Master’s Thesis in Computer Engineering

Smart Contract Analysis and Visualization Software

Supervisors:

Prof. Valentina Gatteschi
Eng. Emanuele Antonio Napoli

Candidate:

Lorenzo Gangemi

Politecnico di Torino

Contents

1. Introduction and motivations
2. System Architecture
3. Frontend
4. Results
5. Conclusions

Introduction and Motivations

Politecnico di Torino

Introduction and Motivations

Challenges

- Smart contracts' **complexity** represents a barrier for non-technical users.
- **Errors** in deployed smart contracts lead to significant financial losses.
- Existing tools fail to provide intuitive, user-friendly **interfaces**.

Goals

- Provide a user-friendly tool for **reading**, **editing**, and **analyzing** smart contracts.
- **Reduce risks** through security analysis and **better code understanding**.

Introduction and Motivations

Three key features

- **Graphical Representation** to abstract the code into a visual format.
- **Security Analysis** to detect vulnerabilities and reduce the risk of errors.
- **LLM Integration** for human readable explanations, functional relationships, and functions highlighting.

Smart Contract Analysis and Visualization Software

System Architecture

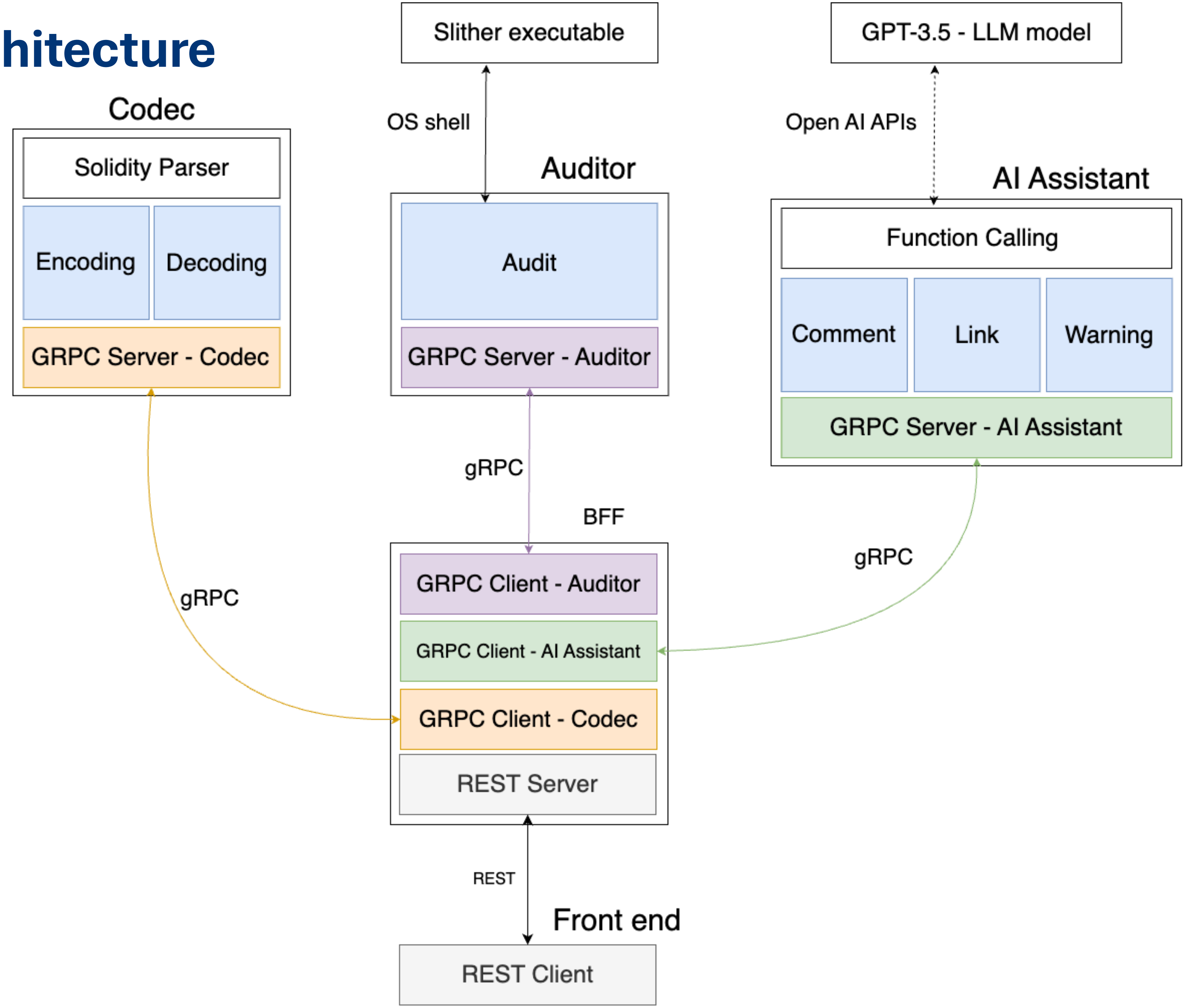
Politecnico di Torino

System Architecture

Microservices architecture

- Codec
- Auditor
- AI Assistant

- BFF

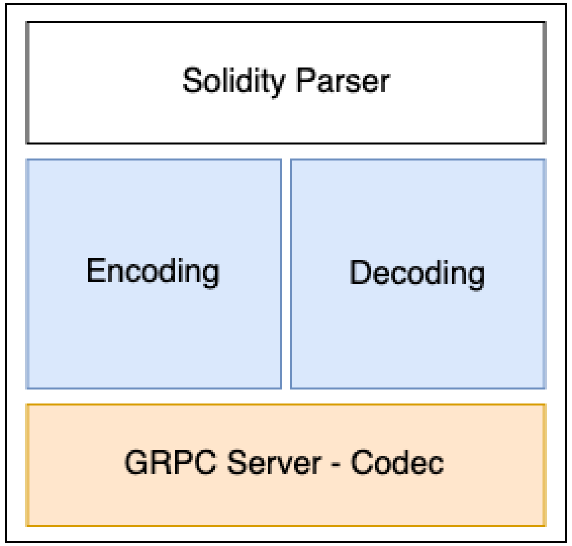


System Architecture

Codec

- **Encodes** the Solidity Smart Contract into the JSON structure used by the whole system.
- **Decodes** a modified JSON structure back into a Solidity Smart Contract.

