BLOCKCHAIN LABORATORY

SavingsVault | Simple ETH Smart Contract

Solidity Blockchain Dev

Student name	Email	Github repo link
Guilherme Gomes de Andrade	andrade2002g@gmail.com	https://github.com/adr-g/solidity-smart-contract-assignment

Beograd, 2025

Content

1 Application Description

This project consists of a very simple smart contract called SavingsVault.

The idea is straightforward: the contract allows ETH to be deposited and later withdrawn. Any user can deposit ETH, but only the contract owner (defined at deployment) can withdraw the funds.

The goal of this project is not to build a complex application, but to demonstrate the basic use of Solidity and how to deploy and interact with a smart contract on the Ethereum blockchain.

2 Implementation

2.1 List of Functions

constructor()

- Function name: constructor
- Purpose: Sets the address that deploys the contract as the owner.
- Input Parameters: None
- Return Type: None
- Access modifier: n/a

deposit()

- Function name: deposit
- Purpose: Allows ETH to be sent to the contract.
- Input Parameters: None (ETH is sent using msg.value)
- Return Type: None
- Access modifier: external payable

withdrawAll()

- Function name: withdrawAll
- Purpose: Allows the owner to withdraw all ETH stored in the contract.
- Input Parameters: None
- Return Type: None
- Access modifier: external

getVaultBalance()

- Function name: getVaultBalance
- Purpose: Returns the current balance of the contract.
- Input Parameters: None
- Return Type: uint256
- Access modifier: external view

2.2 Events & Validations

Events

- FundsDeposited
  Emitted when ETH is deposited into the contract.
- FundsWithdrawn
  Emitted when ETH is withdrawn by the owner.

Validations

The require() function is used to keep the contract logic simple and safe:

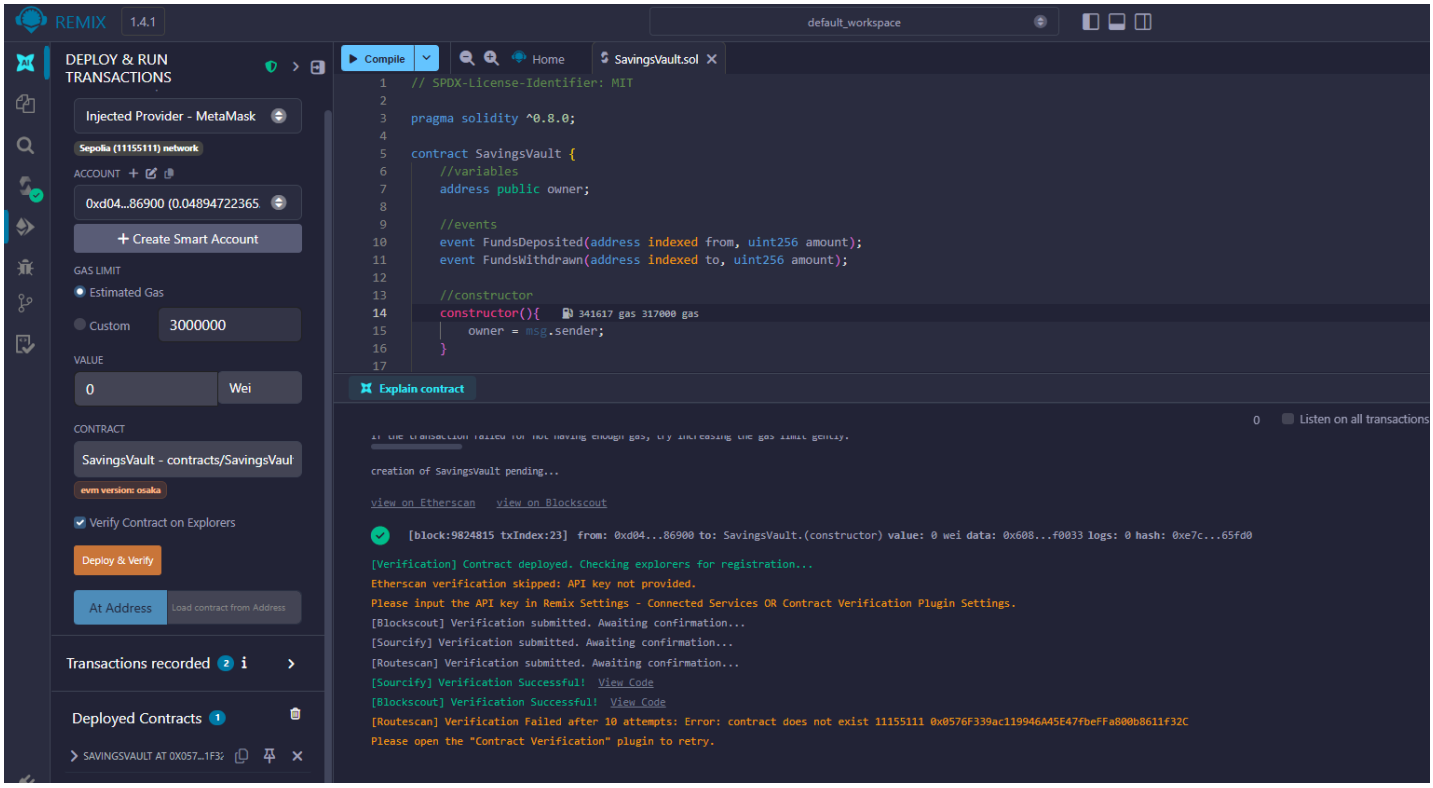
- To avoid depositing zero ETH;
- To make sure only the owner can withdraw from the contract;
- To avoid withdrawing when there are no funds available.

