

DECLARATION

I hereby declare that the report titled "FundHub" is a result of my own work, and all the content included has been developed with utmost sincerity and dedication. The data and insights provided in the report are accurate to the best of my knowledge and have been collected from reliable sources. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable. This report is submitted as part of my academic or professional endeavour and is intended solely for informational purposes.

Name: Harshit Mishra
Roll No: 2300290140072
Branch: MCA

Name: Himanshu Bhardwaj
Roll No: 2300290140076
Branch: MCA

Name: Fauwaz Ayub
Roll No: 2300290140063
Branch: MCA

CERTIFICATE

Certified that **Harshit Mishra 2300290140072, Himanshu Bhardwaj 2300290140076, Fauwaz Ayub 2300290140063** have carried out the project work having “**FundHub**” for **Master of Computer Applications** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the students themselves and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Harshit Mishra 2300290140072
Himanshu Bhardwaj 2300290140076
Fauwaz Ayub 2300290140063

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Prashant Agrawal
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi
Dean & Professor
Department of Computer Application
KIET Group of Institutions, Ghaziabad

FundHub

Harshit Mishra

Himanshu Bhardwaj

Fauwaz Ayub

ABSTRACT

FundHub is an innovative crowdfunding platform that leverages blockchain technology to redefine how creators and funders connect. By integrating decentralized payment systems, FundHub ensures transparency, security, and efficiency in financial transactions. The platform empowers project creators by providing a reliable avenue to showcase their ideas and receive funding while offering contributors full visibility and trust in how their contributions are utilized.

With a focus on inclusivity and technological advancement, FundHub eliminates traditional intermediaries, reduces costs, and fosters global collaboration. This project aims to transform the crowdfunding landscape by creating a seamless and trustworthy ecosystem that enables impactful projects to come to life.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Mr. Prashant Agrawal (Associate Professor)** sir for his invaluable guidance and insights, which have been instrumental in shaping this project, Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Dean & Professor, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time.

Harshit Mishra (2300290140072)

Himanshu Bhardwaj (2300290140076)

Fauwaz Ayub (2300290140063)

TABLE OF CONTENTS

Declaration	2
Certificate	3
Abstract	4
Acknowledgements	5
Table of Contents	6
List of Diagrams	7
1 Introduction	8-9
1.1 Overview	8
1.2 Objective	8
1.3 Project Scope	8
1.4 Hardware / Software used in Project	9
2 Feasibility Study	10-11
2.1 Economical Feasibility	10
2.2 Technical Feasibility	10
2.3 Operational Feasibility	11
3 Design and Planning	12-19
3.1 SDLC	12
3.1.1 Agile model	13
3.1.2 Incremental model	14
3.2 Use Case Diagram	15-16
3.3 ER Diagram	17
3.4 Data Flow Diagram	18
3.5 Flow Chart	19
4 Project Screenshot	20-23
4.1 Meta mask Connectivity	20
4.2 Create Campaign	21
4.3 Campaign	22
4.4 Amount Donation	23
5 Code	24-36
6 Testing	37-38
6.1 Introduction of Testing	37
6.2 Scope of Testing	37
6.3 Types of Testing	38
7 Conclusion	39
8 Future Scope	40-41
9 References	42

LIST OF DIAGRAM

Figure No.	Name of Figure	Page No.
3.1.1	Agile Model	12
3.1.2	Incremental Model	14
3.2	Use Case Diagram	15-16
3.3	ER Diagram	17
3.4	DFD	18
3.5	Flowchart	19
4.1	Meta mask Connectivity	20
4.2	Create Campaign	21
4.3	Campaign	22
4.4	Amount Donation	23

CHAPTER 1

INTRODUCTION

1.1 Overview

FundHub is an innovative platform designed to revolutionize the way individuals, organizations, and communities raise capital. The core concept revolves around crowdfunding, enabling users to pool small contributions from a large number of people to support various causes, projects, or ventures.

1.2 Objective

The objective of FundHub is to simplify the process of raising funds by providing a user-friendly platform where individuals, organizations, and communities can connect with donors. It aims to enhance transparency through real-time tracking and reporting, making the fundraising process trustworthy and reliable. FundHub focuses on accessibility, ensuring that anyone, regardless of their location or background, can use the platform to support their cause. By fostering collaboration and building a sense of community, it seeks to encourage collective efforts toward meaningful projects. Ultimately, FundHub aspires to create a positive impact by empowering users to turn their ideas into reality with the help of collective support.

1.3 Project Scope

The scope of FundHub covers the development and operation of a crowdfunding platform designed to cater to various fundraising needs. It targets a diverse audience, including students seeking funds for education or projects, startups needing seed capital, non-profit organizations, and individuals pursuing personal or creative ventures. The platform includes features such as secure user registration, customizable campaign pages, real-time contribution tracking, transparent reporting, and multiple payment options for ease of access. Initially focused on local and regional users, the platform has the potential to expand globally.

1.4 Hardware / Software Used in Project

Servers

High-performance servers with adequate storage to support blockchain nodes and user transactions.

User Devices

The platform must be accessible on various devices, including desktops, laptops, tablets, and smartphones.

Blockchain Technology

A blockchain framework such as Ethereum or Binance Smart Chain is essential to enable secure, decentralized transactions and smart contracts.

Web Development Frameworks

React, Node.js, and Express.js will be used for the frontend and backend development of the platform.

Database Management Systems:

MySQL will be used to store non- blockchain-related data such as user information and project details.

IDE (Integrated Development Environment)

Visual Studio code can be used for code analysis, debugging, and testing, among other things. It is particularly useful for web creation using web application frameworks like Flask.

Browser

Chrome, Edge

CHAPTER 2

FEASIBILITY STUDY

A feasibility study is a crucial step in assessing the viability of a project or system before investing time and resources into its development.

2.1 Economical Feasibility

An economical feasibility study for FundHub looks at how much money it will cost to build and run the platform and whether it can make enough money to be profitable. The costs to start FundHub include building the website and app, designing how it looks, testing everything, and paying for cloud storage. Ongoing costs include paying for hosting, maintenance, staff, marketing, and making sure the platform follows legal rules.

FundHub can make money by charging a small fee (like 3-5%) on each donation or by offering extra features for a subscription fee. It can also earn money from ads or partnerships with companies. For example, if FundHub helps raise \$100,000 a month and charges a 3% fee, it would make \$3,000 per month.

To figure out when FundHub will start making a profit, we compare the total costs to the money it makes from users. Once FundHub covers its costs and starts earning more money, it will become profitable.

2.2 Technical Feasibility

The technical feasibility of FundHub revolves around selecting the appropriate technologies and ensuring they are suitable for handling the platform's requirements. Node.js is an ideal choice for FundHub's backend due to its asynchronous, event-driven architecture, which makes it highly efficient for handling many simultaneous connections, crucial for real-time updates on donations and campaign status. It's also known for its fast performance and scalability, which are essential for a crowdfunding platform. Using frameworks like Express.js helps streamline the development process, and Node.js supports a wide variety of libraries, enhancing development efficiency.

