Battleship

Group 18: Andrew Mitchell, Evan Kuehr, Nate Jones, Obaid Ebadi

Description

Battleship is a classic two player game in which players use strategy to to search for the enemy's fleet of ships and try to take them down one by one. The winner of the game is the player that can take out the entire enemy's fleet before them. There are some modifications we made to the game, including the concept of a "captains quarters", underwater ship placement, and a player being able to move their entire fleet.

Development Process

We took an iterative development approach when designing this project. Each development phase consists of first writing code modules and appropriate tests, followed by a round of integration with existing code and further testing. At this point, refactoring takes place as needed by the integration process, which allows code to stay readable and modular throughout development.

Requirements

The basis of the original Battleship game requires two players, their own boards that the other player cannot see, and a way to keep track of each player's actions. The benefit of developing this game for a computer is that additional features can be easily supported which is why we added multiple features that branch far from the original game such as special attacks that affect more than a single coordinate of the opponent's board as well as a mechanic that regulates special attacks within gameplay.

The Basics

The first requirement we had was the basic structure of the game which is why we developed game classes that would optimally interact with each other by following effective object-oriented designs learned in class. These included an overall game class that would govern the progression of the game, player classes that would represent the players and house their respective boards, fleets, and additional classes required for features we added to the game.

Regulating Gameplay
Once our basic game was supported by the required classes, we understood that a gameplay mechanic would need to be developed to regulate the players' ability to utilize our special attacks. So we developed a card based system where a player can choose to either use a basic attack (strike a single coordinate of the opponent's board), get a card (which represented a special attack) and not attack, or spend (use and discard) a special attack represented by a card that was previously acquired. This new card class and the way it interacted with the game allowed for a whole new layer of strategy now that players could use special attacks later at the expense of not striking immediately.

Usability
Our final consideration was how someone would actually be able to play the game. We knew that we wanted to accommodate for two players to play the game in the traditional sense (ie. no peeking at your opponent's board) but we also understood the time restraint we had on the development of the UI. We decided that since our game was developed in Java, we would utilize the Spring framework to develop a REST API in tandem to a simple React framework client. This allows anyone with the source code to host the game on a server and anyone in the world to play via their browser on their own screen. It also allows the server to host multiple games at once.

