CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Crowdfunding has become a powerful tool for individuals, startups, and communities to raise funds and turn new ideas into reality. However, most of the money spent in the sector relies on centralized institutions, leading to high transaction costs, lack of transparency, and limited implementation. These challenges often prevent founders and investors from realizing the full potential of funding. Blockchain provides transparency by recording all transactions immutably, while smart contracts reduce the risk of fraud and conflict by automating processes such as fund distribution and goal setting. Web3 continues to evolve this model by allowing users to interact with decentralized platforms through digital wallets, removing intermediaries and creating a trustless environment. By integrating blockchain, smart contracts, and Web3 technology, it provides developers with a secure, transparent, and efficient platform that can connect directly with users to drive innovation and economic growth.

1.2 NEED AND SIGNIFICANCE

The traditional financial model, despite its widespread use, faces inherent problems such as high transaction costs, delays in processing funds, and lack of transparency. These limitations create problems for planners seeking support and investors seeking a trust period. Furthermore, relying on intermediaries increases operating costs and reduces accessibility, especially for international audiences. This project solves these problems by using blockchain technology and Web3 to create a shared network. Its main goal is to provide a transparent, secure, and automated ecosystem that manages funds through immutable smart contracts. Developers can present their ideas on an international stage, while investors are guaranteed transparency and security. The platform reduces costs by eliminating the middleman, enables faster transactions and increases trust between participants. It offers the best features such as revenue and performance tracking to promote better interaction and financial growth. The project embodies the future of crowdfunding by empowering creators and investors through decentralized governance.

1.3 OBJECTIVE

The main goal of the project is to create a shared network using blockchain technology and Web3 tools. The platform aims to connect project developers with potential investors by providing a secure, transparent and automated ecosystem. The system eliminates intermediaries, reduces transaction costs and increases user trust through immutable smart contracts and real-time verification of participation. Integrating features such as revenue and advertising, the platform aims to support innovation while providing a great customer experience.

1.4 PURPOSE

The aim of this project is to solve the problem of inefficiency of the traditional fin ancial aid system. Existing platforms often include high costs, slow performance, and la ck of transparency, which discourages both developers and investors. The project levera ges the decentralized nature of blockchain to provide direct relationships between creato rs and investors without the need for intermediaries. It allows developers to present their ideas globally, while allowing investors to secure and finance projects. The platform au tomates the process through smart contracts, ensuring fair distribution of funds and redu cing the risk of fraud.

1.5 INTENDED USER

The platform is designed for two primary user groups:

- Project Creators: Entrepreneurs, innovators, and organizations looking to raise funds for their ideas or initiatives.
- Investors: Individuals or entities seeking to support promising projects and gain potential returns.
- Additionally, blockchain enthusiasts and cryptocurrency holders can also engage with the platform, making it a versatile tool for various stakeholders in the decentralized finance ecosystem.

1.6 APPLICABILITY

This platform is applicable across multiple domains, such as:

- Startups and Entrepreneurs: Seeking seed funding for innovative ideas.
- Creative Projects: Artists, filmmakers, and authors requiring financial backing.
- Community Initiatives: Fundraising for social or environmental causes.
- Research and Development: Supporting scientific or technological advancements. Its borderless nature enables global participation, making it ideal for projects requiring diverse funding sources and audience engagement.

1.7 COMPONENTS

The project comprises the following components:

- Smart Contracts: Written in Solidity, these automate campaign creation, funding, and fund disbursement.
- Web3 Integration: Facilitates seamless interaction with blockchain via a decentralized user interface.

- Frontend Interface: Built using React and JavaScript, it allows users to view campaigns, contribute funds, and create projects.
- Blockchain Network: Ethereum serves as the backbone, ensuring transparency and security.
- Wallet Integration: Enables users to connect wallets for transactions in cryptocurrencies like Ethereum.
- Reporting Tools: Allow users to generate campaign performance reports and track growth metrics.

1.8 LIMITATIONS

- High Gas Fees: Transactions on Ethereum can be expensive during network congestion.
- User Adoption: Non-technical users may face challenges interacting with Web3 tools.
- Regulatory Concerns: Cryptocurrencies are subject to varying regulations, which may affect platform adoption.
- Dependency on Internet Access: Requires reliable internet for seamless interaction with blockchain networks.
- Smart Contract Vulnerabilities: Although secure, smart contracts are prone to bugs if not rigorously tested.

1.9 FEASIBLITY STUDY

The feasibility of this project is analysed as follows:

- Technical Feasibility: Utilizing proven technologies like Ethereum, Solidity, and Web3.js ensures technical viability. The tools are well-documented and widely adopted in the blockchain ecosystem.
- Economic Feasibility: The decentralized nature reduces operational costs by eliminating intermediaries. Initial development costs are outweighed by long-term savings and revenue potential.
- Operational Feasibility: Users can interact via an intuitive frontend, ensuring ease of use. Clear documentation and tutorials further enhance usability.
- Legal Feasibility: Although cryptocurrency regulations vary, the platform can adapt by supporting compliance mechanisms. Future upgrades may incorporate KYC processes to meet legal standards. This project is both practical and impactful, offering a sustainable solution to the limitations of traditional crowdfunding.

CHAPTER 2

PROBLEM STATEMENT AND LITERATURE REVIEW

2.1 PROBLEM STATEMENT

While there are always many users who are able to connect well with creators and backers, there are also some disadvantages, benefits, and limitations. Centralized systems dominate the crowdfunding space, leading to high operating costs, lack of transparency, and poor usability. These platforms often charge high fees, which reduces the funds available to complete the project. The importance of these platforms also shows a single flaw, which makes them vulnerable to fraud, data manipulation, and auditing. Sponsors often have little understanding of how their money will be spent, leading to scepticism and reluctance.

On the other hand, the project developer faces problems in international expansion due to regional limitations and mediocre collaboration. The manual process of allocating funds can also be slow and confusing, further increasing user confidence. For example, funds may be distributed to developers without clear evidence of success or significant achievement. This lack of accountability creates risk for sponsors and hinders innovation. However, mass adoption of these platforms is still in its infancy. Many existing solutions lack user-friendly interfaces and lack important features such as financial, reporting performance, and real-time transparency.

A platform to solve inefficiencies. Such platforms can empower creators and supporters by eliminating middlemen, reducing costs, and providing a transparent, secure, and automated environment. The goal is to close this gap by creating solutions that increase accountability, foster trust, and encourage international collaboration on innovation and user goods.

2.2 LITERATURE REVIEW

Crowdfunding has evolved from centralized methods such as Kickstarter and GoFundMe to alternative methods supported by blockchain technology. This study shows that the centralized system faces problems such as high costs, lack of transparency, and limited usability. Blockchain-based solutions solve these problems by providing immutability, transparency, and financial control through smart contracts. Research on Web3 technologies shows their potential to enable trustless interactions and global collaboration in financial aid. However, existing platforms often lack user-friendly interfaces, effective advertising tools, and effective hierarchical accounting tools. The

plan builds on these findings by integrating blockchain and Web3 to create a secure, transparent, and user-centric crowdfunding platform designed for developers and investors worldwide.

S.No.	Year	Author Name	Contribution
1	2014	Paul Belleflamme [2]	The paper explores preferences between pre-ordering and profit-sharing crowdfunding, highlighting implications for early-stage managerial decisions and community building.
2	2021	Alexa Böckel [2]	The article reviews crowdfunding and sustainability research, identifies gaps, and recommends focusing on environmental and post-funding impacts for sustainable development.
3	2017	Douglas J. Cumming[2]	The paper reveals that cleantech crowdfunding is influenced by cultural and economic factors, with higher success tied to campaign characteristics like video pitches.
4	2016	Benjamin Gaddy [3]	The paper assesses the high-risk, low-return profile of cleantech VC investments and advocates for broader support and policy measures to foster innovation.
5	2021	Maria Manganiello1 [1]	The paper analyzes the impact of COVID-19 on green equity crowdfunding success, highlighting sustainability's growing investor appeal.
6	2019	Md Nazmus Saadat [4]	Explores blockchain-based crowdfunding systems emphasizing transparency, security, and efficiency in fundraising.
7	2024	Tobias Guggenberger [4]	Discusses designing blockchain tokens for equity crowdfunding to enhance investment security and scalability.
8	2018	Estrin [2]	Examines the evolution of equity crowdfunding and its impact on entrepreneurs and investors entering new markets.
9	2010	Carree [2]	Analyzes the relationship between entrepreneurship and economic growth through comprehensive research.
10	2018	Chen [1]	Explores how blockchain tokens democratize entrepreneurship by enabling innovative and decentralized funding models.
11	2020	Nagel [3]	Benchmarks global trends and challenges in lending, equity, and alternative finance models.
12	2019	Kranz [4]	Reviews blockchain token sales and their impact on modern fundraising and entrepreneurial ecosystems.
13	2019	Hartmann [6]	Compares success factors of blockchain-based crowdfunding versus traditional methods, emphasizing transparency.
14	2021	Treiblmaier [10]	Discusses future blockchain research directions, covering innovation and integration in various industries.
15	2020	Yang [5]	Investigates the application of blockchain in construction for integrating business processes and improving data management.

Table: 2.1 Literature review

CHAPTER 3

PROBLEM REQUIREMENT AND ANALYSIS

3.1 GANTT CHART

The Gantt chart shows the key stages such as requirements analysis, design, smart contract, front-end integration, testing, and deployment. By mapping these stages, it provides a way to complete the project within the allotted time. It helps identify conflicts, operational dependencies, and potential clashes so that adjustments can be made to reduce delay. communication. This transparency ensures accountability and adherence to project goals. It follows the agile development process by supporting progress tracking and dynamic change.

