SPORTS FANTASY APPLICATION - DISSERTATION SEZG628T

BITS Pilani
Pilani Campus

JASPREET SINGH
2021MT93340

Problem

Less Power to User

Less variety of types of game play available

Centralised servers

No transparency in terms of transactions happening on backend

Less interactions with live games

Real-time gameplay predictions not available

Solution

innovate achieve lead

Transparent Reward Process

Instant Verifiable Rewards distributed using Blockchain based Smart Contracts.

Real-time Data Monitoring

Backend running for live score interactions using APIs.

Immutable database

Blockchain transactions are immutable and no one can change it

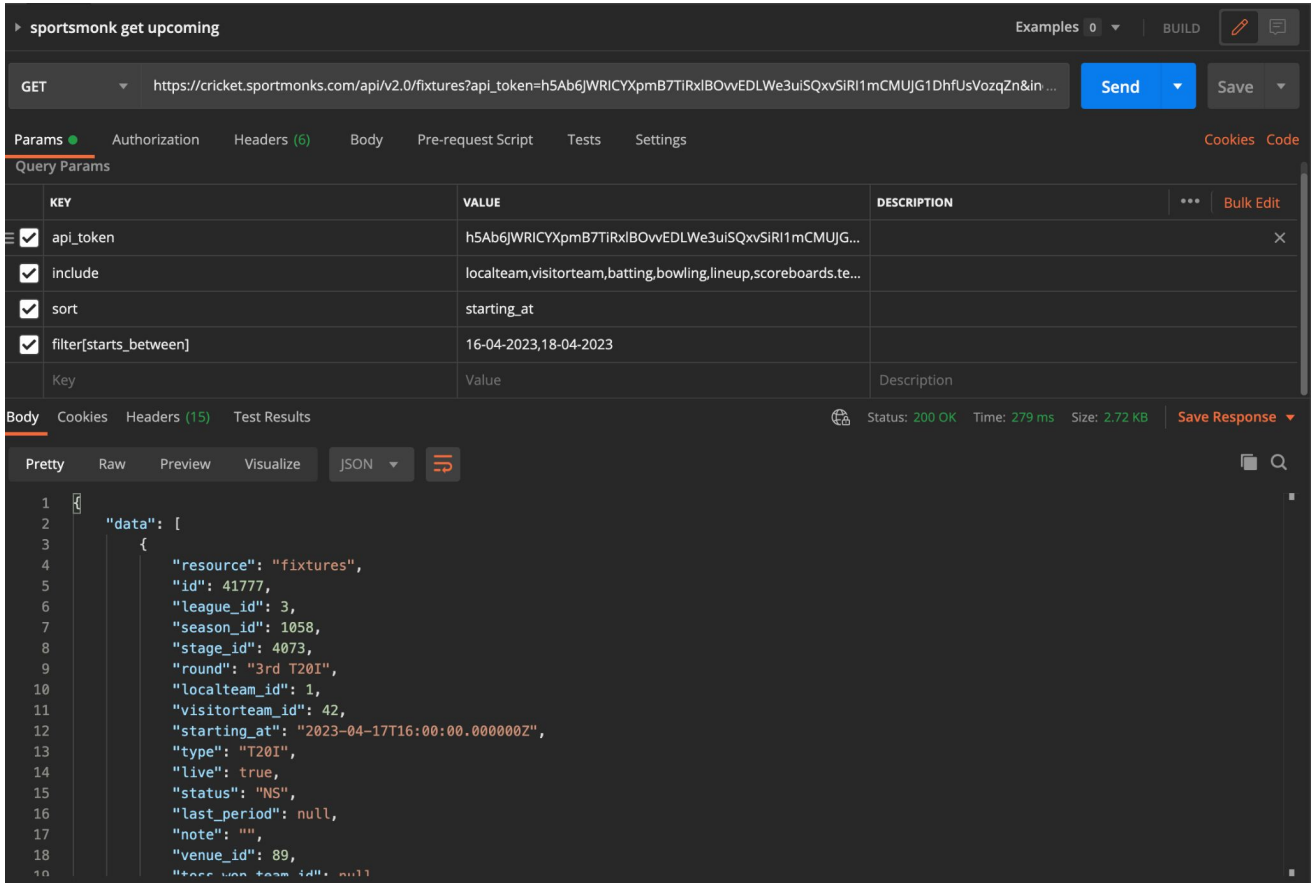
Machine Learning model for prediction

Confidence score for prediction on every option