MetaMask integration is technically feasible as it simplifies the use of cryptocurrency transactions. By allowing users to donate via Ethereum, FundHub can cater to the growing interest in decentralized finance. The MetaMask wallet can be easily connected to the platform using JavaScript libraries like web3.js or ethers.js, enabling secure donation

handling through blockchain technology. MetaMask also supports smart contracts, which could be used to automate donation processing or fund release once specific conditions are met. However, FundHub must be mindful of the potential gas fees associated with Ethereum transactions, as they could affect donor experience, particularly if transaction costs are high during periods of network congestion.

For file storage, Pinata offers decentralized storage via the InterPlanetary File System (IPFS), which is an optimal choice for FundHub. Storing media files like images, videos, and documents on IPFS ensures data integrity and availability since files are distributed across multiple nodes, making them less likely to be lost or corrupted. Pinata simplifies this process by providing an API to upload and “pin” files, ensuring they are always available on the decentralized network. Using IPFS guarantees that media files related to fundraising campaigns remain intact and transparent, providing confidence to both donors and campaign organizers.

2.3 Operational Feasibility

Operational feasibility for FundHub looks at whether the platform can be developed, maintained, and operated successfully, considering factors like resources, technical skills, user support, and long-term sustainability.

The development of FundHub using Node.js, MetaMask, and Pinata is operationally feasible due to the availability of skilled developers familiar with these technologies. Node.js is widely used, and there is a large community of developers with expertise in JavaScript and backend development, making it easy to find the required talent for building the platform. Furthermore, Node.js has many pre-built libraries and frameworks, like Express.js, which can accelerate the development process and reduce the complexity of creating the platform’s core features.

CHAPTER 3

DESIGN AND PLANNING

3.1 Software development life cycle

3.1.1 Agile model

Agile model allows for flexibility to adapt to changes based on user feedback, Provides early delivery of functional components, such as campaign creation or donation systems. Ensures continuous improvement with iterative cycles and testing.

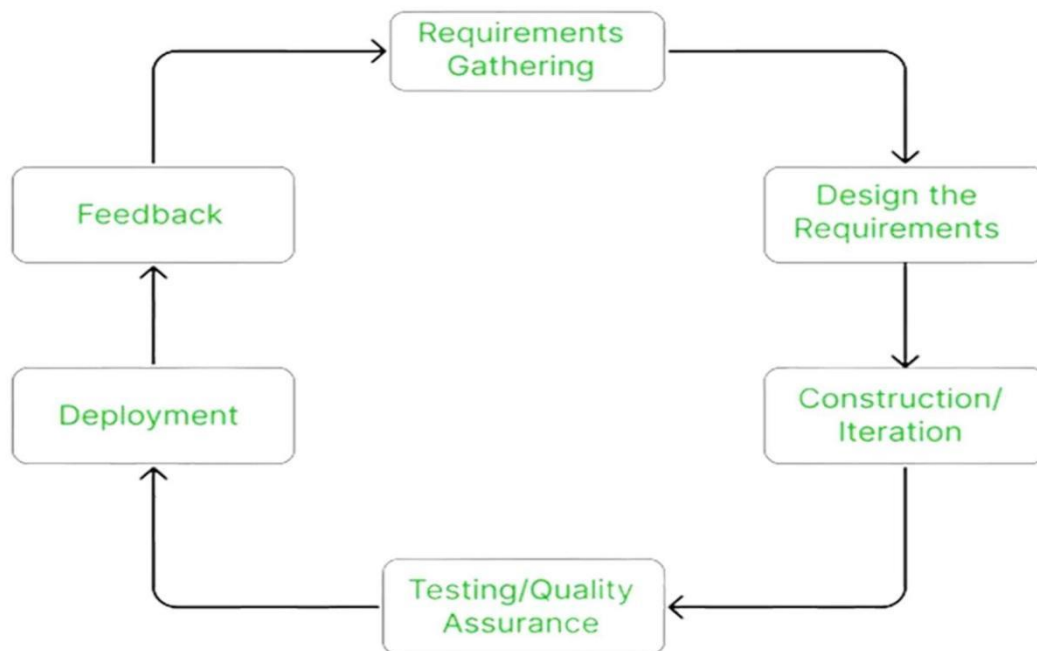


Fig : 3.1.1 Agile Model

- i. **Requirement Gathering:-** In this step, the development team must gather the requirements, by interaction with the customer. Development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.

- ii. Design the Requirements:-** In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.
- iii. Construction / Iteration:-** In this step, development team members start working on their project, which aims to deploy a working product.
- iv. Testing / Quality Assurance:-** Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:
- Unit Testing
 - Integration Testing
 - System Testing
- v. Deployment:-** In this step, the development team will deploy the working project to end users.
- vi. Feedback:-** This is the last step of the **Agile Model**. In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

3.1.2 Incremental model

The incremental model is a software development life cycle (SDLC) model that breaks the process of software creation into smaller, manageable chunks or increments. Each increment delivers a part of the complete functionality.

The product grows as more increments are added over time, ultimately leading to the final system. Enables FundHub to be developed and delivered in increments, starting with core features like campaign management, followed by payment systems and advanced functionalities (e.g., MetaMask integration).

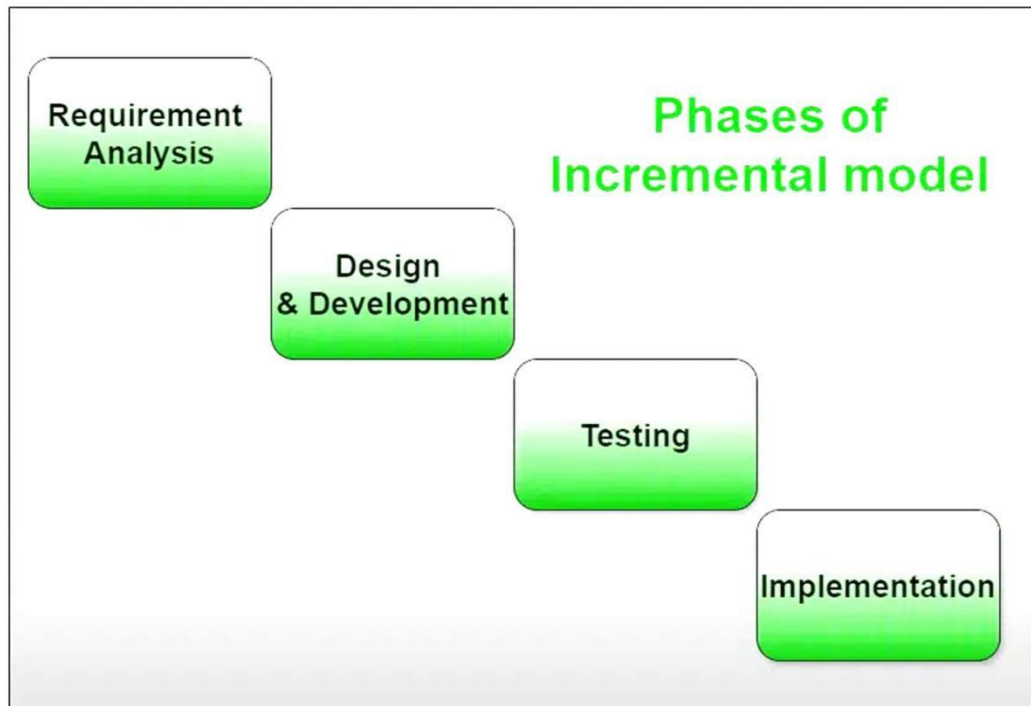


Fig 3.1.2 Incremental Model

- i. **Requirement analysis:** In Requirement Analysis At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer.
- ii. **Design & Development:** At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer. The Development Team first undertakes to develop core features (these do not need services from other features) of the system. Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.
- iii. **Deployment and Testing:** After Requirements gathering requirements then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. in development and Testing the product is checked and tested for the actual process of the model.

