



ASTANA IT UNIVERSITY

PredictionChain: Decentralized Prediction Crowdfunding dApp



BY DILNAZ & MEREKE



What is PredictionChain?

A decentralized platform where users:

- choose outcomes
- place bets using test ETH
- receive rewards based on results
- earn ERC-20 participation tokens (PRED)



Architecture

User → Frontend → Smart Contracts → Ethereum Testnet
(MetaMask + ERC-20 labels)

Components

- Frontend (Web App): UI + interaction with contracts
- Smart Contracts: Event creation, contributions, payouts
- MetaMask: Wallet connection & transaction signing
- ERC-20 Token Contract: Participation/reputation token

Key Technologies

Solidity • Ethers.js • MetaMask • Sepolia/Holesky testnet

PredictionChain

Simple prediction market demo (Sepolia + MetaMask)

Wallet

[Connect MetaMask](#)

[Refresh](#)

Address

0x5F225f7A8962163f348A52FE6F7741dd
4BC4A2Ad

ETH balance

0.690229060685098164 ETH

PRED balance (ERC-20)

10.0 PRED

MetaMask Connected

Create Event

Title

Will GTA 6 release before 2027?

Deadline (unix timestamp)

Options count

e.g. 1767225600

2

[Create](#)

Finalize / Claim

Event ID

0

Winning option

0

ASTANA IT UNIVERSITY

Event Structure

EVERY EVENT INCLUDES:

- TITLE
- OUTCOMES (A/B/C)
- DEADLINE
- TOTAL ETH POOL
- POOLS PER OUTCOME
- CREATOR
- STATUS

How Users Participate

1. USER CONNECTS METAMASK
2. CHOOSES AN EVENT OUTCOME
 3. SENDS TEST ETH (VALUE)
4. CONTRACT STORES THE BET
5. ERC-20 PRED TOKENS ARE MINTED AUTOMATICALLY

The screenshot shows the REMIX Ethereum IDE interface. On the left, there's a sidebar with various icons. The main area has tabs for 'DEPLOY & RUN TRANSACTIONS' and 'EXPLAIN CONTRACT'. The code editor on the right contains three files:

```

DEPLOY & RUN TRANSACTIONS
Both 'receive' and 'fallback' functions are not defined

PREDICTIONCHAIN AT 0xAB...
Balance: 0 ETH
createEvent "Will GTA 6 release before 2018?" 0
PLACEBET
eventid: 0
outcomeIndex: 0
Calldata Parameters transact
resolveEvent uint256 eventid, uint8 winner
bets uint256 , address , uint8
events uint256
getEventsCount()
o: uint256: 1
outcomePools uint256 , uint8
rewardToken
TOKENS_PER...

Low level interactions
CALDATA
Transact
  
```

The 'EXPLAIN CONTRACT' tab is active, showing the ABI for the PredictionChain contract. It lists methods like getEventsCount, resolveEvent, bets, events, getEventsCount, o, outcomePools, rewardToken, and TOKENS_PER... with their respective parameters and descriptions.

The transaction history pane on the right shows a recent call to PredictionChain.getEventsCount with the following details:

- call to PredictionChain.getEventsCount
- [call] from: 0xae28110738aced0e17b27c4ba1ba8bf1be4802f2 to: PredictionChain.getEventsCount() data: 0x303...66d5f
- from: 0xE28110738acEd0e17B27C4bA1Ba8BF1Be4802F2
- to: PredictionChain.getEventsCount() 0xAB9D9E4B1484C7B47BFc4508d375066b9ee5D9a9
- input: 0x303...66d5f
- output: 00000000000000000000000000000001
- decoded input: {}
- decoded output: {} "0": "uint256: 1"
- logs: []
- raw logs: []

At the bottom, it says 'transact to PredictionChain.placeBet pending ...'