Rationale

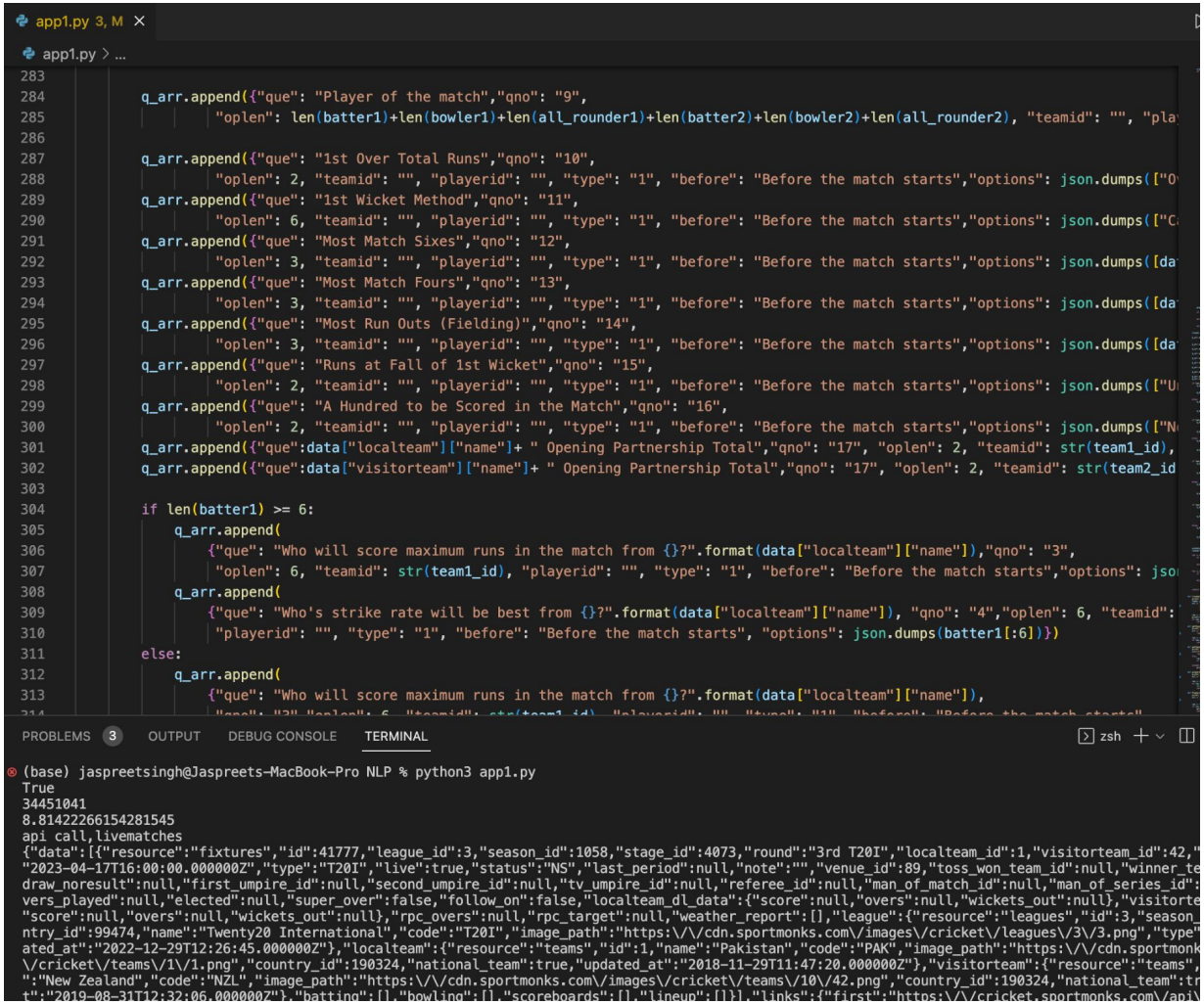
1) Generate quiz questions on-chain by a Python script. Any user can also create custom quiz.
2) Users participate in match questions on-chain using tokens via public key in Metamask.
3) Declare results automatically after the match finishes using Python scripts that use the SportsMonk free APIs for proof of concept.
4) Winnings are calculated based on the answers chosen by users while participating in the quiz.
5) Fetch the total winnings using their account address for transparency.
6) Retrieve user winnings in tokens.

SportsMonk APIs



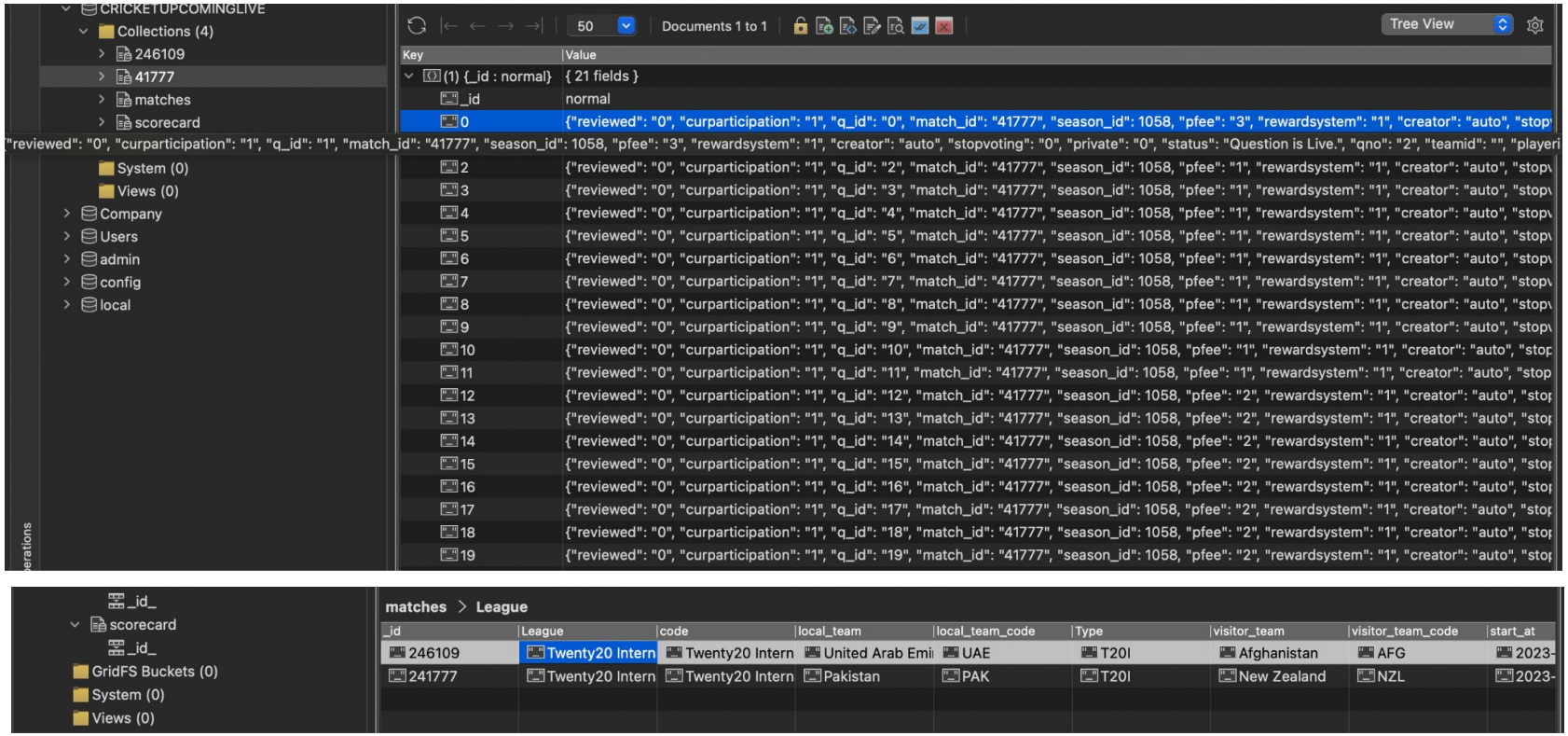
- Sportsmonk APIs are used to get upcoming cricket matches and all the data related.
- These APIs are used in minimal way to fetch and store data.

