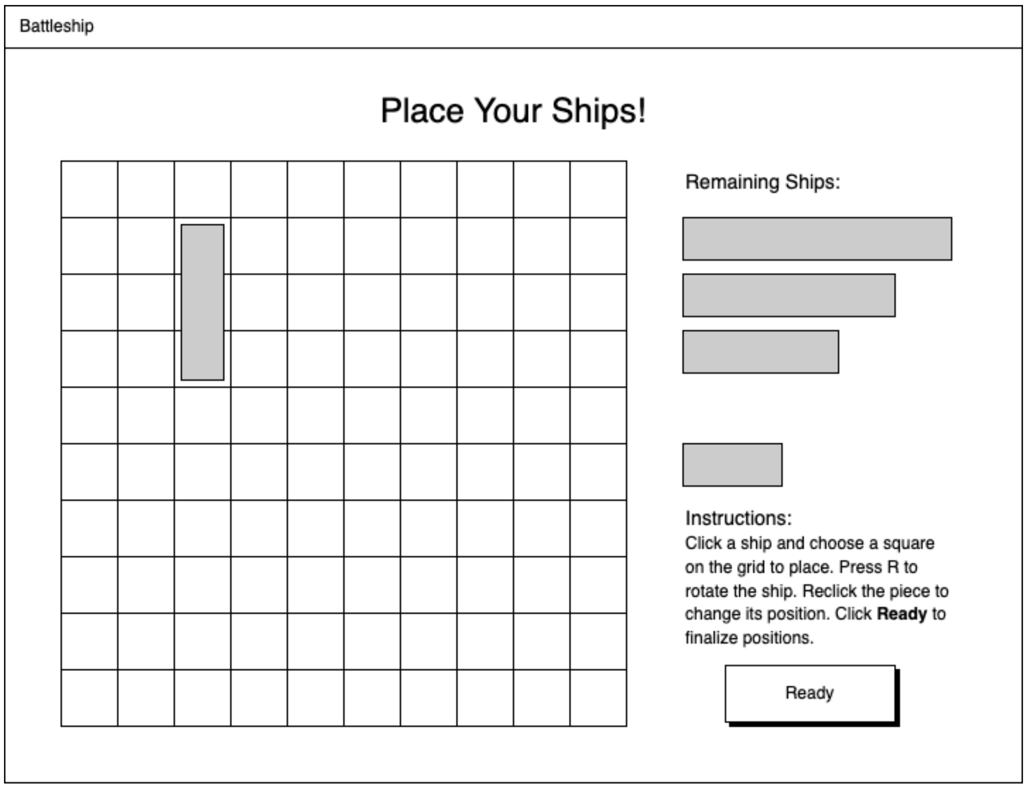
1. *A Login screen:*
   - i. *Visible when starting the game client*
   - ii. *Includes input fields for connection setup to the game server and a field to choose a username*
   - iii. *Features a button to join a game*
2. *The Setup screen:*
   - i. V*isible when successfully joined a game*
   - ii. *Allows player to place its battleships on the grid*
   - iii. *Displays own battlefield as a grid and remaining ships ot place*
   - iv. *Shows a "ready" button that can be presssed as soon as all ships are placed*
3. *The game screen:*
   - i. *Visible when a player joined a game successfully and completed the setup.*
   - ii. *Shows own battlefield and an initially empty one of the opponent*
   - iii. *Also displays scoreboard with information about remaining ships and number of successful hits*
4. *The end screen:*
   - i. *Visible when the game ends (sucessfully)*
   - ii. *Displays which user won and congratulates the winner*
   - iii. *Asks to restart game or end game*


Battleship

Connect to server

Server address	localhost
Server port	53053
Player name	Bob

Figure 2: Login screen



Click a ship and choose a square on the grid to place. Press R to rotate the ship. Reclick the piece to change its position. Click **Ready** to finalize positions.

