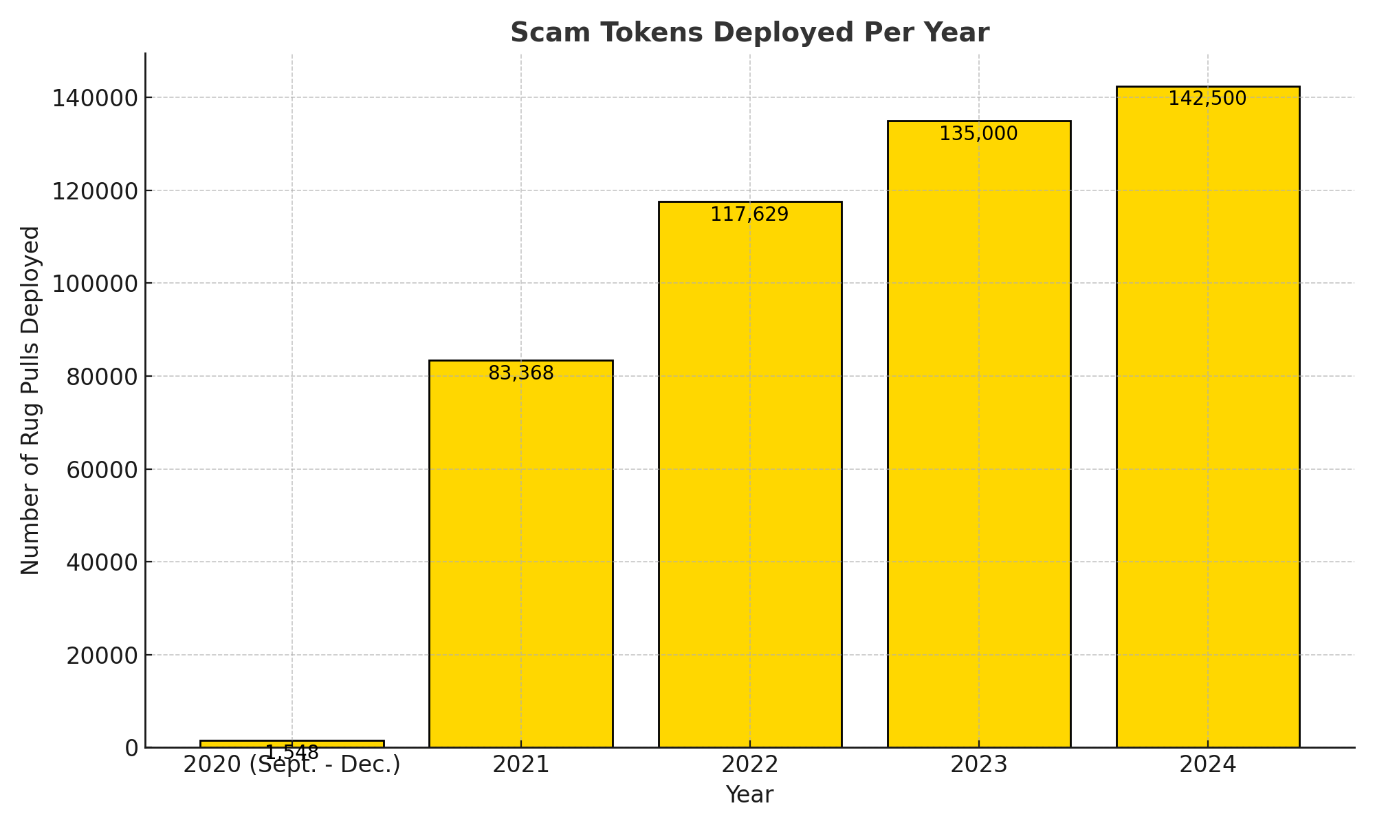
TEAM: Unearthly

Title: Identification of Rug Pull via User Learning Platform  
  
Problem it solves: It helps identify users and spread awareness by practical implementation via hands on learning on rug pull concepts. Hence it provides financial safety by guiding users decision with precision and factual knowledge through previously gathered data and 3 major metrices (1.fake volume, 2.recent transactions, 3.company backed up by which major company)