Python script creates questions



- Python script calls Sportsmonk APIs to get upcoming matches
- The same set of questions are created for every big match.
- MongoDB stores the data of questions.

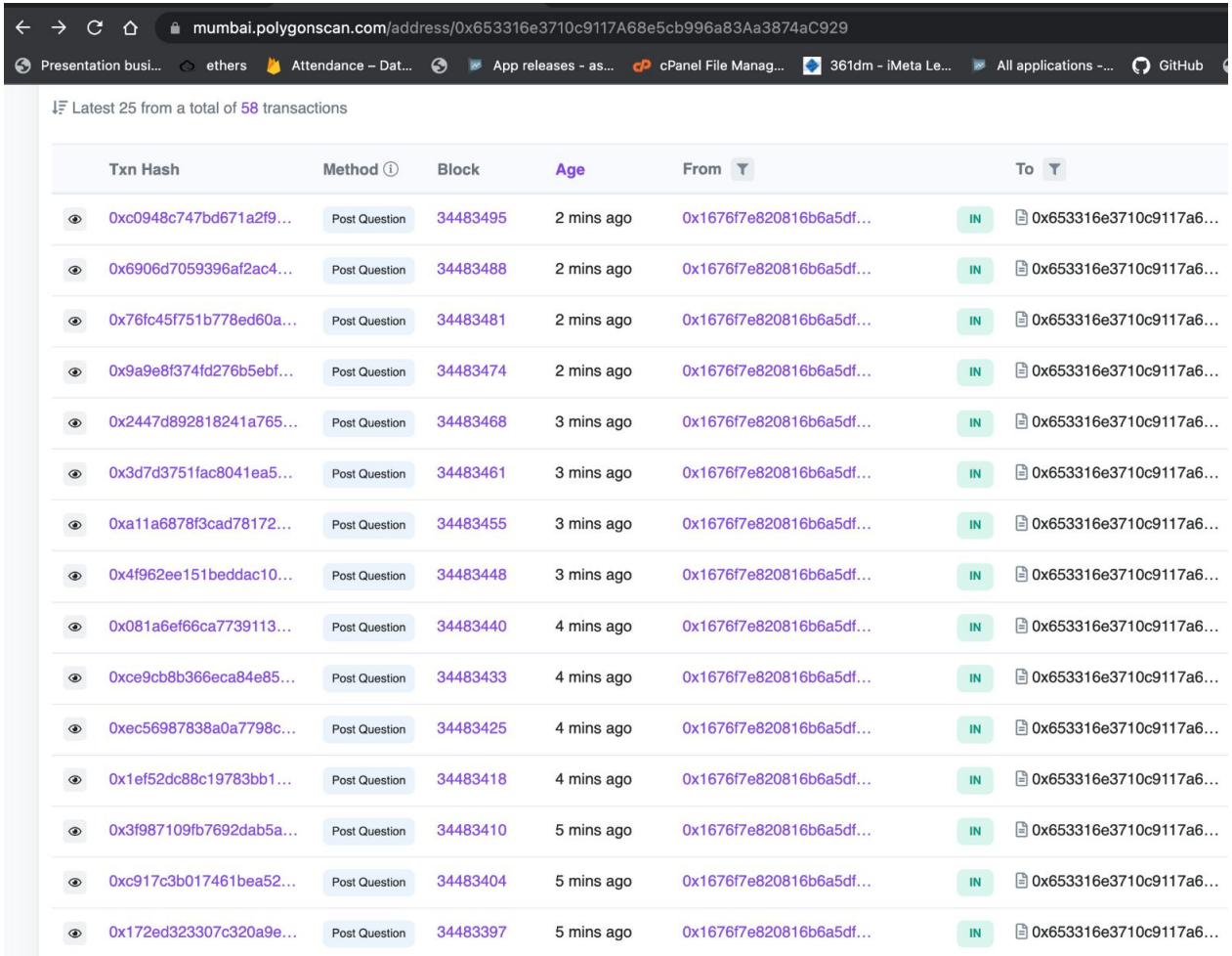
Offchain MongoDB



CRICKETUPCOMINGLIVE Database of MongoDB collections:

- match_id (storing data of match details) - keeps on adding collections.
- matches - contains all match ids of upcoming matches to be shown in app
- scorecard - to keep track of scores of live matches

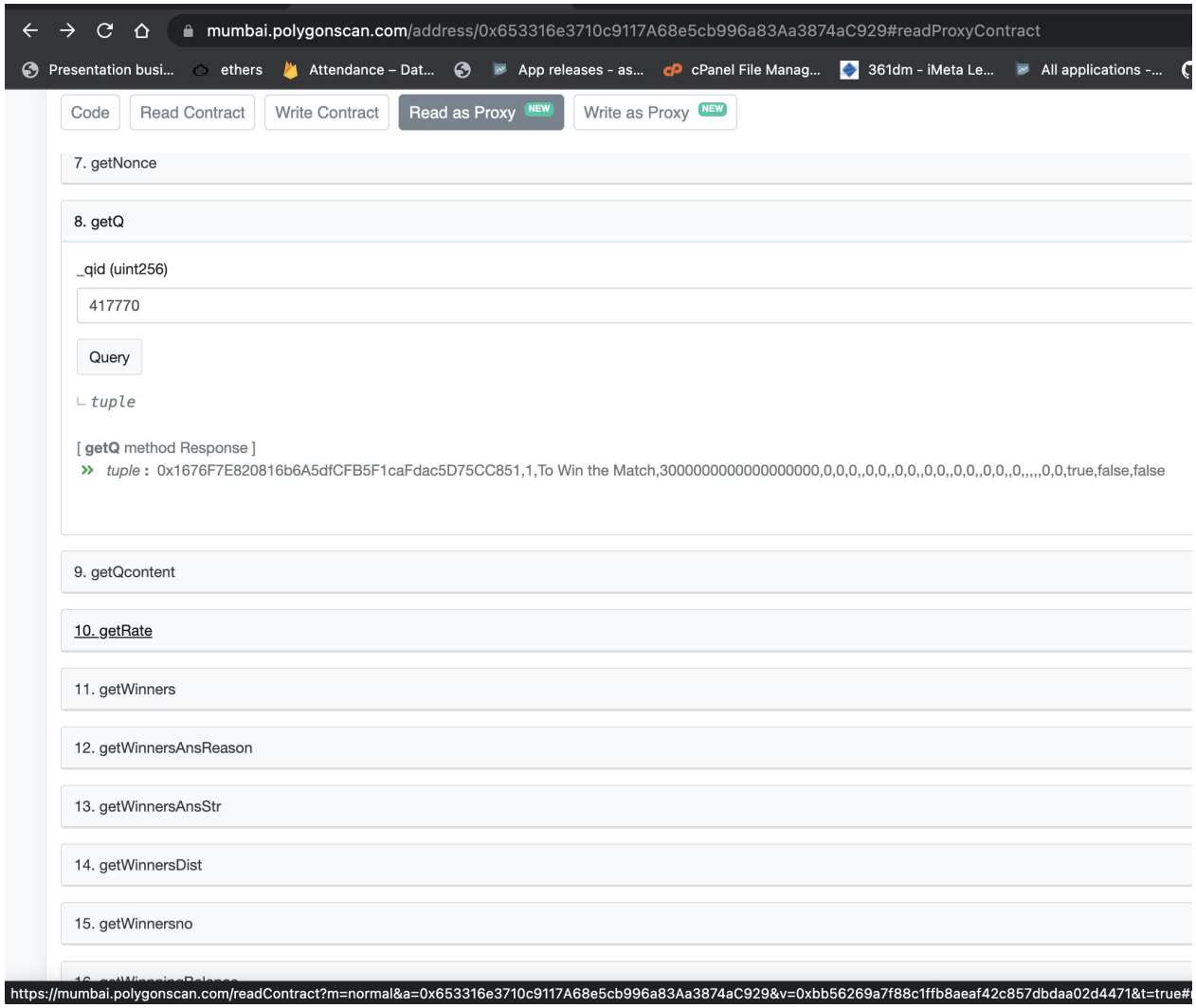
Questions created onchain



Txn Hash	Method	Block	Age	From		To
0xc0948c747bd671a2f9...	Post Question	34483495	2 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x6906d7059396af2ac4...	Post Question	34483488	2 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x76fc45f751b778ed60a...	Post Question	34483481	2 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x9a9e8f374fd276b5ebf...	Post Question	34483474	2 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x2447d892818241a765...	Post Question	34483468	3 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x3d7d3751fac8041ea5...	Post Question	34483461	3 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0xa11a6878f3cad78172...	Post Question	34483455	3 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x4f962ee151beddac10...	Post Question	34483448	3 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x081a6ef66ca7739113...	Post Question	34483440	4 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0xce9cb8b366eca84e85...	Post Question	34483433	4 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0xec56987838a0a7798c...	Post Question	34483425	4 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x1ef52dc88c19783bb1...	Post Question	34483418	4 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x3f987109fb7692dab5a...	Post Question	34483410	5 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0xc917c3b017461bea52...	Post Question	34483404	5 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...
0x172ed323307c320a9e...	Post Question	34483397	5 mins ago	0x1676f7e820816b6a5df...	IN	0x653316e3710c9117a6...