3 Testing

The contract was tested using the Remix IDE and deployed on the Sepolia test network with MetaMask.

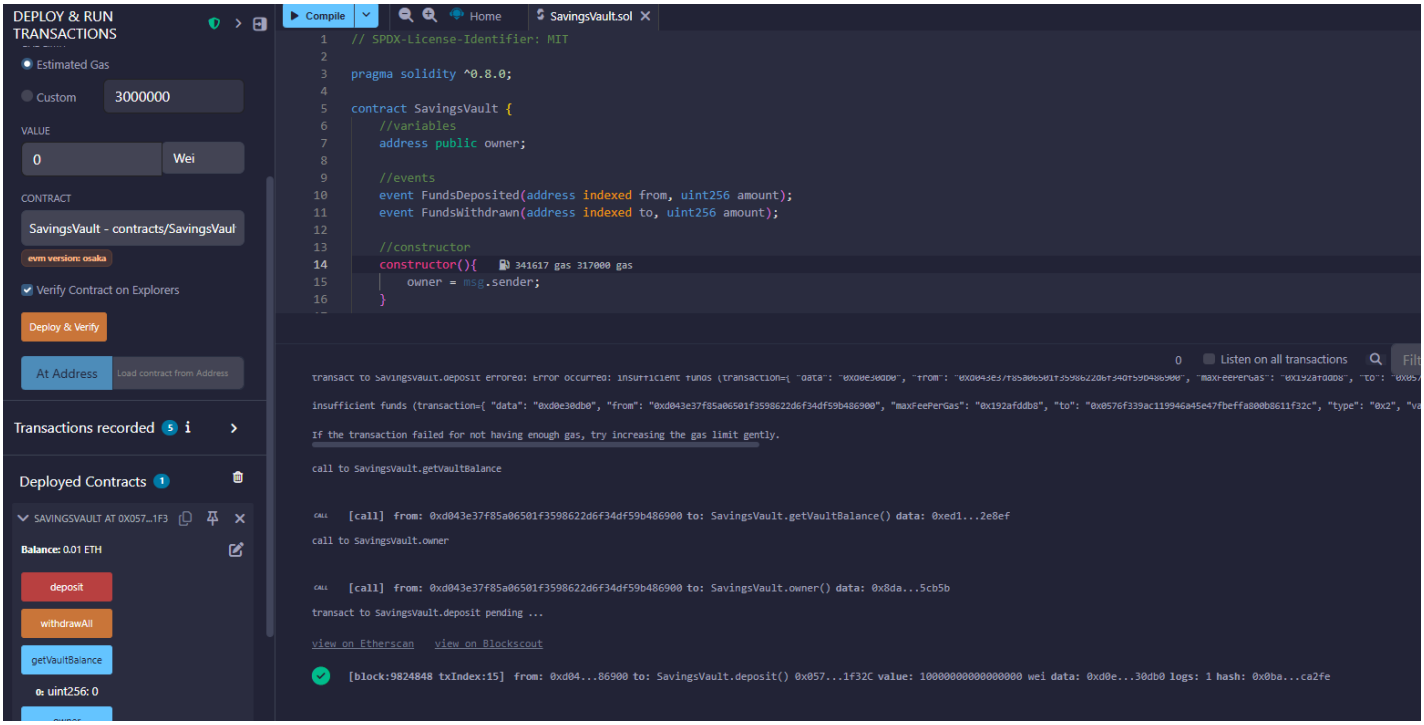
Testing included:

- Compiling and deploying the contract;