TASK	START DATE	END DATE	DURATION
Project Title	9-Sep-24	9-Sep-24	1
Project Planning	9-Sep-24	10-Sep-24	2
Resources & info Collection	10-Sep-24	11-Sep-24	2
Synopsis & PPT work	10-Sep-24	16-Sep-24	7
Synopsis Submission & Presentation 1	17-Sep-24	17-Sep-24	1
MERN Stack Setup	17-Sep-24	18-Sep-24	2
Smart Contract Development	18-Sep-24	28-Sep-24	11
Blockchain Integration	18-Sep-24	28-Sep-24	11
Frontend and Backend Development	28-Sep-24	10-Oct-24	13
Report and Presentaion Work	10-Oct-24	15-Oct-24	6
Project and Presentation 2	15-Oct-24	15-Oct-24	1
Testing and Deployement	15-Oct-24	20-Oct-24	6
Completing Final Report	20-Oct-24	30-Oct-24	10
Research Paper	30-Oct-24	5-Nov-24	7
Internal Presentation	10-Nov-24	10-Nov-24	1

Table: 3.1 Gantt Chart

TASK	DURATION							
	1st Week	2nd week	3rd week	4th week	5th week	6th week	7th week	8th week
Project Title								
Project Planning								
Resources & info Collection								
Synopsis & PPT work								
Synopsis Submission & Presentation 1								
MERN Stack Setup								
Smart Contract Development								
Blockchain Integration								
Frontend and Backend Development								
Report and Presentaion Work								
Project and Presentation 2								
Testing and Deployement								
Completing Final Report								
Research Paper								
Internal Presentation								

3.2 TECHNOLOGY REQUIRED

3.2.1 Frontend development

The frontend is built using React, a JavaScript library for building user interfaces. Vite is chosen for its fast bundling and hot reloading capabilities. Key technologies and tools used are:

- React: Provides component-based architecture, enabling the efficient development of reusable UI components.
- Vite: Offers fast build times and optimized development experience for the React framework.
- JavaScript (JS): Handles the interactive elements of the platform, such as user inputs, event handling, and integration with the blockchain.
- HTML & CSS: HTML structures the application while CSS is used for styling, ensuring a responsive, user-friendly interface.

The frontend interacts with the backend via smart contracts, allowing users to create, view, and contribute to campaigns in a secure and decentralized manner.

3.2.2 Backend development (Blockchain)

The backend logic is implemented using **Solidity**, the programming language for writing smart contracts on the Ethereum blockchain. The decentralized nature of blockchain ensures:

- Security: The use of Ethereum blockchain guarantees that funds are secure and that contributors have full transparency over fund allocation.
- Smart Contracts: These contracts manage the entire lifecycle of crowdfunding campaigns, from creation to fund allocation and withdrawal, based on predefined conditions.

The smart contract handles:

- Campaign creation and validation.
- Receiving contributions from users.
- Ensuring withdrawal conditions are met before releasing funds to the campaign creator.

3.3.3 Integration

The frontend and backend are integrated through web3.js or ethers.js, libraries that allow React to communicate with the Ethereum blockchain. Users interact with the frontend, which in turn sends requests to the smart contracts deployed on the blockchain. This interaction ensures that all transactions are recorded and validated on the blockchain, providing transparency and immutability.

CHAPTER 4

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

This DFD is for a crowdfunding platform where Contributors can add and deploy campaigns, and Investors can fund campaigns and view transactions. The Crowdfund entity likely represents the core platform where campaigns are managed. This is a very high-level view (Level 0 DFD) showing only the main entities and their interactions.

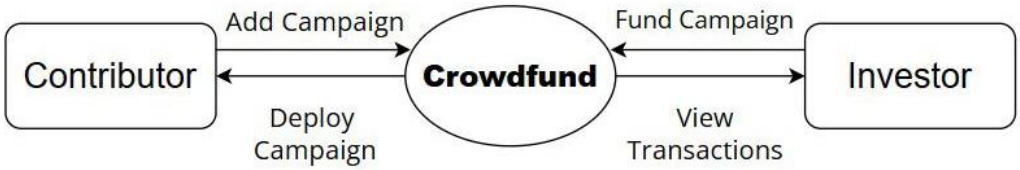


Fig 4.1: DFD level 0

This DFD shows a crowdfunding platform where Contributors can add and deploy campaigns, and Investors can fund campaigns and view transactions. The Crowdfund entity is likely the central platform for managing campaigns. Both Contributors and Investors can view all campaigns. This is a detailed view (Level 1 DFD) showing internal processes and data flows within the system.

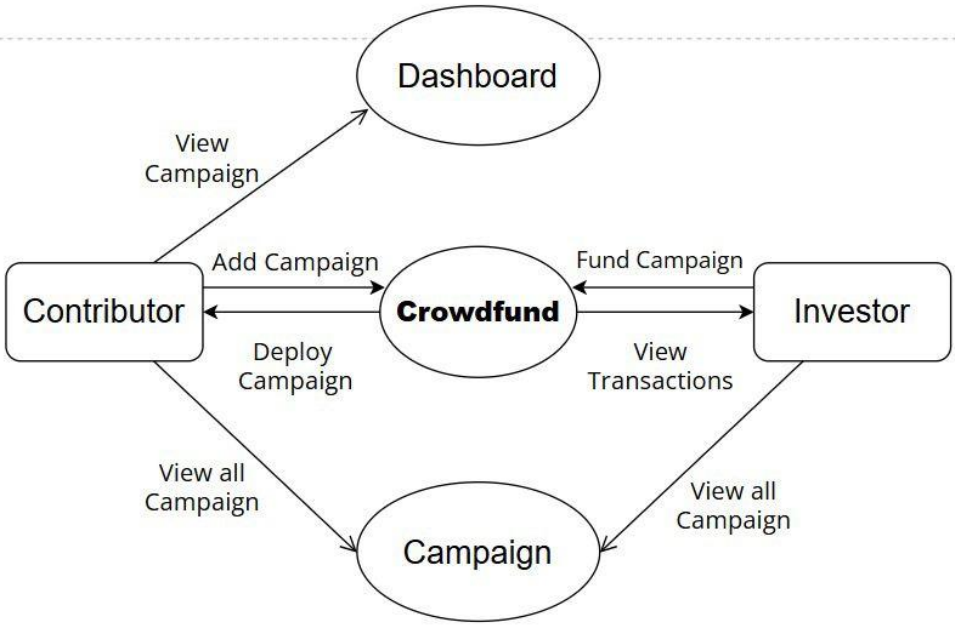


Fig 4.2: DFD level 1

The DFD illustrates a crowdfunding platform. Contributors can browse campaigns to learn about them. Investors can browse campaigns and possibly fund them. Again, this is a high-level view, and a more detailed DFD would illustrate internal processes and data flows.

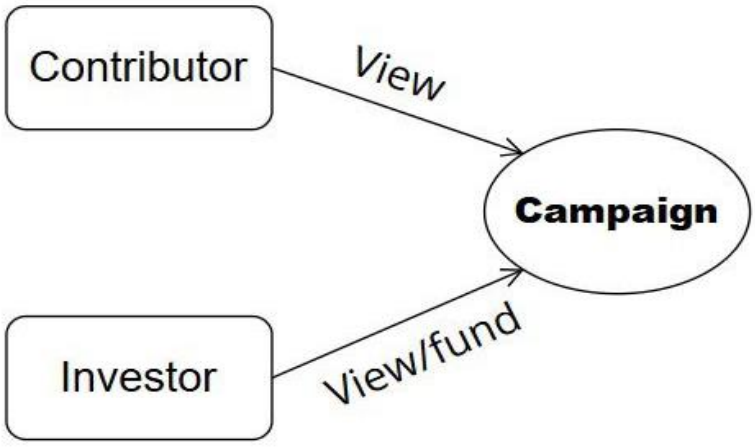


Fig 4.3: DFD level 2 Campaign

This DFD represents a crowdfunding platform where Contributors interact with the Dashboard. They may add campaigns and connect to their Wallet. The Dashboard also interfaces with Smart Contracts, Tiers, and allows for deployment. This is a high-level view, and a more detailed DFD would show internal processes and data flows.

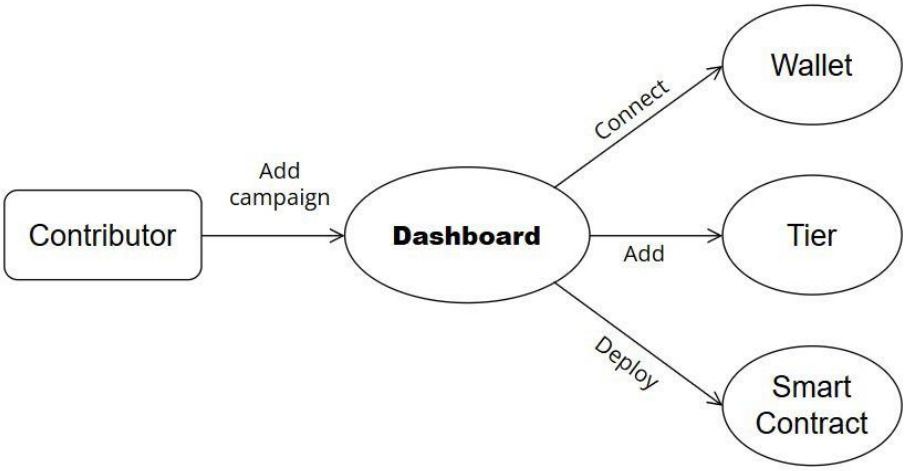


Fig 4.4: DFD level 2 Dashboard

4.2 USE CASE DIAGRAM

Actors:

- User: A user is a type of participant who interacts with the crowdfunding application. They can either create campaigns to raise funds or invest in existing campaigns.
- Blockchain System: The underlying blockchain network where smart contracts are executed, transactions are processed, and campaign details are stored.
- MetaMask Wallet: is a type of cryptocurrency wallet that connects people to the blockchain through safe transactions and monetary handling.

Use Cases:

- Create Campaign: The user can initiate a fundraising campaign by giving input like the goal amount, description, and deadline. This is an action which interacts with the blockchain in order to deploy or register a smart contract.
- Browse Campaign: Users can browse through campaigns hosted on the platform.
- View Campaign Details: Users can see particular details of a campaign, such as its current funding status, creator, and purpose.
- Invest in Campaign: Users can invest in the campaign by transferring funds via their connected wallet, for example, MetaMask. The transaction will be safely recorded on the blockchain.
- Withdraw Funds: Campaign owners can withdraw the raised amounts after the funding goal of their campaign is met.
- Administer Transactions: All funding and withdrawal transactions are channelled through the blockchain system.
- View Fundraising Status: Users can view the level of campaigns' progress, which may include the amount raised based on the target. Investors will be able to track their campaigns with updates on investments.
- Execute Smart Contracts: Smart contracts are executed within the blockchain to ensure automation, transparency, and security of funds.

- Transfer Funds: Blockchain is utilized between the user and campaign in transferring funds, supported by MetaMask.
- Connect Wallet: Users connect their MetaMask wallet with the platform to enable safe transactions and blockchain interactions.

Relations:

The most use cases in the application are interacted with by the User.
The Blockchain System is used in functionalities that require secure and decentralized storage or processing, such as transactions handling and executing smart contracts.
The MetaMask Wallet is mainly used to connect the users to the blockchain and transfer funds.

