**Software Implementation and Testing Document**

**For**

**Group Phaze 5**

Version 3.0

**Authors**:

Faith Miller

Mackenzie Knight

Eduardo Antonini

Tristan Garcia

Katie Rombeiro

# **1.** **Programming Languages (5 points)**

Our proposal was to create a web app -- a multiplayer game that would run on a web browser. Even though we had a change of plans halfway through our implementation, we stuck with our original choice of developing a game for a browser environment. For this, we wrote our backend code using JavaScript, and used the React framework to integrate the backend with the frontend. JavaScript is a powerful programming language, that allows for great flexibility while it saves time by absorbing inconsistencies that would be considered errors in other languages. JavaScript’s power and flexibility, combined with its intuitiveness, were decisive factors when choosing what language we would write our project in.

# **2.** **Platforms, APIs, Databases, and other technologies used (5 points)**

For this part of the iteration we still ran all of our code through Atom. We tested the game by continuously playing it through a local host on one of our computers. For this iteration, no other technologies were used.

# **3.** **Execution-based Functional Testing (10 points)**

Execution-based functional testing was performed in this iteration by using node package manager (npm) in order to compile and run our React Web App. Since our frontend and backend code implementations were connected in this iteration, both were tested together. Functional testing was repeated after each addition of code switching over the backend game implementation code to working with input from the React Web App. In order to debug our program, we recompiled and executed the program after each addition/deletion of code in order to play the game. We repeated the debugging process until a full game of Phaze 5 could be properly completed.

# **4.** **Execution-based Non-Functional Testing (10 points)**

Some execution-based non-functional testing we performed in our project was a continuation of our use of the inspect aspect within google chrome. It allowed us to make sure we were grabbing user input. There was not much non-functional testing, as much of our project ran through the necessary execution-based functional testing. We did not focus on speed, however, our code only had two if statements most of the time for each section of the modal and therefore allowed for a quicker processing for our code. Security, at the end, wasn’t a main issue, as the only personal/private information stored was a user’s first name, where they still had an opportunity to put a nickname if they felt like it instead.

# **5.** **Non-Execution-based Testing (10 points)**

In this iteration, we performed non-execution based testing consisting of code reviews, inspections, and walkthroughs. For starters, we made sure that the front-end teammates checked each other’s code to ensure that there were no mistakes or bugs. We also made sure that all of the code involved in the connection between the back-end game implementation and front end was checked by teammates the following day to ensure there were no mistakes or issues. We all would meet up as a team on a ZOOM call almost every day of the iteration to try and play the Phaze 5 game and watch for any errors or bugs in the game. We would then fix all issues to make sure the game played correctly. We made sure that all teammates inspected another person’s code to ensure it was all well written code with comments. In addition, we made sure to try and resolve all warnings that popped up in the terminal to ensure there would be no issues with our game or web-app.