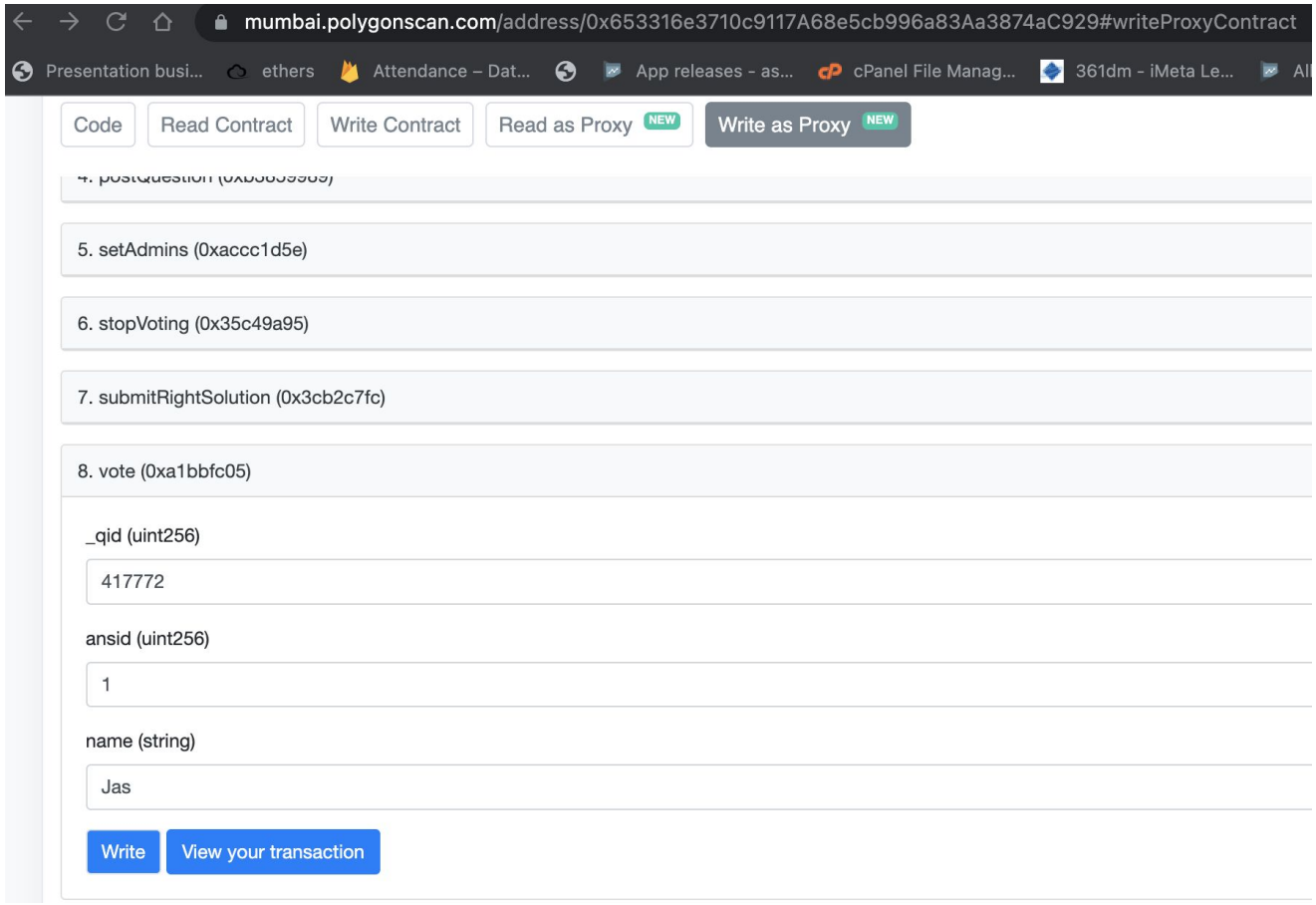
- Questions are created on smart contract using Web3 modules.
- Python script app1.py hits postQuestion API which calls smart contract function.
- Transactions can be viewed by anyone using explorer at Link with timestamps.