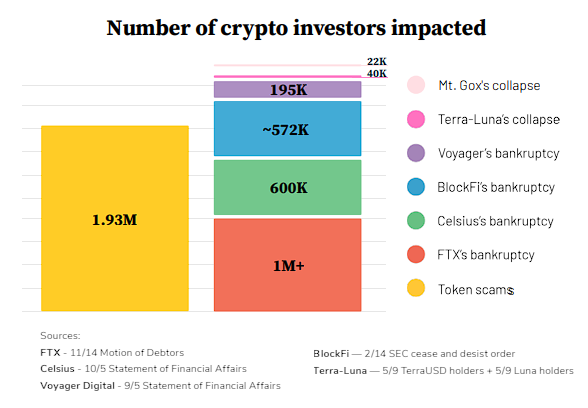
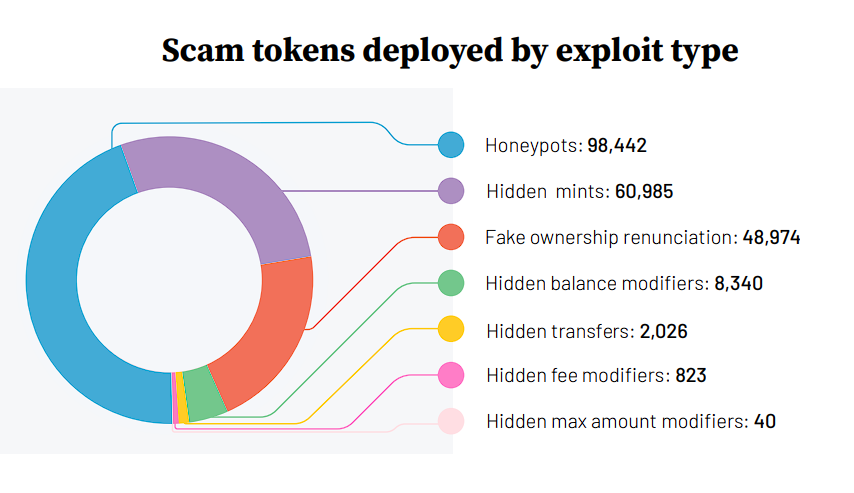
Question) Ok, What is Rug Pull?  
A rug pull is when a scammer develops a crypto token, deploys it on a blockchain, convinces users to buy it, and then liquidates his or her holdings without warning, leaving investors in the lurch.

There are basically two types in it, one being Soft Rug pull where there is no harm/malicious activity in the code but scam happen after token being falsely advertised/popularized while **Hard Rug Pull** has malicious code itself.

