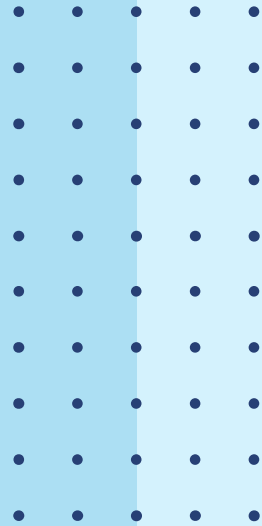
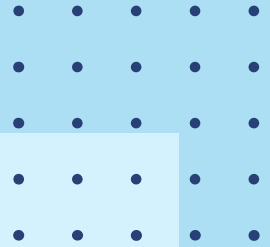


Blockchain's impact on the sports industry and fan engagement

Drishti Samvedi - D17B/63

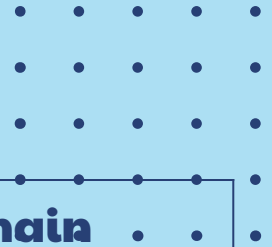


Introduction



- Blockchain is revolutionizing the sports industry by enhancing transparency, security, and efficiency across multiple facets.
- It has introduced innovative concepts like fan tokenization, allowing fans to engage actively with their favorite teams.
- Despite facing challenges, including regulatory hurdles, blockchain's potential to transform how sports organizations operate and interact with fans is increasingly evident, promising a more decentralized, transparent, and fan-centric future for the sports industry

Fan Engagement in Sports



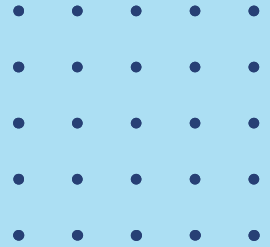
Parameter	Traditional Approach	Using Blockchain
Merchandise	Purchase physical items (eg: jerseys)	Digital collectibles
Ticketing	Traditional ticketing systems	Transparent and secure ticketing
Fan Tokens	Not applicable	Voting rights, access to exclusive content/events
Smart Contracts	Not applicable	Automated for fair athlete payments
Global Reach	Local and regional focus	Global reach, breaking geographical barriers
Interactivity	Limited direct interaction	Direct and interactive fan-team relationships

Fan Tokens

- Type of cryptocurrency that entitles owners to many fan-related membership benefits.
- Ownership and Voting Rights on various club decisions.
- Exclusive content, merchandise, events, and experiences related to the sports club.
- Fan token markets have grown rapidly, with significant market capitalization.
- Fan-centric approach, allowing fans to actively participate in club-related decisions and activities.
- Enable clubs to connect with their global fan base, irrespective of geographical location.



How do Fan Tokens work?



1. Issuance by Sports Clubs
2. Token Sale
3. Ownership and Wallets
4. Blockchain Technology
5. Voting Rights, Exclusive Content and Benefits
6. Token Trading
7. Regulatory Considerations



Blockchain-based Fan Token Framework

Utility

- Access & membership
- Governance
- Engagement / participation
- Value creation

Empowerment Dimension

Financialization

- Investment & speculation
- Funding & issuance
- Tokenomics
- Trading & secondary markets

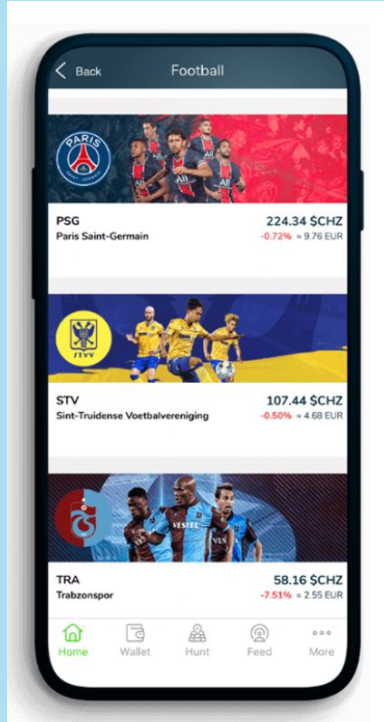
Trust and Efficiency Layer

- Integrity
- Transparency
- Censorship-free
- Automation

Socios Fan Token Platform

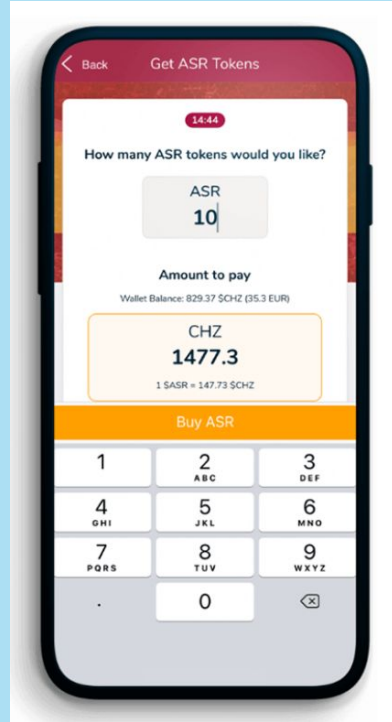


How Socios works?