Figure 4: Setup screen

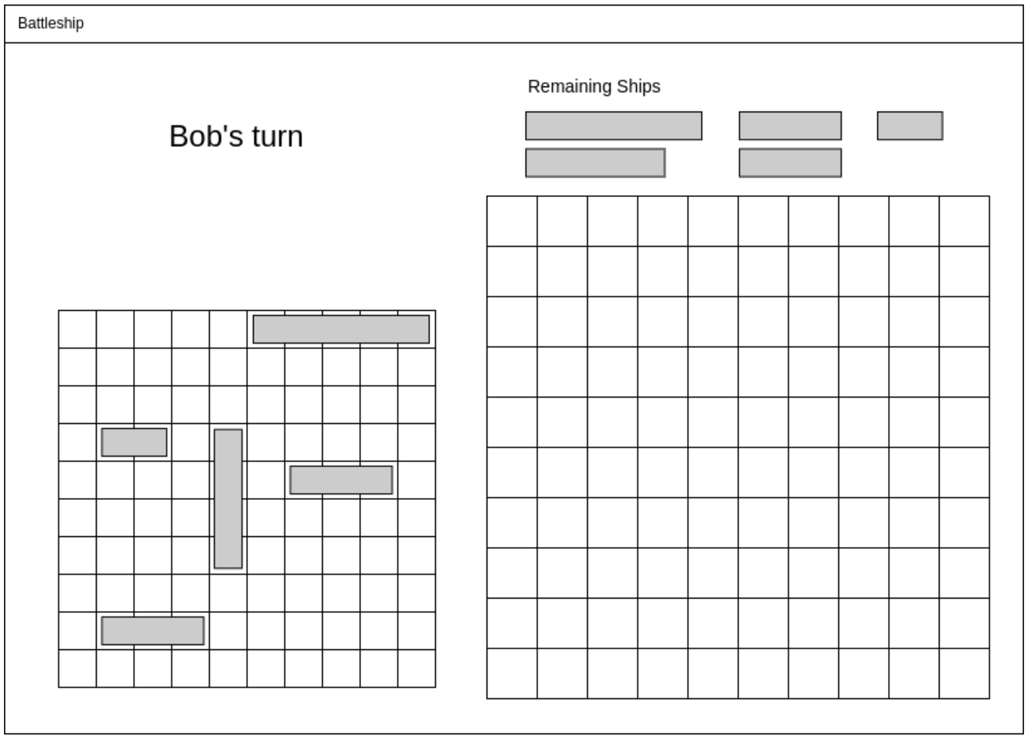


Figure 3: Game screen

Software Interfaces

Communications Interfaces

The system will implement client-server communication between one server and two clients. Server client communication will use TCP to ensure correct, reliable and in-order communication between both parties. The data will be encoded in the JSON format.

System Requirements

Functional Requirements

FREQ-1: Connecting clients

The system should be able to provide a connection between two clients through a server.

Description: *The server should provide a connection between the endhosts of both participating clients and allow users to connect only if there are not already two people connected. If there are sufficient players in the server, an error message should be displayed.*
User Priority(5/5): *This is mandatory for all users.*
Technical Priority(3/5): *This requirement is mandatory for all parts of the application that require communication between the clients, including all gameplay. One server and port is considered one lobby or one game since Battleship is a two-player game, but other technical implementations are possible.*

FREQ-2: Starting the Game

The allow players to join the game with their name and bring them to the setup screen.

Description: *To join the game, a player must insert their name after the connection info and click "Connect."*
User Priority(5/5): *This is mandatory for all users.*
Technical Priority(5/5): *This requirement is mandatory for all parts of the application within the game.*

FREQ-3: Initializing states and objects for each player

The system must associate initial states and objects with each player.

Description: *The system will associate a Gameboard object associated with each player. The Gameboard is meant to model a Battleship game unit. It will include five "plastic" ships (the destroyer, submarine, cruiser, battleship, and carrier), 42 red "hit" pegs, 84 white "miss" pegs, and ocean grid, and a target grid. Each grid will be empty. The gameboard will always be visible to the player. It will change states as the game progresses.*
User Priority(5/5): *This is mandatory. All users need a Gameboard to play the game.*
Technical Priority(5/5): *This is mandatory for all future game states.*

FREQ-4: Initializing states for the collective game

The system should associate states with the entire game during gameplay.

Description: *These states will include the total number of ships sunk by each player, the cumulative number of hits of each player, and the player names. The game will begin with no ships on the board and and no pegs in play. As the game is played, these shared statistics will be updated and will be viewable for all players.*
User Priority(2/5): *This is not strictly necessary to play the game. Users technically only need to see their own gameboard, not keep track of the total progres of the game. But, it will help users keep track of the game's progress.*
Technical Priority(2/5): *This is not strictly necessary to play the game, but will be of benefit to the players.*

FREQ-5: Keeping track of player turn

The system should keep track of whose turn it is so that a player cannot make two consecutive turns.

Description: *A player must make a move on their turn and cannot play two consecutive turns. The system will disable invalid turn sequencing by forbidding players from submitting empty turns or taking consecutive turns.*
User Priority(5/5): *Turn structure is a key component of Battleship gameplay and the game cannot be played appropriately without it.*
Technical Priority(2/5): *Technically, the system itself does not have to keep track of player turns; players can manually avoid submitting empy turns or consecutive turns. However, this feature will help the system safeguard against invalid gameplay.*

FREQ-6: Placing Ships during Battle Preparation

The system should allow the two players to concurrently place ships on their own ocean grid before the main phase of the game begins.

Description: *Each player, potentially at the same time, will place their five ships on their own ocean grid. The system should allow the player to select a ship and drag or place it onto their ocean grid. It should also allow the player to switch between horizontal and vertical orientation by pressing the "R" button on the keyboard. The system will ensure that no ships overlap, that all ships are within the grid of the board, and that all ships are placed either horizontally or vertically. The player should be able to again change the position of a ship after placing it down. The system will not allow ship placement to be changed once the game has begun, and will inform users of this.*
User Priority(5/5): *The selection of ship positions is a central component of the gameplay. The game cannot be played without this process.*
Technical Priority(5/5): *Again, the game cannot be played without this process.*

FREQ-7: Ending Battle Preparation Stage

The system should allow players to indicate when they have finished placing their ships in their initial positions and wait until the other user has as well.

Description: *Players should have the ability to submit their initial ship positions, indicating that they are ready to proceed with the game. Once both players have submitted their ship positions, the system will know that gameplay is ready to begin.*
User Priority(5/5): *Users need to be able to indicate when they are ready to play. They also need to know when their opponent is ready to begin playing the game.*
Technical Priority(5/5): *Further gameplay is not possible unless players submit initial positions.*

FREQ-8: Deciding who will go first

The system should randomly choose which player will have the first turn.

