**Batch: A2 Roll No.: 16010121045**

**Mini Project**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |  |
| --- | --- |
| **Title:** | **Mini Project** |

**Implementation Details:**

*// SPDX-License-Identifier: UNLICENSED*

*pragma* solidity ^0.8.9;

contract CrowdFunding {

struct Campaign {

address owner;

string title;

string description;

uint256 target;

uint256 deadline;

uint256 amountCollected;

string image;

uint256[] donations;

address[] donators;

}

*// helps to use campaigns[0] in solidity*

mapping(uint256 => Campaign) public campaigns;

*// global variable*

uint256 public numberOfCampaigns = 0;

*// create a new campaign*

function createCampaign(

address \_owner,

string memory \_title,

string memory \_description,

uint256 \_target,

uint256 \_deadline,

string memory \_image

) public *returns* (uint256) {

Campaign storage campaign = campaigns[numberOfCampaigns];

*require*(\_deadline < *block*.timestamp , "The deadline should be an upcoming date.");

campaign.owner = \_owner;

campaign.title = \_title;

campaign.description = \_description;

campaign.target = \_target;

campaign.deadline = \_deadline;

campaign.image = \_image;

numberOfCampaigns++;

*return* numberOfCampaigns-1;

}

*// donate to a campaign*

function donateToCampaign(uint256 \_id) public payable{

uint256 amount = *msg*.value;

Campaign storage campaign = campaigns[\_id];

campaign.donators.push(*msg.sender*);

campaign.donations.push(amount);

(bool sent,) = payable(campaign.owner).call{value: amount}("");

*if*(sent){

campaign.amountCollected = campaign.amountCollected + amount;

}

}

*// get donators*

function getDonators(uint256 \_id) public view *returns*(address[] memory, uint256[] memory){

*return* (campaigns[\_id].donators , campaigns[\_id].donations);

}

*// get all campaigns*

function getCampaigns() public view *returns* (Campaign[] memory) {

Campaign[] memory \_campaigns = *new* Campaign[](numberOfCampaigns);

*for*(uint256 i = 0; i < numberOfCampaigns; i++){

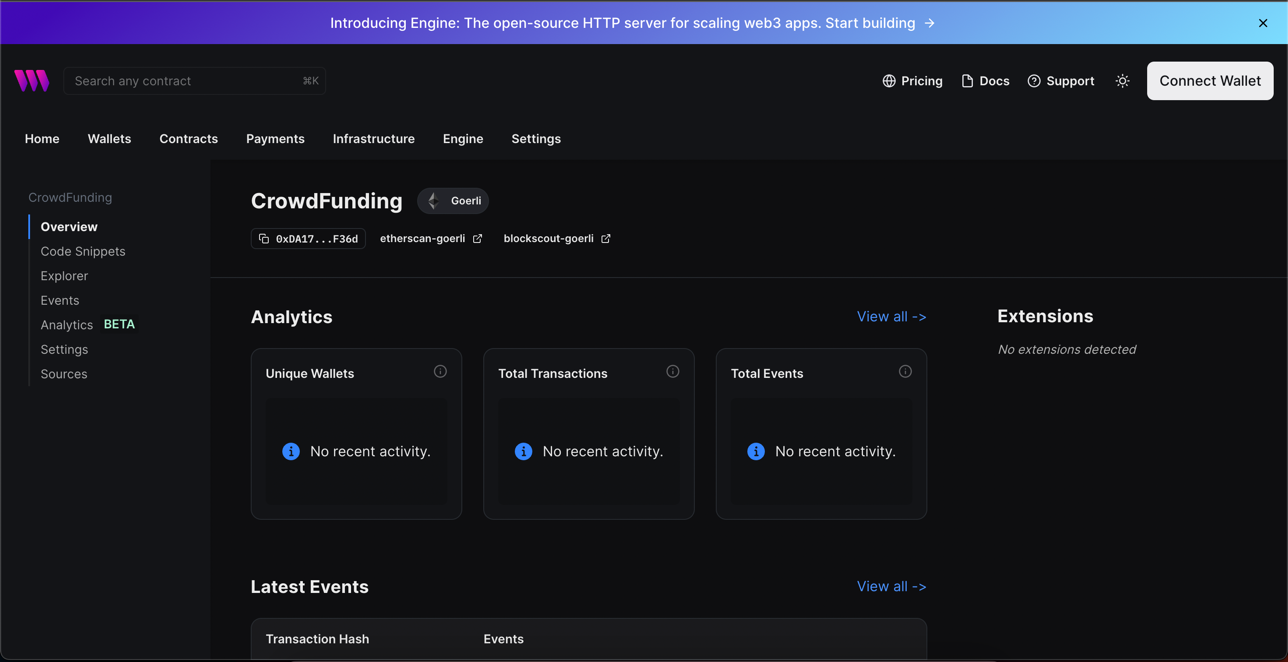
\_campaigns[i] = campaigns[i];

}

*return* \_campaigns;

}

}

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Conclusion:-**

In this experiment, we learnt about eploy your first Solidity smart contract with Remix IDE.