Smart Contract Logic

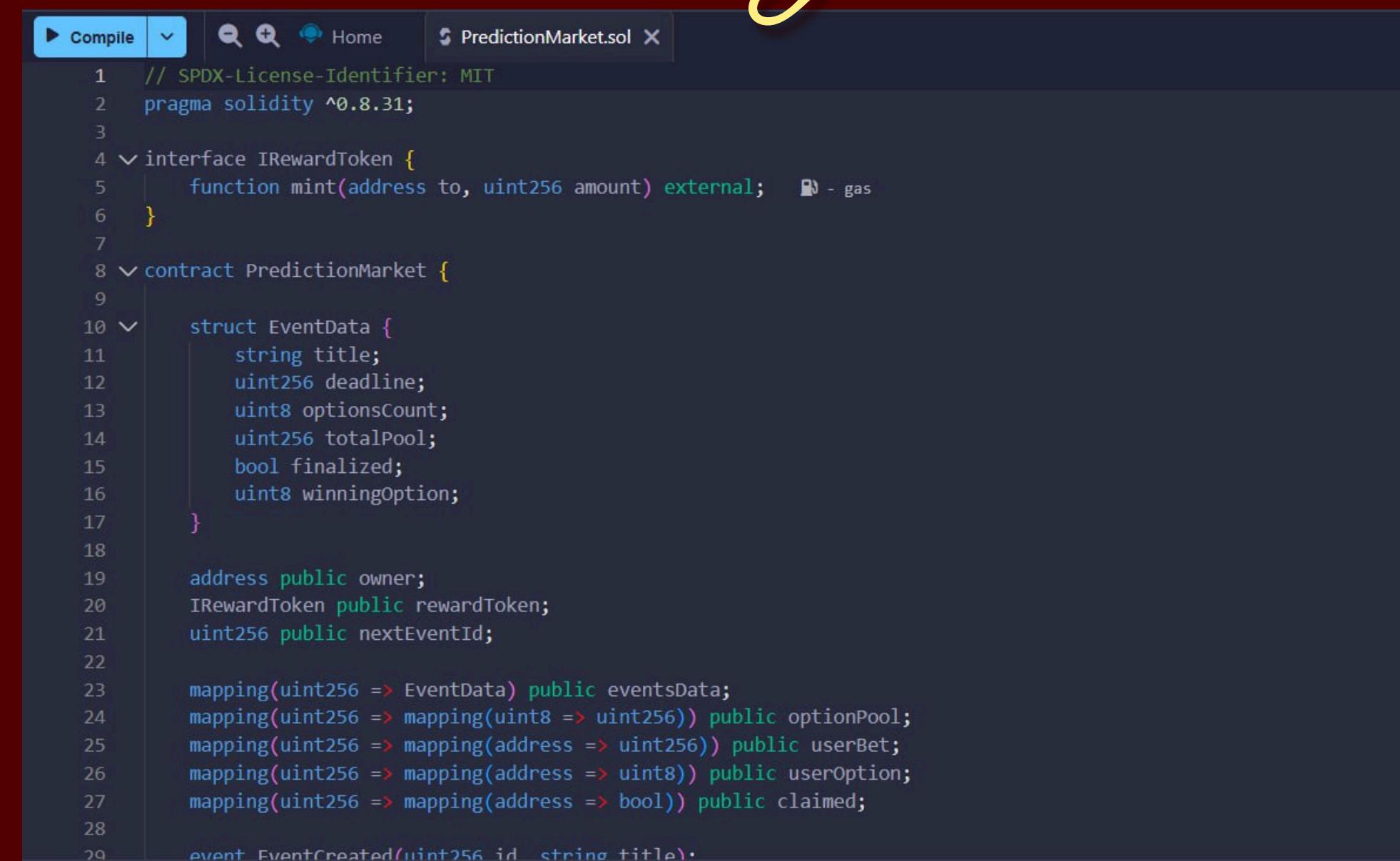
PredictionMarket

- `createEvent()`
- `bet()`
- `finalize()`
- `claim()`

RewardToken

- ERC20
- `mint()`
- `setMinter()`

•



The screenshot shows a Solidity code editor interface with the following details:

- Toolbar:** Includes "Compile" (highlighted in blue), search icons, and tabs for "Home" and "PredictionMarket.sol".
- Code Area:** Displays the `PredictionMarket.sol` file content.
- Code Content:**