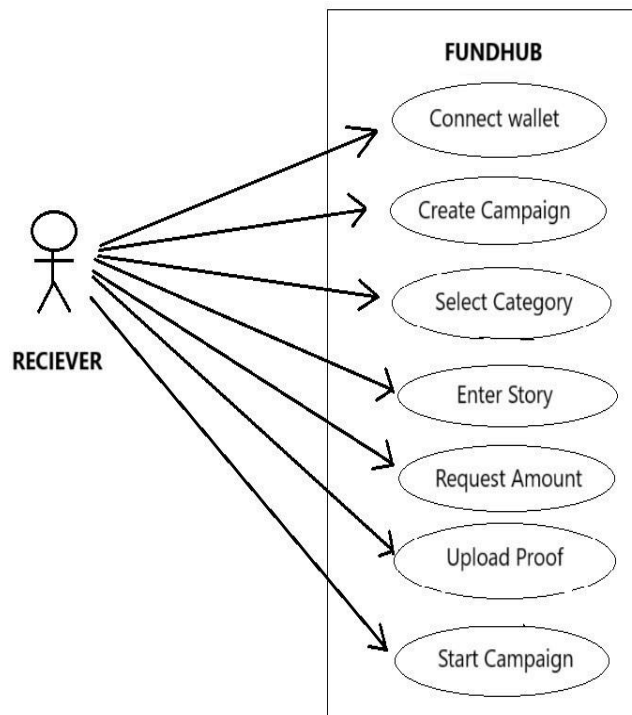
3.2 Use Case Diagram

Use-case diagrams model the behaviour of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

A use case diagram is used to represent the dynamic behaviour of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high level functionality of a system and also tells how the user handles a system.

Purposes of a use case diagram given below:

- . It gathers the system's needs.
- . It depicts the external view of the system.
- . It recognizes the internal as well as external factors that influence the system.
- . It represents the interaction between the actors.



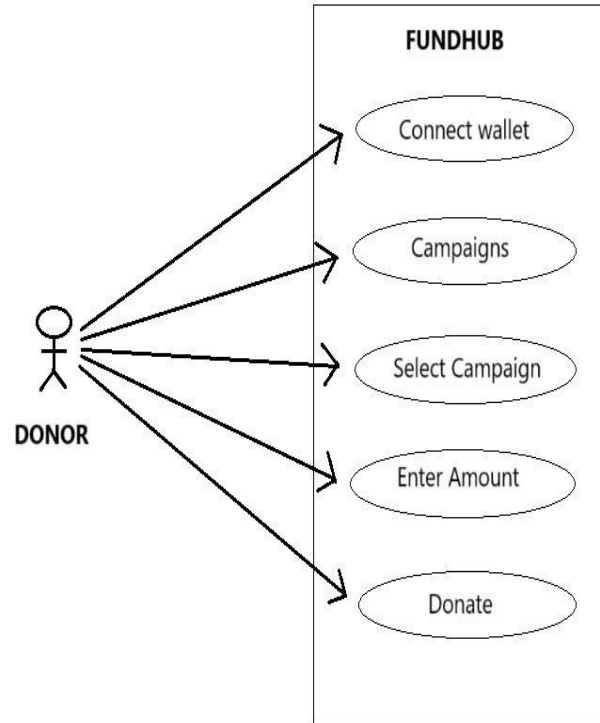


Fig : 3.2 Use Case Diagram

3.3 ER(Entity Relationship) Diagram

An ER Diagram (Entity-Relationship Diagram) is a visual representation of the entities and relationships in a database system. It is used to model the data structure and the relationships between different entities within a system, making it a key tool in database design.

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity relationship diagram.