- This is achieved by using a **custom parser built with ANTLR**, specifically designed for the thesis.



Solidity Parser

Encoding

Decoding

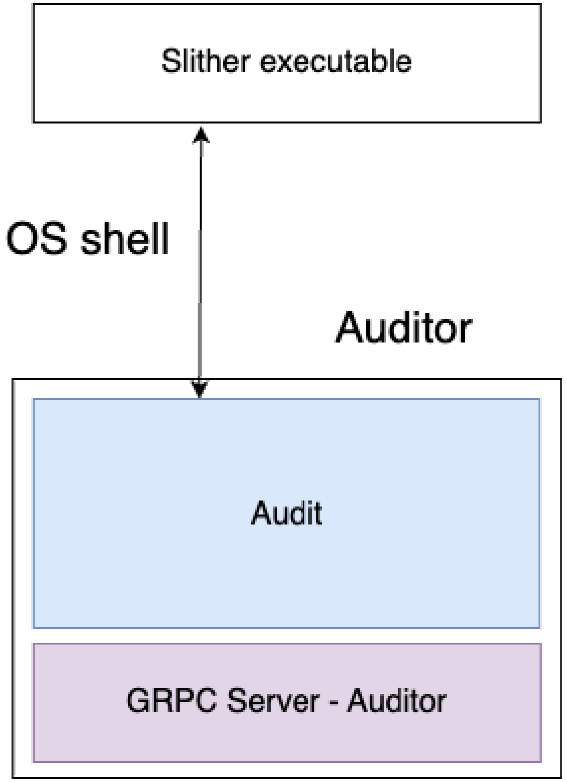
GRPC Server - Codec

System Architecture

Auditor

- Performs a **Static analysis** on the Smart Contract for Vulnerability Detection.

- The analysis is performed using the **Slither** executable.

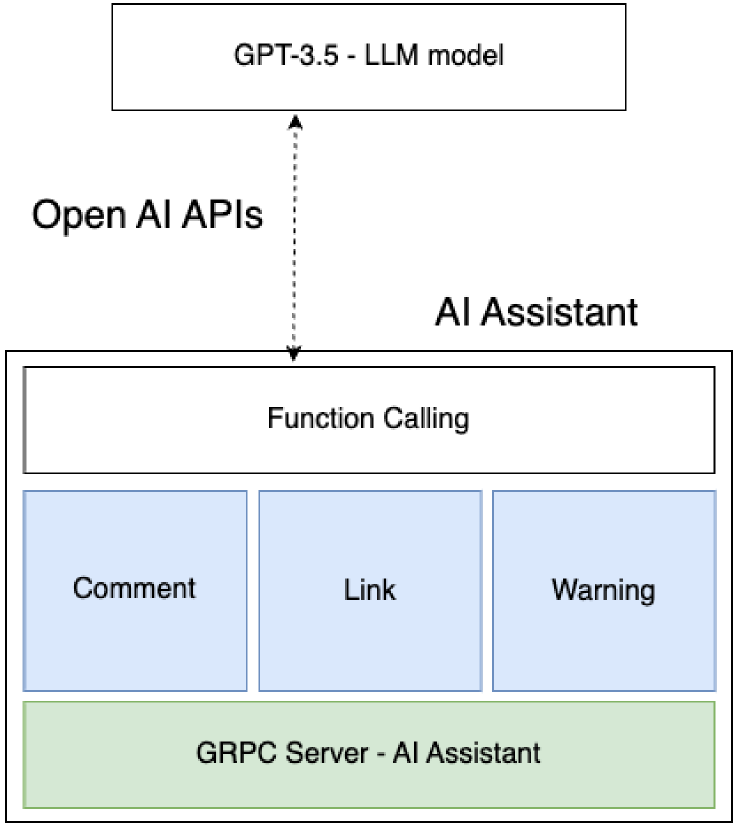


System Architecture

AI Assistant

- **Comments** on each of the Smart Contract components with a description.
- Identifies **links** that represent the functional relationships between elements.
- Highlights with **warnings** the functions that could be improved.

- These operations are performed by the **GPT 3.5** OpenAI model.

