1. Created a minimum viable product that emulates the EA simulation feature in FIFA(21), and better predicts the outcome of a particular match, by curtailing certain impactful factors such as the 'home advantage'.

The FIFA's sim feature relies heavily on the venue of a particular match. There exists a certain advantage to the teams playing at home even in real life, but the same emulation is overdone in the video games. Teams in the top of the bottom half of the table (11th,12th,13th) playing at home against a team in the top half will have very little chance of losing. This is not so in real life. Most teams would get a comfortable win against the bottom half teams.

Following figure shows the simulation of a bottom half team (Newcastle) playing at home against a top half team (Chelsea). Even though Chelsea is a better team, and most likely would have gotten the three points rather comfortably in a real match, it is not so in the FIFA simulation due to the home factor. In the simulation, Newcastle goes on to win the penalties.

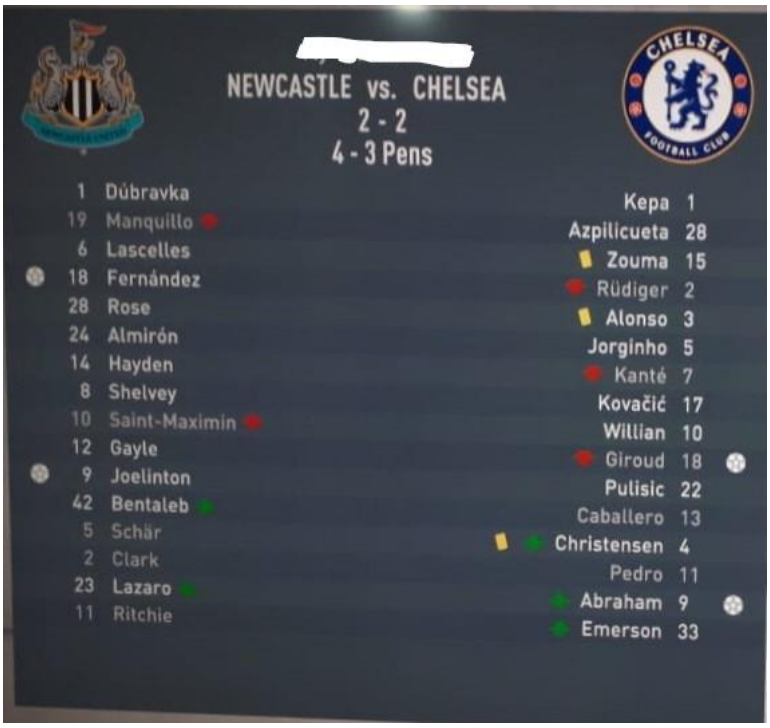

Figure: FIFA simulation of Newcastle United[H] vs Chelsea[A].

```
In [13]: runfile('C:/Users/farha/Desktop/egamingwithbookingsinjuries.py', wdir='C:/
Users/farha/Desktop')

Chelsea home? no
25' Newcastle United 0 - 1 Chelsea
27' Newcastle United 0 - 2 Chelsea
30' Newcastle United 0 - 3 Chelsea

In [14]: runfile('C:/Users/farha/Desktop/egamingwithbookingsinjuries.py', wdir='C:/
Users/farha/Desktop')

Chelsea home? no
48' Newcastle United 0 - 1 Chelsea
50' Newcastle United 0 - 2 Chelsea
62' Newcastle United 1 - 2 Chelsea

In [15]: runfile('C:/Users/farha/Desktop/egamingwithbookingsinjuries.py', wdir='C:/
Users/farha/Desktop')

Chelsea home? no
I. Hayden has been booked
20' Newcastle United 0 - 1 Chelsea
37' Newcastle United 0 - 2 Chelsea

In [16]: runfile('C:/Users/farha/Desktop/egamingwithbookingsinjuries.py', wdir='C:/
Users/farha/Desktop')

Chelsea home? no
33' Newcastle United 1 - 0 Chelsea
```

Figure: Product simulation of Newcastle United[H] vs Chelsea[A].

Simulating the same teams at the same venue using the product, yields the results as depicted in the second figure. Four out of the five matches are won by Chelsea (the away team), thereby reducing the home advantage factor down to a relatively realistic level.