Description: *The system will randomly select a player to start first. Once it has done so, it will indicate who the starting player is to both players. The gameboard of the player who is not starting first will be frozen until it is their turn.*
User Priority(2/5): *This is not explicitly necessary for the game to run. Users could technically decide on their own or rely on some sort of other non-random system.*
Technical Priority(2/5): *Again, this is not explicitly necessary for the game, but it provides a sense of fairness to the product.*

FREQ-9: Calling a shot

The system should allow players to call a shot on their turn. Calling a shot means that a player makes a guess as to where a battleship lies on their opponent's ocean grid.

Description: *The player will call a shot by specifying a coordinate location on their target grid. For example, a player could specify the coordinate "A1" to guess that a battleship lies on the top leftmost tile of their opponent's ocean grid.*
User Priority(5/5): *It is impossible to play the game without this feature.*
Technical Priority(5/5): *It is impossible to play the game without this feature.*

FREQ-10: Keeping the scoreboard updated

The system should always make sure that the scoreboard is properly updated.

Description: *As described earlier (FREQ-4), the scoreboard consists of the following information: total number of ships sunk by each player, the cumulative number of hits of each player, and the player names. Upon completion of each turn, the system should update the scoreboard appropriately. The scoreboard's current state should always be visible to both players.*
User Priority(1/5): *Although helpful, the scoreboard is not essential for gameplay. Users can keep track of this information on their own or infer it from the current state of their Gameboard.*
Technical Priority(1/5): *This is not essential to play the game and is only a feature added for user convenience and product expressiveness.*

FREQ-11: Encountering a hit

The system should notify players when a ship has been hit.

Description: *When a player accurately calls a hit on an enemy ship, the system should update the game board associated with the opponent to indicate that the targeted tile has been hit by placing a red "hit" peg on the corresponding coordinate on the ocean grid. The system should also notify the player that they have hit an enemy ship. The system should also update the scoreboard by increasing the number of hits of the relevant player. The system should also prompt the user to call another shot.*
User Priority(4/5): *It is critical to the game of Battleship that players are able to accurately track hits on enemy ships as that informs the player on their progress in the game and impacts their playing strategy. However, users can also keep track of hits on their own after being informed of their success.*
Technical Priority(4/5): *Accurately tracking hits on enemy ships is a key feature of the game, and is necessary for gameplay to function correctly.*

FREQ-12: Encountering a miss

The system should notify users when they have missed their shot.

Description: *When a player does not accurately call a hit on an enemy ship, the system should update the game board associated with the opponent to indicate that the targeted tile has been missed by placing a white "miss" peg on the corresponding coordinate on the ocean grid. The system should also notify the player that they have missed, and switch to the opponent's turn.*
User Priority(5/5): *It is critical to the game of Battleship that players are able to accurately track misses on enemy ships.*
Technical Priority(4/5): *Accurately tracking misses on enemy ships is a key feature of the game, and is necessary for gameplay to function correctly.*

FREQ-13: Sinking a ship

The system should notify players when a ship has been sunk.

Description: *When a player accurately hits all of the tiles on an enemy ship, the system should update the game board associated with the opponent to indicate that the ship has been sunk by placing additional red "hit" pegs on the corresponding coordinates for the remaining parts of the ship. The system should also notify the player that they have sunk an enemy ship, and update the remaining number of ships that the opponent has in play. If this is the last ship that the opponent has in play, the system should end the game and declare the current player as the winner.*
User Priority(5/5): *Sinking ships is a key gameplay mechanic in the game of Battleship, and is critical for players to be able to achieve victory.*
Technical Priority(5/5): *Sinking ships is a core feature of the game, and is necessary for gameplay to function correctly.*

FREQ-14: Winning the game

The system should track whether a user has sunk all of their enemy's ships, notify the players, and end the game.

Description: *The system should end the game and declare the current player as the winner when that player successfully sinks all of the ships on the opponent's game board. The system should display a victory message to the winning player and a defeat message to the losing player. The system should also provide an option to start a new game.*
User Priority(5/5): *Winning the game is the ultimate goal of playing Battleship, and is critical for player engagement and satisfaction.*
Technical Priority(5/5): *Winning the game is the end goal of the game, and is necessary for gameplay to function correctly.*

FREQ-15: Playing again

The system should allow users to start game play again after a game has ended.

Description: *After a game has ended, the system should provide an option to start a new game. This option should be clearly visible and easily accessible to the player. When the player selects this option, the system should reset the game board, clear all pegs from the previous game, and randomize the location of the ships. The system should also allow the player to choose to play against the same opponent or a new opponent.*
User Priority(3/5): *Providing the ability to play again is important for player satisfaction and engagement, but is not essential to the core gameplay experience.*

Technical Priority(2/5): *Providing the ability to play again is a desirable feature, but is not critical to the basic functionality of the game.*

FREQ-16: Response to abrupt player exit

The system should handle a player exiting before the game has finished without crashing.