Specifications

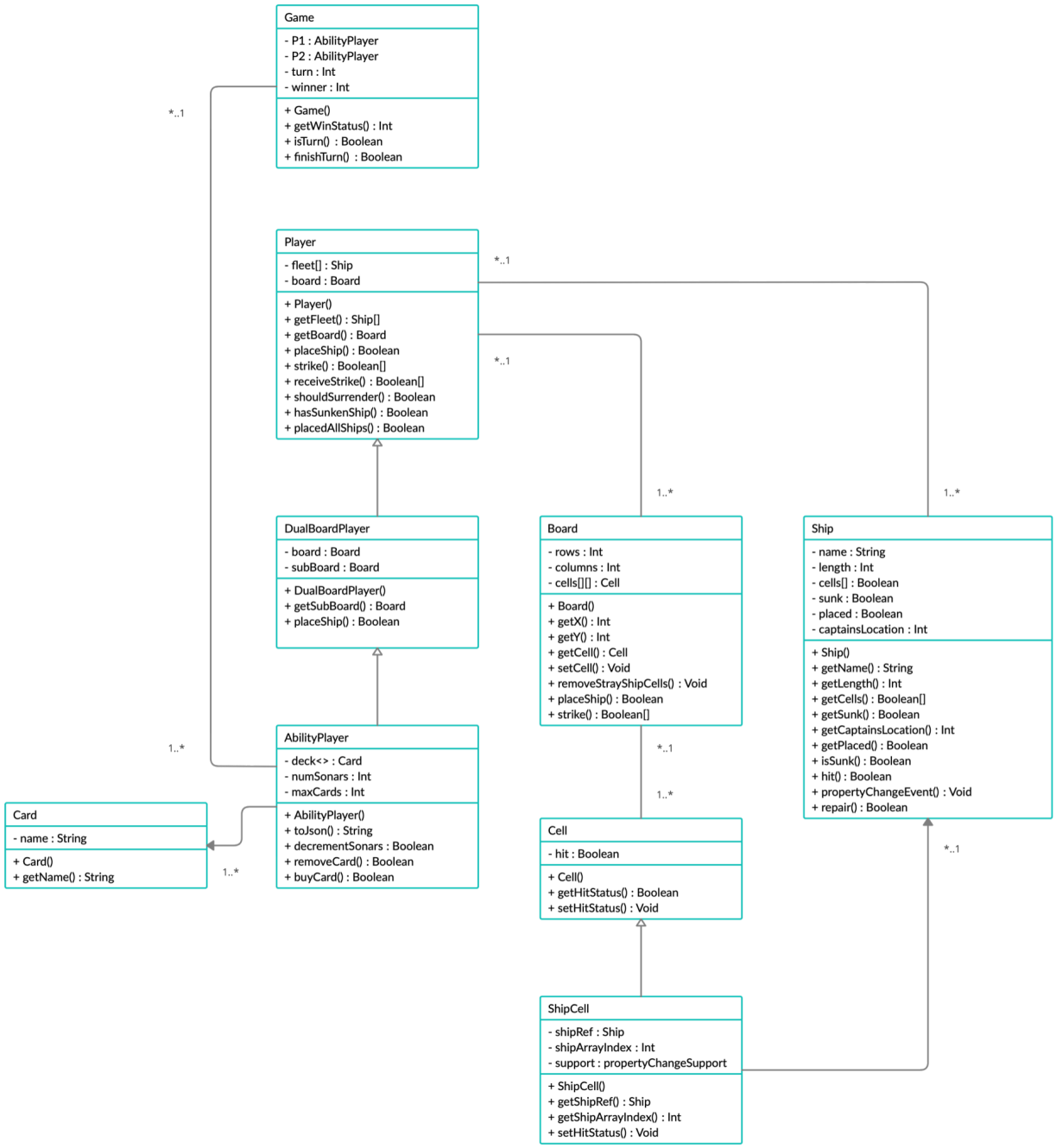
- Java, Spring Framework
- Node.js, Preact Framework (lightweight React alternative)
- minimum 1 computer to run the Java based game server and x2 browser tabs to run each game.

Future/Potential Specifications

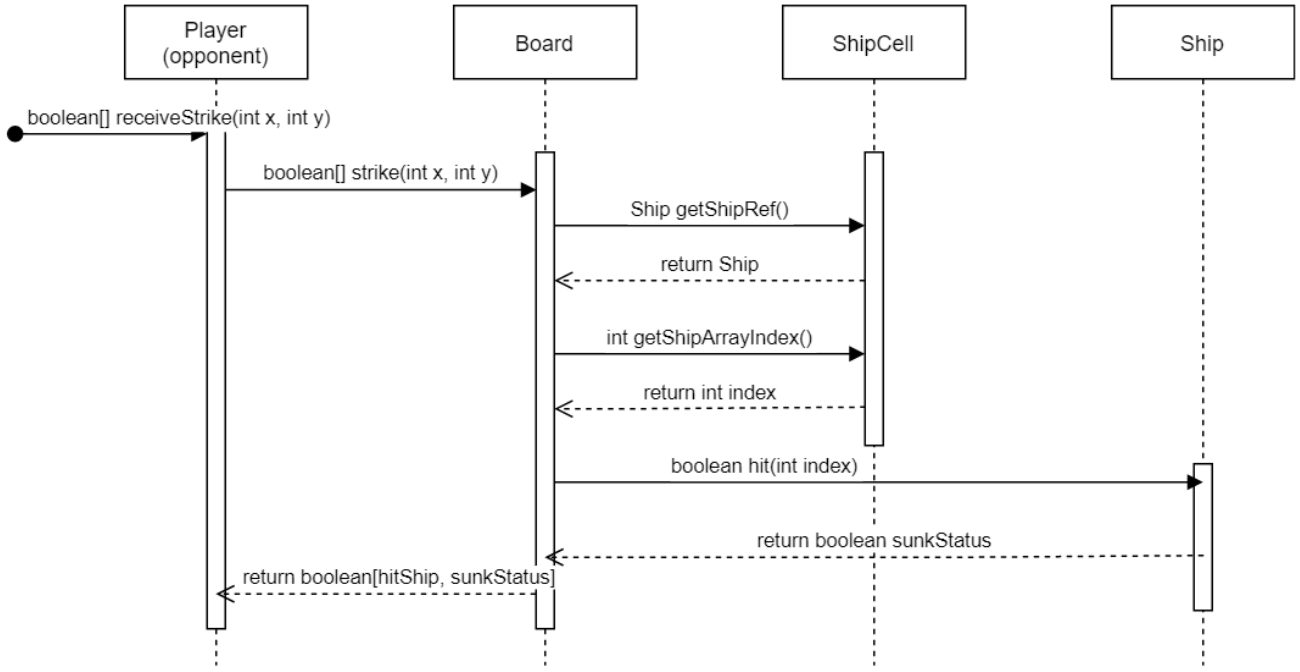
- ngrok (the current specifications allows for games to be played across a local network but a reverse proxy like ngrok or nginx would be required to run the game server remotely.)
- Vercel or any other client hosting platforms to access the client without having to run it locally.