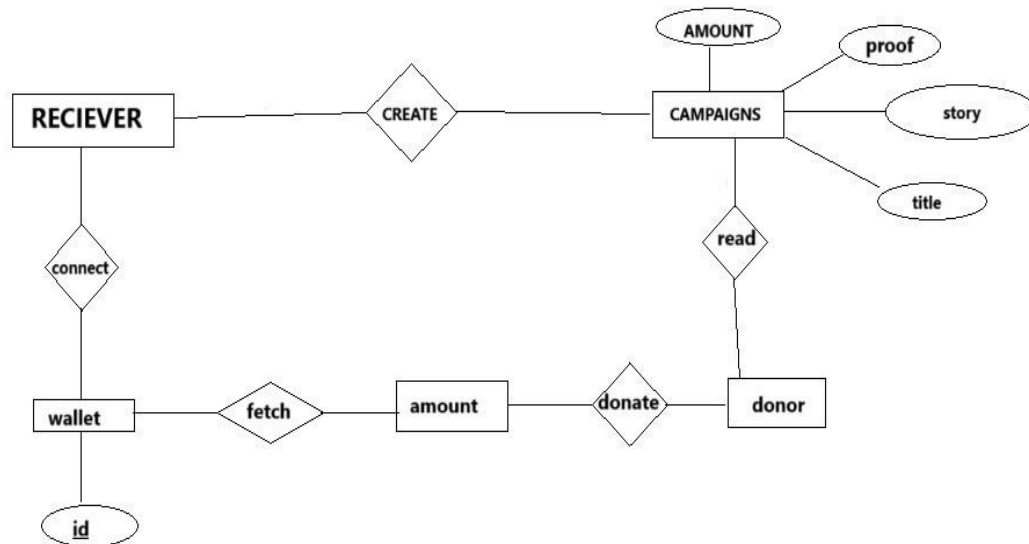


Fig 3.3 ER Diagram

3.4 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

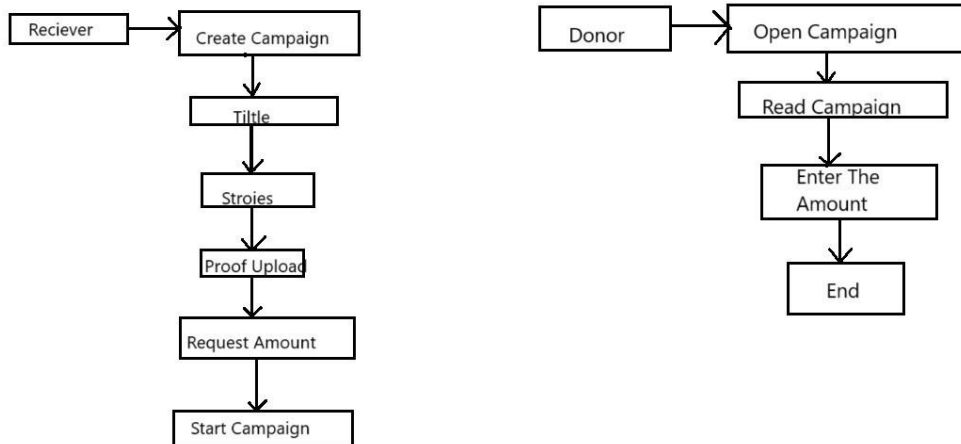


Fig 3.4 DFD

3.5 Flowchart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.

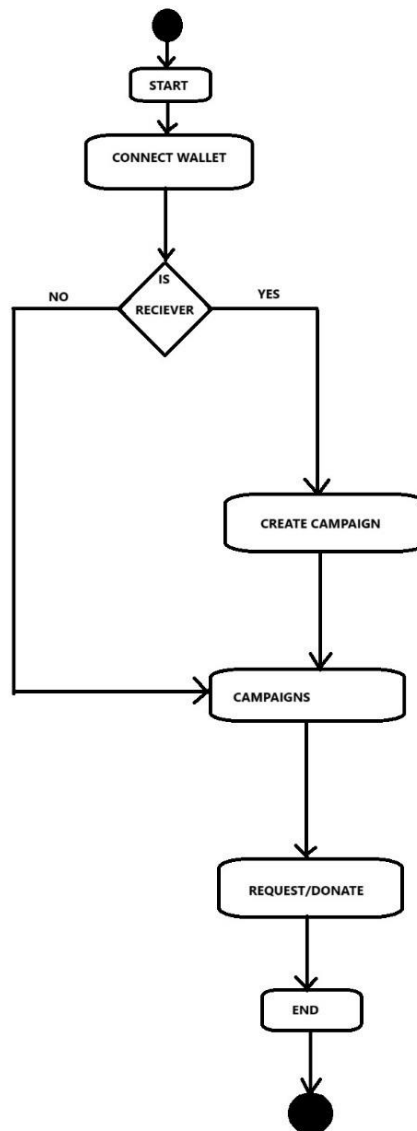
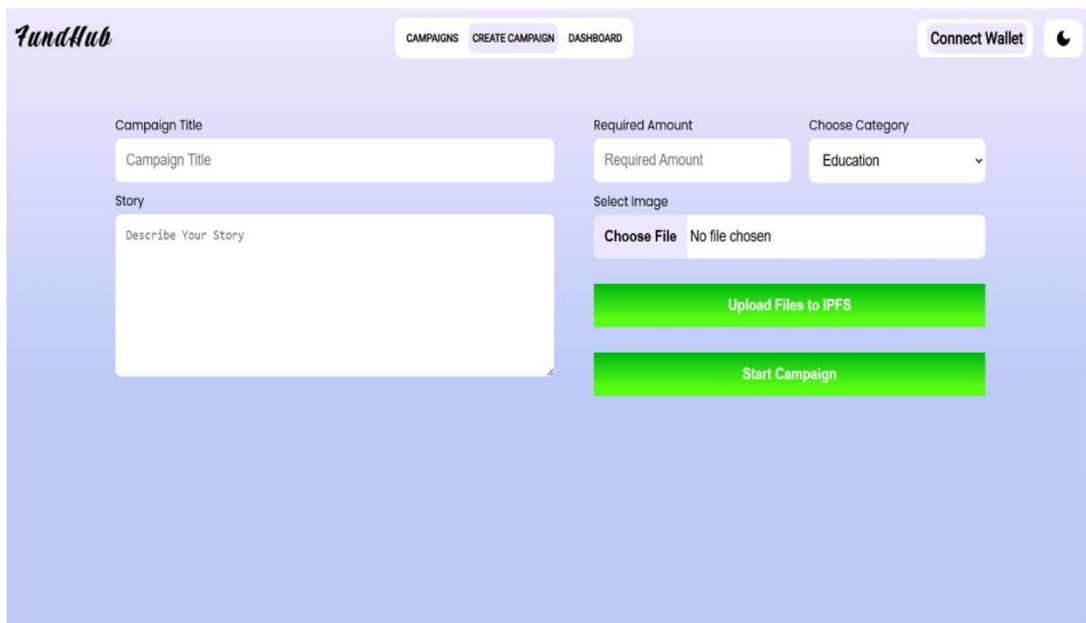


Fig 3.5 Flowchart

CHAPTER 4

PROJECT SCREENSHOTS

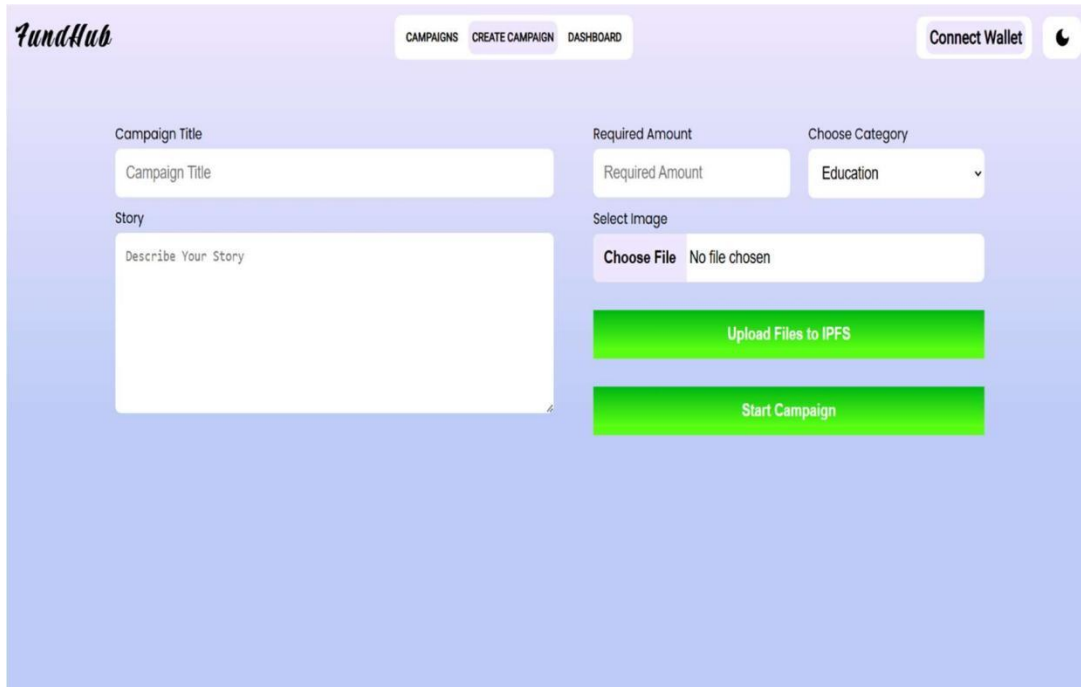
4.1 Meta mask Connectivity



The screenshot displays the 'FundHub' 'CREATE CAMPAIGN' interface. The top navigation bar includes links for 'CAMPAIGNS', 'CREATE CAMPAIGN', and 'DASHBOARD', along with a 'Connect Wallet' button and a user profile icon. The main form is divided into two columns. The left column contains a 'Campaign Title' input field with the placeholder text 'Campaign Title', and a 'Story' section with a large text area and the placeholder 'Describe Your Story'. The right column features a 'Required Amount' input field with the placeholder 'Required Amount', a 'Choose Category' dropdown menu currently set to 'Education', and a 'Select Image' section with a 'Choose File' button and the text 'No file chosen'. At the bottom of the right column are two prominent green buttons: 'Upload Files to IPFS' and 'Start Campaign'.