Description: *If a player exits the game abruptly, such as by closing the browser or turning off the device, the system should handle the situation without crashing. The system should save the current state of the game and notify the other player that their opponent has disconnected. The system should allow the remaining player to continue the game in single-player mode, or exit the game and return to the main menu.*
User Priority(3/5): *Handling abrupt player exits is important for providing a good user experience, but is not critical to the core gameplay experience.*
Technical Priority(1/5): *Handling abrupt player exits is a desirable feature, but is not critical to the basic functionality of the game.*

FREQ-17: Terminating connection through the server

The system should handle a poor or terminated connection without crashing.

Description: *The system should provide a mechanism for gracefully terminating the connection between players in the event of unexpected errors or other issues. If the system detects that a player has lost connectivity or has exited the game unexpectedly, the system should notify the other player and terminate the connection. The system should also provide a mechanism for players to manually terminate the connection, such as through a "forfeit" button or menu option.*
User Priority(1/5): *Providing a mechanism for gracefully terminating connections is important for providing a good user experience, but is not critical to the core gameplay experience.*
Technical Priority(3/5): *Providing a mechanism for gracefully terminating connections is a desirable feature, but is not critical to the basic functionality of the game.*

FREQ-18: Responding to user errors regarding ship placement

The system should only allow valid ship placement.

Description: *The system should validate the placement of ships on the gameboard to prevent user errors, such as overlapping ships or placing ships outside of the gameboard boundaries. If a user attempts to place a ship in an invalid position, the system should provide a visual indication of the error, such as highlighting the invalid area or displaying an error message. The system should prevent the user from placing the ship in the invalid position and prompt the user to choose a valid position.*
User Priority(3/5): *Validating ship placement is important for preventing user errors and ensuring a fair gameplay experience but users can also be careful to not make errors.*
Technical Priority(4/5): *Validating ship placement is a necessary feature for preventing game-breaking errors and ensuring the stability of the game.*

FREQ-19: Performing validity checks on every game state

The system should always have a consistent game state. The system should validate that no illegal moves or states are being created.

Description: *The system should perform validity checks on every game state to ensure that the game is progressing according to the rules and that no illegal moves or states are being created. These checks should include verifying that ships are not overlapping, that shots are being taken on valid positions, that a player cannot take multiple turns in a row, and that the game is not over prematurely. If the system detects an*

invalid game state, it should provide a visual indication of the error and prompt the user to take appropriate action
User Priority(5/5): *Ensuring that every game state is valid is critical to maintaining the integrity and fairness of the game.*
Technical Priority(5/5): *Performing validity checks on every game state is a necessary feature for preventing game-breaking errors and ensuring the stability of the game.*

FREQ-20: Game GUI

The system should provide a GUI during the whole game for each user, displaying the game state (players, game board and score board) and moves.

Description: *The system should provide a graphical user interface (GUI) for the Battleship game that displays the gameboard, ships, pegs, and other game elements in a visually appealing and user-friendly manner. The GUI should allow the user to interact with the gameboard by clicking on grid cells to place ships and call shots, and by displaying feedback on game events such as hits, misses, sunk ships, and the end of the game. The GUI should also provide menus or buttons for starting a new game, saving and loading games, and adjusting game settings.*
User Priority(5/5): *A visually appealing and user-friendly GUI is essential for providing a satisfying and enjoyable gaming experience as well as keep track of the game state.*
Technical Priority(3/5): *Implementing a GUI for the Battleship game is a nice feature for providing a user-friendly and intuitive interface for interacting with the game elements, but there are other technical implementations such as text-based.*

FREQ-21: Basic player-player communication

The system should allow players to have basic interpersonal communication, such as through the use of emotes.

Description: *The system should have a mechanism to facilitate interpersonal communication through prewritten dialogue.*
User Priority(1/5): *Online player to player communication is unnecessary for the gameplay.*
Technical Priority(1/5): *Implementing emotes is not necessary for the basic functionality of gameplay.*

