**Part II: Status Report**

Working Features:

1. A main landing page, where users can either log in or sign up for the game.
2. A sign up/registration module, which successfully adds unique users to the game’s database.
   1. Uses PHP and JavaScript.
3. Registration validation, which prevents users from signing up with an already-taken username and ensures the user is inputting their desired password (passwords need to be inputted twice and match for verification)
   1. Uses PHP.
4. Php/JavaScript lobby page to display available servers in a dropdown menu.
   1. If server is unavailable, the server will be displayed in the dropdown but will not be selectable.
   2. On server selection, the lobby will display a list of players joined to each team on the given server.
   3. Shows teams available when a game server is selected.
   4. Each team’s table is a clickable region, and when selected, joins the user to that team, and the user will appear in that team’s roster. The region will highlight when the user drags a mouse over it.
   5. When a player has joined a team, the user can switch teams by simply clicking on the other team
   6. If a team has 5 players, no further players can join that team.
   7. Only when a user has joined a game, an option to “leave” the game entirely will appear. Player will be removed from the game but remain signed in, still able to join a game or select a server.
   8. A “start game’ button will only appear if at least 1 player is joined to each team.
   9. “Start Game” button will only appear to players who have joined the game -- outside onlookers are unable to start the game.
   10. User can log out by pressing “log out” button -- user is returned to home/login screen.
   11. Implemented functionality that detects if a game server is alive and inactive, disabling inputs if a server is unavailable.
   12. Hid the "start game" button if a server is not available.
   13. Automatically redirect all players to the game board if someone clicks "start game".
5. Modified password management so that passwords are hashed when saved (using the php password\_hash function), and incoming passwords are compared against the hash (using the php password\_verify function).
6. Wrote a NodeJS server application that manages game state across multiple clients
   1. On connection, broadcasts a message to the connecting client informing them of the state of the game, and if it is actively in use
   2. If the server disconnects, broadcasts a disconnect message to all clients. If the server becomes available again, clients will automatically attempt to reconnect to it.
   3. The first phase of the node server is complete, and the remaining features (such as paddle management, ball position, collision management) will be handled in an upcoming sprint. To date, the node server is about 25% complete.
7. Implemented a test rig that can emulate the paddle movement of other users, to assist with game testing in future sprints. This is a standalone node server application that accepts the following arguments on start:
   1. The URL of the node server
   2. The game id
   3. a comma delimited string that contains a list of player ids
   4. The program creates a separate, infinite loop for each player id in the list, with each iteration executing every 60 ms. On the first iteration, the program emits an object to the game server that contains the player id, and a random number between 0 and 600 (the height of the game board). On each subsequent iteration, the program will use an algorithm to determine a new location to transmit to the game server, and emit that new position and player id to the game server.
   5. When a user is playing a game with multiple other paddles on the screen, this emulates other players moving their paddles.
8. Figured out how to run our NodeJS Server on amazon elastic beanstalk, which allows us to handle a considerable amount of traffic.

Partially-Working Features:

1. The lobby page dynamically updates joined players whenever a player joins a team. (90%)
   1. Speed varies.
   2. Will sometimes display incorrect information before correcting itself.
   3. Performance needs improvement.

Next phase of project:

1. Build the UI elements of the game board, including:
   1. player names and colors
   2. ball behavior when a point is obtained (e.g. sliding the ball off the screen)
   3. Score view.
   4. user / paddle interaction
   5. web socket interaction with the game server.
   6. "smooth" paddle movement (currently, paddle movement can appear choppy to other users over slower or higher latency connections, we want to “smooth out” the motion of the paddle so it appears more natural)
2. Game Start:
   1. Create a function that checks that both teams have registered players ready to play. No team should be empty. If a team does not contain any players, it will prevent the start of the game play.
3. Game play:
   1. Randomize starting location for ball with a random vector.
4. Node Server:
   1. Implement the ability to manage all aspects of game state, including paddle locations of all players, current ball position/vector/speed, and game score.
   2. Have the ability for the server to transmit game state to clients if they disconnect and reconnect, if they have an intermittent connection.
5. The “Game Over” screen, which shows the final score of the game, and the name of the winning team.