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.31;
3
4 interface IRewardToken {
5     function mint(address to, uint256 amount) external; - gas
6 }
7
8 contract PredictionMarket {
9
10    struct EventData {
11        string title;
12        uint256 deadline;
13        uint8 optionsCount;
14        uint256 totalPool;
15        bool finalized;
16        uint8 winningOption;
17    }
18
19    address public owner;
20    IRewardToken public rewardToken;
21    uint256 public nextEventId;
22
23    mapping(uint256 => EventData) public eventsData;
24    mapping(uint256 => mapping(uint8 => uint256)) public optionPool;
25    mapping(uint256 => mapping(address => uint256)) public userBet;
26    mapping(uint256 => mapping(address => uint8)) public userOption;
27    mapping(uint256 => mapping(address => bool)) public claimed;
28
29    event EventCreated(uint256 id, string title);

```

Tokenomics

Account 1

0.315 SepoliaETH
+\$0.00 (+0.00%) Discover

Buy Swap Send Receive

Tokens DeFi NFTs Activity

Sepolia

Feb 5, 2026

| | |
|----------------------|-----------------------|
| Place Bet | -<0.000001 SepoliaETH |
| S Confirmed | -<0.000001 SepoliaETH |
| Contract interaction | -0 SepoliaETH |
| S Confirmed | -0 SepoliaETH |
| Set Minter | -0 SepoliaETH |
| S Confirmed | -0 SepoliaETH |
| Contract deployment | -0 SepoliaETH |
| S Confirmed | -0 SepoliaETH |
| Contract deployment | -0 SepoliaETH |
| S Confirmed | -0 SepoliaETH |

PRED = PredictionChain Participation Token

- ERC-20 standard
- 18 decimals
- minted only by PredictionChain
- 1 wei = 1 PRED
- used as reputation/participation token



ASTANA IT UNIVERSITY

Deployment

Deployed Contracts:

- 1)PredictionToken
- 2)PredictionChain

Tools used:

Remix IDE

MetaMask

Sepolia Faucet

The screenshot shows the Remix IDE interface for deploying a smart contract. The left sidebar has icons for AI, Deploy, Run, Transactions, Environment, Accounts, Gas Limit, Value, and Contract. The main area is titled "DEPLOY & RUN TRANSACTIONS". It shows the environment set to "Injected Provider - MetaMask" and the network set to "Sepolia (11155111) network". The account selected is "0xae2...802f2 (0.3153717286)". The gas limit is set to "Estimated Gas" with a value of "3000000". The value is set to "0 Gwei". The contract selected is "Ownable - @openzeppelin/contract" with "evm version: osaka". A checkbox "Verify Contract on Explorers" is checked. At the bottom is a "Deploy & Verify" button. To the right, the source code of the contract is displayed in a syntax-highlighted editor. The code includes event definitions and a function for getting the number of events.

```
EventData memory newEvent {
    title: title,
    deadline: deadline,
    outcomeCount: outcomeCount,
    isResolved: isResolved,
    winningOutcome: winningOutcome,
    creator: msg.sender,
    totalPool: totalPool
};

