**[ WORK IN PROGRESS ]**

**Software Requirements Document**

For CS372 Software Construction, Project 1

Group 5: *Cody Gaines*, *Joshua Guerrero*

**Table of Contents**

1. **Introduction**
   1. **Purpose**
   2. **Scope**
   3. **References**
2. **Design**
   1. **User Stories**
3. **Functional Requirements**
   1. **Initialization**
   2. **Running**
   3. **Input Validation**
   4. **Class Objects and Related Functions**
      1. **Player**
      2. **Board**
   5. **Computer Opponent**
4. **Nonfunctional Requirements**
   1. **Battleship Rules**
5. **Introduction**
   1. **Purpose**

The main objective of this document is to illustrate the software requirements for the Battleship programming project. This document provides the scope and intended end goal of the project while also providing details on both the functional and nonfunctional requirements of the project. The main purpose of this project is to be able to create program that runs the game of Battleship which allows a user to compete against a computer-run opponent.

* 1. **Scope**

The project is designed to be operated by a single user and run from the command line terminal which shall work as a complete interface for the user. The game itself will follow standard Battleship rules where players take turns selecting coordinates, destroying all of the opponent's ships in order to achieve victory. The user’s opponent will be the computer which shall be programmed to follow the rules of the game. At the end of each game, the user shall be able to choose to start a new game or quit out of the program.

* 1. **References**

1. **Design**
   1. **User Stories**
2. **Functional Requirements**
   1. **Initialization**
      * Program will create the board object and set loop flags to appropriate boolean values.
      * Player will be prompted for input for setting up game board and choosing opponent type (human or computer).
      * After initialization is complete, program jumps into a while loop that runs all game code.
   2. **Running**
      * While loop is created that runs until a *gameOver* flag is detected to be true.
      * Current player is asked to give a coordinate which is saved as a string.
      * Program checks string for validity of coordinate
      * If valid, a function is called to change the state of the board
      * Once board state is changed, the state of the board is outputted into the terminal for the user. The board shall be represented with ASCII characters.
      * Once the board is displayed, the state of the player is checked. If player is not defeated, program continues and restarts *while* loop. If player is defeated, *gameOver* flag is set to *false* and program breaks out of the *while* loop.
   3. **Input Validation**
   4. **Classes Objects and Related Functions**
3. **Player**
   * All information about the player shall be stored in a *Player* class object. The code for *Player* class objects is referenced in the header file *player.h*.
   * *player.h* holds constructors for the object, *get* and *set* functions, and private member variables for all *Player* objects.
4. **Board**
   * Every coordinate is represented as a *Board* class object. Each player’s board information shall be stored in a 10 by 10 two-dimensional array of type *Board*. The code for *Board* class objects is referenced in the header file *board.h*
   * *board.h* holds all functions related to the board such as construction, *get* and *set* functions, and displaying board output.
   * All *Board* objects shall contain a private data member that stores the name of the coordinate and flags indicating if it is occupied by a ship or if it has been a chosen coordinate yet.
   1. **Computer Opponent (IDEAS)**
      * Uses function to randomly choose an opponents coordinate
      * If coordinate contains ship, tries all available squares around until another hit is registered.
      * Continue until ship returns a *dead* flag.
5. **Nonfunctional Requirements**
   1. **Battleship Rules (IDEAS)**
      * Must be played on a 10 by 10 board
      * Each player gets one move per turn
      * No player is allowed to see the others board
      * Each player is allowed to see their board and a board showing which coordinates they have previously chosen to attack.
   2. **Ease of Use?**