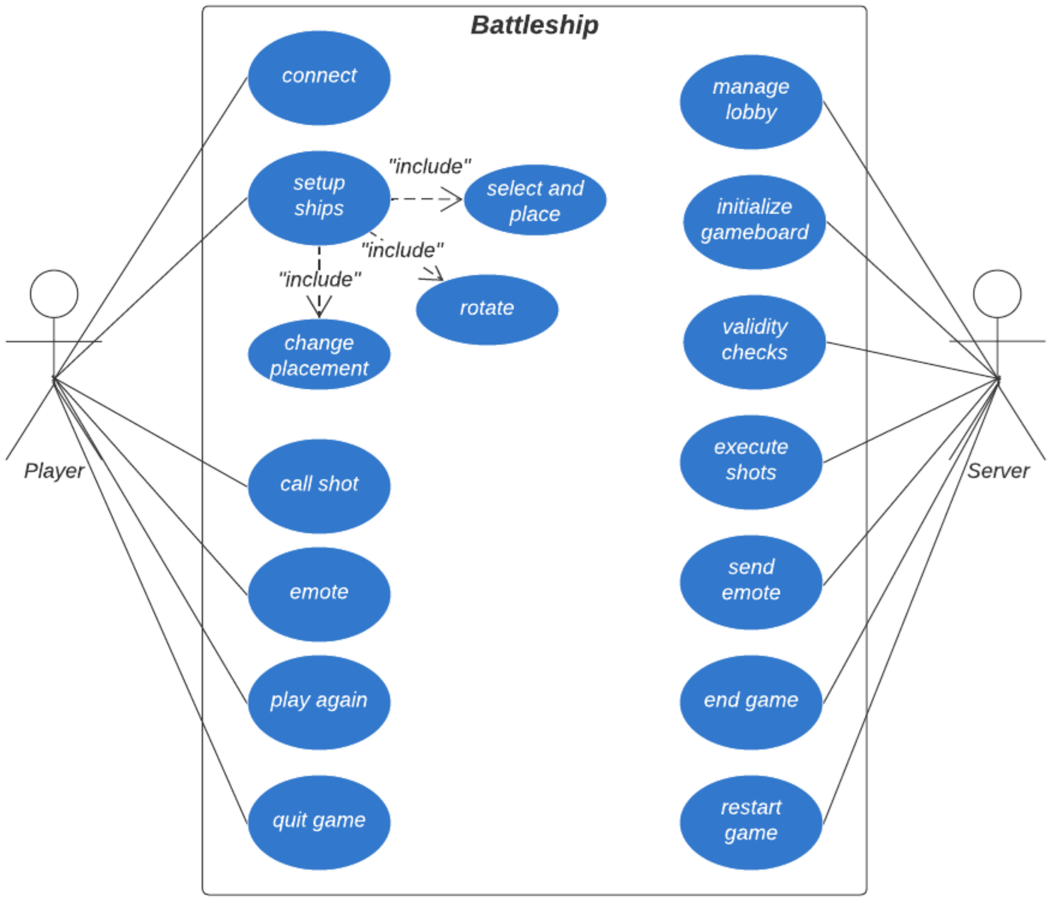
FREQ-22: Adhering to appropriate external interfaces

The system should adhere to appropriate external interfaces for communicating with other software systems or external devices.

Description: *The system should conform to industry standards and procools for data exchange and also provide error handling and logging mechanisms to detect and report any issues with external interface communication to allow product and interface extensibility.*
User Priority(2/5): *While users will not directly interact with external interfaces for gameplay, some importance is placed to ensure smooth gameplay experience on external devices or software systems.*
Technical Priority(4/5): *Adhering to appropriate external interface specifications is a critical feature for ensuring the system can communicate with other software systems in a consistant, reliable, and secure manner.*

System Scenarios

Use-case Diagrams

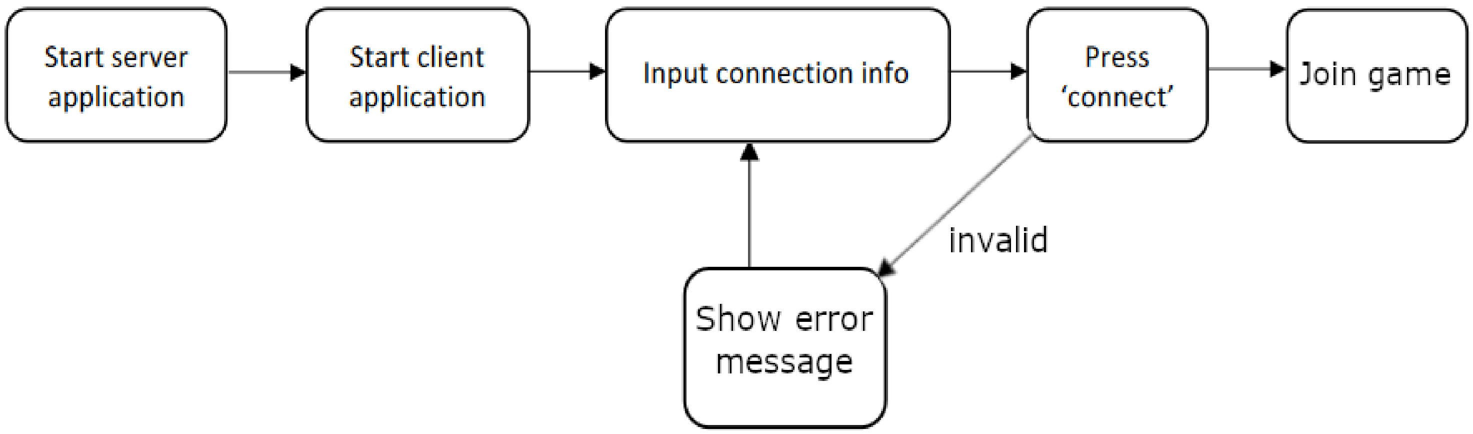


Scenarios

SCN-1: Setting up a game

FREQ reference	*1, 2, 3, 19, 20*	
NFREQ reference	*1, 3, 5, 6*	
Short Description:	*Alice and Bob want to play a game of battleship. Alice starts the game server. Afterwards, both players start their game clients, fill in the connection information and chose a username. When ready, they press the 'connect' button.*	
Activation action:	*Alice starts the game server; Alice and Bob start their clients.*	
Precondition:	*Both players have Battleship installed on their computer and have a stable internet connection.*	
Basic flow: Setting up a game		
Step	**User action**	**System response**
1	*Alice runs the server application.*	*Starts server, displays the connection information (IP and port).*
2	*Players run the client application.*	*Starts clients, displays the login screen.*
3	*Players fill in connection information.*	
4	Players press 'connect'.	*Connects clients to the server, displays setup screen.*
Post-condition:	*Both players connected to the server and are now in the setup phase.*	

