Programming Assessment

The purpose of this exam is to provide a better understanding of your design, programming, and analytical skills. As with any programming project, you will need to balance the amount of time spent on development against the quality of your result. We ask you not to expend more than two **calendar days**.

Please email a Github, Dropbox or Google Drive link with your solution and a readme file explaining if it is required to follow any additional step to run it. It would be great to run your solution as quickly as possible with very few additional steps.

You must use React or Angular (1.x or 2.x) for the front-end.

For the back-end, it is up to you to work with the technology you want. If you have previous experience with Node or ASP.NET, we encourage you to go down this path.

You can add any library/framework and use the database you prefer.

Game of drones game specification

In *Game of Drones* there are two players trying to conquer each other.

Players take turns to make their move, choosing Paper, Rock or Scissors. Each move beats another, just like the game "Paper, rock, scissors".

Like so:

- Paper beats Rock
- Rock beats scissors
- Scissors beat Paper

The first player to beat the other player 3 times wins the battle.

The website must have the following behavior:

1. Inputs for each player to enter his name. (Only two players) and a start button to begin the game.
2. Once the game begins, each player choose one of the possible moves.
    a. First, player1 pick his move, then player2.
    b. The system computes the result of the play.
        i. (The game happens on the same computer for both players. It is not required to create a true online game. Both players share the computer, and the system asks each player for their move assuming the other player looks away while the other selects the move)
    c. The result of each round should be displayed somewhere in the screen, so that players can know the game score while they are playing.
3. Step #2 repeats until one of the players wins three times. This player will be the winner of the game.
4. Once the game has finished, a Play Again button shows to start a new game.

Optional features

Certifications and Courses

The result of each game should be stored somewhere to keep track of games won by each player. We would like to know how many games a player has won and show them in a page.

Rules configuration

We would like to be able to change the possible moves in runtime. This means that after a game completes, one could change the default move rules and have more moves added or change the way each move beats the other. We are interested in a simple solution easy to implement.

Here is an example of a typical game:

```
{
    moves: [
        { move: "paper", kills: "rock"},
        { move: "rock", kills: "scissors"},
        { move: "scissors", kills: "paper"}
    ]
}
```

However, you could change to the following configuration:

```
{
    moves: [
        { move: "paper", kills: "rock"},
        { move: "rock", kills: "scissors"},
        { move: "scissors", kills: "string"},
        { move: "string", kills: "dog"},
        { move: "dog", kills: "paper"},
    ]
}
```

It might happen that no player wins a round. Nothing happens in that case and the game continues to the next round.

Evaluation

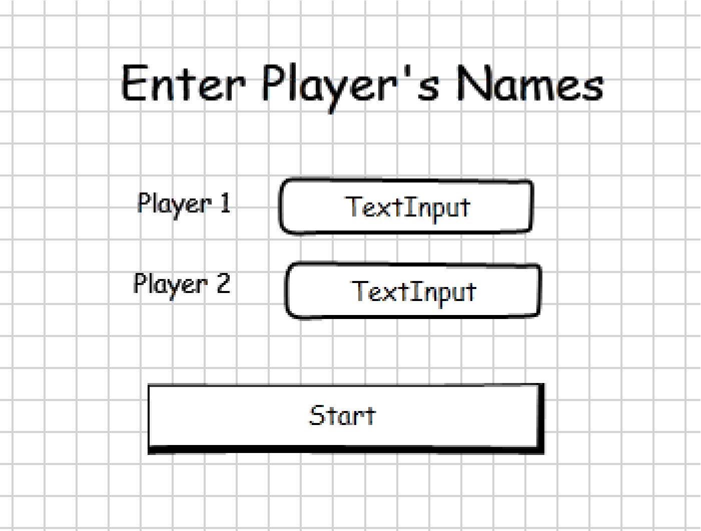
We are aware that the time for developing the solution is short, so you will have to find a balance between the following aspects:

- Front-end and back-end architecture
- Best use of Angular features
- UI design
- Clean code
- Unit and E2E tests
- Easy startup
- Optional features

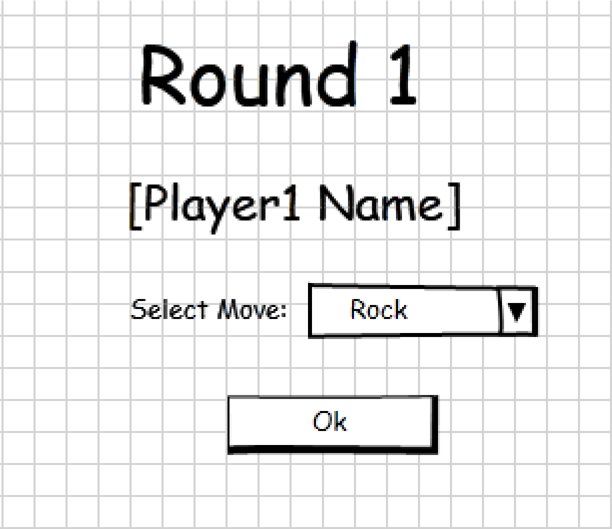
Mocks

Here is a mock of each screen. You are free to modify the look and feel of the screen as you please.

To start a game, the system asks for the name of each player.



The game starts with Round1. The system asks Player1 for a move. Replace [Player1 Name] with player’s name.