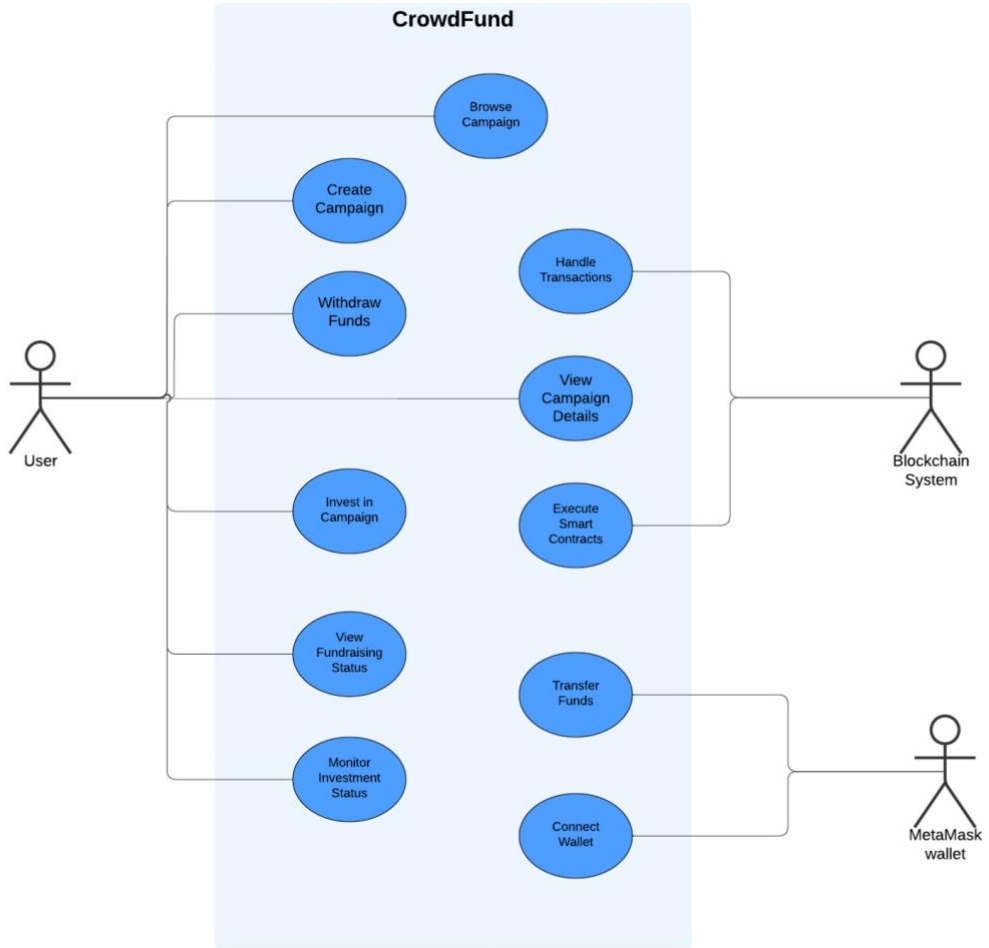


Fig 4.5: Use Case diagram

4.3 CLASS DIAGRAM

The UML class diagram for the crowdfunding platform highlights four main classes: CrowdfundingFactory, Crowdfunding, Tier, and Backer. The CrowdfundingFactory class oversees creating and managing campaigns, storing all

campaigns and user-specific mappings, and providing methods for campaign creation and state management. The Crowdfunding class represents individual campaigns, storing details like name, description, goal, tiers, and backers, while offering methods for funding, tier management, state updates, withdrawals, and refunds. The Tier class defines funding levels, storing the tier name and required amount. The Backer class represents supporters, storing their addresses and contributions. The diagram showcases the relationships between these classes, illustrating the system's core structure and functionality.