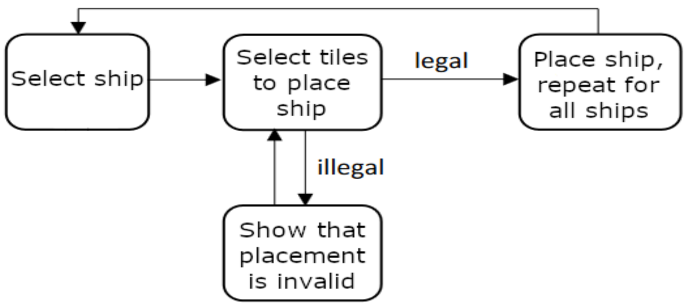
Scenario Diagram for SCN-1 Setting up a game



SCN-2: Placing the ships

FREQ reference	*3, 6, 7, 16, 17, 18 , 19*	
NFREQ reference	*1, 3, 4, 5, 6*	
Short Description:	*Alice connected to the server. She now can place her 5 ships in any order by placing them on her ocean grid. She tries to place a ship such that it overlaps with one she has already placed, the game does not let her do that and indicates visually that this action is illegal. She chooses a different tile and the ship is placed successfully. After she is done placing all her ships, she presses 'ready'.*	
Activation action:	*Automatic transition when connecting to the server.*	
Precondition:	*Alice has successfully connected to the same Battleship server and is in the setup phase.*	
Basic flow: Placing the ships		
Step	**User action**	**System response**
1	*Alice selects the cruiser on the right hand side.*	*Shows the selected cruiser ship next to the mouse cursor.*
2	*Hovers the cursor over an already occupied tile on the ocean grid.*	*Determines that this tile is not a possible option. The cruiser ship preview changes its appearance to indicate this. Disables the ability to place the ship down.*
3	*Hovers the cursor over a different tile instead.*	*Determines that this tile is a valid option. Indicates this again through the appearance of the ship preview.*
4	*Switches rotation of the cruiser between vertical and horizontal by pressing "R" on the keyboard*	*Determines that this rotation is valid (every occupied tile within the grid and not already covered by another ship). Indicates this again through the appearance of the ship preview.*
5	*Clicks left mouse button.*	*Places the ship on the selected position. Updates the Gameboard.*
6	*Repeats these steps for the remaining 4 ships.*	*Repeats*
7	*Presses the 'ready' button.*	*Ends the setup phase for this user. Saves the ship formation in the Gameboard.*
Post-condition:	*Alice can no longer interact with the setup screen. She is marked as ready.*	

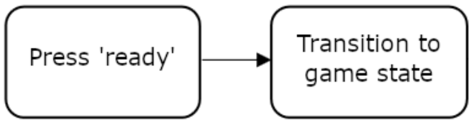
Scenario Diagram for SCN-2 Placing the ships


Select ship
Select tiles to place ship
legal
Place ship, repeat for all ships
illegal
Show that placement is invalid

SCN-3: Starting a game

FREQ reference	*3, 4, 5, 7, 8, 10, 16, 17, 18, 19*	
NFREQ reference	*1, 3, 5, 6*	
Short Description:	*Alice and Bob both pressed 'ready' on their setup screens. The system then starts the game. Both players now see the game screen with their own ships placed as chosen during the setup phase. The system randomly decides who will go first and starts this players turn.*	
Activation action:	*Bob presses 'ready'.*	
Precondition:	*Both players have placed all their ships in a valid formation. Alice has already pressed the 'ready' button before and is marked as 'ready' by the server.*	
Basic flow: Starting a game		
Step	**User action**	**System response**
1	*Bob presses 'ready'.*	*System sees that now both players are ready and transitions to game state. Displays game screen to players. Randomly determines who will go first.*
Post-condition:	*It is now Alice's turn.*	

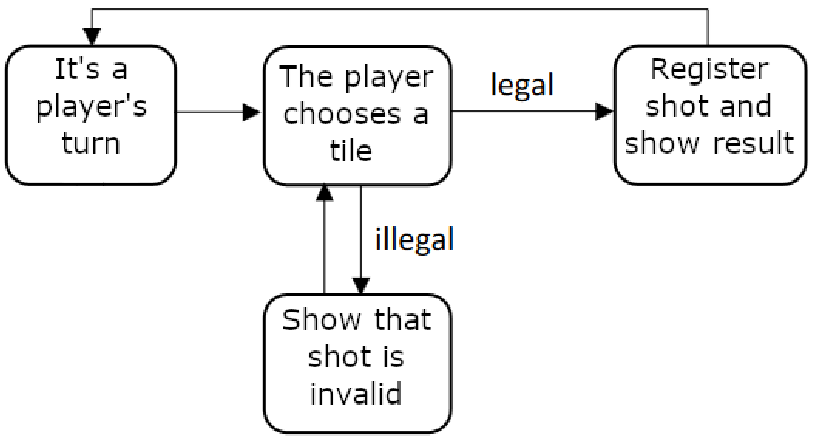
Scenario Diagram for SCN-3 Starting a game



SCN-4: Calling an unsuccessful shot (miss)