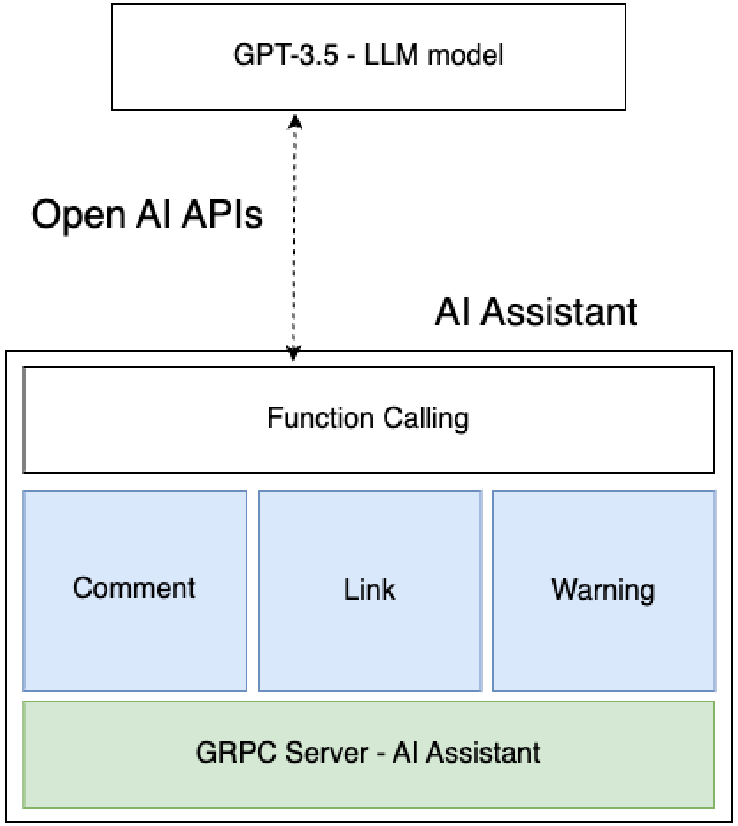


System Architecture

Function calling

Allows the LLM model to return the **results of a conversation in a predefined structured format**, ready to be processed by software.

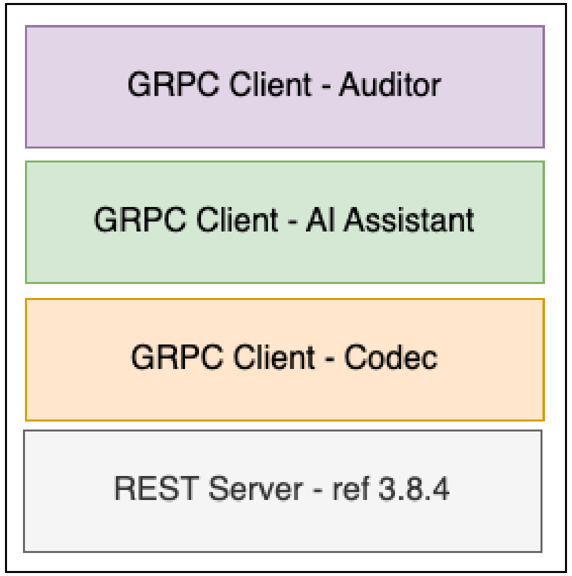
Requires **configurations from** the code.
Requires **specific prompts** that refer to the implemented configuration.



System Architecture

BFF

- **Orchestrates** the gRPC requests between the services.
- **Maintains the state** of the ongoing operations, as a Task object.
- **Interacts with the client** through a REST server.



