EECS 448 – Software Engineering I

Project 02

System Documentation

Group 08:
Huzaifa Zahid
Mir Shazil Faisal
Junyi Zhao
Zhenzhou Wang

<u>Story Point Analysis of previous EECS projects:</u>

1	**2**	**3**	**5**	**8**	**13**
🟩 Lab 1 – Simple Printing Exercise	🟩 Lab 6 – Array reversal, file I/O	🟧 Lab 3 – Lidar Sensor	🟩 Lab 10 – Intro to OOP	🟩 Lab 11 – DMV Class	🟦 Lab 7 - Recursion & Backtracking
🟩 Lab 4 – Fibonacci numbers, ASCII Conversion, etc.	🟦 Lab 1 – Command-line argument file manipulation	🟦 Lab 6 – Recursion exercises	🟦 Lab 4 – Elevator (Queues & Stack)	🟦 Lab 9,10 - Binary Search Tree	⬜ EECS 448 – Project 1
🟨 Homework Assignment 1	🟦 Lab 8 – Time Complexities	🟨 Homework Assignment 2	🟨 Homework Assignment 3	🟦 Lab 5 – Browser Tracking (Lists)	⬜ EECS 448 – Project 2

<u>Legend:</u>

🟩	EECS 168 (Spring 2021)
🟦	EECS 268 (Fall 2021)
🟨	EECS 368 (Spring 2022)
🟧	EECS 388 (Spring 2022)
⬜	EECS 448 (Spring 2022)

<u>Breakdown of EECS 448 Project 2:</u>

1	**2**	**3**
Playing and understanding other group's game*	Documentation	Adding additional feature/s
Estimating hours	Debugging and stress testing	Implementing the AI logic
Outlining the AI logic	-	-

* Understanding other programmers' code is usually a much more difficult task, but since the project is written in the same language and platform that all our group members are familiar with, and worked with on the first project, it is comparatively an easier task for us.

<u>3 hours/story point:</u>

Provided we all are comfortable with C++ and had a fully functional project 1 from the other team, we decided to allocate 3 hours per story point because we felt like it is a sufficient estimation with room for error correction too, if we come across any errors. Additionally, we also divided the more complicated tasks into smaller more doable tasks (with help from helper functions) which makes the 3-hour per story point estimation valid.

<u>Custom Additions to the project:</u>

In totality we have made three custom additions to the game:

1. In a standard battleship game, the turn is switched after every move. However, we included a custom addition where we ask the user/s after how many moves would they like to switch turns (the maximum moves they can have in one turn is 3). This mode is available with both, Player VS AI, and Player VS. Player modes.
2. In the Player vs. AI mode with the standard 1-attack/turn mode, we have added a bonus move after every five turns for both, the player, and the AI.
3. It is now shown on the terminal which attack number the players or the AI are going to play. However, the Attack number does not include the bonus moves.

<u>Design Paradigm used by Team 10 for Project 01:</u>

The Project 1 that we got seems to be using Event Driven Architecture – a design paradigm that uses events to trigger and communicate between decoupled services – because the team used a lot of helper functions throughout their project. For example, when they were placing ships, as soon as the user entered in the coordinates, the team handed over control to a helper function – pathValid() – to check if the coordinates entered are valid and legal. Similarly, inside the sinkShip() function, they hand over control to the getPositionArr(), etc.

For our project 02, we continued the same approach to this design paradigm. Like for our custom additions, if the player chooses to play the AI vs. Player mode on the easy difficulty with 1-attack/move option, our code allows both, the player and AI, to have bonus moves after every fifth turn, otherwise there are no bonus moves and the game is played according to the mode selected.