FREQ reference	*9, 10, 12*	
NFREQ reference	*1, 2, 3, 4, 5, 6*	
Short Description:	*It's Alice's turn. She tries to shoot a tile she has already shot at. The game shows that this isn't a valid input and lets Alice try again. She chooses a different tile, one she hasn't shot at before. This time the shot is executed. The shot misses. The missed shot is indicated on the game screen.*	
Activation action:	*It is Alice's turn.*	
Precondition:	*Alice and Bob are playing a game of Battleship. It's Alice's turn.*	
Basic flow: Calling am unsuccessful shot (miss)		
Step	**User action**	**System response**
1	*Alice clicks on a tile she has shot at before.*	*Determines that this is an illegal move, and shows this visually. The input is ignored and the system waits for a valid shot.*
2	*Alice clicks on a different tile that she hasn't shot at before.*	*Registers the shot and displays it for both players on the correct grid.*
3		*Indicate to both players that the shot was a miss.*
4		*Register that since Alice missed, it is now Bob's turn.*
Post-condition:	*It is now Bob's turn.*	

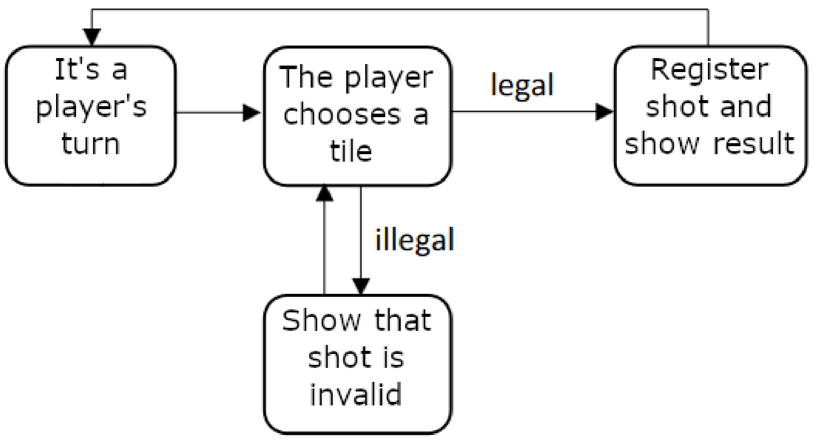
Scenario Diagram for SCN-4 Calling an unsuccessful shot (miss)


It's a player's turn
The player chooses a tile
legal
Register shot and show result
illegal
Show that shot is invalid

SCN-5: Calling a successful shot (hit)

FREQ reference	*9, 10, 11, 13*	
NFREQ reference	*1, 2, 3, 4, 5, 6*	
Short Description:	*It's Alice's turn. She chooses a tile she hasn't shot at before. The shot hits one of Bob's ships and sinks it. Since Alice hit a ship, it's still her turn.*	
Activation action:	*It's Alice's turn.*	
Precondition:	*Alice and Bob are playing a game of Battleship. It's Alice's turn and she only needs one shot to sink one of Bob's ships*	
Basic flow: Calling a successful shot (hit)		
Step	**User action**	**System response**
1	*Alice clicks on a tile that she hasn't shot at before.*	*Registers the shot and displays it for both players on the correct grid.*
2		*Indicate to both players that the shot hit one of Bob's ships and sank it.*
		Show Alice which type of boat she sank and update the list she sees of Bob's ships that are still swimming. Update the scoreboard.
3		*Register that since Alice hit a ship, it's still her turn.*
Post-condition:	*It's still Alice's turn.*	

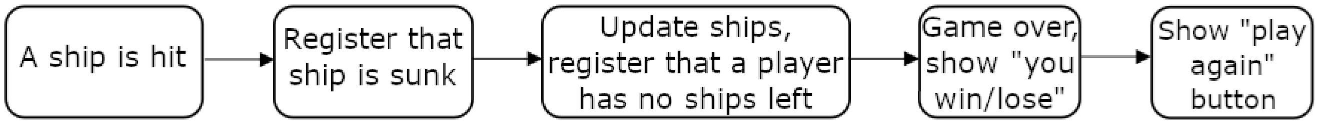
Scenario Diagram for SCN-5 Calling a successful shot (hit)


It's a player's turn
The player chooses a tile
legal
Register shot and show result
illegal
Show that shot is invalid

SCN-6: Ending a game

FREQ reference	*9, 10, 11, 13, 14*	
NFREQ reference	*1, 3, 5, 6*	
Short Description:	*Alice and Bob are playing a game of Battleship. Alice sinks the last of Bob's five ships. The game ends with Alice winning and Bob losing. Then, Bob closes the game client.*	
Activation action:	*Alice sinks Bob's last ship.*	
Precondition:	*Alice and Bob are playing a game of Battleship. Alice has sunk four of Bob's ships and only needs one more hit to win.*	
Basic flow: End of the game		
Step	**User action**	**System response**
1	*Alice hits the last part of Bob's last ship*	*Shows where Alice placed her shot and that this sank Bob's ship .*
2		*Updates number of ships that are still swimming, determines that Bob has no ships left and that the game is over.*
3		*Shows "you win" to Alice and "you lose" to Bob. Display a button 'play again' to both.*
4	Bob closes the game client.	*Closes Bob's client, all other instances of the system keep running.*
Post-condition:	*Alice sees the "you win" message.*	

