**Software Requirements and Design Document**

**For**

**Group Phaze 5**

Version 3.0

**Authors**:

Faith Miller

Mackenzie Knight

Eduardo Antonini

Tristan Garcia

Katie Rombeiro

**1.** **Overview (5 points)**

A simplified version of the game Phase 10 put into a web-game version that allows for 2-6 players to play. This new game is called Phaze 5 and it includes the game play of two of the phases from the original Phase 10 game. The game is run through localhost using the Atom platform. The programming languages React, Javascript, HTML and CSS make up the back-end game implementation and the front-end implementation.

# **2.** **Functional Requirements (10 points)**

For the frontend, the first functional requirement for this iteration is having buttons on the screen (1). This is high priority because it allows for the user to click on the Play game button and enter the players’ usernames and physically play the game. The other button on the screen is the how to play button, which directs the user to the screen explaining all the rules for the game. The next functional requirement for the frontend in this iteration is having the connection to the backend game implementation (2). This is also high priority because this allows for us to take in user input and actually play the game according to the game implementation functions. Another functional requirement in the frontend is having the play game button pop up into a dynamic modal where the whole game is played. This modal must take in all user input in the form of input text boxes and communicate the input to the backend game implementation (3). This allows for us to submit all user input/game plays and allows the user to actually play the game. We also created the game to be able to be played with 2-6 players (7). This is a high priority because we wanted our game to be just like the real game of Phase 10 and we wanted the game to be able to be played by a range of 2-6 players. We created a few functions to allow 2-6 players to have the ability to play the game and choose their own usernames.

For the connection between the frontend and backend, the first functional requirement is taking in the information entered by players and prompting the user with the next input question based on their choice (4). This is a high priority requirement because without prompting each player with the proper sequence of questions during their turn, the game cannot function properly. One example of this is when a player chooses to either make their initial phase or put down cards. If the player decides to make their initial phase, the program must generate the corresponding 6 text boxes to allow the player to input their phase. If the player decides to put down cards, the program must generate the next question asking if the player would like to put down cards to an existing board or make their final discard for that turn. Another functional requirement that goes hand in hand with this is taking in information and storing it when necessary in the proper storage container (5). This is also a high priority for the connection between the frontend and backend, because without the proper storage of information based on a player’s turn, the game cannot function properly and will never be completed. One example of this is when a player chooses to draw from either the discard pile or from the deck. In either case, the program must be able to store the new card into the player’s hand. Otherwise, the player cannot work towards completing the phase for that round.

For the backend alone, the only additional functional requirement in this iteration is that the game implementation switched over to working with the React App must work the same as it did through CodePen in iteration 2 (6). This is a high priority because the game must function properly and mimic the actual game of Phase 10 as it did in our CodePen demo in iteration 2. Note that all of the functional requirements for the backend listed in iteration 2 still stand. Please refer to the backend section of the functional requirements in iteration 2 for more specific details about the backend game implementation requirements.