Fig: 4.1 Meta mask Connectivity

4.2 Create Campaign



The screenshot displays the 'FundHub' 'Create Campaign' interface. At the top, a navigation bar includes the 'FundHub' logo, a menu with 'CAMPAIGNS', 'CREATE CAMPAIGN' (highlighted), and 'DASHBOARD', a 'Connect Wallet' button, and a user profile icon. The main form area is divided into two columns. The left column contains a 'Campaign Title' text input field with the placeholder 'Campaign Title', and a 'Story' text area with the placeholder 'Describe Your Story'. The right column contains a 'Required Amount' text input field with the placeholder 'Required Amount', a 'Choose Category' dropdown menu currently set to 'Education', and a 'Select Image' section with a 'Choose File' button and the text 'No file chosen'. Below these fields are two prominent green buttons: 'Upload Files to IPFS' and 'Start Campaign'.

Fig. 4.2 Create Campaign

4.3 Campaign

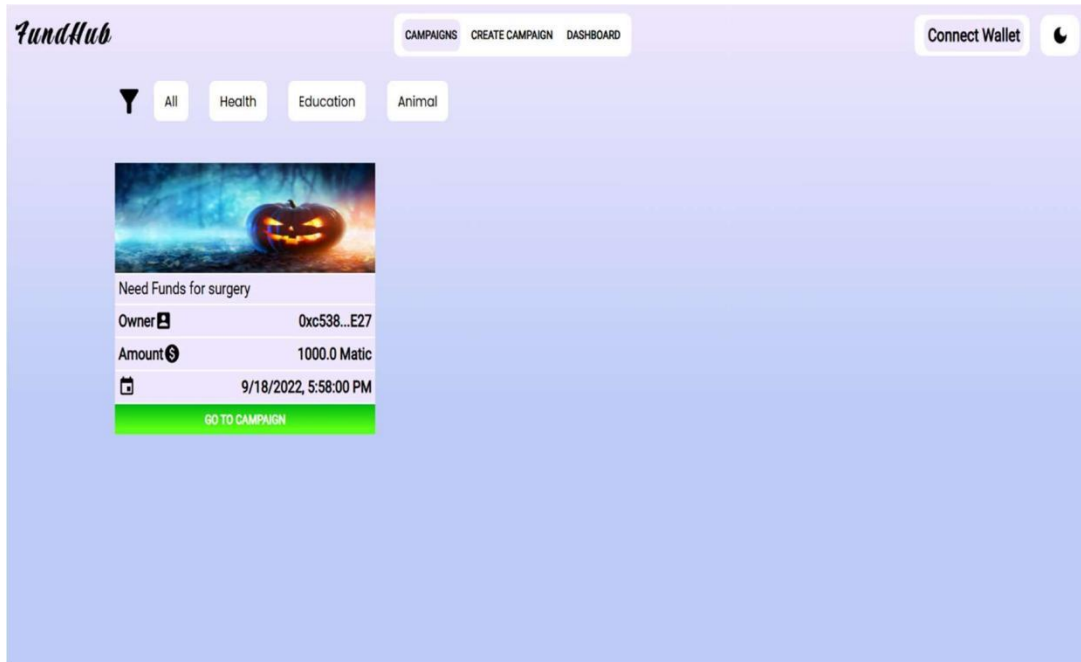
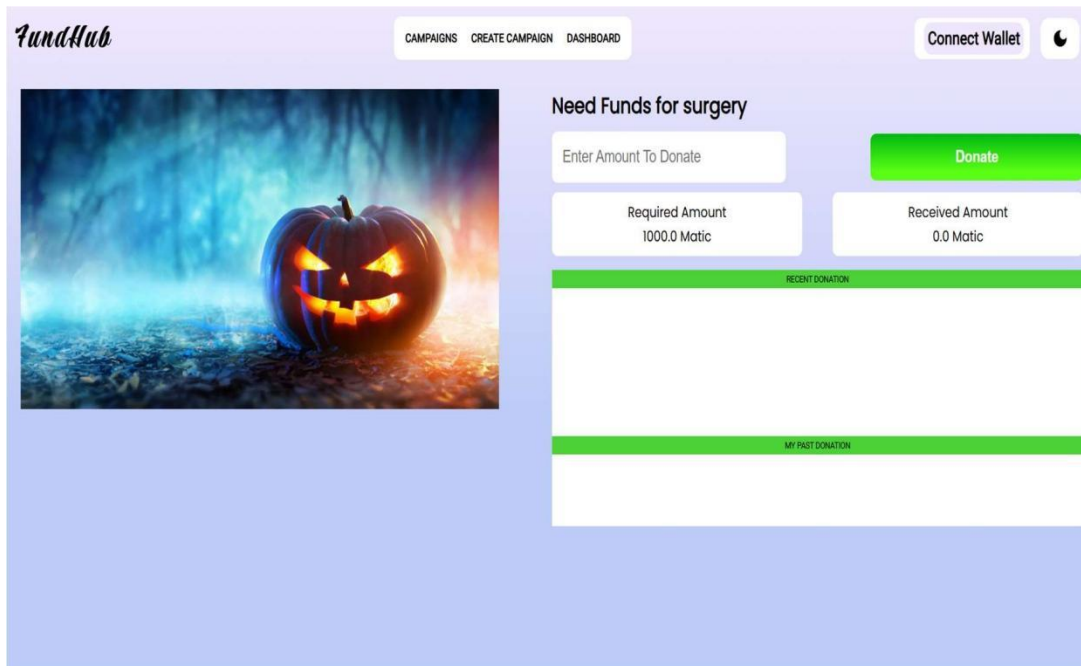


Fig : 4.3 Campaign

4.4 Amount Donation



The screenshot displays the FundHub interface for a campaign titled "Need Funds for surgery". The top navigation bar includes links for "CAMPAIGNS", "CREATE CAMPAIGN", and "DASHBOARD", along with a "Connect Wallet" button and a user profile icon. On the left, there is a featured image of a glowing jack-o'-lantern in a dark, misty setting. The main content area on the right features a "Donate" button and a form to "Enter Amount To Donate". Below this, two boxes show the "Required Amount" as 1000.0 Motic and the "Received Amount" as 0.0 Motic. At the bottom, there are two empty sections labeled "RECENT DONATION" and "MY FIRST DONATION".