Smart Contract Analysis and Visualization Software

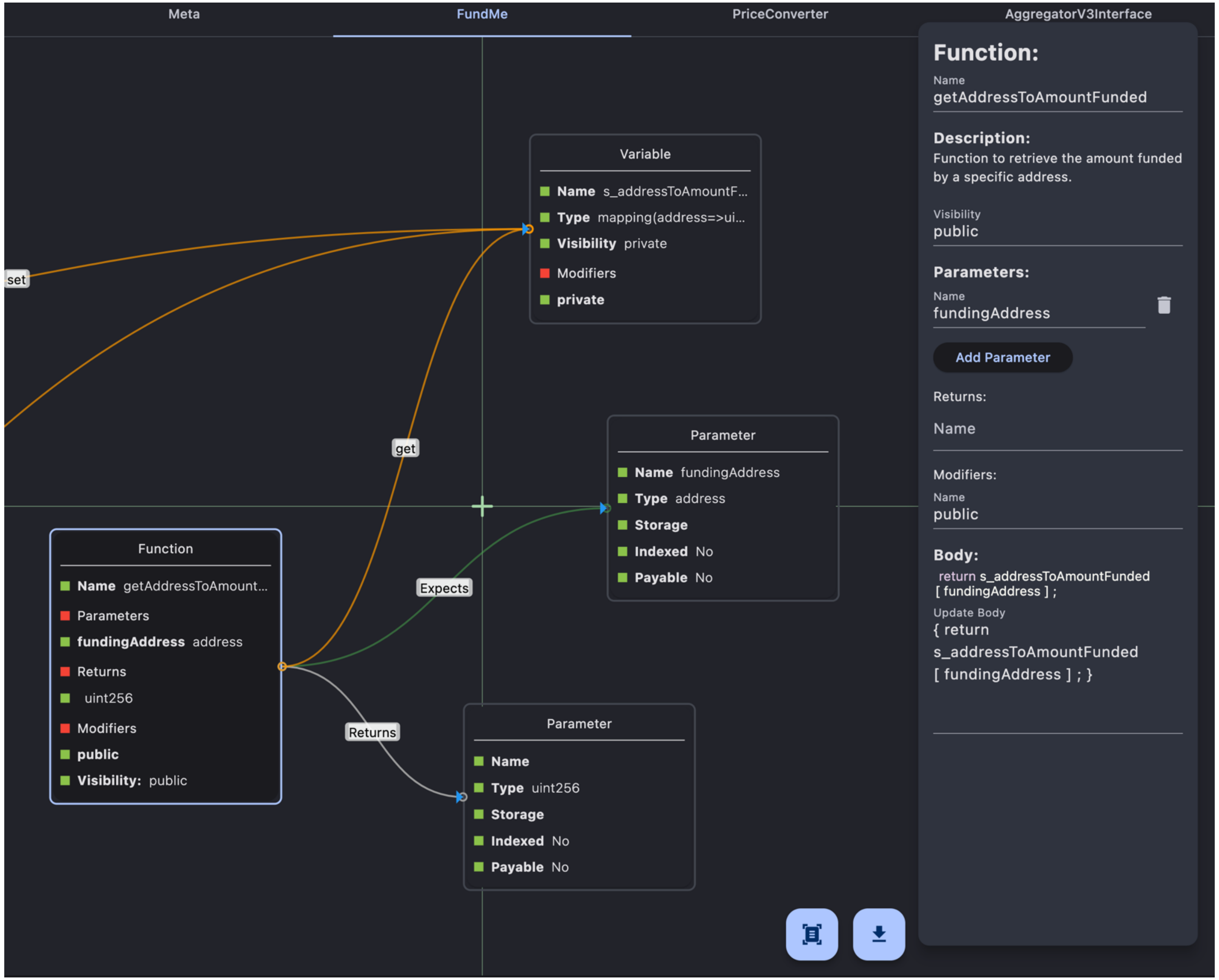
Frontend

Frontend

Editor Grid

The main client component, **displays the results** of the Smart Contract analysis.

Allows to **move around** freely, **interact with elements** and **apply changes**.



Variable

- 🟩 **Name** s_addressToAmountF...
- 🟩 **Type** mapping(address=>ui...
- 🟩 **Visibility** private
- 🟥 Modifiers
- 🟩 **private**

Parameter

- 🟩 **Name** fundingAddress
- 🟩 **Type** address
- 🟩 **Storage**
- 🟩 **Indexed** No
- 🟩 **Payable** No

Function

- 🟩 **Name** getAddressToAmount...
- 🟥 Parameters
- 🟩 **fundingAddress** address
- 🟥 Returns
- 🟩 uint256
- 🟥 Modifiers
- 🟩 **public**
- 🟩 **Visibility:** public

Parameter

- 🟩 **Name**
- 🟩 **Type** uint256
- 🟩 **Storage**
- 🟩 **Indexed** No
- 🟩 **Payable** No

Function:

Name
getAddressToAmountFunded

Description:
Function to retrieve the amount funded by a specific address.

Visibility
public

Parameters:

Name
fundingAddress

Add Parameter

Returns:

Name

Modifiers:

Name
public

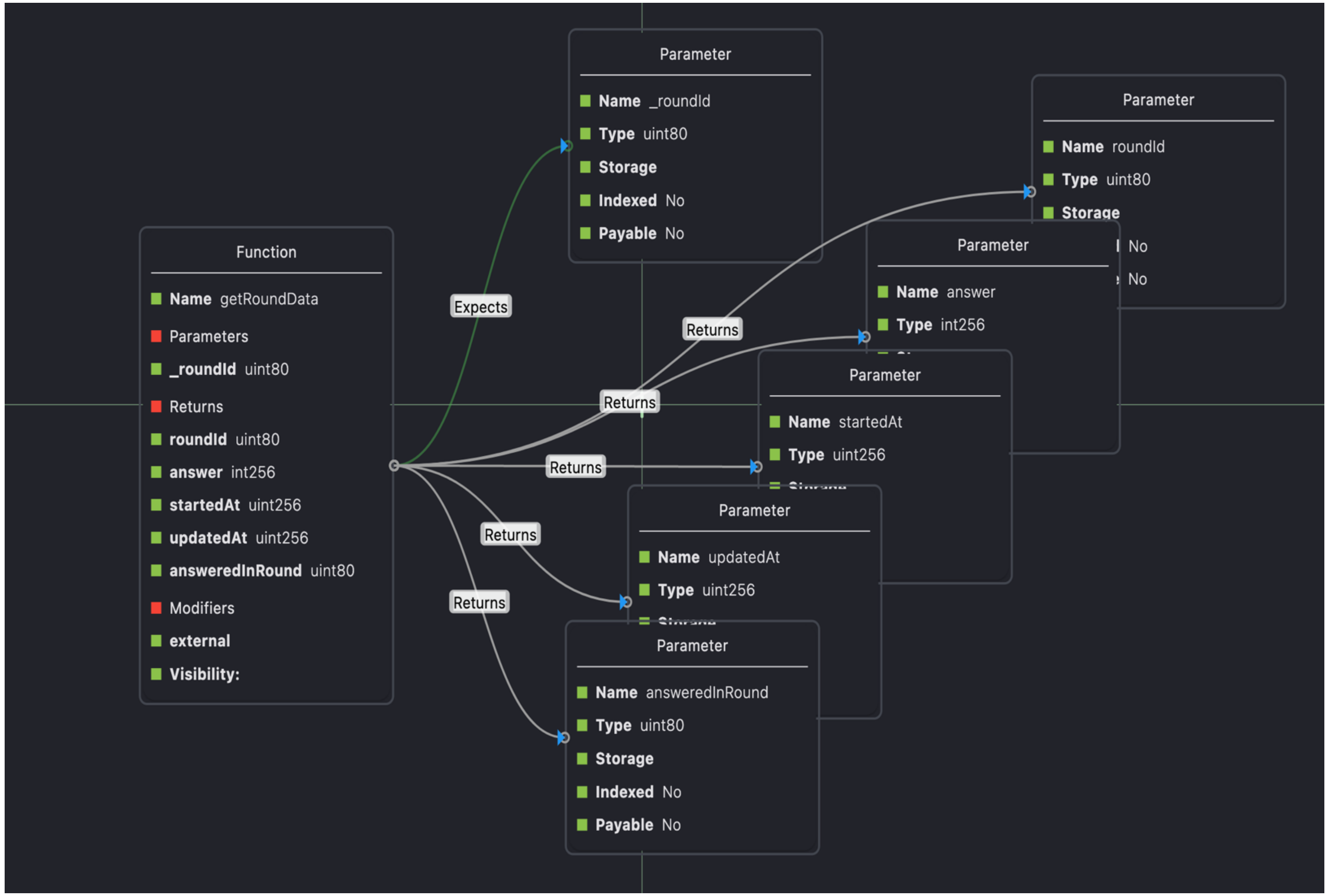
Body:
return s_addressToAmountFunded
[fundingAddress] ;

