Team 20 Project Backlog

Team 20 | Thomas Chen, Kelvin Choi, Scott Merritt, Aaron Althoff, Dan Morton

Problem Statement

The classic game Battleship is a grid based strategy game where two opponents fight against each other by calling out coordinates to sink the other opponents ship. An interesting problem arises when viewing Battleship through a game theory lens. Many strategies exist for this game, but we propose creating a special variation of the game that enables a drastically larger strategy space. We achieve this by making the following changes to the vanilla game:

1. **Expanded Rule-Set** - Expanding the Strategy Space of the Game
2. **Multiple Players** - Transforming the Strategy Space of the Game

Expanded Ruleset

The base functionality of this platform will be a fully functionally online implementation of our variation of Battleship. Our variation follows the following uniquely modified rule-set to enable a more complex strategy space:

- Game state will be a larger than vanilla square grid (e.g, 15x15, 20x20, etc) with a set of predetermined length ships per side (eg., [2,3,3,4,5])
- Players may place their ships on the grid but no ships may overlap
- Each side takes turns and are allotted one turn per move, the turn can be one of the following actions:
    - Hit a space on the opponents grid:
        - Action 1: Select which opponent to target
        - Action 2: Select which grid position to target
        - Result:
            - Either hit or miss
            - All opponents see result of shot
    - Move a ship
        - Action 1: Select ship to move
        - Action 2: Select direction to move, ships cannot turn so the ship can only go in 2 possible places (forwards or backwards)
        - Result: Ship moves one grid square in specified direction
            - Ships are not allowed to turn
            - Once a ship is hit, it may not move
- Each side takes turn firing one shot, and receives result of shot (hit, miss, sunk)
- Game ends when only 1 player has ships remaining

Stretch Functionality

On top of the core functionality listed above, we propose three stretch features:

1) Procedurally generated maps - maps will be ~40% land (grid spaces where ships are not allowed to be placed) generated in a procedural fashion
2) User defined AI - users will click through various UI fields and an AI following their inputted strategy will play for them
3) User implemented AI - users will implement a Java interface that defines a control strategy and then upload their code. This fully defined AI will play for them.

Background Info

This game has an intended audience of both a casual user who enjoys “tinkering” and programmers who want to build out more features artificial intelligence for strategies. This application will have two ways to create a strategy, one through user interface for the casual user and the other through a programming interface for the technical programmer.

Similar Platforms: Robocode

Environment

Frontend

Application Development Platform: Angular
Testing:

- JS testing tool: Jasmine
- Functional testing tool: Selenium
- Cross-Browser testing tool: Browsera (automated)
- CSS testing tool: Needle (automated)

Backend

Language: Java
Framework: Maven + Spring (REST API)
Database: PostgreSQL
Database Connection: Java Persistence API or JDBC
Testing/CICD:

- Test Coverage: Codecov (Automated Tool)
- CICD Pipeline (Build, Test Execution, Deployment): TravisCI (Automated Tool)
- Test Library: JUnit5
- Linter: Maven Checkstyle
    - Java Standard: Google Java Style Standard

Development Tools: IntelliJ, Postman, Sketch

Functional Requirements

ID	Functional Requirement	Hours	Status	Priority
1	As a user, I would like to access this game via a website	3	Unplanned	Core
2	As a user, I can be authenticated	5	Unplanned	Core
3	As a user, I can login/logout of the game platform	5	Unplanned	Core
4	As a user, I have intuitive account controls (e.g, login, logout, settings, etc)	5	Unplanned	Core
5	As a user, I can play against a default bot	8	Unplanned	Core
6	As a user, I can make an attack against the other player (bot)	2	Unplanned	Core
7	As a user, I can see if my attack hits or misses on the game board	5	Unplanned	Core
8	As a user, I would like visual feedback regarding my actions	3	Unplanned	Core
9	As a user, I can choose to move un-hit ships instead of making an attack	2	Unplanned	Core
10	As a user, I can play on larger grids (15 X 15, 20 X 20)	5	Unplanned	Core
11	As a user, I can play against another player/bots	8	Unplanned	Core
12	As a user, I can play against multiple other players/bots	8	Unplanned	Core
13	As a user, I can differentiate between my opponents ships that have sunk	2	Unplanned	Core
14	As a user, I can play against a leveled default strategy (easy/medium/hard)	8	Unplanned	Important
15	As a user, I can see if I have been eliminated	2	Unplanned	Core
16	As a user, I can see if I have eliminated another player	2	Unplanned	Core
17	As a user, I can see if I have won the game	2	Unplanned	Core
18	As a user, I can set my own custom settings for	3	Unplanned	Core