The creation of the product can be divided into five functions:

i. kickoff
ii. probs
iii. midfield
iv. attack
v. score

i. kickoff:

This function determines which team will get more of the ball, and more chances to go through midfield, attack and score. It makes use of the "overall" attribute of the best eleven players of each team to determine who gets more of the ball.

ii. probs:

This function determines the probability of an outcome (such as, the probability of a team getting through the opposition's midfield and into their defence). The entire code of product depends on this function. It makes use of the numpy package available in python, since the simulation of any match depends on the probability with a certain bias.

iii. midfield:

This function determines if a certain team will be able to get past the opposition's midfield and into the final-third of the stadium. It works by calling on the 'probs' function with certain biases. These biases depend on the player's attributes pertaining to midfield position such as short and long passing. These are checked against the opposition team's player attribute pertaining to defending the midfield position such as marking and interceptions. Based on this, a bias is created.

iv. attack:

This function determines if a team's attackers (or forwards) will be able to get through the opposition's defence. This function is only invoked if the 'midfield' is true and the team has gotten through the opposition's midfield. It also works by calling 'probs' with a bias. The attributes used for this function are dribbling, control, crossing, acceleration for the attacking team and sliding tackle, standing tackle, power, strength and marking for the defending team.

v. score:

This function determines if a team has scored a goal. This is only invoked if the 'attack' is true. The attributes used for this function include shot power, long shots, volleys, finishing etc for the attacking team, and goalkeeper diving, handling, positioning, speed etc for the defending team.

All five of the functions are user-defined but make use of in-built functions in the numpy package (in python) to predict the probability of a certain event happening. Needless to say, the entire structure of product relies heavily on the probability factor since a simulation works on probability anyway. The biases for the probability depend on the function described above.

2. The created product also fields the best possible eleven in a random simulation, unlike the EA sim feature.

The product adheres to the principle that a player is better if he has a higher "overall" attribute than others. The process of selecting the best eleven depends on this principle. For now, the default formation for all the teams that the product selects, is 4-3-3.

Since majority of the clubs follow the 4-3-3 formation or its variation, like 4-2-3-1, 4-1-3-2 or 4-3-2-1, there is little room for deviating from the real-life formation that a particular team follows.

The product assumes that all the teams would need one goalkeeper, four defenders, three midfielders and three forwards, it ignores the players whose position has already been filled, even though their overall attribute is better.

The full backs, and full wing backs are considered defenders; central defensive and attacking players are considered midfielders; and centre forwards, supporting strikers and wingers are considered as strikers by the product.

3. Created a product that simulates one full season of a league (premier league) and predicts the position of each club along with the points achieved by the end of the season.

In order to ascertain the accuracy of product in regards to the real-life football, the English Premier League has been simulated for an entire season. Since the player's ratings and attributes are similar to the 2019-20 season of the EPL, the final table of this season is used to check for similarity and variances. The demoted teams of 2019-20 season (Bournemouth, Watford and Norwich City) have been replaced with the promoted teams of 2020-21 season (West Bromwich Albion, Leeds United and Fulham). Upon three iterations, following are the accuracy rates:

Table: Accuracy of product simulation.

	Club Name	Accuracy (%)
1	Liverpool	100
2	Manchester City	100
3	Manchester United	95
4	Chelsea	95
5	Leicester City	95
6	Tottenham Hotspur	85
7	Wolverhampton Wanderers	90
8	Arsenal	90
9	Sheffield United	50
10	Burnley	80
11	Southampton	80
12	Everton	95
13	Newcastle United	85
14	Crystal Palace	75
15	Brighton & Hove Albion	75
16	West Ham United	75
17	Aston Villa	90
18	Bournemouth	85

	Club Name	Accuracy (%)
19	Watford	90
20	Norwich City	80

The table is ranked based on the English Premier League 2019-20 season. It shows the accuracy of teams finishing in that particular place by product simulation (three iterations). Half the teams have more than 90% of accuracy rate. Most of the teams' position was predicted correctly by the product (over three iterations), **averaging the accuracy to 85.5%.**