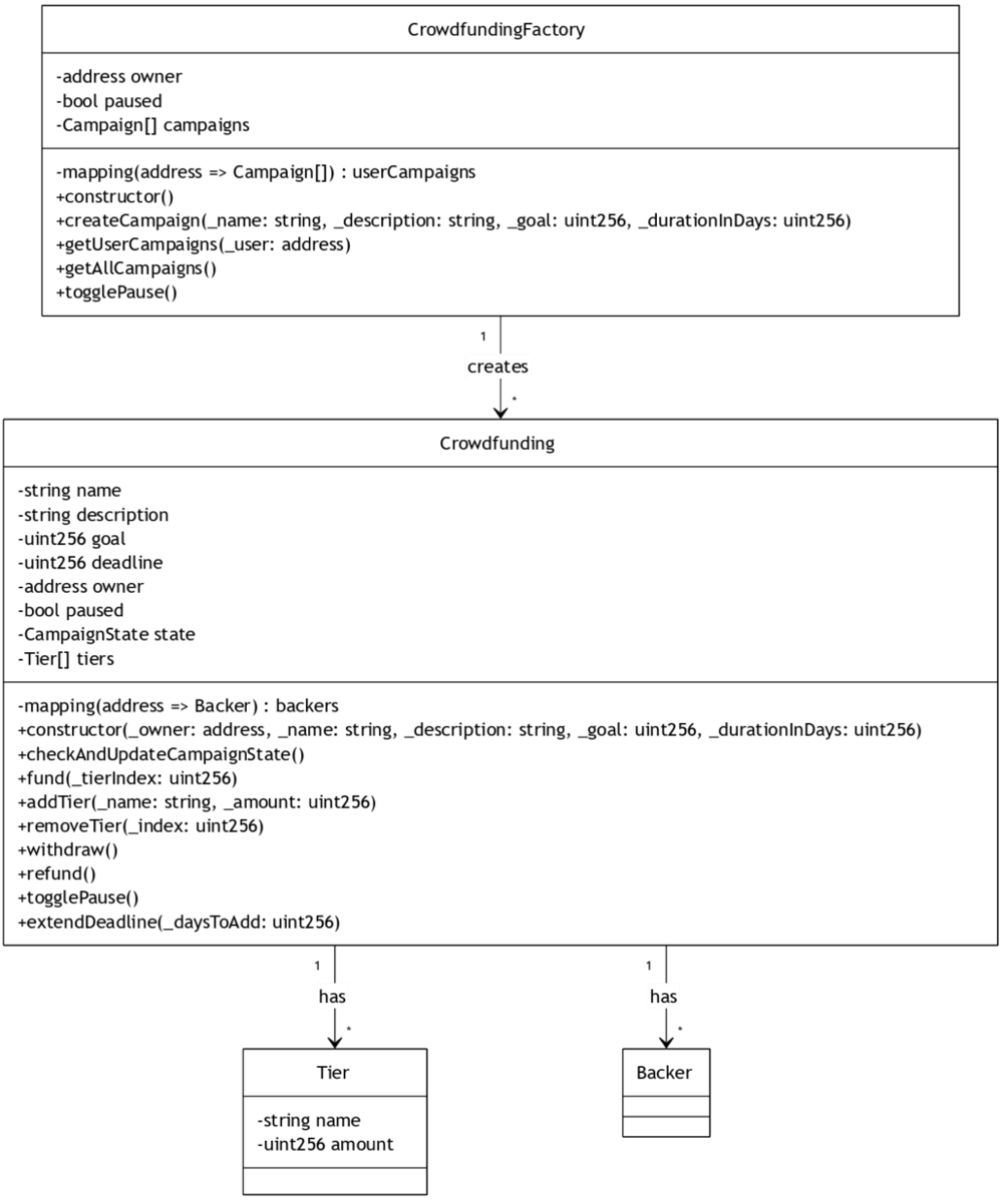


Fig 4.6: Class diagram

CHAPTER 5

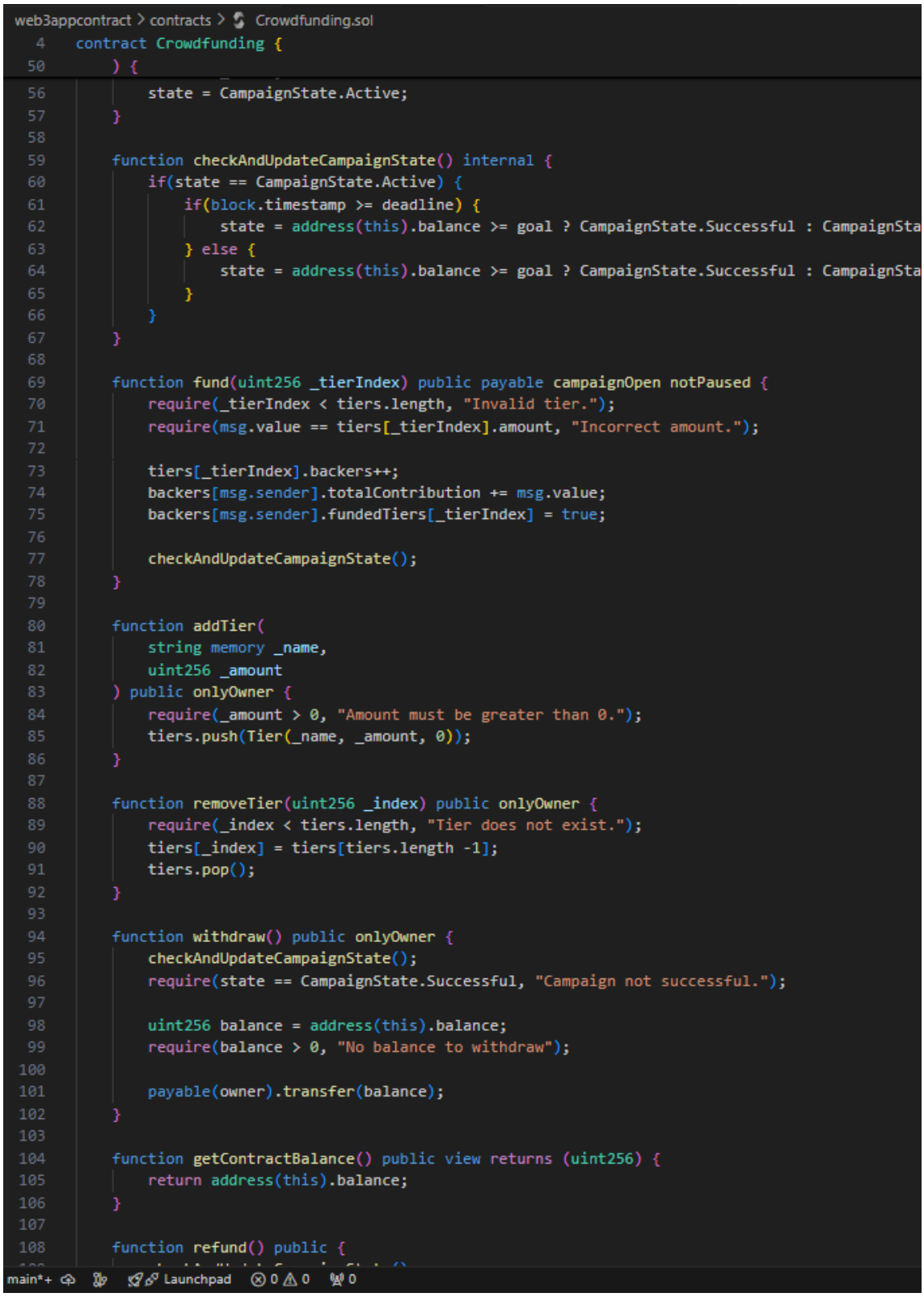
IMPLEMENTATION AND CODING

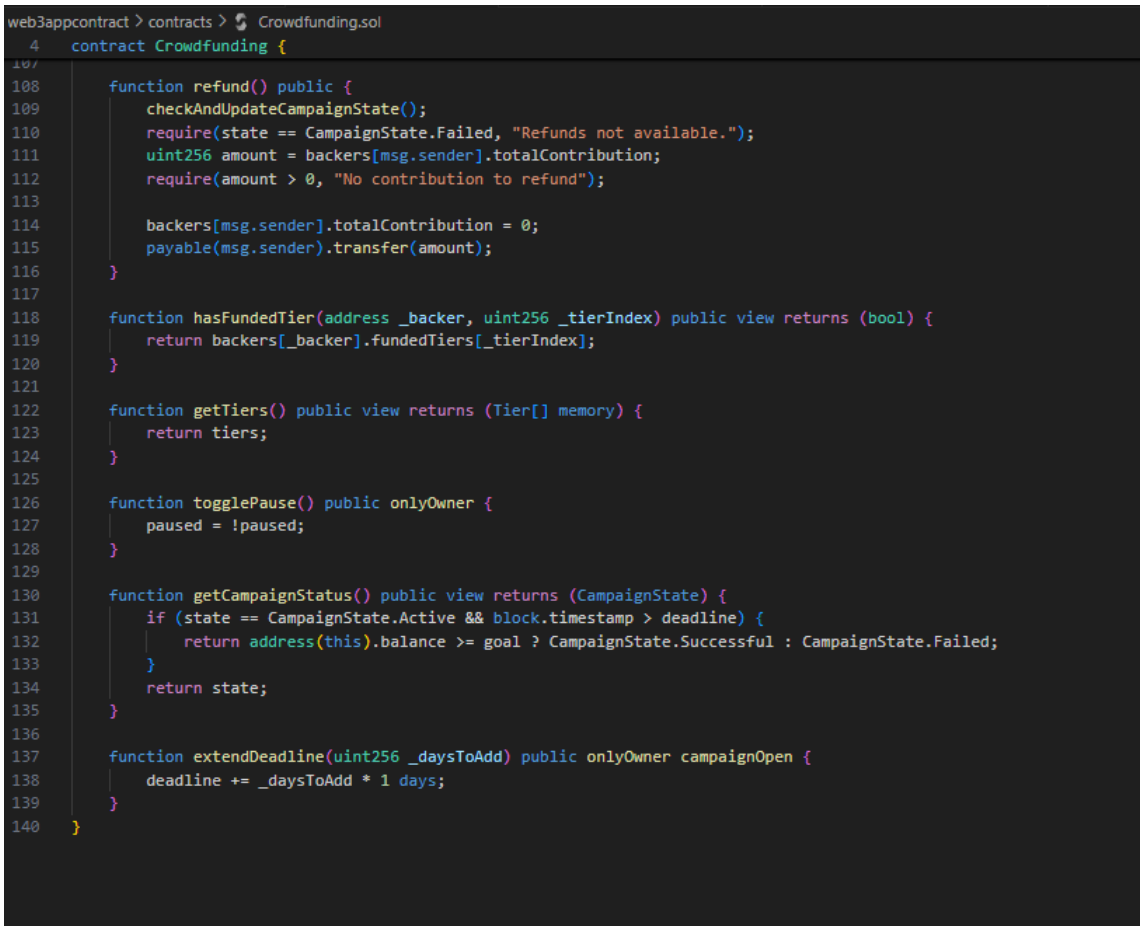
5.1 CODING DETAILS

Crowdfunding

web3appcontract > contracts > Crowdfunding.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    string public name;
    string public description;
    uint256 public goal;
    uint256 public deadline;
    address public owner;
    bool public paused;

    enum CampaignState { Active, Successful, Failed }
    CampaignState public state;

    struct Tier {
        string name;
        uint256 amount;
        uint256 backers;
    }

    struct Backer {
        uint256 totalContribution;
        mapping(uint256 => bool) fundedTiers;
    }

    Tier[] public tiers;
    mapping(address => Backer) public backers;

    modifier onlyOwner() {
        require(msg.sender == owner, "Not the owner");
        _;
    }

    modifier campaignOpen() {
        require(state == CampaignState.Active, "Campaign is not active.");
        _;
    }

    modifier notPaused() {
        require(!paused, "Contract is paused.");
        _;
    }

    constructor(
        address _owner,
        string memory _name,
        string memory _description,
        uint256 _goal,
        uint256 _duratyionInDays
    ) {
        name = _name;
        description = _description;
        goal = _goal;
        deadline = block.timestamp + (_duratyionInDays * 1 days);
        owner = _owner;
```


web3appcontract > contracts > Crowdfunding.sol
contract Crowdfunding {
) {
state = CampaignState.Active;
}
function checkAndUpdateCampaignState() internal {
if(state == CampaignState.Active) {
if(block.timestamp >= deadline) {
state = address(this).balance >= goal ? CampaignState.Successful : CampaignSta
} else {
state = address(this).balance >= goal ? CampaignState.Successful : CampaignSta
}
}
}
function fund(uint256 _tierIndex) public payable campaignOpen notPaused {
require(_tierIndex < tiers.length, "Invalid tier.");
require(msg.value == tiers[_tierIndex].amount, "Incorrect amount.");
tiers[_tierIndex].backers++;
backers[msg.sender].totalContribution += msg.value;
backers[msg.sender].fundedTiers[_tierIndex] = true;
checkAndUpdateCampaignState();
}
function addTier(
string memory _name,
uint256 _amount
) public onlyOwner {
require(_amount > 0, "Amount must be greater than 0.");
tiers.push(Tier(_name, _amount, 0));
}
function removeTier(uint256 _index) public onlyOwner {
require(_index < tiers.length, "Tier does not exist.");
tiers[_index] = tiers[tiers.length -1];
tiers.pop();
}
function withdraw() public onlyOwner {
checkAndUpdateCampaignState();
require(state == CampaignState.Successful, "Campaign not successful.");
uint256 balance = address(this).balance;
require(balance > 0, "No balance to withdraw");
payable(owner).transfer(balance);
}
function getContractBalance() public view returns (uint256) {
return address(this).balance;
}
function refund() public {
main*+ Launchpad 0 0 0


web3appcontract > contracts > Crowdfunding.sol
4 contract Crowdfunding {
108 function refund() public {
109 checkAndUpdateCampaignState();
110 require(state == CampaignState.Failed, "Refunds not available.");
111 uint256 amount = backers[msg.sender].totalContribution;
112 require(amount > 0, "No contribution to refund");
113
114 backers[msg.sender].totalContribution = 0;
115 payable(msg.sender).transfer(amount);
116 }
117
118 function hasFundedTier(address _backer, uint256 _tierIndex) public view returns (bool) {
119 return backers[_backer].fundedTiers[_tierIndex];
120 }
121
122 function getTiers() public view returns (Tier[] memory) {
123 return tiers;
124 }
125
126 function togglePause() public onlyOwner {
127 paused = !paused;
128 }
129
130 function getCampaignStatus() public view returns (CampaignState) {
131 if (state == CampaignState.Active && block.timestamp > deadline) {
132 return address(this).balance >= goal ? CampaignState.Successful : CampaignState.Failed;
133 }
134 return state;
135 }
136
137 function extendDeadline(uint256 _daysToAdd) public onlyOwner campaignOpen {
138 deadline += _daysToAdd * 1 days;
139 }
140 }

Crowdfunding factory

web3appcontract > contracts > CrowdfundingFactory.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import {Crowdfunding} from "./Crowdfunding.sol";

contract CrowdfundingFactory {
    address public owner;
    bool public paused;

    struct Campaign {
        address campaignAddress;
        address owner;
        string name;
        uint256 creationTime;
    }

    Campaign[] public campaigns;
    mapping(address => Campaign[]) public userCampaigns;

    modifier onlyOwner() {
        require(msg.sender == owner, "Not owner.");
        _;
    }

    modifier notPaused() {
        require(!paused, "Factory is paused");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    function createCampaign(
        string memory _name,
        string memory _description,
        uint256 _goal,
        uint256 _durationInDays
    ) external notPaused {
        Crowdfunding newCampaign = new Crowdfunding(
            msg.sender,
            _name,
            _description,
            _goal,
            _durationInDays
        );
        address campaignAddress = address(newCampaign);

        Campaign memory campaign = Campaign({
            campaignAddress: campaignAddress,
            owner: msg.sender,
            name: _name,
            creationTime: block.timestamp
        });