Update Body
{ return
s_addressToAmountFunded
[fundingAddress] ; }

Frontend

Links

Functional relationships between elements are displayed as dynamic connections.



Function

🟩 **Name** getRoundData
🟥 Parameters
🟩 **_roundId** uint80
🟥 Returns
🟩 **roundId** uint80
🟩 **answer** int256
🟩 **startedAt** uint256
🟩 **updatedAt** uint256
🟩 **answeredInRound** uint80
🟥 Modifiers
🟩 **external**
🟩 **Visibility:**

Parameter

🟩 **Name** _roundId
🟩 **Type** uint80
🟩 **Storage**
🟩 **Indexed** No
🟩 **Payable** No

Parameter

🟩 **Name** roundId
🟩 **Type** uint80
🟩 **Storage**
No
No

Parameter

🟩 **Name** answer
🟩 **Type** int256

Parameter

🟩 **Name** startedAt
🟩 **Type** uint256

Parameter

🟩 **Name** updatedAt
🟩 **Type** uint256

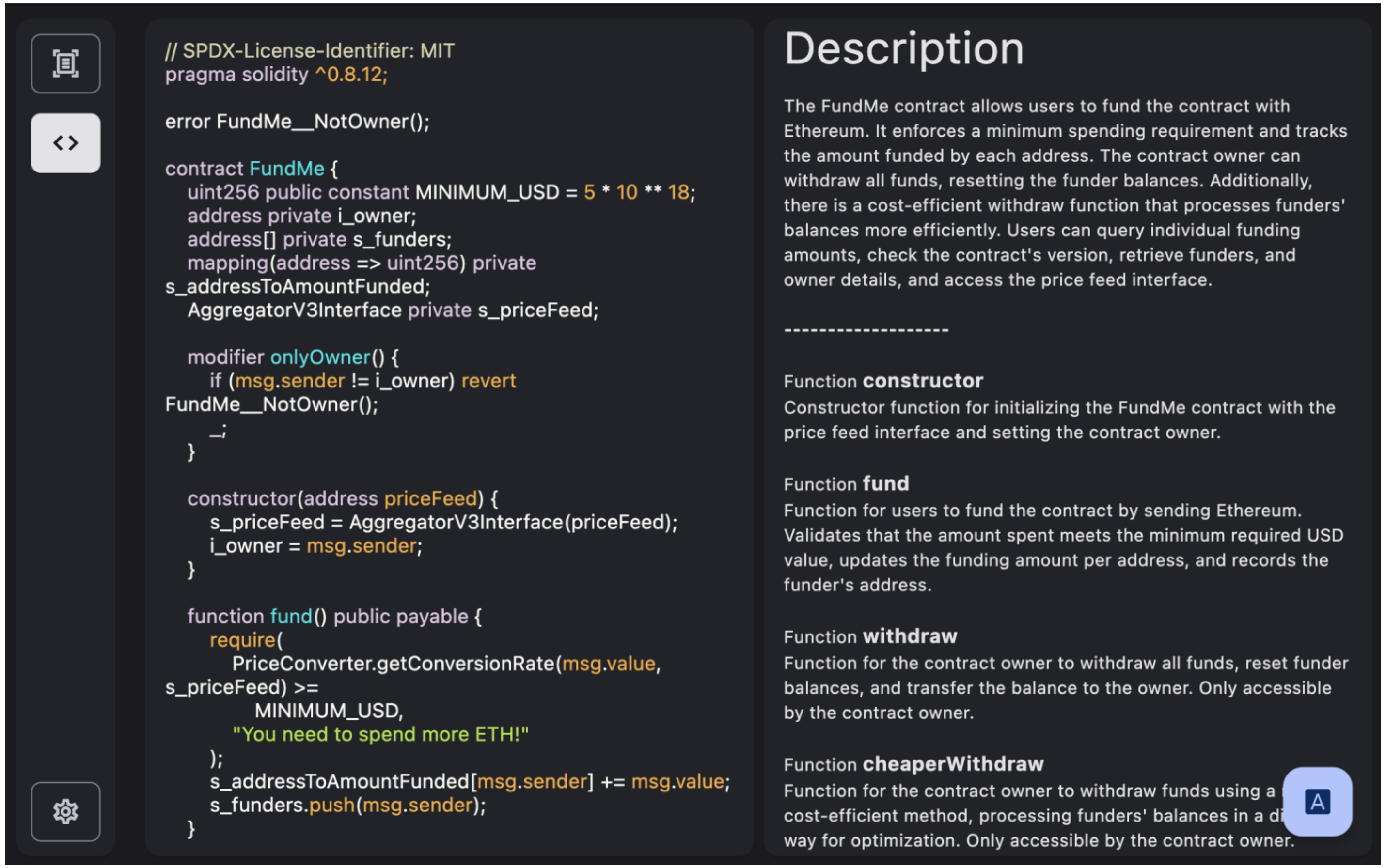
Parameter

🟩 **Name** answeredInRound
🟩 **Type** uint80
🟩 **Storage**
🟩 **Indexed** No
🟩 **Payable** No

Frontend

Code descriptions

The AI generated elements descriptions are listed in this page beside the Smart Contract code.