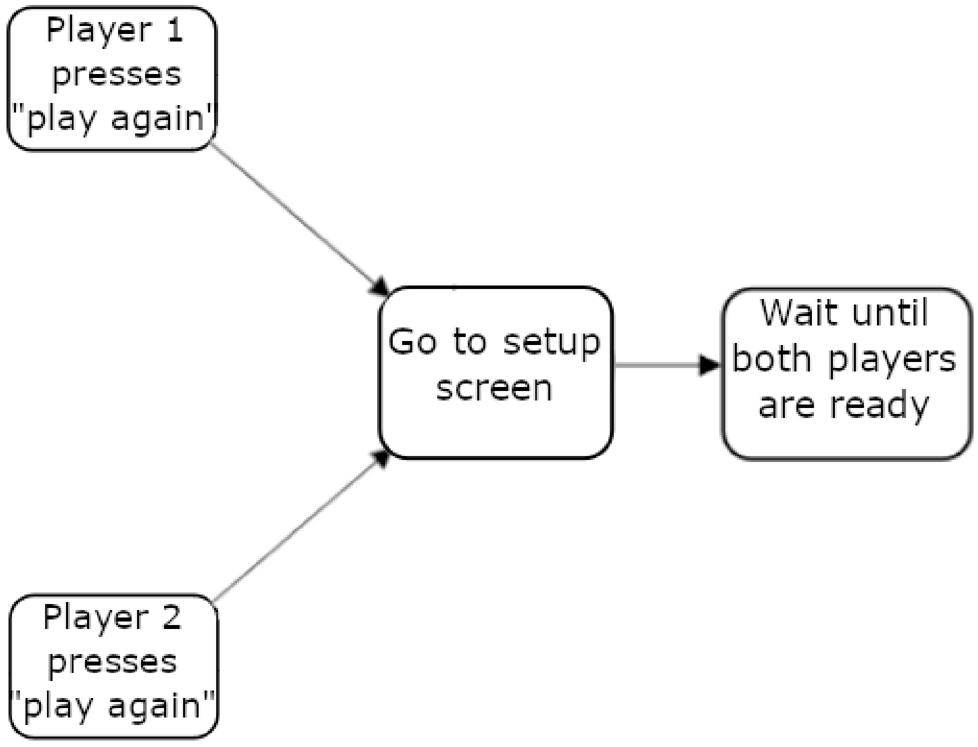
Scenario Diagram for SCN-6 Ending a game


A ship is hit
Register that ship is sunk
Update ships, register that a player has no ships left
Game over, show "you win/lose"
Show "play again" button

SCN-7: Playing again

FREQ reference	*3, 4, 10, 15, 16, 19*	
NFREQ reference	*1, 2, 3, 5, 6*	
Short Description:	*Alice and Bob finished playing a game of Battleship. Both players see the end screen and decide to press ‘play again’. The Gameboard resets and both players are directly put back into the setup phase to chose a new formation for their ships. They see the setup screen.*	
Activation action:	*Both players pressed ‘play again’.*	
Precondition:	*Alice and Bob finished a game of Battleship.*	
Basic flow: Playing again		
Step	**User action**	**System response**
1	*Alice presses ‘play again’.*	*The system marks Alice as ‘wants to play again’*
2	*Bob presses ‘play again’.*	*Sees that both players want to play again. Resets the Gameboard. Displays the setup screen to both players.*
		Waits for the players to complete their setup and to press ‘ready’.
Post-condition:	*Alice and Bob are again in the setup phase (SCN-2).*	

Scenario Diagram for SCN-7 *Playing again*


Player 1 presses "play again"
Go to setup screen
Wait until both players are ready
Player 2 presses "play again"

System Constraints

Important Nonfunctional Requirements

NFREQ-1: Language

The system should display the texts in English language.

User Priority(3/5): *Allows for a wide user range to play the game*
Technical Priority(1/5): *System is not dependent on NFREQ-1 to work*

NFREQ-2: Performance

The waiting time for a player to be able to make a move after the opponent finished his turn should no be longer than a second.

User Priority(4/5): *Games that have long waiting times are less likely to be played. This NFREQ has a strong influence on the user experience.*
Technical Priority(2/5)*: This is not a hard requirement for the system to work.*

NFREQ-3: User interface simplicity

The user interface of the client should be self-explanatory s.t. no extra tutorial page or similar is needed.

User Priority(4/5): A simple user interface is key to get users started and keep them playing.
Technical Priority(1/5)*: This is not a necessary technical requirement.*

NFREQ-4: Userface memorability

The diagrams for the ships and symbols on the grid should be easy to understand even when you the play the game after a long time again.

User Priority(3/5): It is in our interest to keep user frustration (e.g. caused by an unnecessarily complicated interface for a simple game) low.
Technical Priority(1/5)*: This is not a necessary technical requirement.*

NFREQ-5: Reliability

The server should not crash and the clients should not lose connection to the server.

User Priority(5/5): *Essential to be able to play the game.*
Technical Priority(5/5)*: Essential for a working client-server communication.*

NFREQ-6: Failure management

The system should have a failure management such that in case of an error, the clients and the server can be simply restarted and a new game can be started.

User Priority(3/5): *Even though we strive for a reliable game, the user should be able to restart clients and server in case of technical difficulties.*
Technical Priority(4/5)*: It is necessary that new clients and server can be started as usual even if a previous game encountered a technical failure.*

Other Requirements

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To Be Determined List