```

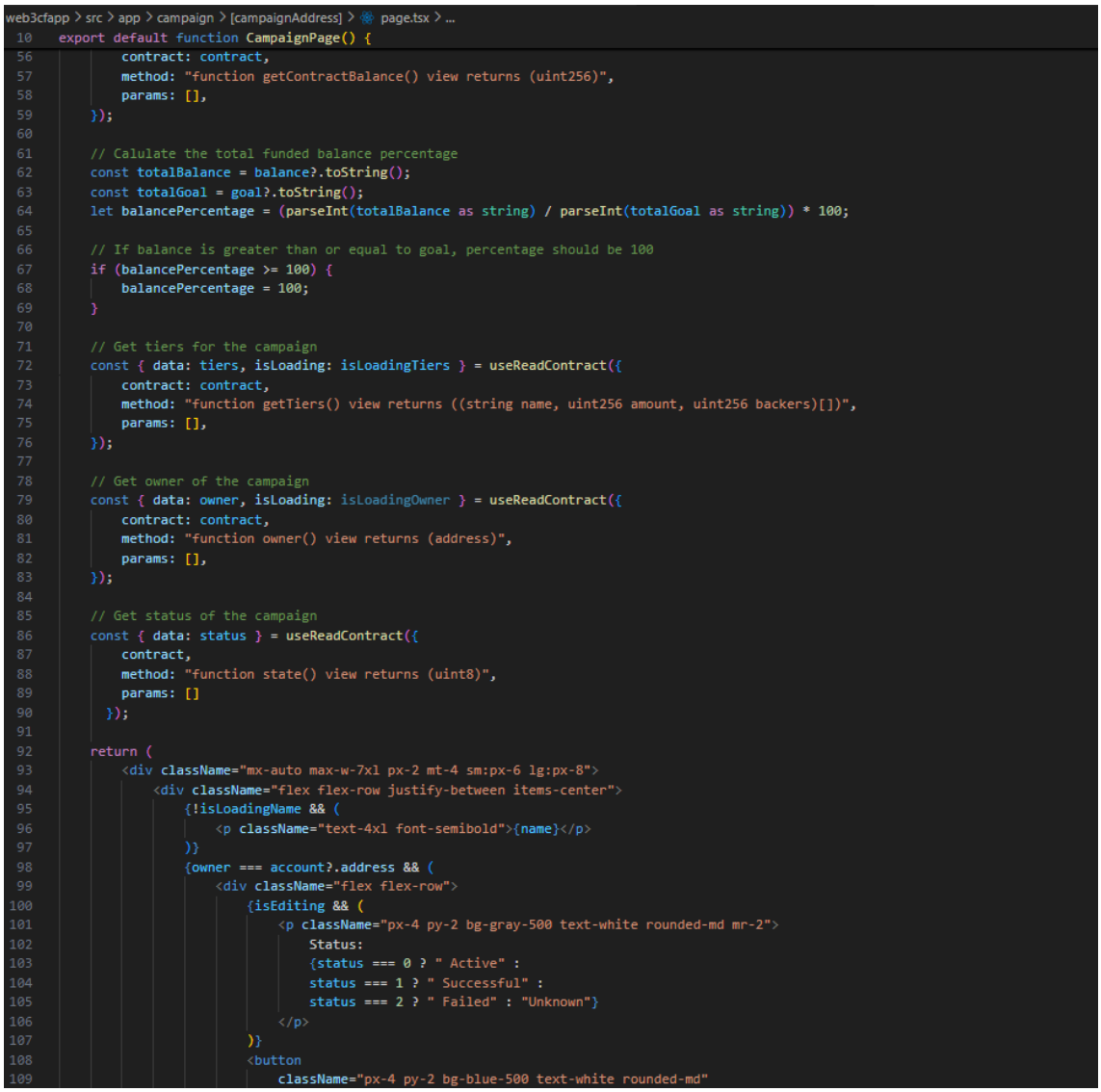
web3appcontract > contracts > CrowdfundingFactory.sol

```
6   contract CrowdfundingFactory {
39      ) external notPaused {
54          });
55
56          campaigns.push(campaign);
57          userCampaigns[msg.sender].push(campaign);
58      }
59
60      function getUserCampaigns(address _user) external view returns (Campaign[] memory) {
61          return userCampaigns[_user];
62      }
63
64      function getAllCampaigns() external view returns (Campaign[] memory) {
65          return campaigns;
66      }
67
68      function togglePause() external onlyOwner {
69          paused = !paused;
70      }
71  }
```

Page.tsx

web3cfapp > src > app > campaign > [campaignAddress] > page.tsx > ...

```
1   'use client';
2   import { client } from "@/app/client";
3   import { TierCard } from "../../components/TierCard";
4   import { useParams } from "next/navigation";
5   import { useState } from "react";
6   import { getContract, prepareContractCall, ThirdwebContract } from "thirdweb";
7   import { lineaSepolia } from "thirdweb/chains";
8   import { lightTheme, TransactionButton, useActiveAccount, useReadContract } from "thirdweb/react";
9
10  export default function CampaignPage() {
11      const account = useActiveAccount();
12      const { campaignAddress } = useParams();
13      const [isEditing, setIsEditing] = useState<boolean>(false);
14      const [isModalOpen, setIsModalOpen] = useState<boolean>(false);
15
16      const contract = getContract({
17          client: client,
18          chain: lineaSepolia,
19          address: campaignAddress as string,
20      });
21
22      // Name of the campaign
23      const { data: name, isLoading: isLoadingName } = useReadContract({
24          contract: contract,
25          method: "function name() view returns (string)",
26          params: [],
27      });
28
29      // Description of the campaign
30      const { data: description } = useReadContract({
31          contract,
32          method: "function description() view returns (string)",
33          params: []
34        });
35
36      // Campaign deadline
37      const { data: deadline, isLoading: isLoadingDeadline } = useReadContract({
38          contract: contract,
39          method: "function deadline() view returns (uint256)",
40          params: [],
41      });
42      // Convert deadline to a date
43      const deadlineDate = new Date(parseInt(deadline?.toString() as string) * 1000);
44      // Check if deadline has passed
45      const hasDeadlinePassed = deadlineDate < new Date();
46
47      // Goal amount of the campaign
48      const { data: goal, isLoading: isLoadingGoal } = useReadContract({
49          contract: contract,
50          method: "function goal() view returns (uint256)",
51          params: [],
52      });
53
54      // Total funded balance of the campaign
55      const { data: balance, isLoading: isLoadingBalance } = useReadContract({
```

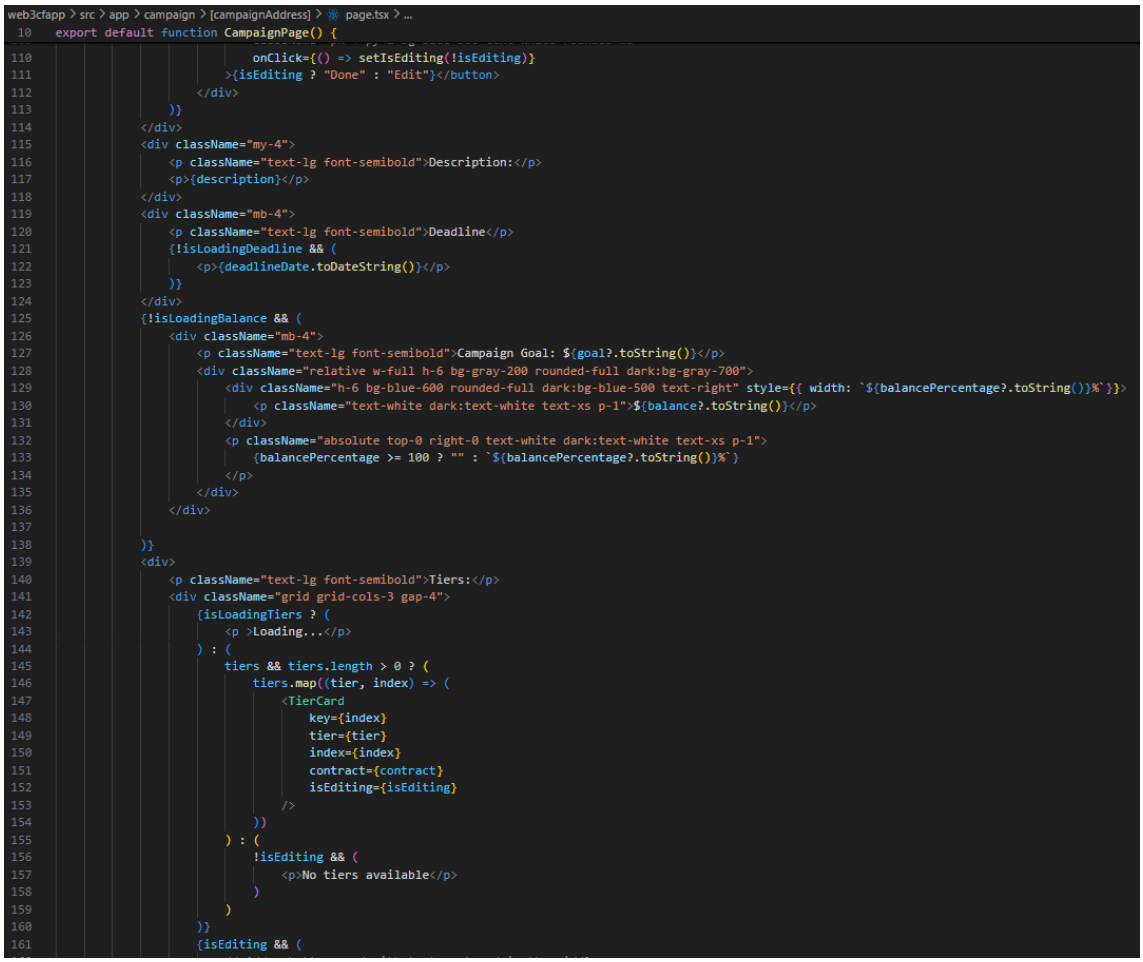

web3cfapp > src > app > campaign > [campaignAddress] > page.tsx > ...
export default function CampaignPage() {
contract: contract,
method: "function getContractBalance() view returns (uint256)",
params: [],
});
// Calulate the total funded balance percentage
const totalBalance = balance?.toString();
const totalGoal = goal?.toString();
let balancePercentage = (parseInt(totalBalance as string) / parseInt(totalGoal as string)) * 100;
// If balance is greater than or equal to goal, percentage should be 100
if (balancePercentage >= 100) {
balancePercentage = 100;
}
// Get tiers for the campaign
const { data: tiers, isLoading: isLoadingTiers } = useReadContract({
contract: contract,
method: "function getTiers() view returns ((string name, uint256 amount, uint256 backers)[])",
params: [],
});
// Get owner of the campaign
const { data: owner, isLoading: isLoadingOwner } = useReadContract({
contract: contract,
method: "function owner() view returns (address)",
params: [],
});
// Get status of the campaign
const { data: status } = useReadContract({
contract,
method: "function state() view returns (uint8)",
params: []
});
return (