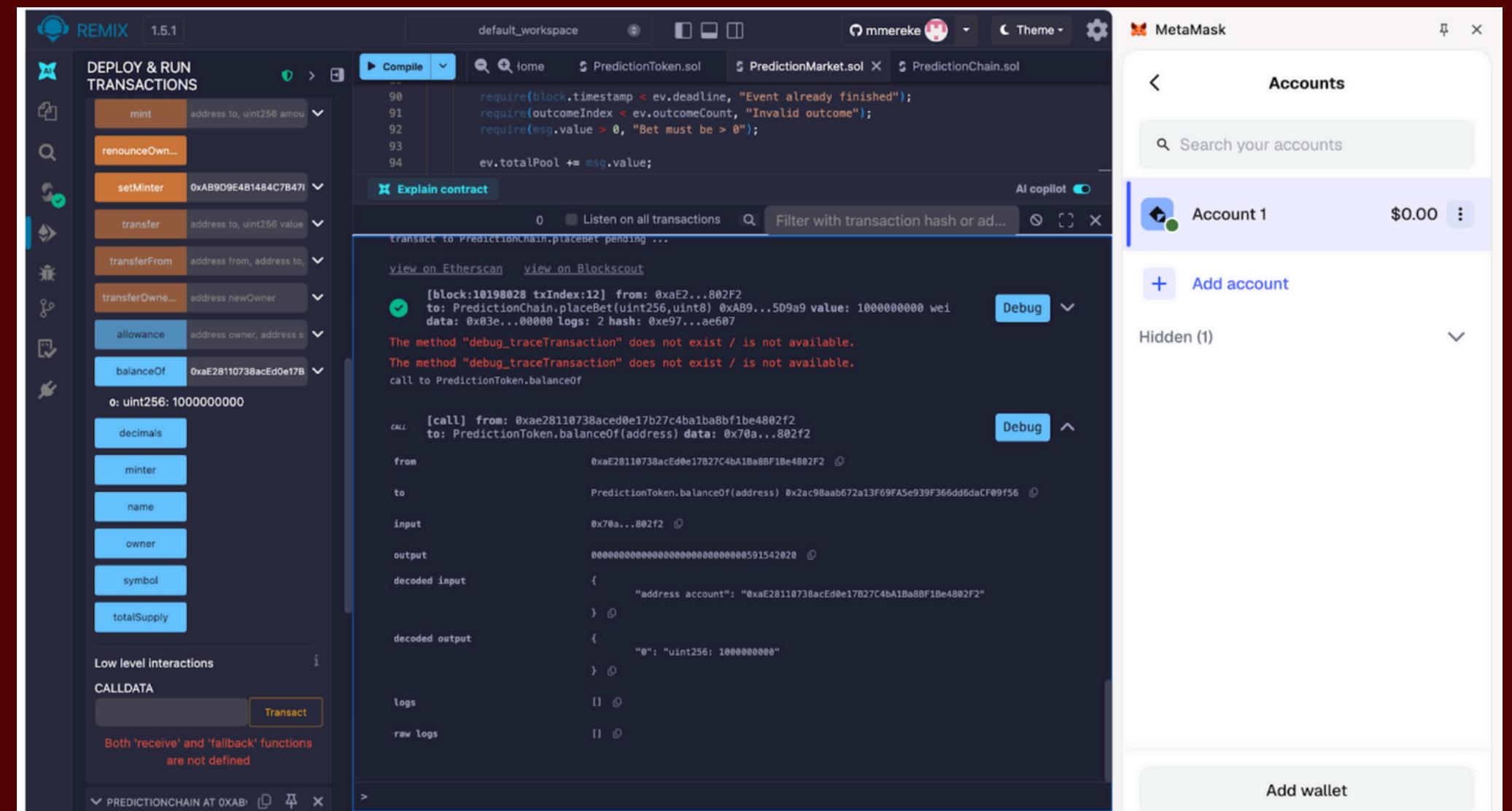
events.push(newEvent);
uint256 eventId = eventID;
emit EventCreated(eventId);

function getEventsCount() external view returns (uint256) {
    return events.length;
}
```

Testing

Manual tests performed:

- Creating events ✨
- Sending bets with value
- Minting PRED tokens
- Checking balances
- Resolving event
- Claiming rewards
- Viewing ETH distribution



The screenshot shows the REMIX IDE interface with a Solidity code editor containing three contracts: PredictionToken.sol, PredictionMarket.sol, and PredictionChain.sol. The code editor displays a transaction trace for a bet placement, showing details like block number, timestamp, address, value, and logs.

REMX 1.5.1

DEPLOY & RUN TRANSACTIONS

- mint: address to, uint256 amount
- renounceOwnership
- setMinter: 0xAB9D9E4B1484C7B471
- transfer: address to, uint256 value
- transferFrom: address from, address to, uint256 value
- transferOwnership: address newOwner
- allowance: address owner, address spender, uint256 value
- balanceOf: address owner, uint256 value

Low level interactions

CALldata

Both 'receive' and 'fallback' functions are not defined

Explained contract

MetaMask

Accounts

Search your accounts

Account 1: \$0.00

Add account

Hidden (1)

Add wallet

STUDIO SHODWE

Thank You

FOR YOUR ATTENTION