Questions data viewable onchain



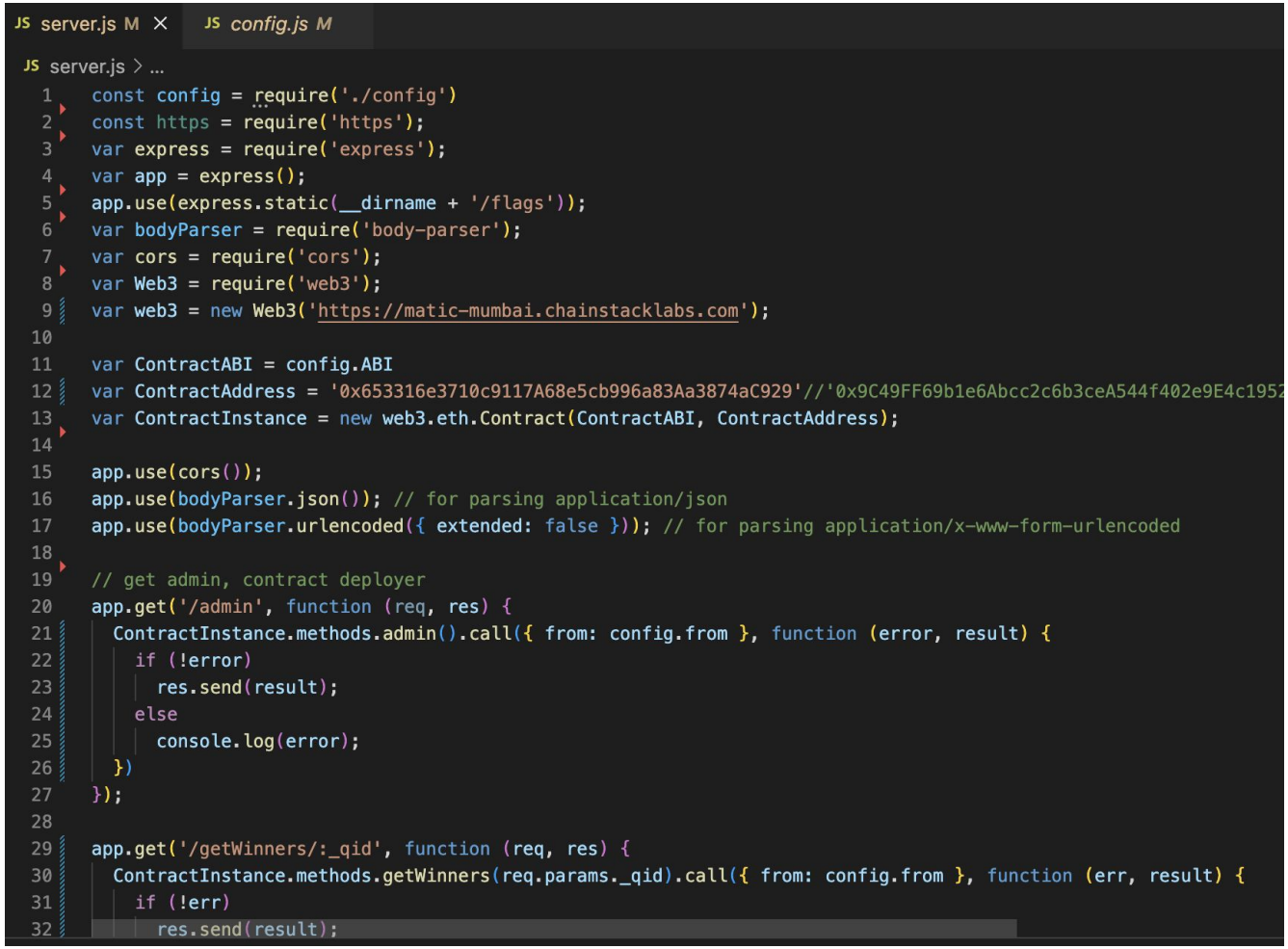
- Questions are visible and can be queried by anyone onchain using explorer
- The winners for each qid and reasons can also be viewed.
- Smart Contract Read in Explorer

Users can participate onchain



- Anyone with SF tokens can approve the smart contract to use it.
- User then calls smart contract function vote after that to participate in question ID. Link
- Participation fee is deducted and user is participated.