{!isLoadingName && (
{name}
)}
{owner === account?.address && (

{isEditing && (

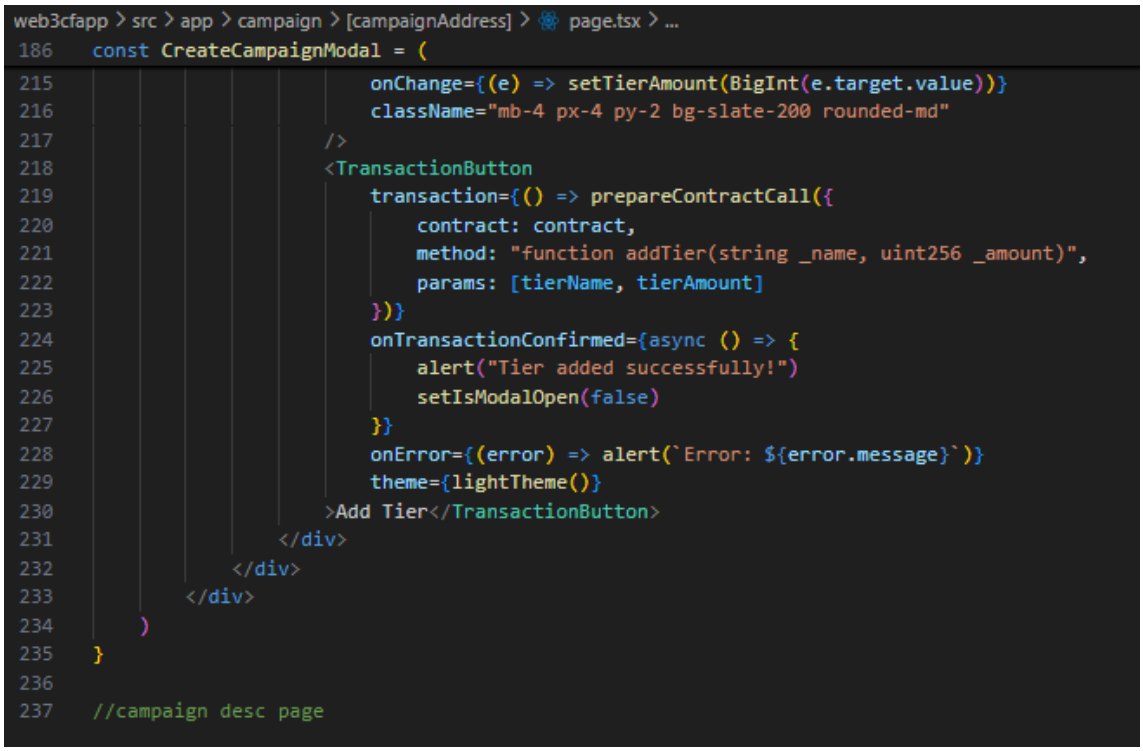
Status:
{status === 0 ? " Active" :
status === 1 ? " Successful" :
status === 2 ? " Failed" : "Unknown"}

)}
<button
className="px-4 py-2 bg-blue-500 text-white rounded-md"



```
web3cfapp > src > app > campaign > [campaignAddress] > page.tsx > ...
10  export default function CampaignPage() {
110                                 onClick={() => setIsEditing(!isEditing)}
111                             >{isEditing ? "Done" : "Edit"}</button>
112                         </div>
113                     )}
114                 </div>
115                 <div className="my-4">
116                     <p className="text-lg font-semibold">Description:</p>
117                     <p>{description}</p>
118                 </div>
119                 <div className="mb-4">
120                     <p className="text-lg font-semibold">Deadline</p>
121                     {!isLoadingDeadline && (
122                         <p>{deadlineDate.toDateString()}</p>
123                     )}
124                 </div>
125                 {!isLoadingBalance && (
126                     <div className="mb-4">
127                         <p className="text-lg font-semibold">Campaign Goal: ${goal?.toString()}</p>
128                         <div className="relative w-full h-6 bg-gray-200 rounded-full dark:bg-gray-700">
129                             <div className="h-6 bg-blue-600 rounded-full dark:bg-blue-500 text-right" style={{ width: `${balancePercentage?.toString()}%`}}>
130                                 <p className="text-white dark:text-white text-xs p-1">${balance?.toString()}</p>
131                             </div>
132                             <p className="absolute top-0 right-0 text-white dark:text-white text-xs p-1">
133                                 {balancePercentage >= 100 ? "" : `${balancePercentage?.toString()}%`}
134                             </p>
135                         </div>
136                     </div>
137 
138                 )}
139                 <div>
140                     <p className="text-lg font-semibold">Tiers:</p>
141                     <div className="grid grid-cols-3 gap-4">
142                         {isLoadingTiers ? (
143                             <p >Loading...</p>
144                         ) : (
145                             tiers && tiers.length > 0 ? (
146                                 tiers.map((tier, index) => (
147                                     <TierCard
148                                         key={index}
149                                         tier={tier}
150                                         index={index}
151                                         contract={contract}
152                                         isEditing={isEditing}
153                                     />
154                                 ))
155                             ) : (
156                                 !isEditing && (
157                                     <p>No tiers available</p>
158                                 )
159                             )
160                         )}
161                         {isEditing && (
```

```
web3cfapp > src > app > campaign > [campaignAddress] > page.tsx > ...
10  export default function CampaignPage() {
162                         // Add a button card with text centered in the middle
163                         <button
164                             className="max-w-sm flex flex-col text-center justify-center items-center font-semibold p-6 bg-blue-500 text-white border border-slate-100 rounded-lg shadow"
165                             onClick={() => setIsModalOpen(true)}
166                         >+ Add Tier</button>
167                     )}
168                 </div>
169             </div>
170 
171             {isModalOpen && (
172                 <CreateCampaignModal
173                     setIsModalOpen={setIsModalOpen}
174                     contract={contract}
175                 />
176             )}
177         </div>
178     );
179 }
180 
181 type CreateTierModalProps = {
182     setIsModalOpen: (value: boolean) => void
183     contract: ThirdwebContract
184 }
185 
186 const CreateCampaignModal = (
187     { setIsModalOpen, contract }: CreateTierModalProps
188 ) => {
189     const [tierName, setTierName] = useState<string>("");
190     const [tierAmount, setTierAmount] = useState<bigint>(1n);
191 
192     return (
193         <div className="fixed inset-0 bg-black bg-opacity-75 flex justify-center items-center backdrop-blur-md">
194             <div className="w-1/2 bg-slate-100 p-6 rounded-md">
195                 <div className="flex justify-between items-center mb-4">
196                     <p className="text-lg font-semibold">Create a Funding Tier</p>
197                     <button
198                         className="text-sm px-4 py-2 bg-slate-600 text-white rounded-md"
199                         onClick={() => setIsModalOpen(false)}
200                     >Close</button>
201                 </div>
202                 <div className="flex flex-col">
203                     <label>Tier Name:</label>
204                     <input
205                         type="text"
206                         value={tierName}
207                         onChange={(e) => setTierName(e.target.value)}
208                         placeholder="Tier Name"
209                         className="mb-4 px-4 py-2 bg-slate-200 rounded-md"
210                     />
211                     <label>Tier Cost:</label>
212                     <input
213                         type="number"
214                         value={parseInt(tierAmount.toString())}
215                         onChange={(e) => setTierAmount(BigInt(e.target.value))}
```


web3cfapp > src > app > campaign > [campaignAddress] > page.tsx > ...
186 const CreateCampaignModal = (
215 onChange={(e) => setTierAmount(BigInt(e.target.value))}
216 className="mb-4 px-4 py-2 bg-slate-200 rounded-md"
217 />
218 <TransactionButton
219 transaction={() => prepareContractCall({
220 contract: contract,
221 method: "function addTier(string _name, uint256 _amount)",
222 params: [tierName, tierAmount]
223 })}
224 onTransactionConfirmed={async () => {
225 alert("Tier added successfully!")
226 setIsModalOpen(false)
227 }}
228 onError={(error) => alert(`Error: ${error.message}`)}
229 theme={lightTheme()}
230 >Add Tier</TransactionButton>
231
232
233
234)
235 }
236
237 //campaign desc page

CHAPTER 6

SOFTWARE TESTING

6.1 TESTING STRATEGY

A strong testing approach is essential to guaranteeing a blockchain-based crowdfunding platform's dependability, security, and functionality

6.1.1 Smart contract testing

- Tools: Truffle Suite: Utilize Truffle for smart contract development, testing, and deployment.
- Hardhat: It is another popular tool for smart contract development and testing with built-in testing environments.
- Testing Approach:
  Unit Testing: Write unit tests using frameworks such as Mocha or Chai to test smart contract functionalities and logic.
  Scenario Testing: It is used to conduct scenario-based testing to validate various contract states and edge cases.

6.1.2 Security audits

- Tools:
  MythX: Use MythX for smart contract security analysis and vulnerability detection.
  Solvent: Use Solvent to detect possible security vulnerabilities and do solidity linting.
- Testing Approach:
  Automated Security Audits: To find security flaws and hazards in smart contracts, run automated security analysis tools.

6.1.3 Documentation and reporting

- Approach: For traceability and future reference, keep thorough test plans, test cases, and test result documentation.
- Provide thorough test reports that detail the coverage of the test, any problems that are discovered, and their fixes.

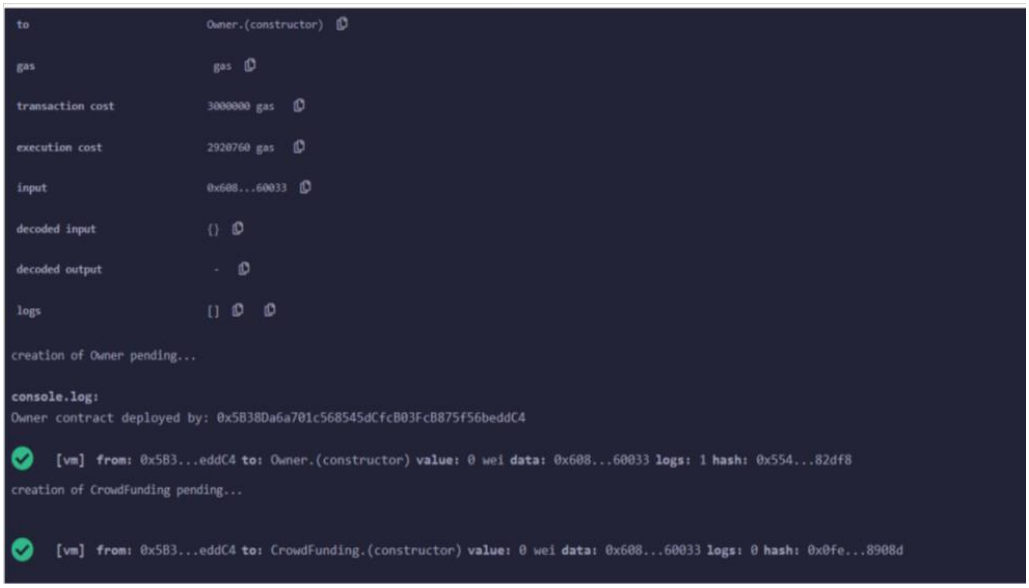


Fig 6.1: Remix IDE Prompt

6.2 TEST CASES AND OUTCOMES

6.2.1 Test case:

Validate campaign creation constraints

Scenario: Try to start a campaign with a past-due deadline.

Expected Outcome: A reply with the subject "The deadline should be a date in the future" is anticipated.

6.2.2 Test case: donate to campaign

Scenario: Send Ether to a campaign as a donation.

Expected Outcome: The successful donation transaction results in the transfer of Ether to the campaign owner and an increase in the total amount raised for the campaign.

6.2.3 Test case: retrieve campaign details

Scenario: Use getCampaigns() or getDonators() to obtain specific campaign details.

Expected Outcome: Information about the campaign, including the title, description, goal amount, deadline, amount raised, donors, and donations, was successfully retrieved.