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.12;

error FundMe__NotOwner();

contract FundMe {
  uint256 public constant MINIMUM_USD = 5 * 10 ** 18;
  address private i_owner;
  address[] private s_funders;
  mapping(address => uint256) private
s_addressToAmountFunded;
  AggregatorV3Interface private s_priceFeed;

  modifier onlyOwner() {
    if (msg.sender != i_owner) revert
FundMe__NotOwner();
    _;
  }

  constructor(address priceFeed) {
    s_priceFeed = AggregatorV3Interface(priceFeed);
    i_owner = msg.sender;
  }

  function fund() public payable {
    require(
      PriceConverter.getConversionRate(msg.value,
s_priceFeed) >=
        MINIMUM_USD,
      "You need to spend more ETH!"
    );
    s_addressToAmountFunded[msg.sender] += msg.value;
    s_funders.push(msg.sender);
  }
```

Description

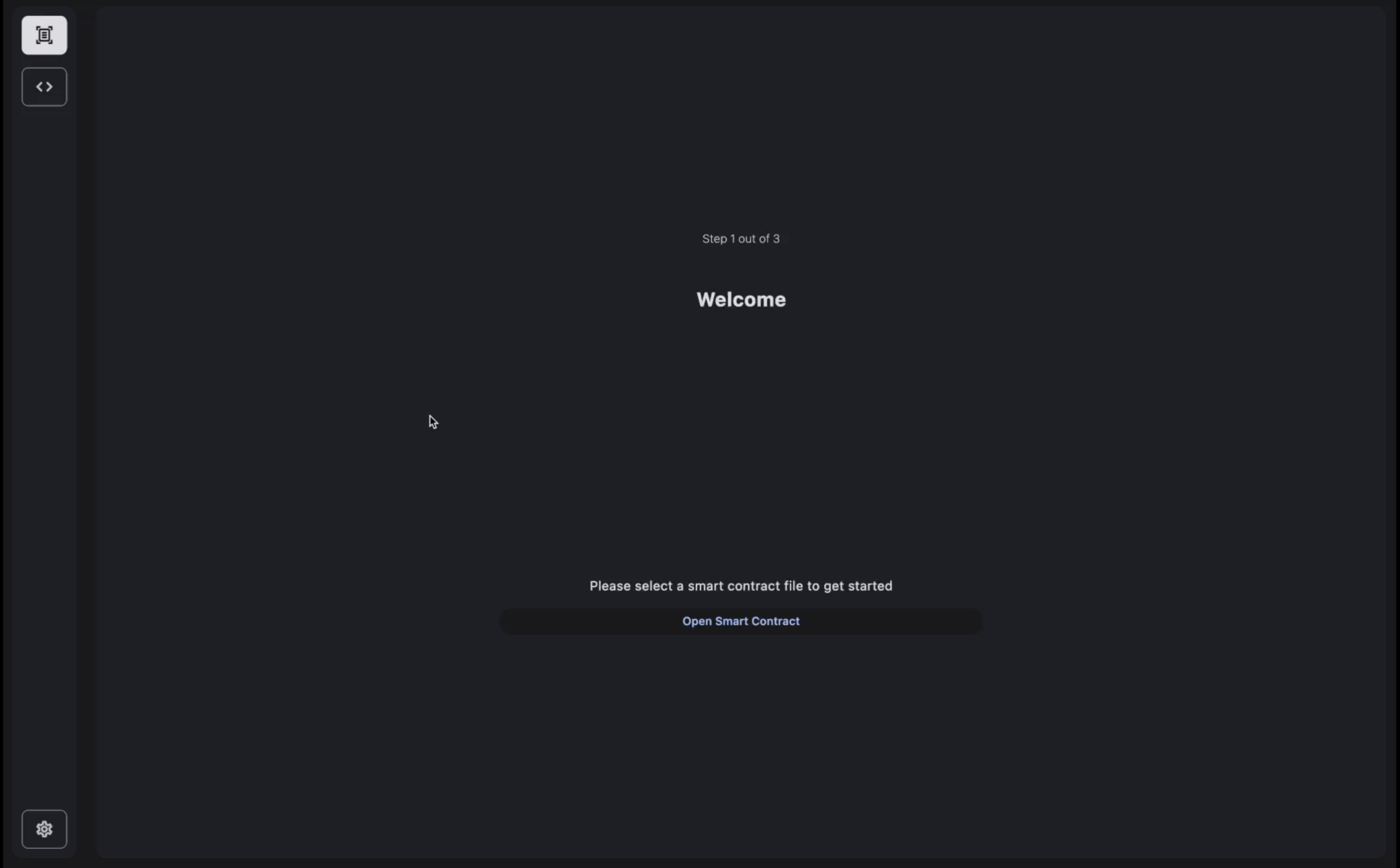
The FundMe contract allows users to fund the contract with Ethereum. It enforces a minimum spending requirement and tracks the amount funded by each address. The contract owner can withdraw all funds, resetting the funder balances. Additionally, there is a cost-efficient withdraw function that processes funders' balances more efficiently. Users can query individual funding amounts, check the contract's version, retrieve funders, and owner details, and access the price feed interface.

Function **constructor**
Constructor function for initializing the FundMe contract with the price feed interface and setting the contract owner.

Function **fund**