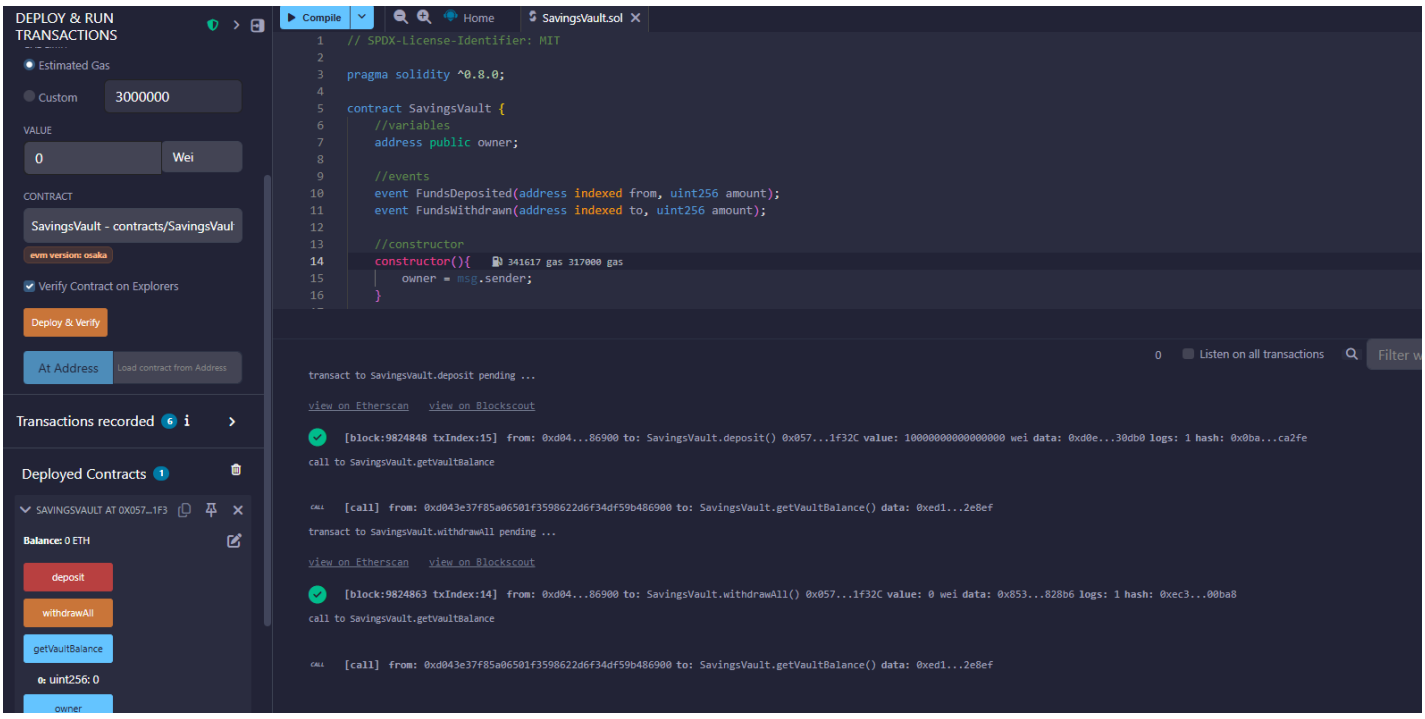
Img 1 - 01_deploy_savingsvault_sepolia

- Calling the deposit() function with test ETH;



