A decentralised and privacy-centric Fantasy Sports Platform

Existing fantasy sports players rely on a central fantasy sports platform such as Dream11 or MPL to play fantasy sports. We look to remove these central services which act as middlemen from this system with the help of a blockchain based service run by the players themselves, while maintaining popular features present in the central platforms.

Scope of the project

The smart contract can be deployed on etherium blockchain. It removes the need of Centralized Authority. If there is a central authority controlling the platform, they charge a commission on reward amount, using block chain we do away with any central authority and hence the participants can get the whole reward amount without deduction. Block chain makes the operations very transparent to ensure that there are no ill-practices or cheating going on. This transparency was not possible in the absence of blockchain.

The smart contract can be used at different scales, such as local games, or international level games such as IPL. This contract can handle users worldwide.

General Structure

We will be running the entire system on an Ethereum blockchain network, which will be maintained by the the participants. The players will decide their team before the game starts. The system will be updated about the team. When the game ends the collected reward amount will be distributed among the participants based in their team selection.

The flow of the app will be as follows-

- The people will bet on the cricket matches like it is done in Dream 11.
- Smart contract will receive the live data of each ball played by cricketers.
- Smart contract will decide who wins the bet and regulate the transaction.

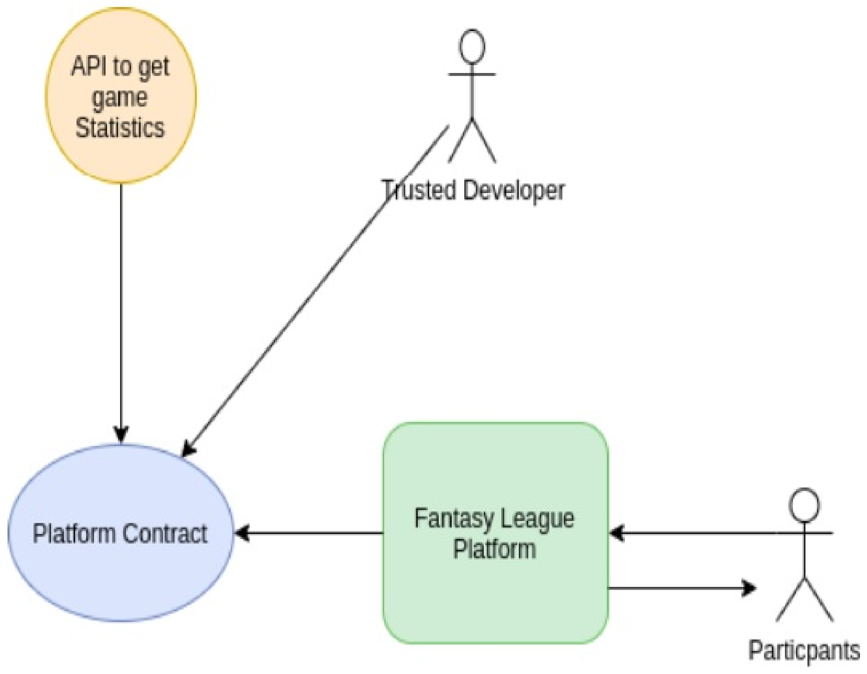

Fig. 2. Architecture [2]

Implementation Details

Start Game

To begin a new game this function is called. In game variables like Team size, and points corresponding to runs and wickets are initialized.

```
function startGame(uint256 teamSize, uint256 runMul, uint256 wicketMul)
```

Add Players To The Game

Players who are going to play in the game are added. The owners can choose from these players for their fantasy team.

```
addPlayerToGame(string memory playerName)
```

List Players

The owners can view the details of the listed player using this function and then create their team.

```
listPlayers()
```

Players Added

Once all players are added this function is called to notify that the owners can now start forming their teams.

```
playersAddedToGame()
```

Add Fantasy Team

Owners who want to play in the fantasy game can form their own team. One owner can only form one team for the game.

```
addFantasyTeam(string memory teamName)
```

Add Players To Fantasy Team

Once the fantasy team is formed the owner can add players to their team. The player ID is passed as an argument.

```
addPlayerToFantasyTeam(uint256 playerID)
```

Set Captain

Once the fantasy team is formed the owner can assign players to be captain and vice captain.The player ID of captain and vice captain is passed as an argument.

```
setCaptains(uint256 captainID, uint256 viceCaptainID)
```

Place Bid

After formin the teams owners can start bidding. The reward for the woner will be proportionate to their placed bid and the score of the team.

```
bid()
```

Begin game

When this is called the team formation and bidding stops and the actual game begins.

```
gameBegins()
```

Player Stats

The stats for each player is updated using the API. This function is called for each player ID and there performance stats are passed as arguments.

```
setplayerStats(uint256 playerID, uint256 runs, uint256 wickets)
```

Stop Game

Once stats of all players are updated the game is stopped using this function.

```
 gameStops()
```

Calculate Score And Distribute Reward

After the game ends. Score for each fantasy team is calculated based on performance of players in the team. Each team owner then recieves a reward amount proportionate to the team score and the submitted bid.

```
calculateTeamsScore()
```

How To Interact

The following are some sample tests on how to use the smart contract. A lot of checking has been done everywhere to make sure that only the authorised users can execute key functions.

