Proves paradox.

1. Randomly choose the door that has the car behind it.
2. Let the contestant choose a door. This is done using a random number generator.
3. Choose the door to open to the contestant. This can’t be the door with the car, or the door that the contestant has chosen. If the user has already chosen the door with the car, then there will be two possible doors that can be opened. In this case chose a door at random from these two, otherwise open the only door that fits these criteria.
4. Allow the contestant to change their selection. This is done by passing a parameter to indicate whether the user always changes their choice, or always sticks with their original choice. If the user in changing their selection, calculate the one remaining door that they can choose, and set their selection to that.

Create an API with following request parameter and generate a simulation as response.

Request format:

{

NumberOfAttempts: 1000;

doSwitch: false;

}

I have added Swagger to get the below functionality:

Then be able to press a button that starts the simulation by calling backend that performs the requested number of simulated games.

I have assumed the response result as it was not that clear in the document:

{

**"noOfSimulation": 1000,**

**"switch": false,**

**"noOfSwitchWin": 672,**

**"noOfStayWin": 328**

}

Future Implementation:

1. Add error handling in the project:
2. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-2.1
3. Do CI/CD AzureDevops pipeline, deployment on Azure Cloud.
4. Add more test cases in Unit test project