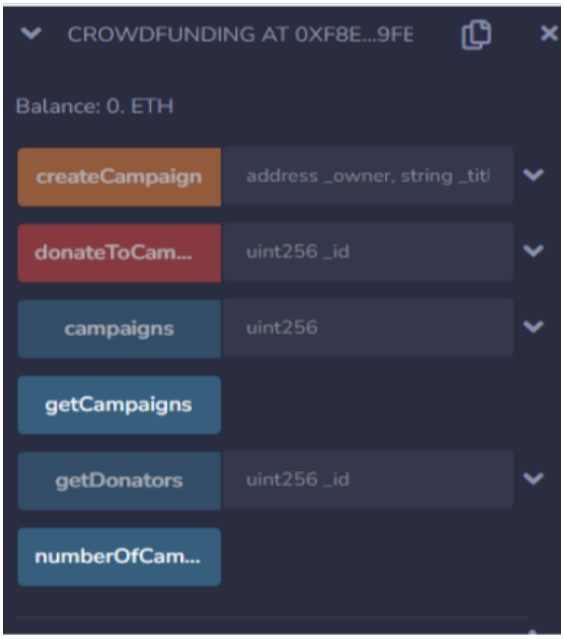


Fig 6.2: Remix IDE Deploy and run portal

6.2.4 Test case:

Multiple donations to the same campaign
Scenario: Different users contribute to the same campaign in different quantities.
Expected Outcome: Precise documentation of every donation and its corresponding donor address in the arrays for donors and donations for the campaign.

6.2.5 Test case: donating zero ether

Scenario: Make an effort to contribute 0 Ether to a campaign.
Expected Outcome: A revert informing that the donation amount needs to be more than zero is the anticipated outcome.

6.2.6 Test case: campaign count

Scenario: After generating numerous campaigns, get the total number of campaigns.
Expected Outcome: The number of campaigns generated and the overall number of campaigns match.

6.2.7 Test case:

Campaign donations
Scenario: Verify the amount donated in relation to the overall money raised for the campaign.
Expected Outcome: The total amount raised for the campaign is equal to the sum of the contributions made by each individual donor.

6.2.8 Test case: retrieve campaigns when none exist

Scenario: If no campaigns have been established, retrieve campaigns.

Expected Outcome: There are no campaigns available, as indicated by the function's empty array result.

6.2.9 Test case: out-of-bounds campaign access

Scenario: Get access to an invalid campaign ID.
Expected Outcome: A revert or exception is anticipated as a result of trying to access a campaign that doesn't exist.

Table 6.1: Smart Contract Testing

Test Case	Expected Outcome	Result
Token Transfer Functionality	Successful transfer	Passed
Gas Fee Estimation Accuracy	Accurate calculation	Passed
Insufficient Balance Handling	Revert transaction	Passed
Invalid Address Error Handling	Revert transaction	Passed
Security Checks (Re-entrancy)	Block unauthorized access	Passed

Table 6.2: Frontend Testing

Test Case	Scenario	Result
Wallet Connection Flow	Connect with METAMASK	Passed
Token Balance Display	Correct display of token balances	Passed
Transaction Confirmation Model	Proper UI rendering	Passed
Cross – Browser Compatibility	Test on Chrom,FireFox, Edge	Passed
Mobile Responsiveness	Tested on iOS and Android	Passed

Table 6.3: Integration Testing

Test Case	Description	Result
Smart Contract Function Calls	Seamless integration	Passed
Event Listening (real-time)	Proper event triggers	Passed
Error Handling in UI	User-friendly error messages	Passed

CHAPTER 7

RESULT AND DISCUSSION

7.1 PROJECT OUTCOME

7.1.1 Enhanced transparency and trust

- By leveraging blockchain, your crowdfunding platform ensures transparency and immutability.
- Backers can verify transactions, track fund usage, and gain confidence in the process.
- The outcome is increased trust between creators and backers.

7.1.2 Direct peer-to-peer transactions

- Web3 allows direct interaction between backers and campaign creators.
- Backers can contribute directly to projects without intermediaries.
- The outcome is financial empowerment and a more efficient funding process.

7.1.3 Global participation and inclusion

- Blockchain transcends borders, enabling global participation.
- Backers from different countries can support projects they believe in.
- The outcome is a diverse and inclusive crowdfunding ecosystem.

7.1.4 Smart contract automation

- Smart contracts automate campaign processes (e.g., fund release, goal tracking).
- Creators can focus on their projects while the platform handles logistics.
- The outcome is streamlined operations and reduced administrative overhead.

7.1.5 Educational impact

- The project can educate users about blockchain, cryptocurrencies, and decentralized finance (DeFi).
- Learning becomes an integral part of the crowdfunding experience.

7.1.6 Community building

- Successful campaigns foster communities around shared interests.
- Backers become advocates, spreading the word about projects.

- The outcome is a supportive network that extends beyond funding.

7.1.7 Scalability and future expansion

- The objective is to build a robust platform that can handle a growing number of campaigns.
- Future enhancements might include NFT-based rewards or decentralized governance.

7.2 SNAPSHOT OF WEBSITE

7.2.1 Home page before wallet connected

This is the home page of the project. The image below displays the front-end interface before a wallet is connected. At this stage, all campaigns are visible, but contributions to them are restricted until the wallet connection is established.

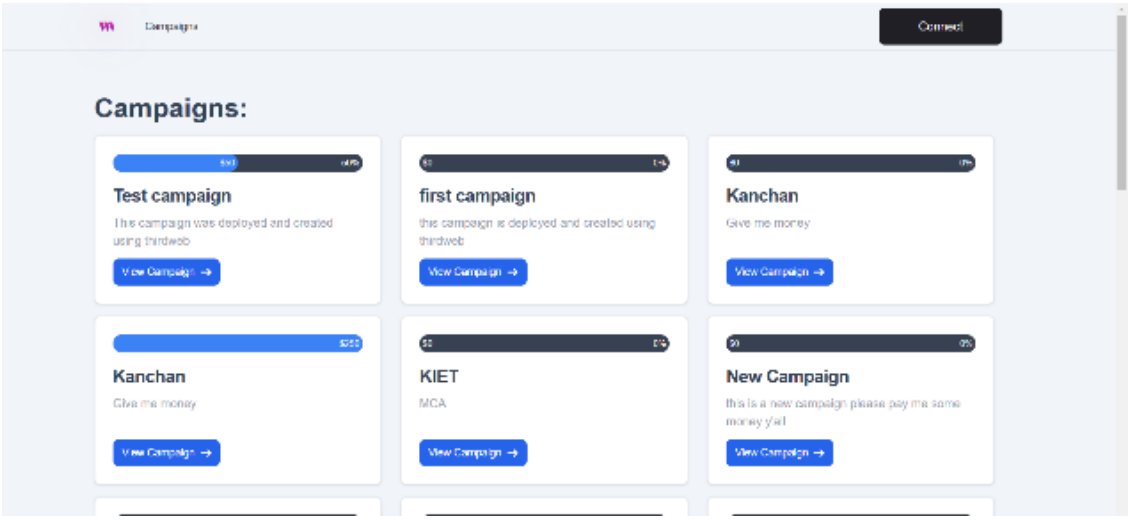


Fig 7.1: Home page before wallet connected

7.2.2 Home page after wallet connected

This is the home page of the project. The image below shows the front-end interface after connecting to the wallet. At this stage, all listed campaigns are visible, and you can now contribute to them.

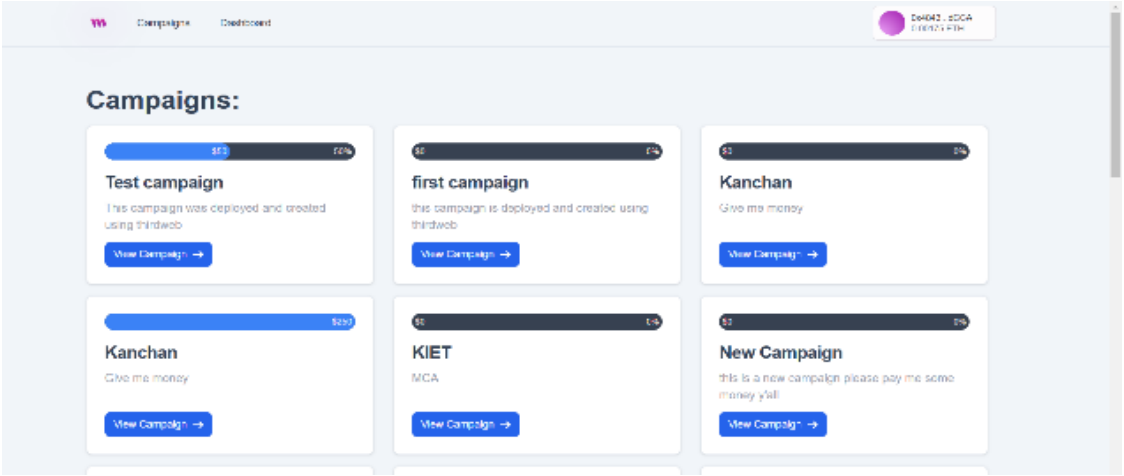


Fig 7.2: Home page after wallet connected

7.2.3 Dashboard page

The below image shows the dashboard page used by the campaign creator. Here, the creator can view the number of campaigns they have created, make edits to existing campaigns, and also create new ones.

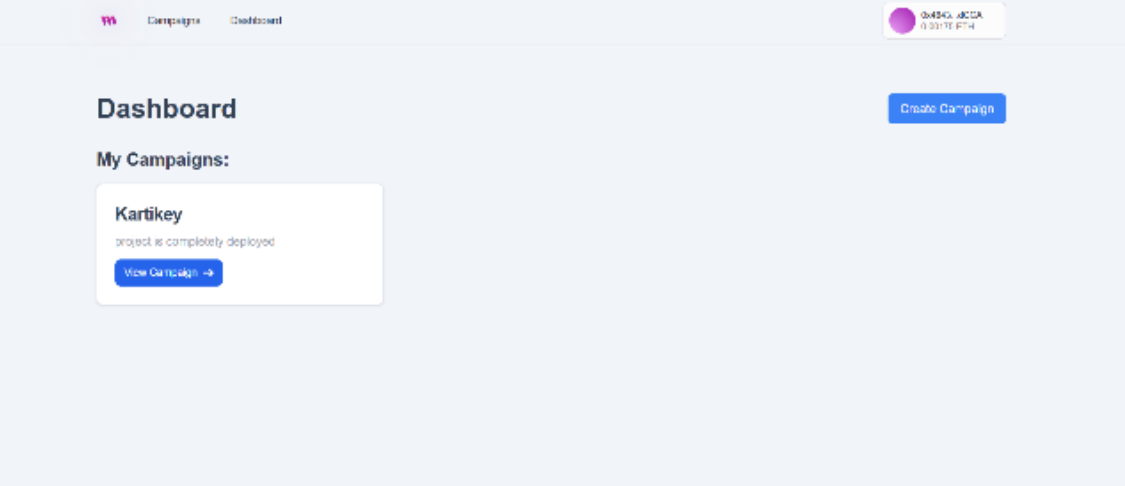


Fig 7.3: Dashboard page

7.2.4 Create campaign form

The above image shows the 'Create Campaign' form, accessible from the dashboard after connecting to your wallet. In this form, you can enter the campaign name, description, goal amount, and the number of days you want the campaign to remain active.

