Requirement Engineering: MoSCoW

Functional Requirements

For our game Naval Battle the requirements regarding functionality and service are grouped under the Functional Requirements and Non-Functional Requirements. The functional requirements are grouped into four categories using the MoSCoW model for prioritizing requirements.

1.1 MUST HAVES

- The player shall be able to play against a remote.
- The game shall ask the player to authenticate himself with a username and hashed password before starting the game.
- The game shall only start when the credentials of the player correspond with the credentials stored in the database.
- The game shall not start a new game for a player when the credentials are not found in the database.
- The game shall allow a new player to register.
- The game shall show an empty board before a new game starts.
- The board of the game shall be a grid with a width of 10 cells and a height of 10 cells.
- The player shall be able to see the board of the opponent and the board of the user side by side.
- The 5 ships shall be:
    - One ship called *Carrier* who occupies 5 cells
    - One ships called *Battleship* who occupies 4 cells
    - One ships called *Cruiser* who occupies 3 cells
    - One ship called *Submarine* who occupies 3 cells
    - One ships called *Destroyer* who occupies 2 cell
- The game shall offer the 5 ships with a specified size and orientation upon starting a new game.
- The game shall be able to generate a board for the remote with random places for the ships keeping the constraint that there shall be at least one block between each ship.
- The player shall be able to drag the 5 ships on the board when starting a new game.
- The game shall not allow the player to place ships outside the board.
- The game shall be able to detect and reject if there is not at least one block between each ship during the placement of the ships on the board by the player.
- The game shall be able to detect and reject if not all 5 ships are present on the board during the placement of the ships on the board by the player.
- The player shall be able to click an untouched cell on the board when it shall be the players turn.
- The game shall not allow a player to click an untouched cell on the board when it shall not be the players turn.
- The game shall not allow a player to click an touched cell on the board regardless whether it shall be the players turn.
- The game shall be able to verify whether a cell, containing (a piece of) a ship, is touched.

- The game shall be able to let the opponent click an untouched cell when the player has not touched a cell containing (a piece of) a ship.
- The game shall be able to let the player click an untouched cell when the player has touched a cell containing (a piece of) a ship.
- The game shall be able to verify that all the cells of a ship are touched.
- The game shall initiate and show the player's & opponent's score at 0.
- The game shall be able to display the amount of points an player has achieved during the game.
- The player shall win when the player has performed a click event on all the ships of the opponent before the opponent has performed a click event on all the ships of the player.
- The player shall lose when the player has not performed a click event on all the ships of the opponent while the opponent has already performed a click event on all the ships of the player.
- The player shall be able to enter his recorded score after completion of a game regardless whether the player has won or lost.
- The game shall be able to store the new score in a database.
- The game shall be able to display the highest 5 scores in the database. (High scores / Top 5)

1.2 SHOULD HAVES

- The game shall mark the cells on which a click event is detected and do not contain (a piece of) a ship with a grey block.
- The game shall mark the cells on which a click event is detected and do contain (a piece of) a ship with that discovered (piece of the) ship.
- The game shall be able to mark the cells with a grey block around the cells containing the ship when on all the cells containing the ship a click event has occurred.
- The player shall be able to start a new game of Naval Battle via a button.
- The player shall be able to stop a game of Naval Battle that is currently in progress.
- The game shall end a game when the player loses the game or stops it.
- The player shall be able to see the click events its opponent is performing.
- The game shall provide a button to show the rules for the placement of the ships.
- The game shall provide a button to show the instructions on how to play the game and how the points are assigned.
- The game shall be able to detect on which part of the ship a click event has occurred.
- The game shall keep track of the player's score using information about the weak places of the ship. The points of a ship divided by the total amount of click events on cells not containing (a piece of) a ship (*number of misses*) during the game shall be assigned to the player when on the whole ship click events have occurred:
    - The ship called *Carrier* shall be worth in total 5000 points. When the first hit is on the front or on the back of the ship 2000 points shall be deducted. When the first hit is on the right or on the left from the middle of the ship 1000 points shall be deducted. When the first hit in exactly in the middle of the ship no points shall be deducted.
    - The ship called *Battleship* shall be worth in total 4000 points. When the first hit is on the front of the ship no points shall be deducted. When the first hit is on the back of the ship 1000 points shall be deducted. When the first hit is on the right or on the left of the middle of the ship 2000 points shall be deducted.