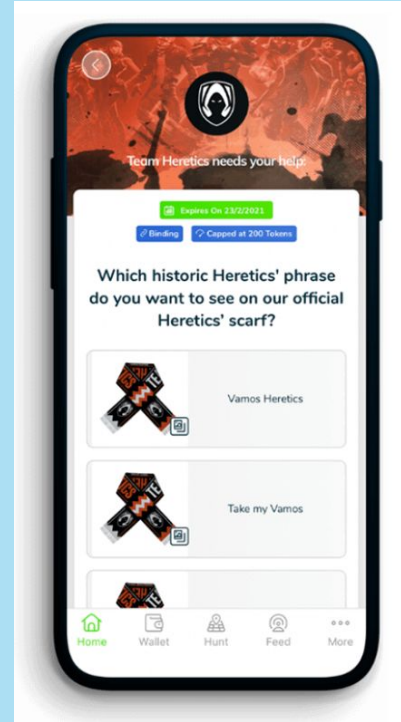
STEP 1

Sign up to Socios.com



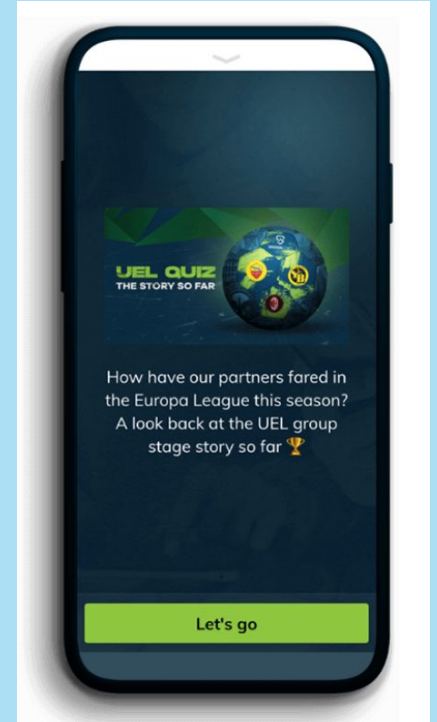
STEP 2

Buy Fan Tokens & hunt free Tokens



STEP 3

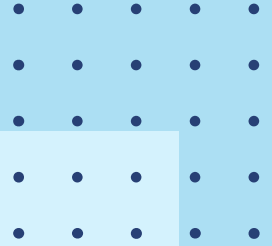
Vote on team polls



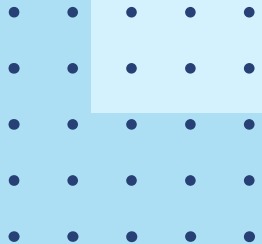
STEP 4

Engage & score rewards

Literature Review

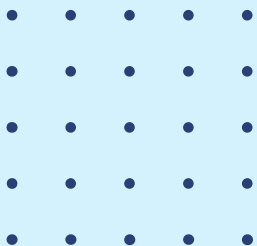


Enhancing Trust, Efficiency, and Empowerment
in Sports: Developing a Blockchain-Based Fan
Token Framework



Conclusion



- Fan tokens represent a remarkable convergence of blockchain technology and sports fandom, offering an innovative and engaging way for fans to connect with their favorite teams and organizations.
- While fan tokens have gained popularity primarily in football (soccer) clubs, their potential extends across various sports and industries.
- They signify a shift towards more interactive and inclusive fan experiences, underlining the ever-evolving relationship between technology and sports fandom.



Experiment 6: Smart Contract 1

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity ^0.8.0;
4
5  contract TicketManagement {
6      address public owner;
7      uint256 public totalTickets;
8      mapping(address => uint256) public ticketBalances;
9
10     event TicketIssued(address indexed recipient, uint256 quantity);
11
12     constructor() { 350642 gas 321000 gas
13         owner = msg.sender;
14         totalTickets = 0;
15     }
16
17     modifier onlyOwner() {
18         require(msg.sender == owner, "Only the owner can perform this action");
19         _;
20     }
21
22     function issueTickets(address recipient, uint256 quantity) public onlyOwner { infinite gas
23         require(quantity > 0, "Quantity must be greater than zero");
24         totalTickets += quantity;
25         ticketBalances[recipient] += quantity;
26         emit TicketIssued(recipient, quantity);
27     }
28
29     function getMyTicketBalance() public view returns (uint256) { 2548 gas
30         return ticketBalances[msg.sender];
31     }
32 }
```

Experiment 6: Smart Contract 2

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity ^0.8.0;
4
5  contract FanEngagementVoting {
6      address public owner;
7      mapping(address => uint256) public votes;
8      uint256 public totalVotes;
9
10     constructor() {  278167 gas 248600 gas
11         owner = msg.sender;
12         totalVotes = 0;
13     }
14
15     modifier onlyOwner() {
16         require(msg.sender == owner, "Only the owner can perform this action");
17         _;
18     }
19
20     function voteForAddress(address candidate) public {  infinite gas
21         require(candidate != address(0), "Invalid candidate address");
22         require(msg.sender != candidate, "You cannot vote for yourself");
23
24         votes[candidate] += 1;
25         totalVotes += 1;
26     }
27 }
```