Inbuilt APIs for users

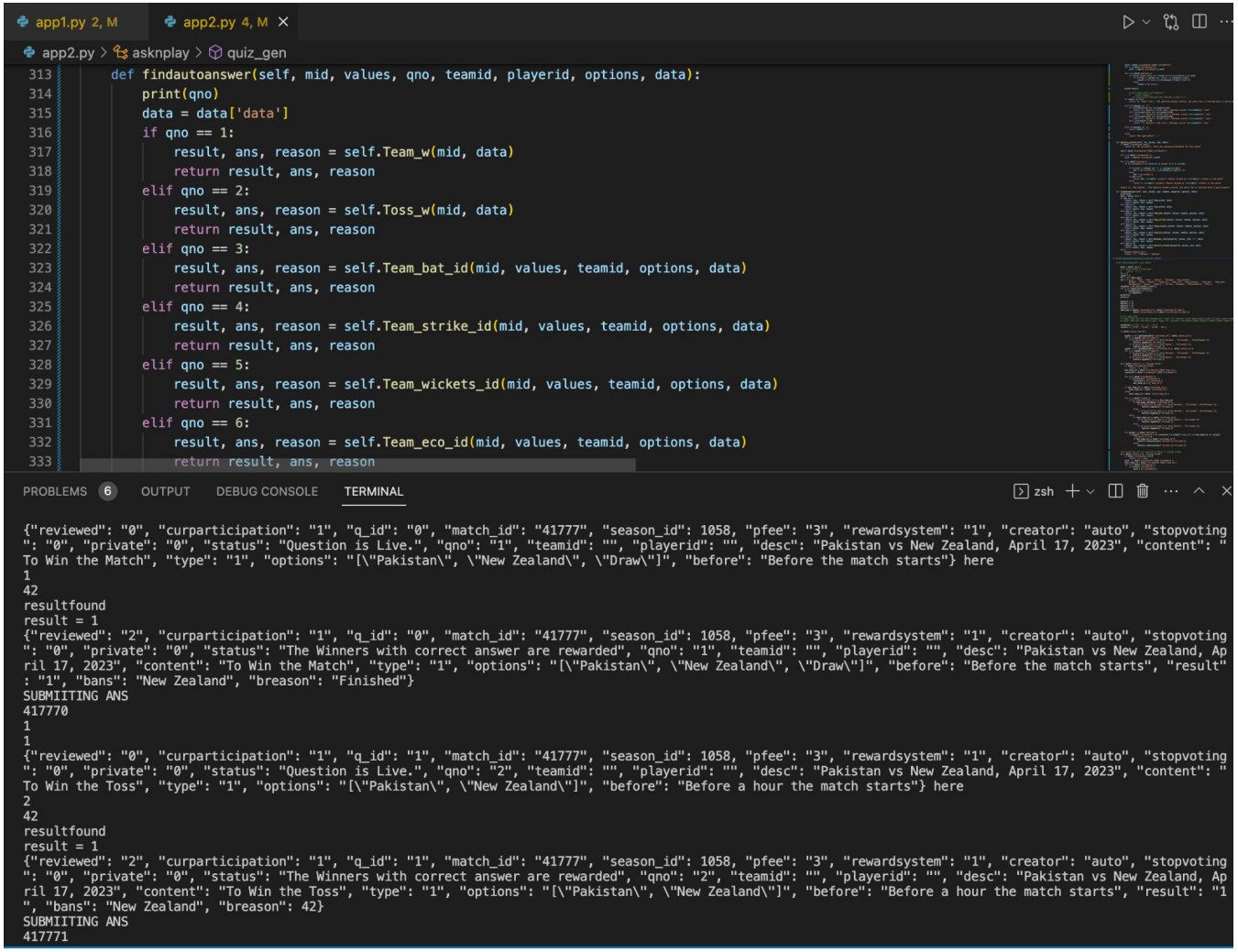


Server is deployed for helping:

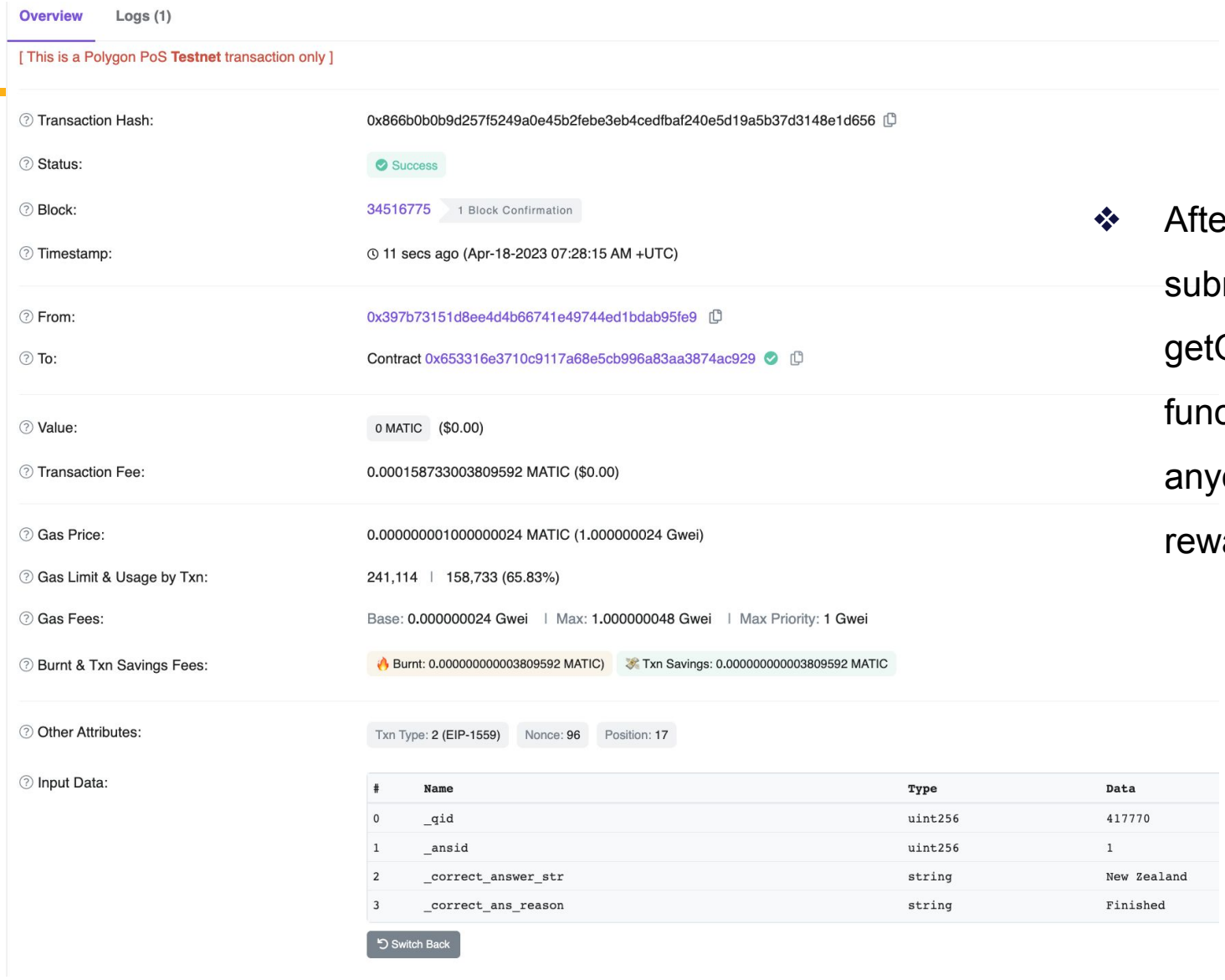
- Create accounts
- Create questions
- Participate in questions
- Stop participation time
- Submitting correct answer
- Retrieve winners

These APIs can be integrated with mobile app.

Python script submits answers onchain



- Python script calls Sportsmonk APIs to get results of matches
- MongoDB stores the data of results in CRICKET DB and removes from CRICKETUPCOMING DB.
- Solution is submitted onchain.