    - The ship called *Cruiser* shall be worth in total 3000 points. When the first hit is on the front of the ship 2000 points shall be deducted. When the first hit is on the back of the ship 1000 points shall be deducted. When the first hit is on the middle of the ship no points shall be deducted.
    - The ship called *Submarine* shall be worth in total 3500 points. When the first hit is on the front of the ship 500 points shall be deducted. When the first hit is on the back of the ship no points shall be deducted. When the first hit is on the middle of the ship 1500 points shall be deducted.
    - The ship called *Destroyer* shall be worth in total 2000 points. When the first hit is on the front of the ship 500 point shall be deducted. When the first hit is on the back of the ship no points shall be deducted.
- The game shall show an empty board when the game has stopped.
- The game shall reset the player's score when a game has ended.
- The game shall show a visible grid on the board.

1.3 COULD HAVES

- The game shall be able to offer boards of different squared sizes.
- The game shall be able to offer boards with different shapes (non-squares) (Classical Mode & Enhanced Mode).
- The player shall be able to select a specific board upon starting a new game.
- The player shall be able to change the orientation of the given ships.
- The game shall not allow the player to place the ships diagonally.
- The game shall have a timer which forces the player to make a click event within a time limit.
- The player shall be able to pause the game while in progress.
- The game shall play a music theme when in progress.
- The game shall play a sound when a click event on a cell has been performed.
- The player shall be able to turn the music and sound of the game on or off.

1.4 WONT HAVES

- The player shall be able to continue his game upon restarting the application.
- The player shall be able to play against another player.
- The player shall be able to change the board's background by choosing an image from his/her computer files.

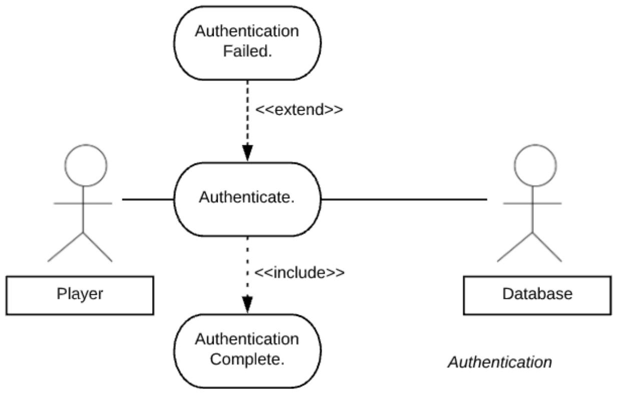
Non-Functional Requirements

- The game shall be playable on Windows (7 or higher), Mac OS X (10.8 or higher), and Linux.
- For the iterations after the delivery of the first fully working version, the Scrum methodology shall be applied.
- The game shall be implemented using the programming language Java.
- The game shall use SQL.
- The game shall use JDBC Driver.
- The game shall use prepared statements in Java to avoid code-injections.

Modeling Use Cases

Must Haves:

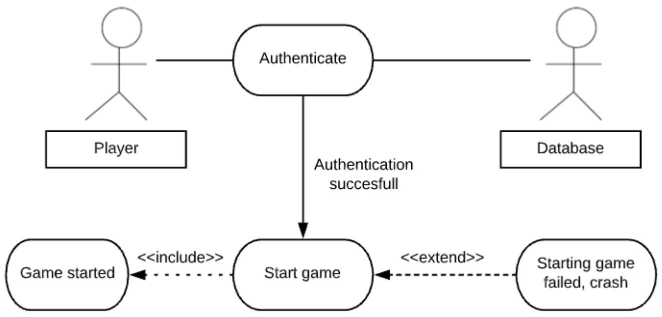
1. **The game shall ask the player to authenticate himself with a username and a hashed password before starting the game.**



Authentication

The game will ask the player to login using their existing credentials found within the database. This only applies to users that have already registered. As indicated in the diagram, the user will have to authenticate themselves. The input of the user in then verified using the database. If it succeeds, the authentication is complete. Otherwise, it fails and they will not be able to continue the game.

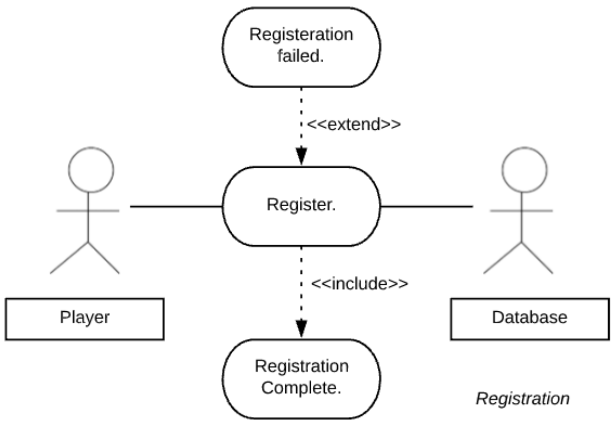
2. **The game shall only start when the credentials of the player correspond with the credentials stored in the database.**



Start game

Linking onto the previous Must-Have, this example links to the actions that happen after a user has successfully undergone authentication. After authentication, the game will either start up successfully or due to a technical difficulty, fail.