Experiment 7: Integration with Metamask

The screenshot displays the Remix IDE interface for deploying a Solidity contract. The left sidebar, titled "DEPLOY & RUN TRANSACTIONS", shows the "Injected Provider - MetaMask" and the "Sepolia (11155111) network". The "ACCOUNT" field displays "0xB3B...1b7a4 (0.278834724)". The "GAS LIMIT" is set to "3000000". The "VALUE" is "0" in "Wei". The "CONTRACT" dropdown shows "TicketManagement - contract1.sol". The "Deploy" button is visible. Below it, there is a "Publish to IPFS" checkbox and an "At Address" button. The main editor displays the Solidity code for the "TicketManagement" contract, which includes a constructor, a modifier, and a function to issue tickets. The right sidebar shows a "MetaMask Notification" dialog for "Sepolia test network" with a "New contract" button. The notification details show the URL "https://remix.ethereum.org", the gas limit "0.001059", and the total amount "0.001059 SepoliaETH". The "Confirm" button is highlighted.

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.8.0;
4
5 contract TicketManagement {
6     address public owner;
7     uint256 public totalTickets;
8     mapping(address => uint256) public ticketBalances;
9
10    event TicketIssued(address indexed recipient, uint256 quantity);
11
12    constructor() {
13        owner = msg.sender;
14        totalTickets = 0;
15    }
16
17    modifier onlyOwner() {
18        require(msg.sender == owner, "Only the owner can perform this action");
19        _;
20    }
21
22    function issueTickets(address recipient, uint256 quantity) public onlyOwner {
23        require(quantity > 0, "Quantity must be greater than zero");
24        totalTickets += quantity;
25        ticketBalances[recipient] += quantity;
26        emit TicketIssued(recipient, quantity);
27    }
28 }
```

MetaMask Notification

Sepolia test network

Account 1 → New contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

DETAILS DATA

Site suggested > i

Gas (estimated) 0.001059

0.001059 SepoliaETH

Very likely in < 15 seconds Max fee: 0.001059 SepoliaETH

Total 0.001059

0.001059 SepoliaETH

Amount + gas fee Max amount: 0.001059 SepoliaETH

Reject Confirm

Transactions recorded 2 1 >

Deployed Contracts

Currently you have no contract instances to interact with.

Search

ENG IN 19:21 06-10-2023

Experiment 8: Deploying with Ganache Account

The screenshot displays the Remix IDE interface for deploying a Solidity contract. The left sidebar shows the deployment configuration:

- ENVIRONMENT:** Injected Provider - MetaMask
- ACCOUNT:** 0x654...2b6cd (100 ether)
- GAS LIMIT:** 3000000
- VALUE:** 0 Wei
- CONTRACT:** TicketManagement - contract1.sol
- evm version:** paris
- Buttons:** Deploy, Publish to IPFS, At Address, Load contract from Address
- Transactions recorded:** 1
- Deployed Contracts:** Currently you have no contract instances to interact with.

The main editor displays the Solidity code for the `TicketManagement` contract:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.8.0;
4
5 contract TicketManagement {
6     address public owner;
7     uint256 public totalTickets;
8     mapping(address => uint256) public ticketBalances;
9
10    event TicketIssued(address indexed recipient, uint256 quantity);
11
12    constructor() {
13        owner = msg.sender;
14        totalTickets = 0;
15    }
16
17    modifier onlyOwner() {
18        require(msg.sender == owner, "Only the owner can perform this action");
19        _;
20    }
21
22    function issueTickets(address recipient, uint256 quantity) public onlyOwner {
23        require(quantity > 0, "Quantity must be greater than zero");
24        totalTickets += quantity;
25        ticketBalances[recipient] += quantity;
26        emit TicketIssued(recipient, quantity);
27    }
28
29    function getMyTicketBalance() public view returns (uint256) {
30        return ticketBalances[msg.sender];
31    }
32}
```

The right sidebar shows the MetaMask confirmation dialog for the deployment:

- Account:** Account 3
- Operation:** New contract
- URL:** https://remix.ethereum.org
- Button:** CONTRACT DEPLOYMENT
- Details:** Network is busy. Gas prices are high and estimates are less accurate.
- Gas (estimated):** 0.00171067 ETH
- Very likely in < 15 seconds**
- Max fee:** 0.00171067 ETH
- Total:** 0.00171067 ETH
- Amount + gas fee:** 0.00171067 ETH
- Buttons:** Reject, Confirm

Experiment 8: Deploying with Ganache Account

The screenshot displays the Remix IDE interface within a Firefox Web Browser. The browser's address bar shows the URL: `https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js`.