FundHub CAMPAIGNS CREATE CAMPAIGN DASHBOARD Connect Wallet

Need Funds for surgery

Enter Amount To Donate **Donate**

Required Amount: 1000.0 Motic
Received Amount: 0.0 Motic

RECENT DONATION

MY FIRST DONATION

Fig: 4.4 Amount Donation

CHAPTER 5

CODE

5.1 SOLITIDY

```
// SPDX-License-Identifier: Unlicensed
```

```
pragma solidity >0.7.0 <=0.9.0;
```

```
contract CampaignFactory {  
    address[] public deployedCampaigns;
```

```
    event campaignCreated(  
        string title,  
        uint requiredAmount,  
        address indexed owner,  
        address campaignAddress,  
        string imgURI,  
        uint indexed timestamp,  
        string indexed category  
    );
```

```
    function createCampaign(  
        string memory campaignTitle,  
        uint requiredCampaignAmount,  
        string memory imgURI,  
        string memory category,  
        string memory storyURI) public  
    {
```

```
        Campaign newCampaign = new Campaign(  
            campaignTitle, requiredCampaignAmount, imgURI, storyURI, msg.sender);
```

```
        deployedCampaigns.push(address(newCampaign));
```

```
        emit campaignCreated(  
            campaignTitle,  
            requiredCampaignAmount,  
            msg.sender,  
            address(newCampaign),  
            imgURI,  
            block.timestamp,  
            category  
        );
```

```

    }
}

contract Campaign {
    string public title;
    uint public requiredAmount;
    string public image;
    string public story;
    address payable public owner;
    uint public receivedAmount;

    event donated(address indexed donar, uint indexed amount, uint indexed timestamp);

    constructor(
        string memory campaignTitle,
        uint requiredCampaignAmount,
        string memory imgURI,
        string memory storyURI,
        address campaignOwner
    ) {
        title = campaignTitle;
        requiredAmount = requiredCampaignAmount;
        image = imgURI;
        story = storyURI;
        owner = payable(campaignOwner);
    }

    function donate() public payable {
        require(requiredAmount > receivedAmount, "required amount fullfilled");
        owner.transfer(msg.value);
        receivedAmount += msg.value;
        emit donated(msg.sender, msg.value, block.timestamp);
    }
}

```


5.2 INDEX

```
import styled, { ThemeProvider } from 'styled-components';
import FilterAltIcon from '@mui/icons-material/FilterAlt';
import AccountBoxIcon from '@mui/icons-material/AccountBox';
import PaidIcon from '@mui/icons-material/Paid';
import EventIcon from '@mui/icons-material/Event';
import Image from 'next/image';
import { ethers } from 'ethers';
import { useState } from 'react';
import Link from 'next/link';
// import CampaignFactory from 'createcampaign.js';
// Main Component
export default function Index({ AllData, HealthData, EducationData, AnimalData }) {
  const [filter, setFilter] = useState(AllData);

  return (
    <ThemeProvider theme={theme}>
      <HomeWrapper>
        { /* Filter Section */ }
        <FilterWrapper>
          <FilterAltIcon style={{ fontSize: 40 }} />
          <Category onClick={() => setFilter(AllData)}>All</Category>
          <Category onClick={() => setFilter(HealthData)}>Health</Category>
          <Category onClick={() => setFilter(EducationData)}>Education</Category>
          <Category onClick={() => setFilter(AnimalData)}>Animal</Category>
        </FilterWrapper>

        { /* Cards Container */ }
        <CardsWrapper>
          { /* Check for empty campaigns */ }
          { filter.length > 0 ? (
            filter.map((e) => {
              return (
                <Card key={e.title}>
                  <CardImg>
                    <Image
                      alt="Crowdfunding dapp"
                      layout="fill"
                      src={"https://gateway.pinata.cloud/ipfs/" + e.image}
                    />
                  </CardImg>
                  <Title>{e.title}</Title>
                  <CardData>
                    <Text>Owner<AccountBoxIcon /></Text>
                    <Text>{e.owner.slice(0, 6)}...{e.owner.slice(39)}</Text>
                  </CardData>
                  <CardData>
                    <Text>Amount<PaidIcon /></Text>
                    <Text>{e.amount} Matic</Text>

```

```

        </CardData>
        <CardData>
          <Text><EventIcon /></Text>
          <Text>{ new Date(e.timeStamp * 1000).toLocaleString()}</Text>
        </CardData>
        <Link passHref href={'/' + e.address}>
          <Button>Go to Campaign</Button>
        </Link>
      </Card>
    );
  })
): (
  <p>No campaigns available in this category.</p>
)
</CardsWrapper>
</HomeWrapper>
</ThemeProvider>
);
}

```

// Static Props to Fetch Data from the Blockchain

```

export async function getStaticProps() {
  try {

```

```

    const provider = new ethers.providers.Web3Provider(window.ethereum);
    const signer = provider.getSigner();
    const address = await signer.getAddress();
    const campaignFactoryAddress =
      "0x03B8cEb7cE097Fe1fEaDAa4a628438a099E05754"; // Add your deployed factory
    address here

```

```

    const CampaignFactory = new ethers.Contract(
      campaignFactoryAddress,
      CampaignFactory.abi,
      signer
    );

```

```

    const getAllCampaigns = contract.filters.campaignCreated();
    const AllCampaigns = await contract.queryFilter(getAllCampaigns);
    const AllData = AllCampaigns.map((e) => ({
      title: e.args.title,
      image: e.args.imgURI,
      owner: e.args.owner,
      timeStamp: parseInt(e.args.timestamp),
      amount: ethers.utils.formatEther(e.args.requiredAmount),
      address: e.args.campaignAddress,
    }));

```

```

    const getHealthCampaigns = contract.filters.campaignCreated(null, null, null, null,
      null, null, 'Health');

```

```

const HealthCampaigns = await contract.queryFilter(getHealthCampaigns);
const HealthData = HealthCampaigns.map((e) => ({
  title: e.args.title,
  image: e.args.imgURI,
  owner: e.args.owner,
  timeStamp: parseInt(e.args.timestamp),
  amount: ethers.utils.formatEther(e.args.requiredAmount),
  address: e.args.campaignAddress,
}));

const getEducationCampaigns = contract.filters.campaignCreated(null, null, null, null,
null, null, 'education');
const EducationCampaigns = await contract.queryFilter(getEducationCampaigns);
const EducationData = EducationCampaigns.map((e) => ({
  title: e.args.title,
  image: e.args.imgURI,
  owner: e.args.owner,
  timeStamp: parseInt(e.args.timestamp),
  amount: ethers.utils.formatEther(e.args.requiredAmount),
  address: e.args.campaignAddress,
}));

const getAnimalCampaigns = contract.filters.campaignCreated(null, null, null, null,
null, null, 'Animal');
const AnimalCampaigns = await contract.queryFilter(getAnimalCampaigns);
const AnimalData = AnimalCampaigns.map((e) => ({
  title: e.args.title,
  image: e.args.imgURI,
  owner: e.args.owner,
  timeStamp: parseInt(e.args.timestamp),
  amount: ethers.utils.formatEther(e.args.requiredAmount),
  address: e.args.campaignAddress,
}));

return {
  props: {
    AllData,
    HealthData,
    EducationData,
    AnimalData,
  },
  revalidate: 10,
};
} catch (error) {
  console.error('Error fetching data:', error);
  return {
    props: {
      AllData: [],
      HealthData: [],
      EducationData: [],
    },
  };
}