	the game.			
19	As a user, I can save my own custom settings to be used at any time.	2	Unplanned	Core
20	As a user, I can view my play history.	8	Unplanned	Core
21	As a user, I can design a strategy through a user interface	21	Unplanned	Stretch
22	As a user, I can create a strategy through a programming interface	21	Unplanned	Stretch
23	As a user, I can save a strategy	8	Unplanned	Stretch
24	As a user, I can delete a strategy	2	Unplanned	Stretch
25	As a user, I can modify a strategy	3	Unplanned	Stretch
26	As a user, I can view other people strategies	8	Unplanned	Stretch
27	As a user, I can play against my own strategies	13	Unplanned	Stretch
28	As a user, I can play my strategy against default strategy	8	Unplanned	Stretch
29	As a user, I can play multiple games in a sequence	2	Unplanned	Stretch
30	As a user, I can see my saved strategies	2	Unplanned	Stretch
31	As a user, I can choose to group players into teams	4	Unplanned	Stretch
	Total	180		

Nonfunctional Requirements

1. Animation: In the game there are many opportunities to take advantage of good animation to help the user experience. This includes animating the movement of ships, marking hits or misses on the grid, displaying victory/defeat for a player.
2. Sign In and Accounts: Using google sign-in for logging into a user profile. This will be fairly simple to implement and secure for users to log into their profiles, allowing for storage of player information with a secured account system.

Use Cases

Case	Action	System Response
Login	1. Choose sign in with google 3. Sign in with google 4. Confirm sign in	2. Prompt that user was successfully logged in 5. Navigate to login page.
Logout	1. Choose logout from the logout menu while authenticated.	2. Prompt that the user was successfully logged out. 3. Navigate to the login page.
Starting a Game	1. Select ‘new game’ button 3. Choose option to play against included AI or to select a certain strategy 4. Choose grid size and number of players	2. Game options will appear 5. Grid will be created with the selected size 6. Bots will be loaded in according to selected strategy
Viewing another opponent’s grid	1. User selects an opponent 3. User selects ‘View Grid’ option	2. Options appear next to opponent’s icon 4. Panel opens to reveal opponent’s grid from the user’s perspective (i.e. shows land and attacks made by any player on that grid)
Making an Attack	1. User will select space to attack	2. System will mark space as a hit or miss on the board
Moving a ship	1. User will select a ship to move 3. User will select position to move ship	2. Grid will display movement options that selected ship can perform 4. Update Grid with new position of that ship
Winning the Game	1. User will have eliminated all other opponents from the game 3. User can choose to see results or play again	2. App will display a victory notification 4. Display results or take user back to new game page
Create a strategy	1. Select create new strategy	2. Open modal view

	button 4. Select strategy design option	3. Present options to design strategy a. UI b. Programming Interface 5. Open corresponding interface
Quit Game	1. Select quit game button. 2. Select either yes or no button as confirmation.	3. Indicate that user has successfully quit the game.
View Results	1. Select View Results button. 3. Select Done button when finished viewing results.	2. Display result from the finished game. 4. Stop displaying results from finished game
View Play History	1. Select play history button from main menu 3. Select Done button when finished viewing the results.	2. Display log of all games played.