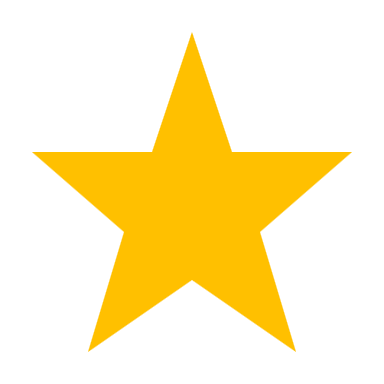
CHART:Increasing Rug Pulls by years



**The Problem is serious, and It needs a raise.**

**** 

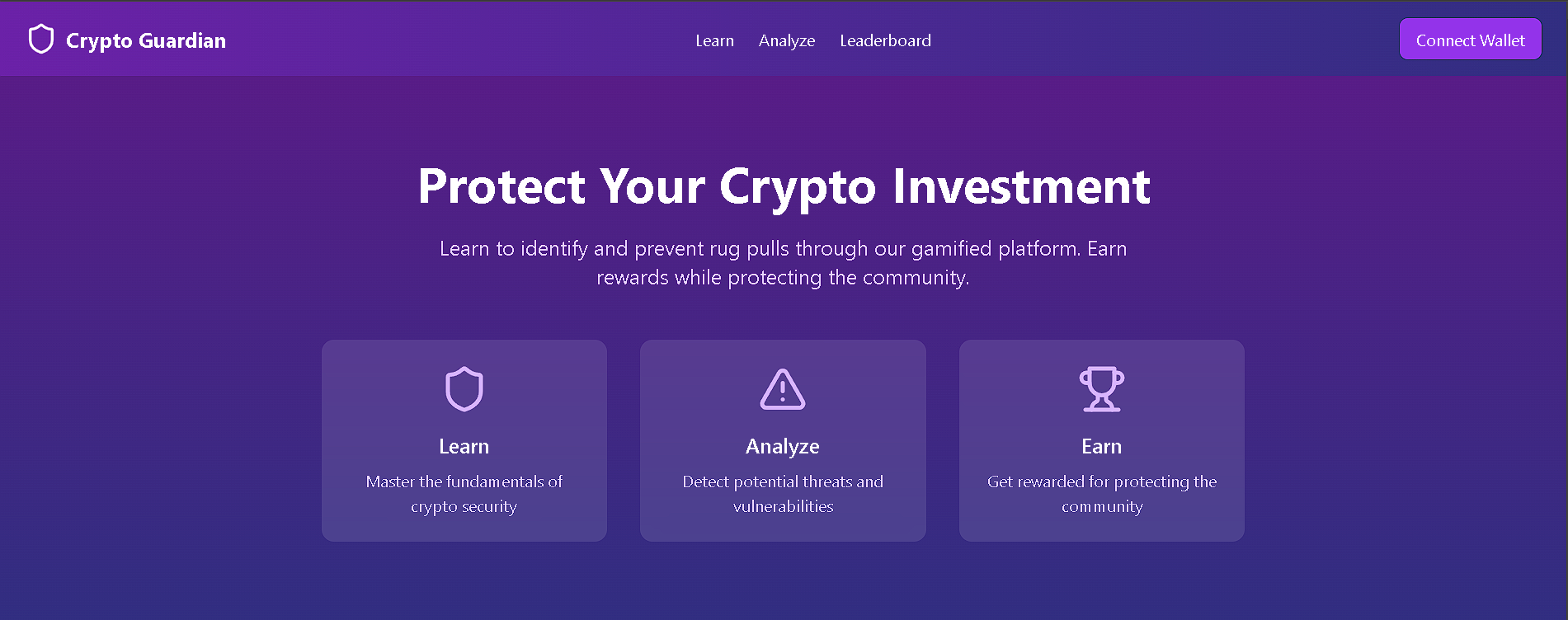
