Team 20 Project Charter

BattleshipAI

Team Members:

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

Problem Statement:

In this application we are creating a web application based version of the popular game *Battleship* with a unique twist. The game is a grid based strategy game that can be implemented well with modern UI components, but our version of Battleship will feature players inputting game models and having their models compete against each other as opposed to just playing the game. This game will be able to be played against our own AI computer models as well as against user submitted (multiplayer) models if time permits. It can serve as an interesting tool for up-and-coming coders to put their skills to the test writing their own models.

Problem Objectives:

- Recreate the popular game of Battleship as a modern web application.
- Build out a control interface and model engine that allows players to implement or write-up simple models that are then interpreted by the engine
- Create an easy-to-understand UI for players to interact with the game.
- Allow users to play against our model, models of their own design, or models created by other users
- Design an effective model to play against users, with varying degrees of difficulty.
- Have the app save user's game in the cloud so users can monitor their progress.

Stakeholders:

- Developers: Scott Merritt, Thomas Chen, Dan Morton, Aaron Althoff, Kelvin Choi
- Projects Owners: Scott Merritt, Thomas Chen, Dan Morton, Aaron Althoff, Kelvin Choi
- Users: People who are interested in machine learning, programming or enjoy the game of battleship and want to try new strategies and algorithms.
- Project Coordinator: Adam Johnston

Project Deliverables:

- Web application with the ability to allow users to play the game of Battleship.
    - The framework being used for this frontend is Angular. NgRx will be used for state management, as the game is heavily state based.

- Backend Server which manages session/game state and exposes public controls endpoints to the user's submitted model
    - Controls API will be a RESTful API written in Java powered by Spring
    - Game state will be a 10x10 grid with (2,3,3,4,5) length ships per side
    - No ships may overlap
    - Each side takes turn firing one shot, and receives result of shot (hit, miss, sunk)
    - Game ends when the ships of one side have all been sunk
- A set of models to have the user play against, each with a different degree of difficulty
    - Models will interface with the controls API and either be a simple config file or if time permits a DSL
- A state management system to save players' data in the could so they can monitor their statistics over time
    - Persisted data will be stored in a SQL database (PostgreSQL or MySQL) and connected to via Java Persistence API (JPA) or JDBC.