```

```

        AnimalData: [],
      },
    };
  }
}

// Styling Components
const HomeWrapper = styled.div`
  display: flex;
  flex-direction: column;
  align-items: center;
  width: 100%;
`;

const FilterWrapper = styled.div`
  display: flex;
  align-items: center;
  width: 80%;
  margin-top: 15px;
`;

const Category = styled.div`
  padding: 10px 15px;..
  background-color: ${(props) => props.theme.bgDiv};
  margin: 0px 15px;
  border-radius: 8px;
  font-family: 'Poppins';
  font-weight: normal;
  cursor: pointer;
`;

const CardsWrapper = styled.div`
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  width: 80%;
  margin-top: 25px;
`;

const Card = styled.div`
  width: 30%;
  margin-top: 20px;
  background-color: ${(props) => props.theme.bgDiv};

  &:hover {
    transform: translateY(-10px);
    transition: transform 0.5s;
  }

  &:not(:hover) {

```

```

    transition: transform 0.5s;
  }
`;

const CardImg = styled.div`
  position: relative;
  height: 200px; // Ensure the parent container has a height
  width: 100%;
`;

const Title = styled.h2`
  font-family: 'Roboto';
  font-size: 18px;
  margin: 2px 0px;
  background-color: ${({props}) => props.theme.bgSubDiv};
  padding: 5px;
  cursor: pointer;
  font-weight: normal;
`;

const CardData = styled.div`
  display: flex;
  justify-content: space-between;
  margin: 2px 0px;
  background-color: ${({props}) => props.theme.bgSubDiv};
  padding: 5px;
  cursor: pointer;
`;

const Text = styled.p`
  display: flex;
  align-items: center;
  margin: 0;
  padding: 0;
  font-family: 'Roboto';
  font-size: 18px;
  font-weight: bold;
`;

const Button = styled.button`
  padding: 8px;
  text-align: center;
  width: 100%;
  background-color: #00b712;
  background-image:
    linear-gradient(180deg, #00b712 0%, #5aff15 80%);
  border: none;
  cursor: pointer;
  font-family: 'Roboto';
  text-transform: uppercase;

```

```
color: #fff;  
font-size: 14px;  
font-weight: bold;  
`;  
  
// Define a theme for the styled-components  
const theme = {  
  bgDiv: '#f0f0f0',  
  bgSubDiv: '#ffffff',  
};
```

5.3 HARDHAT.CONFIG.JS

```
require("@nomiclabs/hardhat-waffle");
require('dotenv').config({ path: './.env.local' });
const { task } = require("hardhat/config");

// Retrieve the private key and RPC URL from environment variables
const privateKey = process.env.NEXT_PUBLIC_PRIVATE_KEY;
const rpcUrl = process.env.NEXT_PUBLIC_RPC_URL;

// Define a custom task to print the list of accounts
task("accounts", "Prints the list of accounts", async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners();

  for (const account of accounts) {
    console.log(account.address);
  }
})

module.exports = {
  solidity: "0.8.10", // Specify the Solidity version
  defaultNetwork: "sepolia", // Set the default network to Sepolia
  networks: {
    hardhat: {}, // Configuration for Hardhat's local network
    sepolia: {
      url: rpcUrl || "https://sepolia.infura.io/v3/84fa584a498a45428a3f4dad697b27bd", //
      Use Infura RPC URL
      accounts: [privateKey] // Private key for the account to deploy
    }
  }
}
```

5.4 DASHBOARD.JS

```
import styled from 'styled-components';
import FilterAltIcon from '@mui/icons-material/FilterAlt';
import AccountBoxIcon from '@mui/icons-material/AccountBox';
import PaidIcon from '@mui/icons-material/Paid';
import EventIcon from '@mui/icons-material/Event';
import Image from 'next/image';
import { ethers } from 'ethers';
import CampaignFactory from '../artifacts/contracts/Campaign.sol/CampaignFactory.json'
import { useEffect, useState } from 'react';
import Link from 'next/link';

export default function Dashboard() {
  const [campaignsData, setCampaignsData] = useState([]);

  useEffect(() => {
    const Request = async () => {
      await window.ethereum.request({ method: 'eth_requestAccounts' });
      const Web3provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = Web3provider.getSigner();
      const Address = await signer.getAddress();

      const provider = new ethers.providers.JsonRpcProvider(
        process.env.NEXT_PUBLIC_RPC_URL
      );

      const contract = new ethers.Contract(
        process.env.NEXT_PUBLIC_ADDRESS,
        CampaignFactory.abi,
        provider
      );

      const getAllCampaigns = contract.filters.campaignCreated(null, null, Address);
      const AllCampaigns = await contract.queryFilter(getAllCampaigns);
      const AllData = AllCampaigns.map((e) => {
        return {
          title: e.args.title,
          image: e.args.imgURI,
          owner: e.args.owner,
          timeStamp: parseInt(e.args.timestamp),
          amount: ethers.utils.formatEther(e.args.requiredAmount),
          address: e.args.campaignAddress
        }
      })
      setCampaignsData(AllData)
    }
    Request();
  }, [])
```



```

return (
  <HomeWrapper>

    {/* Cards Container */}
    <CardsWrapper>

      {/* Card */}
      {campaignsData.map((e) => {
        return (
          <Card key={e.title}>
            <CardImg>
              <Image
                alt="crowdfunding dapp"
                layout='fill'
                src={"https://gateway.pinata.cloud/ipfs/" + e.image}
              />
            </CardImg>
            <Title>
              {e.title}
            </Title>
            <CardData>
              <Text>Owner<AccountBoxIcon /></Text>
              <Text>{e.owner.slice(0,6)}...{e.owner.slice(39)}</Text>
            </CardData>
            <CardData>
              <Text>Amount<PaidIcon /></Text>
              <Text>{e.amount} Matic</Text>
            </CardData>
            <CardData>
              <Text><EventIcon /></Text>
              <Text>{new Date(e.timeStamp * 1000).toLocaleString()}</Text>
            </CardData>
            <Link passHref href={'/' + e.address}><Button>
              Go to Campaign
            </Button></Link>
          </Card>
        )
      })}
      {/* Card */}

    </CardsWrapper>
  </HomeWrapper>
)
}

```

```

const HomeWrapper = styled.div`
  display: flex;
  flex-direction: column;
  align-items: center;

```

```

    width: 100%;
  },

  const CardsWrapper = styled.div`
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
    width: 80%;
    margin-top: 25px;
  },

  const Card = styled.div`
    width: 30%;
    margin-top: 20px;
    background-color: ${props => props.theme.bgDiv};

    &:hover{
      transform: translateY(-10px);
      transition: transform 0.5s;
    }

    &:not(:hover){
      transition: transform 0.5s;
    }
  },

  const CardImg = styled.div`
    position: relative;
    height: 120px;
    width: 100%;
  },

  const Title = styled.h2`
    font-family: 'Roboto';
    font-size: 18px;
    margin: 2px 0px;
    background-color: ${props => props.theme.bgSubDiv};
    padding: 5px;
    cursor: pointer;
    font-weight: normal;
  },

  const CardData = styled.div`
    display: flex;
    justify-content: space-between;
    margin: 2px 0px;
    background-color: ${props => props.theme.bgSubDiv};
    padding: 5px;
    cursor: pointer;
  },

  const Text = styled.p`
    display: flex;
    align-items: center;
    margin: 0;
    padding: 0;
  `