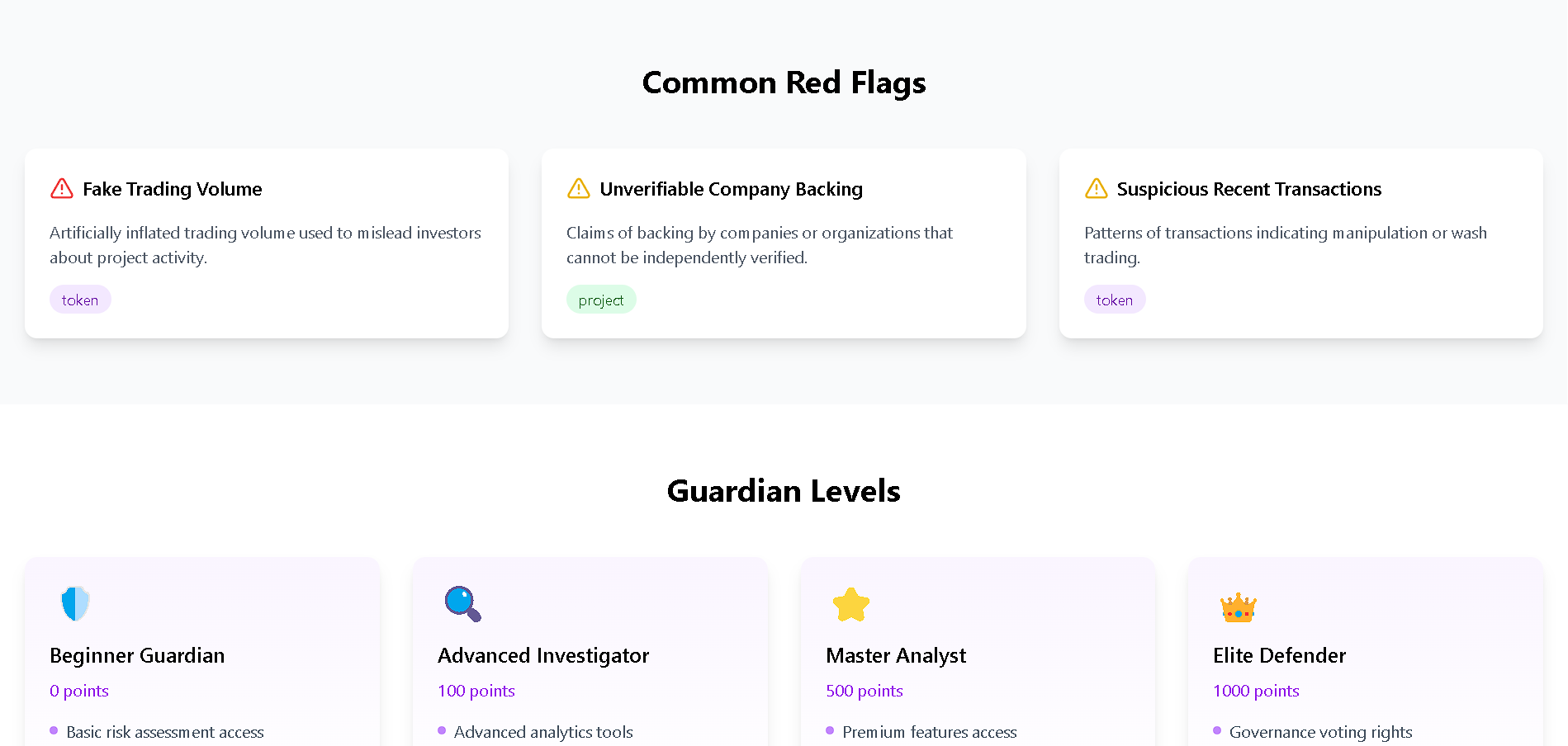
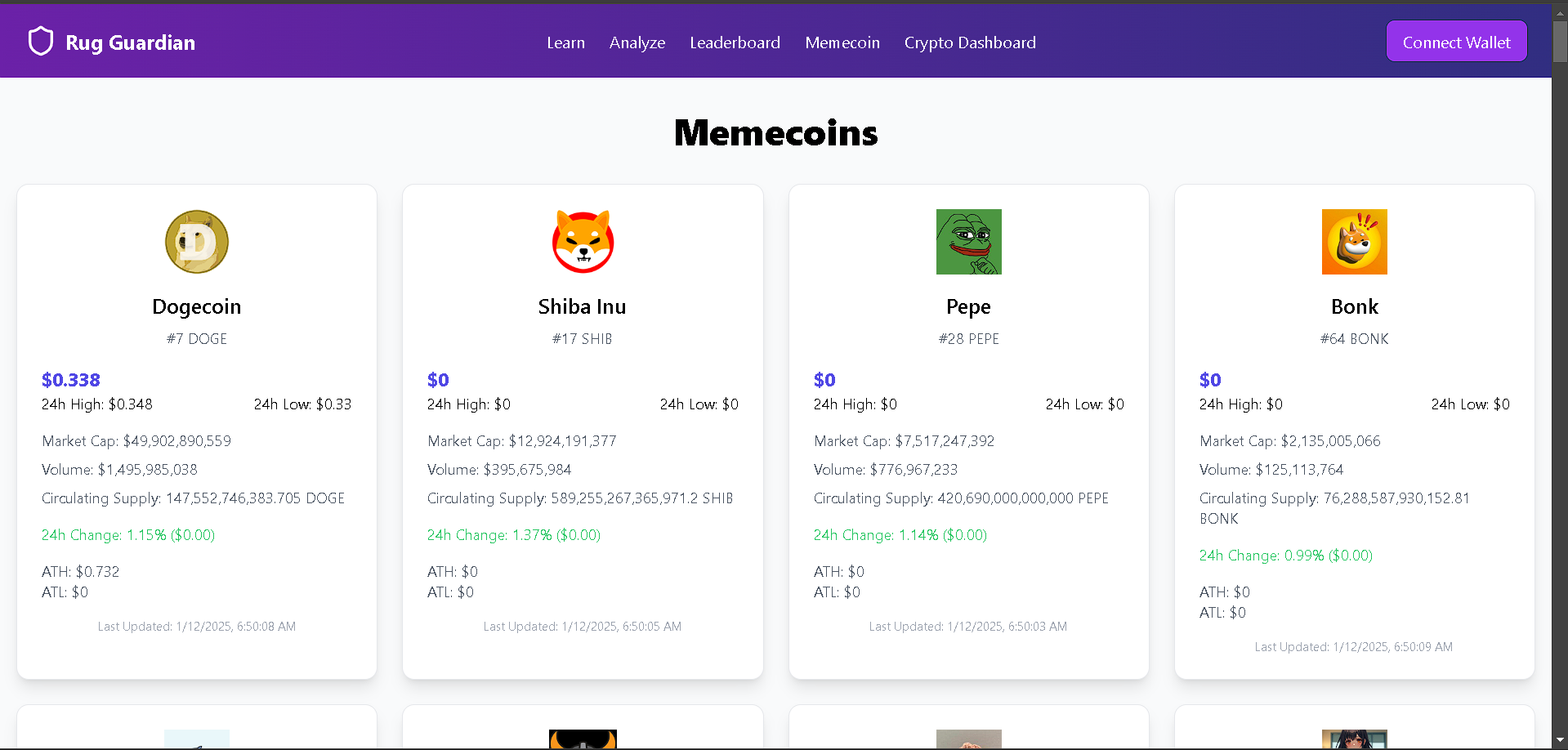
Our solution lies around with creating a learning pathway for users, where they will be able to actually implement, hands-on practically identify rug pulls. Hence It promotes financial security, and create rug-pull free environment.