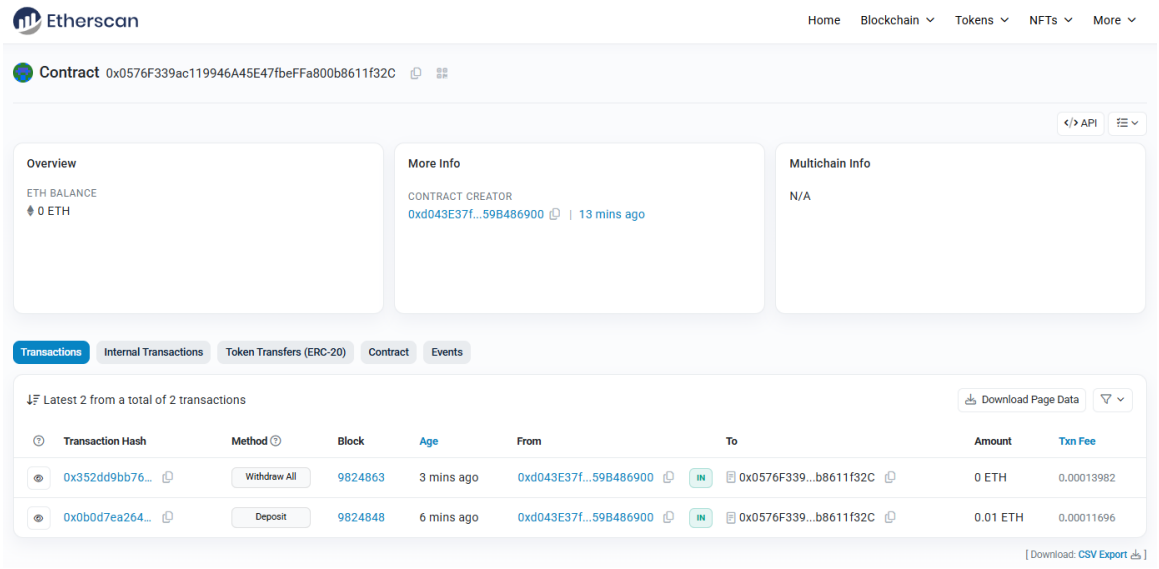
Img 2 - 02_deposit_savingsvault_sepolia

- Calling the withdrawAll() function;

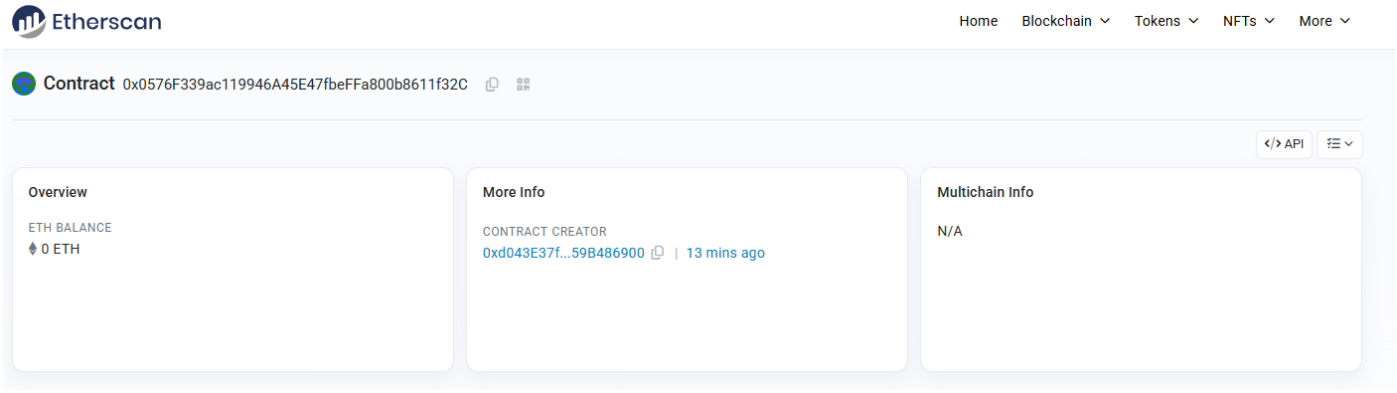


Img 3 - 03_withdrawAll_savingsvault_sepolia

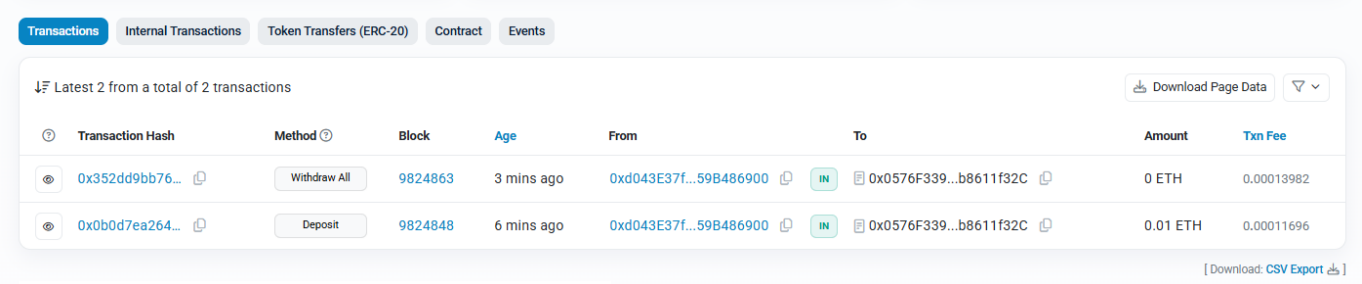
- Verifying transactions on Sepolia Etherscan.



Img 4 - 04_etherscan_savingsvault_sepolia



Img 5 - 05_etherscan_contract_creation



Img 6 - 06_etherscan_transactions