The following goes through an example of how to use the API with the fetch web API in a javascript file.

First we define the base url that is the beginning of all of the endpoints in the API:

```
const baseUrl =  'https://ec2-54-215-217-154.us-west-1.compute.amazonaws.com/'
```

Next we define generic post, put and get objects that will be used to make all of our fetch requests:

```
const postObject = {
    method: 'POST',
    headers: {'Content-Type': 'application/json'},
    credentials: 'include'
}

const getObject = {
    method: 'GET',
    credentials: 'include'
}

const putObject = {
    method: 'PUT',
    headers: {'Content-Type': 'application/json'},
    credentials:  'include'
}
```

Now we define functions that will make the actual fetch requests:

```
async function postJSONResponse(url, postObj, data){
    postObj.body = JSON.stringify(data);
    let response = await fetch(url, postObj);
    return await response.json();
}

async function getJSONResponse(url, getObj){
    let response = await fetch(url, getObj);
    return await response.json();
}

async function putJSONResponse(url, putObj, data){
    putObj.body = JSON.stringify(data);
    let response = await fetch(url, putObj);
    return await response.json();}
```

Let's create a function through which the user will interact with the API:

```
async function userInteraction(userName, password){

}
```

The userName variable is a string with the user's desired username and password is a string with their desired password. The rest of the code in this document should be placed into the body of the userInteraction function.

The first thing a user will want to do is to create an account:

```
let url = baseUrl + 'users/signup'
let data = {
    user_name: userName,
    password: password
}
let jsonResponse = await postJSONResponse(url, postObject, data);
console.log(jsonResponse);
```

This will create an account with the username and password that the user chose (assuming that the username is not already in use). The console should display the following object:

{user_name: (the chosen username)}

This lets the user know that they have successfully created an account with the username and password that they put in.

Next the user will want to login to their account:

```
url = baseUrl + 'users/login';
jsonResponse  = await postJSONResponse(url, postObject, data);
console.log(jsonResponse)
```

The console should display the following object:

{message: 'You are now logged in as (the chosen username)}

This lets the user know that they are logged in.

Now that they are logged in, the user can start a game:

```
url = baseUrl + 'users/startgame'
data = {}
jsonResponse = await postJSONResponse(url, postObject, data);
console.log(jsonResponse);
```

The console should display the following object:

```
{
  gameNumber: 1,
  dateCurrent: “2020-09-14",
  maxDate: "2021-02-23",
  balance: 10000
}
```

This lets the user know that they have successfully started a game and gives them some information about the game.

While being logged in, the user can view their account at any time:

```
url = baseUrl + 'users/account';
jsonResponse = await getJSONResponse(url, getObject);
console.log(jsonResponse);
```

The console should display the following object:

```
{
  userName: (the chosen username),
  bestGame: null,
  latestGameNumber: 1,
 ongoing: true
}
```

Before placing a bet, the user will want to view the odds for some future games:

```
    url = baseUrl + 'users/odds?startDate=2020-10-17&&endDate=2020-10-19';
    jsonResponse =  await getJSONResponse(url, getObject);
    console.log(jsonResponse);
```

The user has requested to view the odds for games that take place between 2020-10-17 and 2020-10-19 inclusive. The console should now display the following array:

```
0: {gameID: 58936, gameDate: "2020-10-17", homeTeam: "Chelsea", awayTeam: "Southampton", homeOdds: "0.36", …}
1: {gameID: 58941, gameDate: "2020-10-17", homeTeam: "Manchester City", awayTeam: "Arsenal", homeOdds: "6.50", …}
2: {gameID: 58938, gameDate: "2020-10-17", homeTeam: "Everton", awayTeam: "Liverpool", homeOdds: "0.75", …}
3: {gameID: 58942, gameDate: "2020-10-17", homeTeam: "Newcastle United", awayTeam: "Manchester United", homeOdds: "0.35", …}
4: {gameID: 58937, gameDate: "2020-10-18", homeTeam: "Crystal Palace", awayTeam: "Brighton and Hove Albion", homeOdds: "0.53", …}
5: {gameID: 58940, gameDate: "2020-10-18", homeTeam: "Leicester City", awayTeam: "Aston Villa", homeOdds: "0.13", …}
6: {gameID: 58944, gameDate: "2020-10-18", homeTeam: "Tottenham Hotspur", awayTeam: "West Ham United", homeOdds: "0.62", …}
7: {gameID: 58943, gameDate: "2020-10-18", homeTeam: "Sheffield United", awayTeam: "Fulham", homeOdds: "0.31", …}
8: {gameID: 58939, gameDate: "2020-10-19", homeTeam: "Leeds United", awayTeam: "Wolverhampton Wanderers", homeOdds: "1.64", …}
9: {gameID: 58945, gameDate: "2020-10-19", homeTeam: "West Bromwich Albion", awayTeam: "Burnley", homeOdds: "2.21", …}
length: 10
__proto__: Array(0)
```