**Version 1.0 (Beta)  
3 basic parameters we foster our educating journey on!  
  
1) Fake Volume Identification Pathway.**

**2) Identification of Project Backed up by which major org/company/Project itself.**

**3)Artificial Transaction Generation (Back and Forth transactions from similar accounts).  
  
By Completing the above 3 tracks, a user will get a basic idea about how to avoid fake chains/tokens/coins etc i.e by differentiating through above factors.**

Our Final Product:

  
  
****

**Our Smart Contract!**

**// SPDX-License-Identifier: MIT**

**pragma solidity ^0.8.0;**

**contract RugPullDetection {**

**// Owner of the contract**

**address public owner;**

**// Event to log detection results**

**event RugPullEvaluated(**

**string projectName,**

**uint256 score,**

**string classification**

**);**

**// Struct to hold project details**

**struct Project {**

**string name;**

**uint256 fakeVolumeScore; // Score based on fake volume creation**

**uint256 recentTransactionsScore; // Score based on recent transactions**

**uint256 backingScore; // Score based on company backing**

**uint256 finalScore; // Final computed score**

**string classification; // "Rug Pull" or "Trustworthy"**

**}**

**// Mapping to store projects**

**mapping(string => Project) public projects;**

**// Modifier to restrict function access to the owner**

**modifier onlyOwner() {**

**require(msg.sender == owner, "Not authorized");**

**\_;**

**}**

**// Constructor to initialize the contract owner**

**constructor() {**

**owner = msg.sender;**

**}**

**/\*\***

**\* @dev Evaluate a project based on given scores**

**\* @param \_name Name of the project**

**\* @param \_fakeVolumeScore