Overview | Logs (1)

[This is a Polygon PoS **Testnet** transaction only]

Transaction Hash:	0x866b0b0b9d257f5249a0e45b2febe3eb4cedfbaf240e5d19a5b37d3148e1d656
Status:	Success
Block:	34516775 1 Block Confirmation
Timestamp:	11 secs ago (Apr-18-2023 07:28:15 AM +UTC)
From:	0x397b73151d8ee4d4b66741e49744ed1bdab95fe9
To:	Contract 0x653316e3710c9117a68e5cb996a83aa3874ac929
Value:	0 MATIC ($0.00)
Transaction Fee:	0.000158733003809592 MATIC ($0.00)
Gas Price:	0.000000001000000024 MATIC (1.000000024 Gwei)
Gas Limit & Usage by Txn:	241,114 | 158,733 (65.83%)
Gas Fees:	Base: 0.000000024 Gwei | Max: 1.000000048 Gwei | Max Priority: 1 Gwei
Burnt & Txn Savings Fees:	Burnt: 0.000000000003809592 MATIC) Txn Savings: 0.000000000003809592 MATIC
Other Attributes:	Txn Type: 2 (EIP-1559) Nonce: 96 Position: 17

Input Data:

#	Name	Type	Data
0	_qid	uint256	417770
1	_ansid	uint256	1
2	_correct_answer_str	string	New Zealand
3	_correct_ans_reason	string	Finished

Switch Back

- After Solution is submitted onchain, getQidWinning function is called by anyone to pull in rewards for winners.

Rewards distributed onchain & event emitted

innovate achieve lead

Transaction Hash: 0x5d3b76a8a7bd947ff76cb3fd6b623daad96d670e98fae890a1f21589ef2e988a

Status: Success

Block: 34517371 1 Block Confirmation

Timestamp: 14 secs ago (Apr-18-2023 07:49:21 AM +UTC)

From: 0x397b73151d8ee4d4b66741e49744ed1bdab95fe9

Interacted With (To): Contract 0x653316e3710c9117a68e5cb996a83aa3874ac929

Tokens Transferred: From 0x653316e3710c9… To 0x397b73151d8ee… For 1 SportsFantas... (SD)

Value: 0 MATIC ($0.00)

Transaction Fee: 0.000181203005254887 MATIC ($0.00)

Gas Price: 0.000000001000000029 MATIC (1.000000029 Gwei)

Gas Limit & Usage by Txn: 275,370 | 181,203 (65.8%)

Gas Fees: Base: 0.000000029 Gwei | Max: 1.000000058 Gwei | Max Priority: 1 Gwei

Burnt & Txn Savings Fees: Burnt: 0.000000000005254887 MATIC) Txn Savings: 0.000000000005254887 MATIC

Other Attributes: Txn Type: 2 (EIP-1559) Nonce: 98 Position: 26

Input Data:

#	Name	Type	Data
0	_qid	uint256	417772
1	participant	address	0x397B73151D8Ee4D4B66741E49744ed1BDAB95fe9

Switch Back

244

Address 0x653316e3710c9117a68e5cb996a83aa3874ac929

Name WinnerDeclared (uint256 qid, string name, address winner, uint256 rewardamt, uint256 ans, uint256 _type) View Source

Topics 0 0x5bcf5bc217a775f034679cdb974a5180ff85b8db07899fc7b7d87dd3ee15b44d

Data

Dec Hex

qid : 417772
name : Jas
winner : 0x397b73151d8ee4d4b66741e49744ed1bdab95fe9
rewardamt : 1000000000000000000
ans : 999
_type : 1

11. getWinners

_qid (uint256)

417772

Query

address[]

[getWinners method Response]

address[] : 0x397B73151D8Ee4D4B66741E49744ed1BDAB95fe9

12. getWinnersAnsReason

_qid (uint256)

417772

Query

string

[getWinnersAnsReason method Response]

string : Tom Latham is MoM

13. getWinnersAnsStr

14. getWinnersDist

15. getWinnersno

16. getWinnningBalance

Prediction of Player's score

```
# Step 1: Data collection
# http://www.howstat.com/cricket/Statistics/Players/PlayerProgressSummary_ODI.asp?PlayerID=3474
data = pd.read_csv('player_data.csv')

# Step 2: Data preprocessing
X = data[['Versus', 'Ground', 'Age']] # features
y = data['Batting Runs'] # target variable
X = pd.get_dummies(X, columns=['Versus', 'Ground']) # one-hot encode categorical features
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=0)
# split into training and test sets

# Step 3: Feature selection
# You can use techniques like correlation analysis, recursive feature elimination, or
# principal component analysis to select relevant features

# Step 4: Model training
rf = RandomForestRegressor(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)

# Step 5: Model evaluation
y_pred = rf.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# # # # Step 6: Prediction
next_match = pd.DataFrame({
    'Versus': ['Sri Lanka'],
    'Ground': ['R Premadasa Stadium'],
    'Age': [12520]
})

# One-hot encode categorical features in next_match
next_match = pd.get_dummies(next_match, columns=['Versus', 'Ground'])
# Add missing columns to next_match (since not all values were present in X_train)
missing_cols = set(X_train.columns) - set(next_match.columns)
for col in missing_cols:
    next_match[col] = 0
next_match = next_match[X_train.columns] # Reorder columns to match X_train

# Predict batting score for next match
next_match_pred = rf.predict(next_match)
print(f'Predicted score for next match: {next_match_pred[0]:.2f}')
```

```
Predicted score for next match: 60.72
```

- Random Forest Regression is used that trains the data based on decision trees.
- Predict the score of player before the match based on past performance metrics on:
  - Versus
  - Ground
  - Age
- Target variable
  - Batting Runs

Thank you