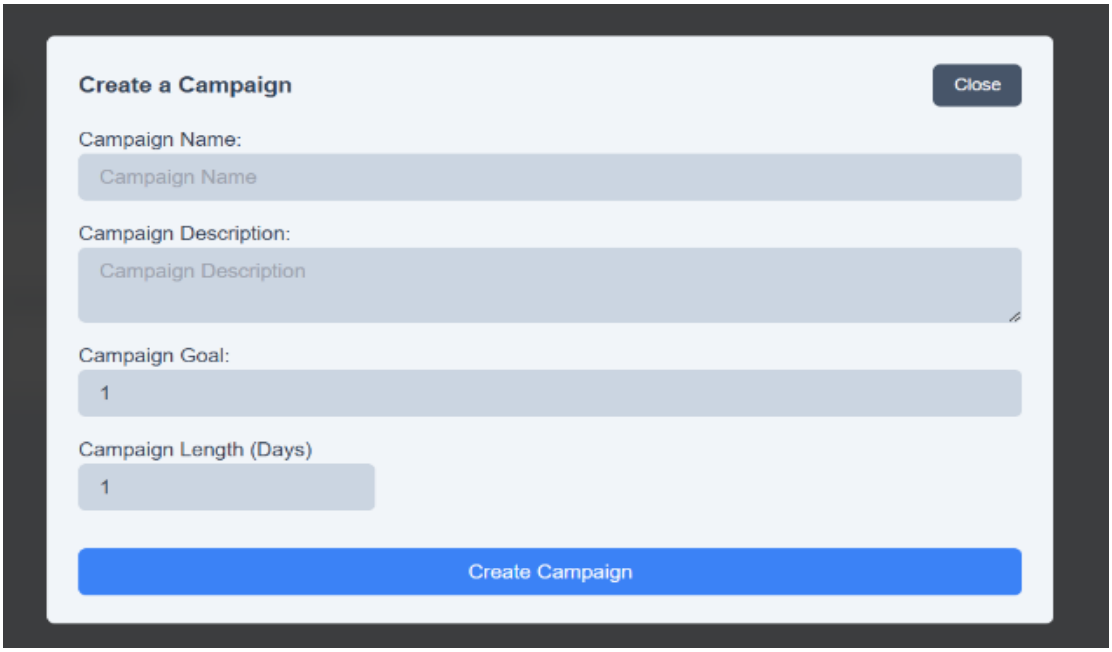


Fig 7.4: Create campaign form

7.2.5 Adding tier in campaign

Above image shows that you can add tier in your campaign. Tier is basically the predefined amount you have declared to your contract so that investor can invest only the

amount the creator have listed on their contract. Investor can select amount from one of them.

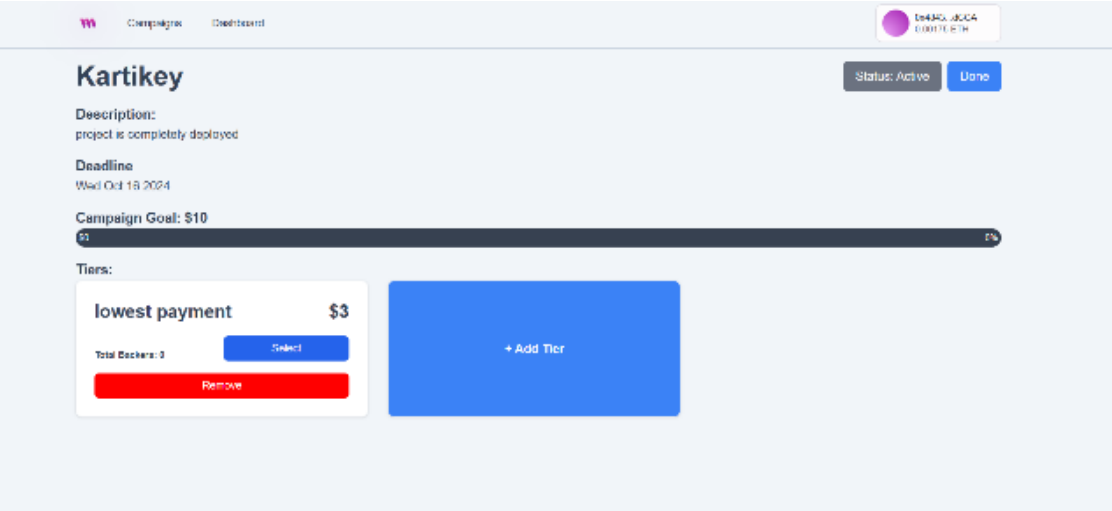


Fig 7.5: Adding tier in campaign

- The images illustrate the user journey on our platform's front end.
- The interface shows the view before and after connecting a wallet, allowing a smooth transition once connected.
- Users can see all available campaigns, and upon connecting a wallet, they gain the option to invest or create their own campaign via the dashboard.
- The platform enables users to add tiers within their campaigns—preset investment amounts that simplify contributions for supporters.
- A clear campaign creation form is also displayed, making it easy for users to set up and customize their projects.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

This decentralized crowdfunding platform, by using the power of Web3 and blockchain technology, is literally changing the very nature of funding mechanisms for secure, transparent, and efficient service for its creators and investors alike. The elimination of middlemen, thus expensing the costs, through smart contracts and cryptocurrency transactions increases the momentum of building trust through immutable records in the real-time transactions. This interface makes the system super user-friendly and thus highly accessible, making users easy to fund or launch with confidence in a decentralized and fully trustless environment. This generally helps open an inclusive ecosystem that supports the innovation and global financial growth it enables both entrepreneurs and investors to participate in transformative opportunities.

Overall, this project demonstrates the potential of blockchain and Web3 technology to transform the crowd by solving the limitations of traditional regulations. The platform provides transparency, security, and cost-effectiveness for creators and investors by leveraging smart contracts and decentralized governance. Developers can reach a global audience through a flexible process, while investors benefit from security, immutability, and automatic rollback if the campaign fails. The implementation and integration of features enhances the user experience, making the platform both practical and efficient. In addition, the ability to generate activity reports provides better insights into growth and financial performance. Compliance, advanced analytics, and advanced integration tools. It not only bridges the gap between founders and investors, but also fosters innovation and trust in the continuous development of blockchain-based financial solutions.

REFERENCES

[1] Belleflamme, P., Lambert, T., & Schwienbacher, A. (2014). Crowdfunding: Tapping the right crowd. Journal of Business Venturing, 29(5), 585–609. https://doi.org/10.1016/j.jbusvent.2013.07.003.

[2] Böckel, A., Hörisch, J., & Tenner, I. (2021). A systematic literature review of crowdfunding and sustainability: Highlighting what really matters. Management Review Quarterly, 71(2), 433–453. https://doi.org/10.1007/s11301-020-00189-3.

[3] Cumming, D. J., Leboeuf, G., & Schwienbacher, A. (2017), Crowdfunding Cleantech. Energy Economics, (accepted manuscript). https://ssrn.com/abstract=2985703.

[4] Gaddy, B. E., Sivaram, V., Jones, T. B., & Wayman, L. (2016). Venture Capital and Cleantech: The Wrong Model for Energy Innovation. SSRN Electronic Journal. Published. https://doi.org/10.2139/ssrn.2788919.

[5] Manganiello, M., & Dragulanescu, I.-V. (2021). Sustainable Equity Crowdfunding Projects: Are They A Driving Force to Revitalise Italy After Global Socio-Economic Consequences of The COVID-19? SHS Web of Conferences, 92(01030). https://www.shs-conferences.org/articles/shsconf/pdf/2021/03/shsconf_glob20_01030.pdf.

[6] Md Nazmus Saadat, Syed Abdul Halim, Husna Osman, Rasheed Mohammad Nassr, Megat F. Zuhairi. Blockchain based crowdfunding systems. Indonesian Journal of Electrical Engineering and Computer Science, Vol. 15, No. 1, July 2019, pp. 409~413.

[7] Tobias Guggenberger, Benjamin Schellinger, Victor von Wachter, Nils Urbach. Kickstarting blockchain: designing blockchain-based tokens for equity crowdfunding. Electronic Commerce Research (2024) 24:239–273. https://doi.org/10.1007/s10660-022-09634-9

[8] Estrin, S., Gozman, D., & Khavul, S. (2018). The evolution and adoption of equity crowdfunding: Entrepreneur and investor entry into a new market. Small Business Economics, 51(2), 425–439.

[9] Carree, M. A., & Thurik, A. R. (2010). The impact of entrepreneurship on economic growth. In Z. J. Acs & D. B. Audretsch (Eds.), Handbook of entrepreneurship research (pp. 557–94). Springer New Yor k

[10] Chen, Y. (2018). Blockchain tokens and the potential democratization of entrepreneurship and inno-vation. Business Horizons, 61(4), 567–75. https:// doi. org/ 10. 1016/j. bushor. 2018. 03. 006.

[11] Cambridge Center for Alternative Finance. (2020). The Global Alternative Finance Market Bench-marking Report: Trends, Opportunities and Challenges for Lending, Equity, and Non-Investment Alternative Finance Models.

[12] Kranz, J., Nagel, E., & Yoo, Y. (2019). Blockchain token sale. Business & Information Systems Engineering, 61(6), 745–753.

[13] Hartmann, F., Grottolo, G., Wang, X., & Lunesu, MI. (2019). Alternative fundraising: success fac-tors for blockchain-based vs. conventional crowdfunding. In Tonelli R (Ed.), alternative fundrais-ing: success factors for blockchain-based vs. conventional crowdfunding (pp. 38–43) Piscataway, NJ. IEEE

[14] Treiblmaier, H., Swan, M., de Filippi, P., Lacity, M., Hardjono, T., & Kim, H. (2021). What's next in blockchain research? SIGMIS Database, 52(1), 27–52. https:// doi. org/ 10. 1145/ 34479 34. 34479 38.

[15] Yang, R., Wakefield, R., Lyu, S., et□al. (2020). Public and private blockchain in construction busi-ness process and information integration. Automation in Construction, 118, 103276. https:// doi. org/ 10. 1016/j. autcon. 2020. 103276