Function for users to fund the contract by sending Ethereum. Validates that the amount spent meets the minimum required USD value, updates the funding amount per address, and records the funder's address.

Function **withdraw**
Function for the contract owner to withdraw all funds, reset funder balances, and transfer the balance to the owner. Only accessible by the contract owner.

Function **cheaperWithdraw**
Function for the contract owner to withdraw funds using a
cost-efficient method, processing funders' balances in a di
way for optimization. Only accessible by the contract owner.



Step 1 out of 3

Welcome

Please select a smart contract file to get started

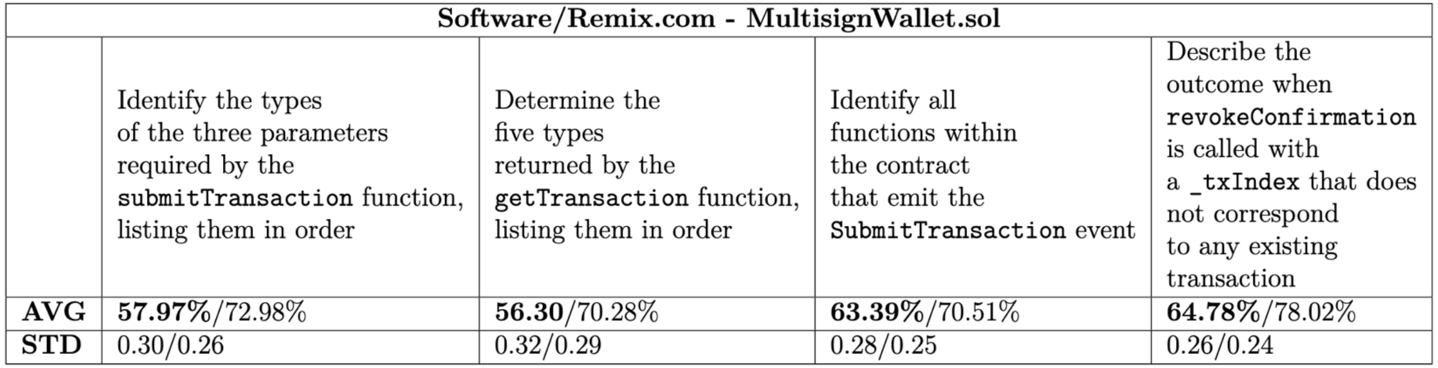
Results

Politecnico
di Torino

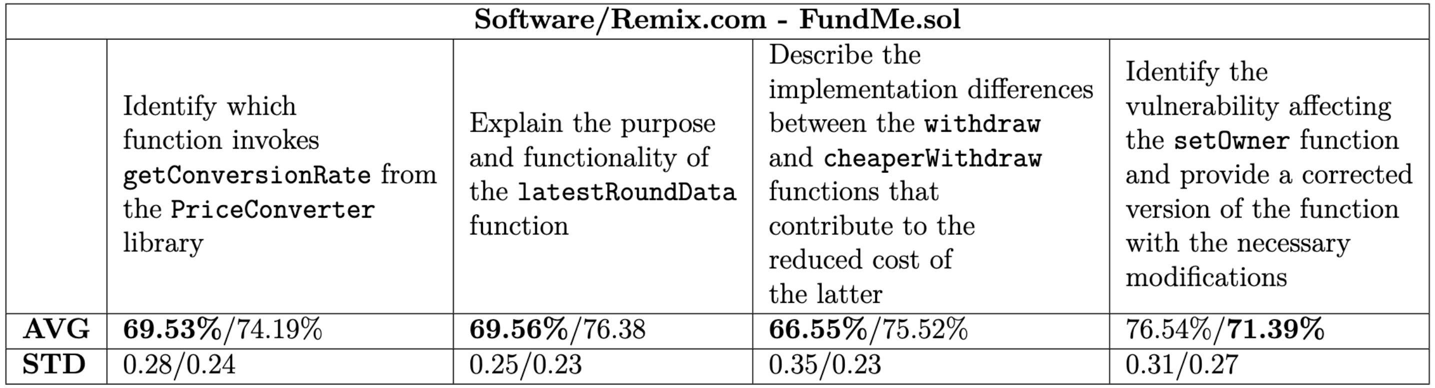
Results

Survey

- **Two groups** of users.
- **Two Smart Contracts,** each with 4 questions.
- Tested against both the **Thesis Software and the Remix IDE.**



	Software/Remix.com - MultisignWallet.sol			
	Identify the types of the three parameters required by the `submitTransaction` function, listing them in order	Determine the five types returned by the `getTransaction` function, listing them in order	Identify all functions within the contract that emit the `SubmitTransaction` event	Describe the outcome when `revokeConfirmation` is called with a `_txIndex` that does not correspond to any existing transaction
AVG	**57.97%**/72.98%	**56.30**/70.28%	**63.39%**/70.51%	**64.78%**/78.02%
STD	0.30/0.26	0.32/0.29	0.28/0.25	0.26/0.24

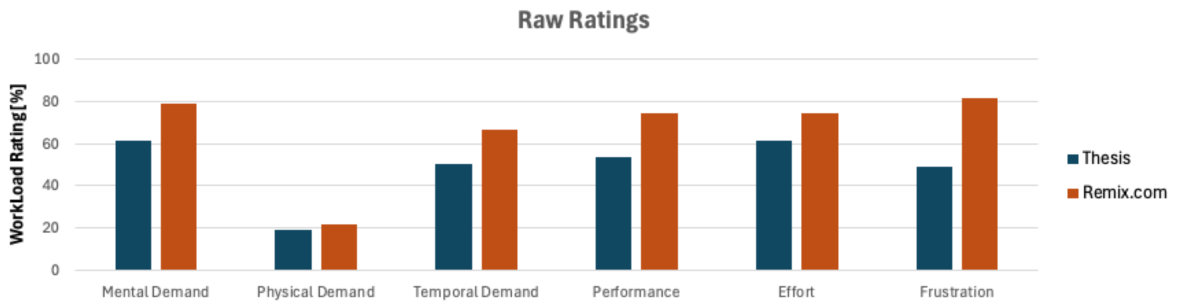


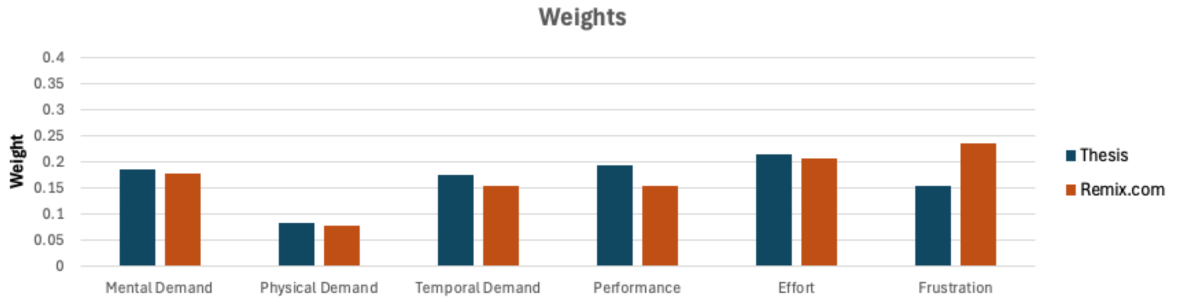
	Software/Remix.com - FundMe.sol			