Score for fake volume creation (0-40)**

**\* @param \_recentTransactionsScore Score for recent transactions (0-30)**

**\* @param \_backingScore Score for major company backing (0-30)**

**\*/**

**function evaluateProject(**

**string memory \_name,**

**uint256 \_fakeVolumeScore,**

**uint256 \_recentTransactionsScore,**

**uint256 \_backingScore**

**) public onlyOwner {**

**require(**

**\_fakeVolumeScore <= 40 &&**

**\_recentTransactionsScore <= 30 &&**

**\_backingScore <= 30,**

**"Invalid scores"**

**);**

**uint256 finalScore = \_fakeVolumeScore +**

**\_recentTransactionsScore +**

**\_backingScore;**

**string memory classification;**

**if (finalScore < 60) {**

**classification = "Most Likely a Rug Pull";**

**} else {**

**classification = "Trustworthy";**

**}**

**// Store the project details**

**projects[\_name] = Project({**

**name: \_name,**

**fakeVolumeScore: \_fakeVolumeScore,**

**recentTransactionsScore: \_recentTransactionsScore,**

**backingScore: \_backingScore,**

**finalScore: finalScore,**

**classification: classification**

**});**

**// Emit the result**

**emit RugPullEvaluated(\_name, finalScore, classification);**

**}**

**/\*\***

**\* @dev Retrieve project details**

**\* @param \_name Name of the project**

**\* @return Project details including scores and classification**

**\*/**

**function getProjectDetails(string memory \_name)**

**public**

**view**

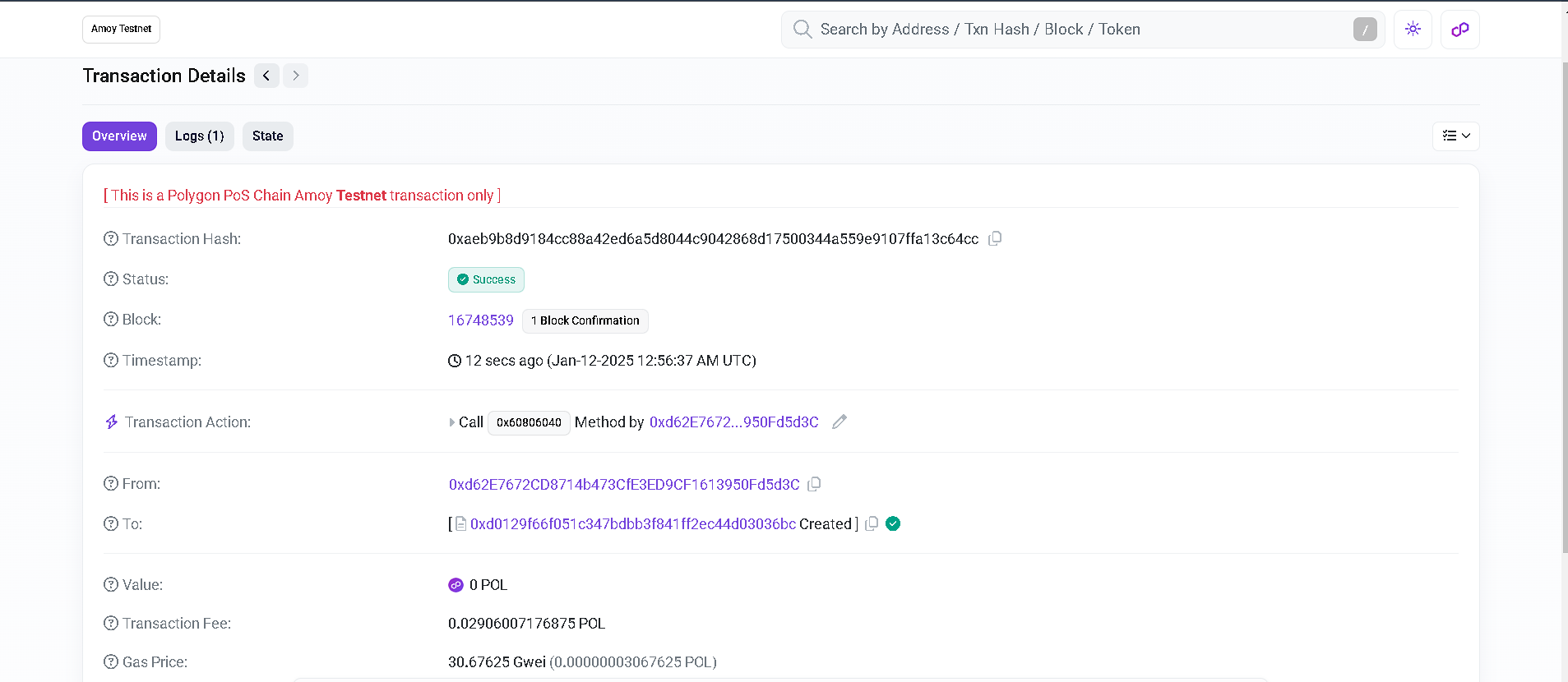
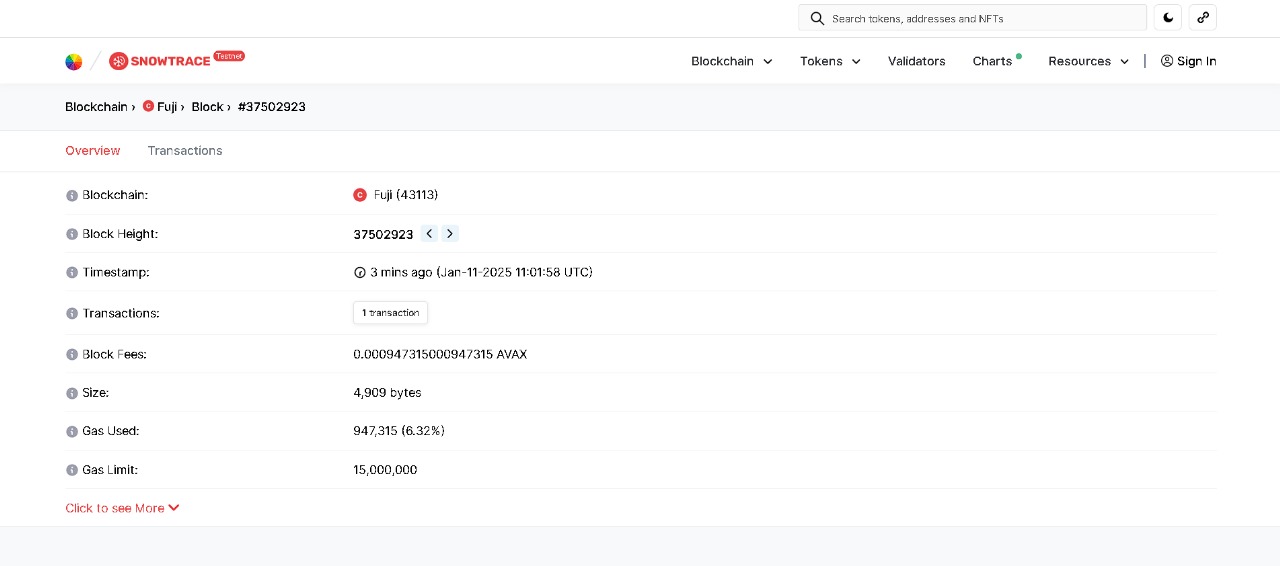
**returns (Project memory)**

**{**

**return projects[\_name];**

**}**

**}**

**Smart Contract deployed on Polygon and Avalanche Testnet.  
  
1)Polygon  
  
2)Avalanche  
  
  
Thank You**