You need to have NodeJS and Truffle installed. Instructions are present in the links attached.

Required Packages

On the terminal, type

```
npm install package.json
```

Steps to Run

On the terminal, type

```
truffle develop
```

In the truffle console,

```
deploy
let instance = await FantasySports.deployed();
```

After creating the instance, you can interact with the smart contract by calling the available functions.

Test 1

On the terminal, type

```
truffle develop
```

In the truffle console,

```
deploy
let instance = await FantasySports.deployed();
instance.startGame(2, 1, 50);
instance.addPlayerToGame("Player1");
instance.addPlayerToGame("Player2");
instance.addPlayerToGame("Player3");
instance.addPlayerToGame("Player4");
instance.playersAddedToGame();
instance.addFantasyTeam("Team1", {from: accounts[0]});
instance.addFantasyTeam("Team2", {from: accounts[1]});
instance.addPlayerToFantasyTeam(0, {from: accounts[0]});
instance.addPlayerToFantasyTeam(1, {from: accounts[0]});
instance.addPlayerToFantasyTeam(2, {from: accounts[1]});
instance.addPlayerToFantasyTeam(3, {from: accounts[1]});
instance.setCaptains(0, 1, {from: accounts[0]});
instance.setCaptains(2, 3, {from: accounts[1]});
instance.bid({from: accounts[0], value: 10000000000000000000});
instance.bid({from: accounts[1], value: 10000000000000000000});
instance.gameBegins();
instance.setplayerStats(0, 10, 20);
instance.setplayerStats(1, 1, 2);
instance.setplayerStats(2, 1, 2);
instance.setplayerStats(3, 1, 2);
instance.gameStops();
instance.calculateTeamsScore();
web3.eth.getBalance(accounts[0]);
web3.eth.getBalance(accounts[1]);
```

This is a very simple test to check if the basic add and view operations work and one can observe that correct reward amount is bieng transferred to the owners.

Test 2

On the terminal, type

```
truffle develop
```

In the truffle console,

```
deploy
let instance = await FantasySports.deployed();
instance.startGame(2, 1, 50);
instance.addPlayerToGame("Player1");
instance.addPlayerToGame("Player2");
instance.addPlayerToGame("Player3");
instance.addPlayerToGame("Player4");
instance.playersAddedToGame();
instance.addFantasyTeam("Team1", {from: accounts[0]});
instance.addFantasyTeam("Team2", {from: accounts[1]});
instance.addPlayerToFantasyTeam(0, {from: accounts[0]});
instance.addPlayerToFantasyTeam(1, {from: accounts[0]});
instance.addPlayerToFantasyTeam(2, {from: accounts[1]});
instance.addPlayerToFantasyTeam(3, {from: accounts[1]});
instance.setCaptains(0, 1, {from: accounts[0]});
instance.setCaptains(2, 3, {from: accounts[1]});
instance.bid({from: accounts[0], value: 10000000000000000000});
instance.bid({from: accounts[1], value: 10000000000000000000});
instance.gameBegins();
instance.setplayerStats(0, 0, 0);
instance.setplayerStats(1, 0, 0);
instance.setplayerStats(2, 0, 0);
instance.setplayerStats(3, 0, 0);
instance.gameStops();
instance.calculateTeamsScore();
web3.eth.getBalance(accounts[0]);
web3.eth.getBalance(accounts[1]);
```

This is a test case where all the fantasy teams score 0 points. In such a case all the owners will be returned their bids.

Test 3

On the terminal, type

```
truffle develop
```

In the truffle console,

```
deploy
let instance = await FantasySports.deployed();
instance.startGame(2, 1, 50);
instance.addPlayerToGame("Player1");
instance.addPlayerToGame("Player2");
instance.addPlayerToGame("Player3");
instance.addPlayerToGame("Player4");
instance.playersAddedToGame();
instance.addFantasyTeam("Team1", {from: accounts[0]});
instance.addFantasyTeam("Team2", {from: accounts[1]});
instance.addPlayerToFantasyTeam(0, {from: accounts[0]});
instance.bid({from: accounts[0], value: 10000000000000000000});
```

In this testcase ann error will be thrown as the owner of Team1 can not bid as the team is still incomplete.

Test 4

On the terminal, type

```
truffle develop
```

In the truffle console,

```
 deploy
let instance = await FantasySports.deployed();
instance.startGame(2, 1, 50);
instance.addPlayerToGame("Player1");
instance.addPlayerToGame("Player2");
instance.addPlayerToGame("Player3");
instance.addPlayerToGame("Player4");
instance.playersAddedToGame();
instance.addFantasyTeam("Team1", {from: accounts[0]});
instance.addFantasyTeam("Team2", {from: accounts[0]});
```

Again in this testcase an error will be thrown as one owner ofcan not own 2 teams.

Team 1

- Manish (2018101073)
- Akshat Goyal (2018101075)
- Tanish Lad (2018114005)