	Identify which function invokes `getConversionRate` from the `PriceConverter` library	Explain the purpose and functionality of the `latestRoundData` function	Describe the implementation differences between the `withdraw` and `cheaperWithdraw` functions that contribute to the reduced cost of the latter	Identify the vulnerability affecting the `setOwner` function and provide a corrected version of the function with the necessary modifications
AVG	**69.53%**/74.19%	**69.56%**/76.38	**66.55%**/75.52%	76.54%/**71.39%**
STD	0.28/0.24	0.25/0.23	0.35/0.23	0.31/0.27

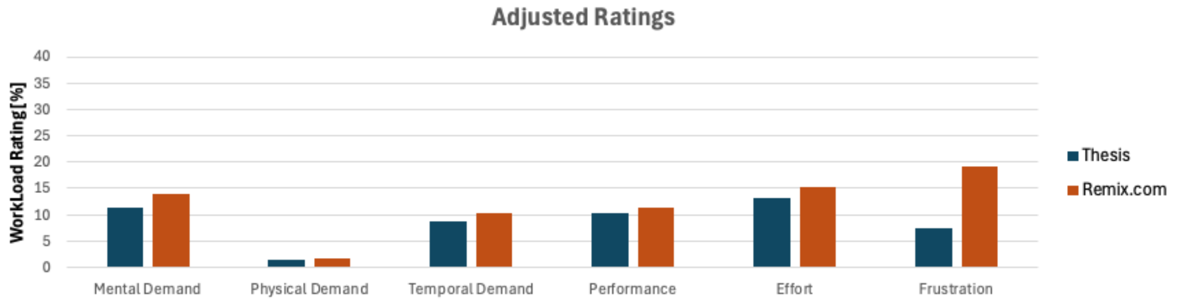
Results

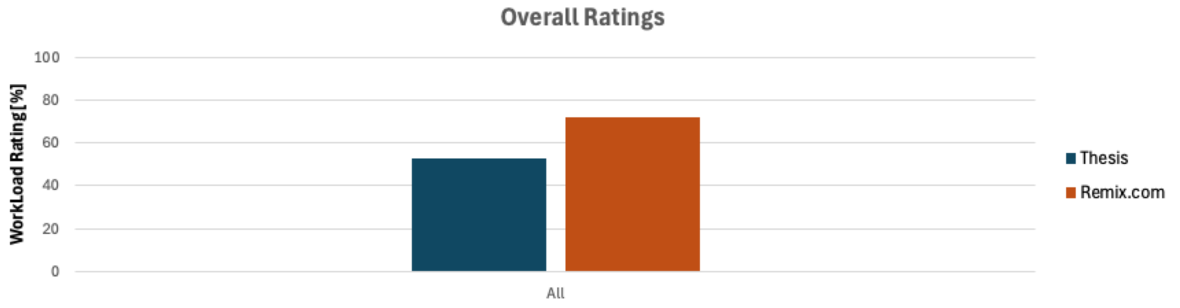
NASA-TLX

- Lower **mental demand, temporal demand** and reduced **effort.**
- Significant reduction in **Frustration.**
- Overall difference in score of 19 points for the Software.


Raw Ratings


Weights


Adjusted Ratings


Overall Ratings

Smart Contract Analysis and Visualization Software

Conclusions

Politecnico di Torino

Conclusions

- The Software met the goals by integrating the three key features.
- The survey resulted in positive results, with average performances better across tasks compared to the same using Remix.
- The NASA-TLX results support the software's potential advantages over Remix.

The system bridges the gap between complex smart contract development and accessibility, by making use of LLMs, Static Analysis tools, a network of microservices and an intuitive graphic interface, providing a solid structure for future enhancements.

Thank you for your time

Politecnico di Torino