```

```
font-family: 'Roboto';
font-size: 18px;
font-weight: bold;
,
const Button = styled.button`
padding: 8px;
text-align: center;
width: 100%;
background-color: #00b712 ;
background-image:
    linear-gradient(180deg, #00b712 0%, #5aff15 80%);
border: none;
cursor: pointer;
font-family: 'Roboto';
text-transform: uppercase;
color: #fff;
font-size: 14px;
font-weight: bold;
,

```

CHAPTER 6

TESTING

6.1 Introduction to Testing

The primary goal of testing is to ensure that the **FundHub platform** operates seamlessly, providing a secure, efficient, and user-friendly experience. Testing is divided into functional, performance, security, and usability categories to cover all critical aspects.

Importance of Testing:

- Identify and fix defects before platform launch.
- Improve platform reliability and performance.
- Ensure compliance with security and data protection standards.
- Build trust among users and backers.

6.2 Scope of Testing:

The FundHub platform is designed to provide a comprehensive and seamless experience for both campaign creators and donors. Its core features include campaign creation, a streamlined donation process, and robust user management. The campaign creation functionality allows users to build detailed pages for their projects or causes. These pages include titles, descriptions, images, and funding goals. Creators can set deadlines to create urgency and organize their campaigns using categories and tags for easy discoverability. Real-time updates, such as the total amount raised and the number of backers, ensure transparency and encourage further participation.

The donation process prioritizes ease and security. Backers can contribute using multiple payment options like credit/debit cards, UPI, and digital wallets. Transactions are encrypted and processed securely through trusted gateways. Donors receive instant acknowledgments and receipts, and an option for anonymous donations provides flexibility. This ensures a smooth experience for all participants, promoting trust and engagement.

User management is another crucial aspect of the platform. It enables users to register via email or social media accounts and manage their campaigns or contributions through personalized dashboards. Campaign creators can track their progress, while donors can

view their donation history. Administrators are equipped with tools to oversee the platform, ensuring compliance, resolving disputes, and addressing fraudulent activities.

6.3 Types of Testing

Unit Testing: This involves testing individual components or units of the FundHub platform to ensure they function correctly in isolation. For FundHub, unit testing might involve validating algorithms for calculating campaign progress, handling payment processing, or sending notifications based on specific user actions.

Integration Testing: Integration testing verifies that different components of the FundHub platform work together as expected. For FundHub, this might involve testing how the payment gateway integrates with the donation process or how the user dashboard interacts with the campaign database and real-time updates.

Functional Testing: Functional testing examines whether the platform meets the specified functional requirements. For FundHub, this might involve testing whether campaigns are created with accurate details, donations are processed seamlessly, and notifications are triggered correctly for actions such as campaign milestones or donation confirmations.

Regression Testing: Regression testing ensures that recent code changes have not adversely affected existing functionalities. For FundHub, regression testing would ensure that updates to features like campaign management or payment systems do not disrupt previously working functionalities such as real-time progress tracking or user authentication.

User Acceptance Testing (UAT): UAT involves testing the platform with real users to ensure it meets their needs and expectations. For FundHub, UAT might involve gathering feedback from users about the ease of campaign creation, donation processes, and the overall user interface to ensure the platform is intuitive and effective.

Performance Testing: Performance testing evaluates the platform's responsiveness, scalability, and stability under various load conditions. For FundHub, performance testing might involve measuring how quickly campaign data is updated and displayed, how efficiently donations are processed during peak usage times, and how well the system handles multiple concurrent users creating campaigns or making donations.

CHAPTER 7

CONCLUSION

In conclusion, FundHub aims to revolutionize the way fundraising is conducted by offering a secure, user-friendly, and scalable platform for campaign creators and backers. Its comprehensive features, including seamless campaign creation, real-time progress tracking, and flexible donation processes, ensure that users can effectively reach their funding goals. With a focus on scalability, FundHub is designed to accommodate a growing number of users and campaigns without compromising performance.

Ultimately, FundHub is more than just a crowdfunding platform—it's a tool that brings communities together to support and fund innovative ideas. It offers a reliable and transparent space for creators to raise funds while ensuring backers can confidently contribute to causes they believe in. The platform's combination of functionality, security, scalability, and usability positions it as a future leader in the crowdfunding space, helping people bring their visions to life with the support of others.

CHAPTER 8

FUTURE SCOPE

The future of **FundHub** is filled with exciting possibilities to improve and grow the platform. Here are some simple ways it could evolve:

- **More Payment Options:** FundHub can add new ways to pay, like cryptocurrencies or mobile wallets, to make it easier for people around the world to donate.
- **Smart Recommendations:** By using technology like AI, FundHub can suggest campaigns to users based on their interests and previous donations, making it easier to find relevant projects.
- **Different Types of Campaigns:** FundHub could expand to support more types of crowdfunding, such as Education, Healthcare etc.
- **Better Social Media Sharing:** By allowing users to share campaigns easily on social media, FundHub can help campaigns reach more people and raise more funds.
- **Improved Tools for Creators:** FundHub can offer better analytics for campaign creators, helping them understand who their backers are and improve their campaigns.
- **Helping Charities:** FundHub could add features for nonprofits, like tax receipts and partnerships with big companies to raise more funds.
- **Peer-to-Peer Fundraising:** Users could create their own fundraising campaigns for causes they care about, helping FundHub reach more people.
- **Global Reach:** FundHub can expand to other countries by offering different , and meeting local laws languages, currencies, making it accessible worldwide.

- **Mobile App:** A FundHub app could make it easier for people to donate and manage their campaigns on the go, with notifications for updates.
- **Blockchain for Transparency:** FundHub could use blockchain technology to ensure that all donations are tracked and transparent, which would build trust among users.

CHAPTER 9

REFERENCES

1. Books :

- i) Duckett, J. (2011). *HTML and CSS: Design and Build Websites*. Wiley.
- ii) Marijn Haverbeke (2018). *Eloquent JavaScript*
- iii) David Herron (2019). *Node.js Web Development*
- iv) Josef R. G., and Robert R. S. (2023). *Blockchain and Metamask*

2. Websites :

- i) **W3Schools: HTML Tutorial**
Website: <https://www.w3schools.com/html/>
- ii) **CSS-Tricks**
Website: <https://css-tricks.com/>
- iii) **W3Schools: JavaScript Tutorial**
Website: <https://www.w3schools.com/js/>
- iv) **W3Schools: Node.js Tutorial**
Website: <https://www.w3schools.com/nodejs/>
- v) **MetaMask Official Website**
Website: <https://metamask.io/>