The interface is divided into three main panels:

- Left Panel (Deploy & Run Transactions):** This panel contains a sidebar with a "Deploy & Run Transactions" section. It includes a "Load contract from Address" button, a "Transactions recorded" section, and a "Deployed Contracts" section. The "Deployed Contracts" section shows a contract named "TICKETMANAGEMENT AT 0xAf8...925". Below this, there is a "Balance: 0 ETH" section and an "IssueTickets" section with a "recipient" field (value: 34082EAdE16461eC45a4810092b6cd) and a "quantity" field (value: 5). There are buttons for "Calldata", "Parameters", and "transact". Below these are buttons for "getMyTicketB...", "owner", "ticketBalances" (with a dropdown menu showing "address"), and "totalTickets". At the bottom, there is a "Low level interactions" section with a "CALLDATA" field and a "Transact" button.
- Center Panel (Code Editor):** This panel displays the Solidity code for the "contract1.sol" file. The code is as follows:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.8.0;
4
5 contract TicketManagement {
6     address public owner;
7     uint256 public totalTickets;
8     mapping(address => uint256) public ticketBalances;
9
10    event TicketIssued(address indexed recipient, uint256 quantity);
11
12    constructor() {
13        owner = msg.sender;
14        totalTickets = 0;
15    }
16
17    modifier onlyOwner() {
18        require(msg.sender == owner, "Only the owner can perform this action");
19        _;
20    }
21
22    function issueTickets(address recipient, uint256 quantity) public onlyOwner {
23        require(quantity > 0, "Quantity must be greater than zero");
24        totalTickets += quantity;
25        ticketBalances[recipient] += quantity;
26        emit TicketIssued(recipient, quantity);
27    }
28
29    function getMyTicketBalance() public view returns (uint256) {
30        return ticketBalances[msg.sender];
31    }
32}
```
- Right Panel (Account 3):** This panel shows the account "Account 3" with a balance of "99.9984 ETH". It includes buttons for "Buy", "Send", "Swap", "Bridge", and "Portfolio". Below these are sections for "Tokens", "NFTs", and "Activity". The "Activity" section shows a list of transactions, including "Contract i..." and "Contract ...", both with a status of "Confirmed" and a value of "-0 ETH". At the bottom, there is a "MetaMask support" button.

The bottom of the interface shows a transaction log with the following details:

- Block: 4 txIndex: 0
- From: 0x654...2b6cd to: TicketManagement.issueTickets(address,uint256) 0xAf8...92926 value: 0 wei
- Data: 0x85b...00005 logs: 1 hash: 0xc36...ffdad

A "Debug" button is visible next to the transaction log.

Experiment 8: Deploying with Ganache Account

Ganache									
ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES			
CURRENT BLOCK 2	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE BLOCKCHAIN-HONORS		SWITCH ⚙️
BLOCK 2	MINED ON 2023-09-25 18:45:38				GAS USED 43724		1 TRANSACTION		
BLOCK 1	MINED ON 2023-09-25 18:36:24				GAS USED 125677		1 TRANSACTION		
BLOCK 0	MINED ON 2023-09-25 17:28:39				GAS USED 0		NO TRANSACTIONS		

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
2

GAS PRICE
20000000000

GAS LIMIT
6721975

HARDFORK
MERGE

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

WORKSPACE
BLOCKCHAIN-HONORS

SWITCH

TX HASH

0x49f974dc7644868cee5628309418d7e357f8c2b5a3db96c4f8eac4790c0d7657

CONTRACT CALL

FROM ADDRESS

0x3BD6c04db63254D2624fe2f8368A7dEd0152c4b4

TO CONTRACT ADDRESS

0xCb2191C283D756580eBF36aa886B166a431D6Eb5

GAS USED

43724

VALUE

0

TX HASH

0x3ef2d624adda27d645e9366e3c3f2c84d9e61fc76e07d396360a439f14d2fe3a

CONTRACT CREATION

FROM ADDRESS

0x3BD6c04db63254D2624fe2f8368A7dEd0152c4b4

CREATED CONTRACT ADDRESS

0xCb2191C283D756580eBF36aa886B166a431D6Eb5

GAS USED

125677

VALUE

0