The user may now want to view additional stats by using the stats endpoints. Then being armed with the stats and the odds, they will probably decide to place a bet:

```
url = baseUrl + 'users/bet'
data = {
    gameID: '58941',
    predictedResult: 1,
    amountBet: 1000
}
jsonResponse = await postJSONResponse(url, postObject, data);
console.log(jsonResponse);
```

Having liked the odds for Manchester City against Arsenal on ‘2020-10-17’, the user has bet 1000 on the home team. The console should now display the following object:

```
amountBet: 1000
beforeBalance: 10000
betNumber: 1
dateMade: "2020-09-14"
gameID: "58941"
gameNumber: 1
predictedResult: 1
```

Having placed a bet, the user can place more bets or move forward in time to a future date:

```
url = baseUrl + 'users/goforward'
data = {date: '2020-10-18'}
jsonResponse = await putJSONResponse(url, putObject, data);
console.log(jsonResponse);
```

The user has requested to move forward to the date '2020-10-18'. The API will now move forward in time and resolve all of the outstanding bets for games that occurred before '2020-10-18'. The console should now display the following object:

```
balance: 16500
dateCurrent: "2020-10-18"
finished: false
gameNumber: 1
maxDate: "2021-02-23"
numberBets: 1
```

While being logged in and playing a game, a user can view the status of their current game at any time:

```
url = baseUrl + 'users/viewgame'
jsonResponse = await getJSONResponse(url, getObject);
console.log(jsonResponse);
```

The console should display the following object:

```
balance: "16500.00"
dateCurrent: "2020-10-18"
finished: false
maxDate: "2021-02-23"
numberBets: 1
```

The user may also view all of the bets that they have made for the current game:

```
url = baseUrl + 'users/viewbets'
jsonResponse = await getJSONResponse(url, getObject);
console.log(jsonResponse);
```

The console should display the following array:

```
0: {betNumber: 1, gameID: 58941, predictedResult: 1, amountBet: "1000", profit: "6500.00", …}
length: 1
__proto__: Array(0)
```

At this point, the user could view more odds and stats and make more bets if they choose to. At any point, the user can end the game by moving forward past the maxDate '2021-02-23':

```
url = baseUrl + 'users/goforward'
data = {
    date: '2021-02-24'
}
jsonResponse = await putJSONResponse(url, putObject, data);
console.log(jsonResponse);
```

The console should display the following object:

```
balance: 16500
dateCurrent: "2021-02-23"
finished: true
gameNumber: 1
maxDate: "2021-02-23"
numberBets: 1
```

The game is finished. The user can verify that the necessary changes were made to his account:

```
url = baseUrl + 'users/account';
jsonResponse = await getJSONResponse(url, getObject);
console.log(jsonResponse);
```

The console should display the following object:

{
  userName: (the chosen username),
  bestGame: 1,
  latestGameNumber: 1,
  ongoing: false
}

Now that the user has finished the game, they are free to observe other players' games with the /observe endpoints, or they may start a new game with the /users/startgame endpoint. If the user decides that they are done using the app for now, they can logout:

```
url = baseUrl + 'users/logout';
jsonResponse = await getJSONResponse(url, getObject);
console.log(jsonResponse);
```

The console should display the following:

{message: "You are now logged out"}

This concludes this example of how to use the API. It was not meant to be exhaustive. For a complete description of all of the endpoints, please reference the documentation.