Architecture and Design

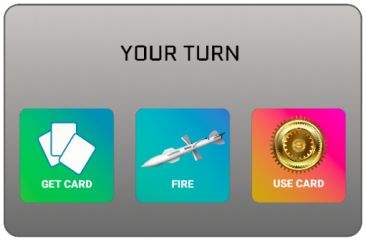
The general design of the system involves an application running a RESTful API which includes the business logic of the game and an interface that clients can use to interact with the game. Clients connect from a browser. React is used to run the client-side code and UI.

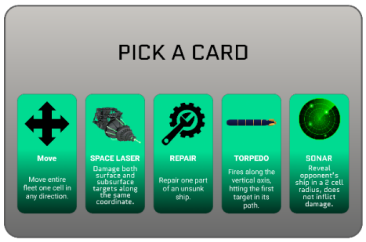


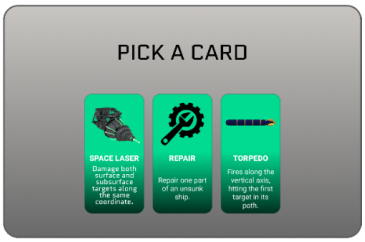
Use Case: Attacking a Ship

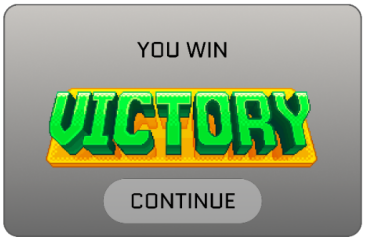


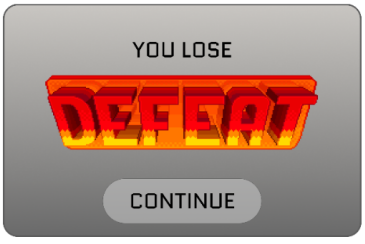
Figma Prototypes











Personal Reflections

Andrew Mitchell
This extensive project allowed me to identify and apply a variety of object-oriented designs and patterns that I personally found useful and are deemed good practice. It also flexed my ability to work in a team and practice good time management efforts. Finally, while developing this project allowed me to practice useful OOAD skills, it also allowed me to practice developing a full game from the ground up in a fun and challenging way.

Evan Kuehr
Through completing this project, I was able to gain a lot of experience with working in a development team and solving problems with design patterns. I had done a small amount of object oriented programming in the past, but not to this extent. I also gained a lot of new skills through our implementation of a user interface. We decided to convert our backend application into a REST API, which I had never done before. I also gained experience using React and building a more complicated web app. This is probably one of the largest projects that I have worked on so far. This new experience will hopefully assist me in my internship this summer and future projects, such as the senior capstone.

Nate Jones
Working on this project has taught me a great deal about not only collaborative development, but the full-stack development process in general. It has given me invaluable experience and many skills that are necessary in the context of industry, such as web development APIs and frameworks (REST). Working effectively with such a large project scope was a challenge at first, but over time this work became more efficient and focused, and everything ultimately came together in the end. The knowledge this experience has given me will definitely help improve my work on future development projects.

Obaid Ebadi
This semester-long project was one that I learned a lot from. Not only did we gain experience with design patterns, we got a lot of experience with object oriented programming. Another thing I gained a lot of experience in was to work with a team. The circumstances of this semester being mostly or all remote made it interesting too, but I think it will definitely help in the future for work.