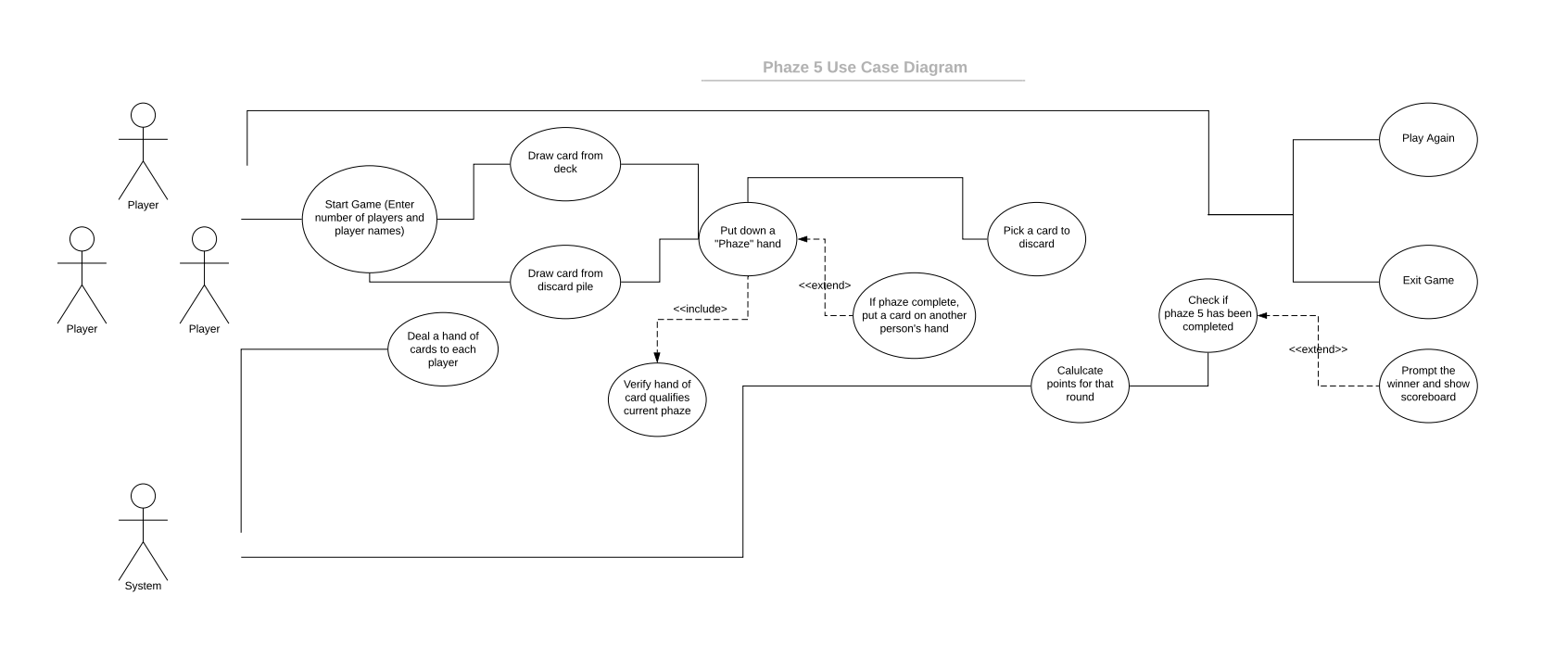
# **3.** **Non-functional Requirements (10 points)**

The performance of our project must be good enough to allow the entire multiplayer game of 2 phases to be completed. There are no specific performance requirements for our game, even though we have been coding as efficiently as possible. The system must be reliable enough to store player information, such as a player’s hand, their board containing their completed phase, the running score against them, and a state variable indicating what phase each player is on in any given round. This is a high priority, because the storage of player information is essential in completing the game properly.

Our project did not need any security or safety because the users would not be inputting any personal information or information that they would not want seen by the public. We ensured that the game would always perform properly and that no issues or bugs would occur during gameplay by playing the game and testing for issues many times. The game is reliable because it will be played on a local host, so we don't have to worry about the server going down or a database not functioning properly.

**4.** **Use Case Diagram (10 points)**

Link to PDF version of diagram: <https://www.lucidchart.com/publicSegments/view/bb20ea11-da9a-4a5b-aa11-6dc4c68ffb69/image.pdf>



**5.** **Class Diagram and/or Sequence Diagrams (15 points)**

**

**6.** **Operating Environment (5 points)**

Phaze 5 is designed to operate in most common web browsers. While running it in the demo, we used Apple Safari and Google Chrome; however, it should have no issues running in other browsers such as Microsoft Edge. Because Phaze 5 was engineered in JavaScript -- a popular language for web apps --, it should be compatible with any web browser capable of interpreting JS files with the React framework. Phaze 5 is compatible with all major operating systems: macOS, Windows, and Linux.

**7.** **Assumptions and Dependencies (5 points)**

The only assumed factor our team had by iteration 3 that could affect the final result of our project was making the connection between the front-end code and the back-end game implementation code. Our switch to 2 phazes in iteration 2 allowed us to go through easier testing, which in the long run was helpful not having to sit through multiple hour long games just to test one function. However, as stated within other documents, we were able to overcome this issue of the connection of the front-end to the back-end code. We still finished with a couple of days to spare before the deadline, which easily allowed us to test our final code with no pressure on our shoulders, therefore taking away the potential time-crunch issue. Our entire team stayed fully connected and communicated throughout the entire iteration 3, therefore keeping us all on track for what was needed and how to start the next step as we went through the coding of the project. We also have a texting group chat where we all spoke of when we each made a push to GitHub so others could pull the updated code.