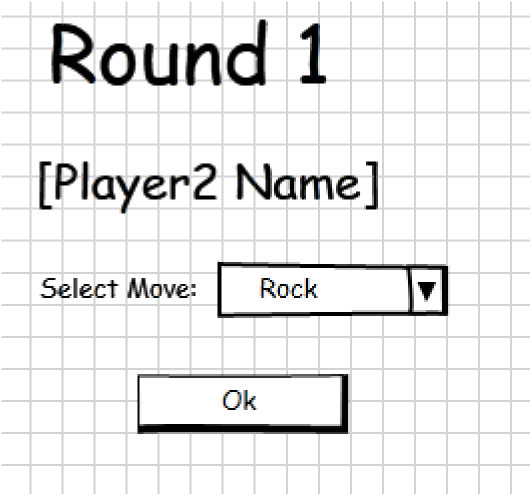


Round 1

[Player1 Name]

Select Move: Rock

After Player1 selects a move, then the system asks Player2 for a move (same round).

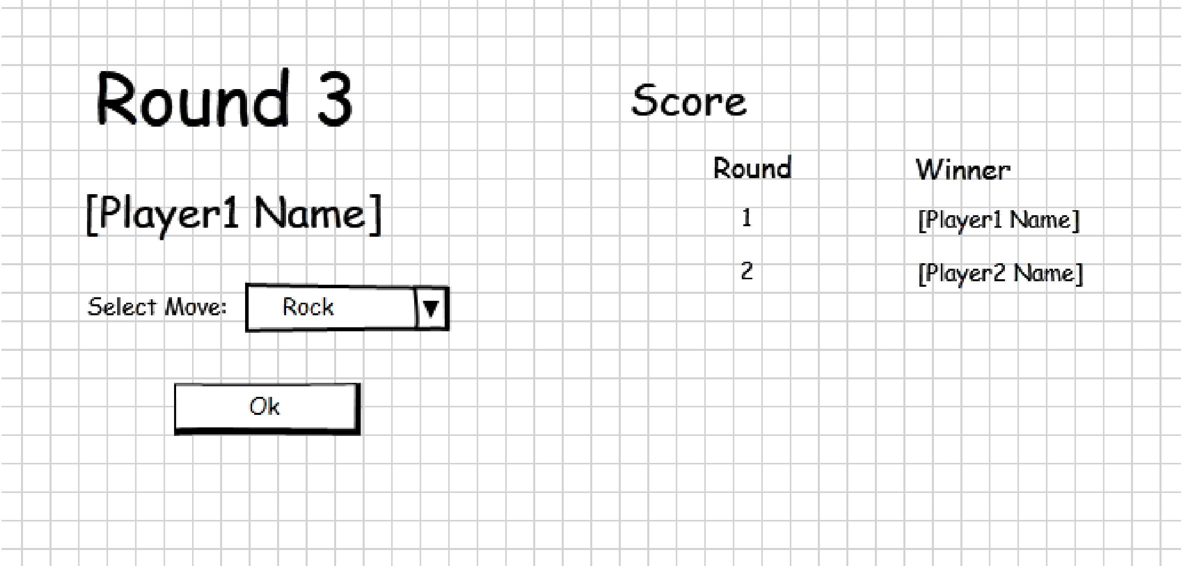


Round 1

[Player2 Name]

Select Move: Rock

Winners of each round are displayed somewhere on the screen. Here we see Round #3, and to the right the score of the previous rounds:



Round 3

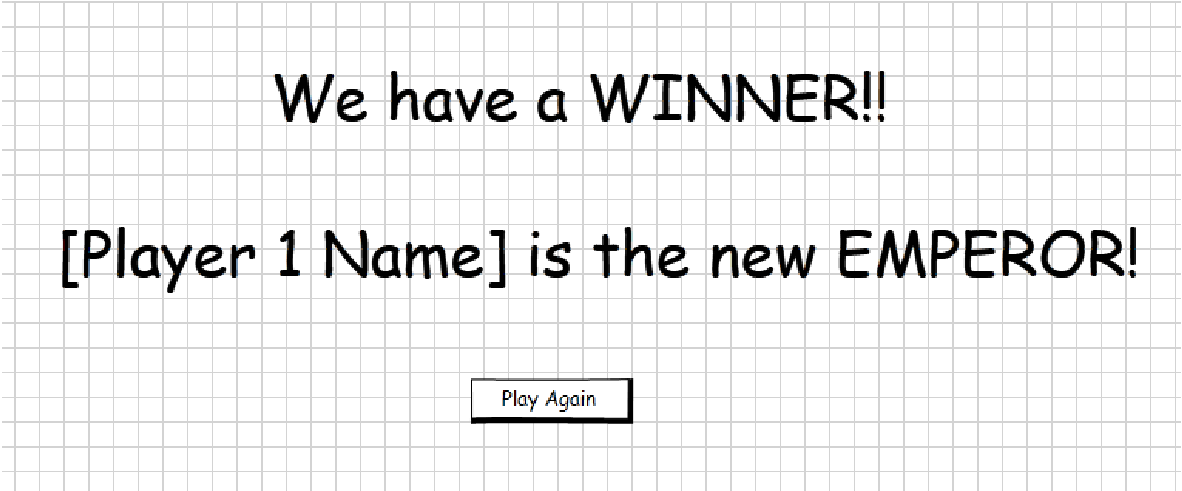
[Player1 Name]

Select Move: Rock

Score

Round	Winner
1	[Player1 Name]
2	[Player2 Name]

When a player reaches three wins, he is the winner, and the following screen displays:



We have a WINNER!!

[Player 1 Name] is the new EMPEROR!

Clicking the Play Again button, the systems takes you to the first screen.