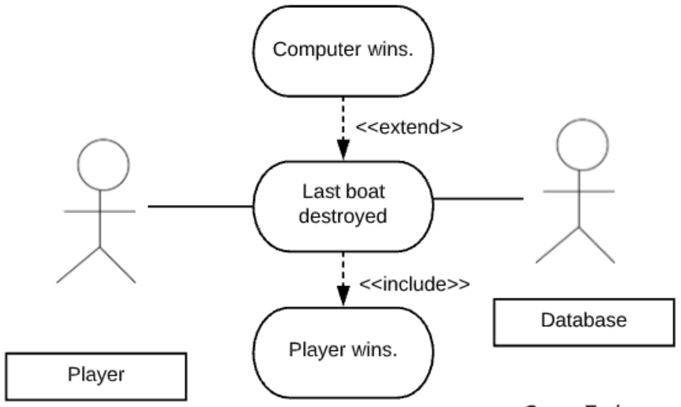
3. **The game shall allow a new player to register.**



Registration

When a player opens the homepage they are given the option to login using their existing credentials. However if the user has not yet registered, the user is required to create an account within the database. This can either result in a successful transaction or a failed one. Once registered, the user's credentials are recorded, enabling them to login the next time instead of creating a different account.

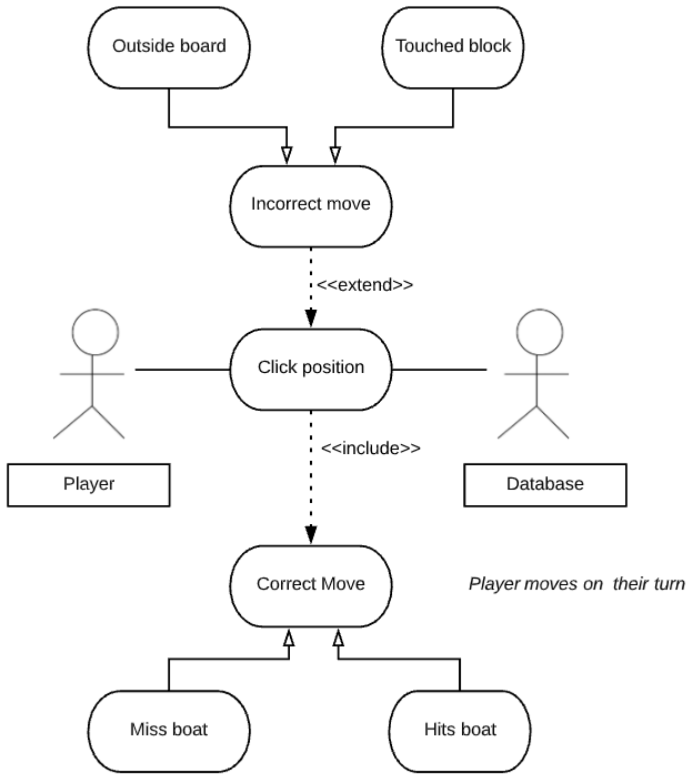
4. **The player shall win when the player has performed a click event on all the ships of the opponent before the opponent has performed a click event on all the ships of the player.**



Game Ends

This Must-Have discusses the situation when a player wins the game. Once the player destroys the final boat of the computer (or other way around), then the player wins the game. There is the possibility of failure that the computer (or the opponent) wins instead. This also involves the database, seeing as highscores have to be recorded at the end of the game.

5. **The game shall be able to let the player click an untouched cell when the player has touched a cell containing (a piece of) a ship.**
6. **The game shall be able to let the opponent click an untouched cell when the player has not touched a cell containing (a piece of) a ship.**



We chose to combine two of our Must-Haves seeing as they share similar UML diagrams. The player has to select a place on the opponent's board to hit. An incorrect move is defined as a move that is outside the opponent's board or has already been hit once. A correct move is defined as a place where nothing has been done yet and it either results in a hit/miss with respect the position of the opponent's ships.