**PROJECT REPORT**

***SathiSahyogi***

*Submitted in partial fulfillment of the requirements*

*for the award of the degree of*

**Bachelor of Computer Application**

**(Blockchain)**

**Under the supervision of:** **Submitted By:**

Dr. Bhawana Vidit Kulshrestha

Assistant Professor (SCSET) E22BCAU0034



**Plot Nos 8, 11, TechZone 2, Greater Noida, Uttar Pradesh 201310**

**2023-24**

**ABSTRACT**

Abstract—SathiSahyogi is a charity crowdfunding platform designed to revolutionize the way charitable causes raise funds. Crowdfunding is a method of fundraising that involves collecting small amounts of money from a large number of people, typically via the internet. There are various types of crowdfunding, including reward-based, equity-based, and donation-based crowdfunding. While traditional crowdfunding platforms often operate on a centralized model, SathiSahyogi leverages blockchain technology and smart contracts to offer a decentralized and transparent approach to fundraising.

Our platform prioritizes user-friendly design and security, ensuring that contributors can contribute to charitable initiatives with confidence and ease. By focusing on charity crowdfunding, SathiSahyogi provides a dedicated space for charitable organizations to showcase their projects and connect with potential beneficiaries. We aim to streamline the donation process, offering an efficient and transparent way for individuals and organizations to support meaningful charitable causes.

Join SathiSahyogi in reshaping the future of charity fundraising, where transparency, efficiency, and community engagement come together to make a lasting impact on society.

**TABLE OF CONTENTS**

**Title**…………………………………………………………………………………………..…….. I

**Abstract**………………………………………………………………………………………….... II

**Contents**…………………………………………………………………….….…….…………... III

**Chapter-1: INTRODUCTION**…………………………………………………………………... 1

1.1 Introduction of your application that you are working on with diagrams………...….…1

1.1 Problem Statement………………………………………………………………...….…1

1.2 Objectives…………………….…………………………...…………………………… 3

**Chapter-2: TECHNOLOGY USED**………………………………………………………………6

2.1 Blockchain (public or private)…...……………………………………………………...6

2.2 Smart contract and its language used.………………..………………………………….7

2.3 Consensus algorithm used in the implemented Blockchain …..…………..…………....7

**Chapter-3: RELATED REVIEW**………………………………………………………………..11

3.1 Write about existing papers related to your work…...………………..………………..11

**Chapter-4: PROPOSED MODEL**……………....……………………………………………….17

4.1 Proposed model name and its description ……………………………………………..17

4.2 Smart contracts working with flow diagrams….……………..………………………..19

**Chapter-5: IMPLEMENTATION**……………....……………………………………………….23

5.1 Tool and technology used (frontend and backend)…………………………….………23

5.2 Results ……………………. …………………………………………………………..23

**Conclusion**………………………………………………………………………………………...42

**Future Scope**………………………………………………………………………………………43

**References**…………………………………………………………………………………………44

**CHAPTER 1**

**INTRODUCTION**

* 1. **Introduction**

In the evolving landscape of fundraising and charitable giving, traditional methods often grapple with issues of transparency, trust, and engagement[18]. These challenges can hinder the potential for impactful philanthropy, leaving both contributors and beneficiaries searching for more reliable and efficient avenues. Enter SathiSahyogi—a revolutionary charity crowdfunding platform designed to address these very concerns and reinvigorate the spirit of giving.

SathiSahyogi combines the principles of decentralized fundraising with the transformative capabilities of blockchain technology to create a transparent, secure, and inclusive platform. Drawing inspiration from the "Research on Charity System Based on Blockchain," our platform leverages blockchain's inherent transparency to record and verify charitable activities, thereby restoring public trust and preventing fraud. By embracing the concept of nonprofit crowdfunding, SathiSahyogi collects small donations from a diverse community of supporters online, making philanthropy affordable and accessible to all. Campaigns on SathiSahyogi set clear goals, engage supporters through shared missions, and foster a sense of community and collaboration.

Choosing SathiSahyogi means opting for a fundraising solution that is not only transparent and affordable but also encourages community engagement and trust. Unlike traditional platforms, SathiSahyogi operates without intermediaries, ensuring that a larger portion of the donations reaches the intended beneficiaries. By integrating the best practices from leading nonprofit crowdfunding platforms like GoFundMe, FundRazr, and Indiegogo, SathiSahyogi offers a user-friendly experience that empowers individuals and organizations to support causes they believe in. Join us in revolutionizing the world of charity crowdfunding, where transparency, efficiency, and community come together to make a lasting impact on society.[1]

* 1. **Problem Statement**

In today's world, many charitable fundraising platforms face serious problems. One big issue is that people often don't trust these platforms because they're not always transparent about where the donated money goes. This lack of trust makes it harder for genuine causes to get the donations they need. Another problem is that these platforms are often controlled by a central authority, which can lead to fraud or misuse of funds. Additionally, the high costs associated with running these centralized platforms mean that less money goes to the causes people want to support.

Given these challenges, there is a clear need for a new kind of charity fundraising platform. This platform should be transparent, so people can see exactly how their donations are being used. It should also be secure, to prevent any fraud or misuse of funds. Lastly, it should be decentralized, meaning it's not controlled by a single authority, to reduce costs and ensure more money reaches the causes people care about.

In response to these identified issues and needs, this research introduces **SathiSahyogi**—a revolutionary charity crowdfunding platform designed to address the challenges by leveraging the power of blockchain technology and decentralized systems.[3][8]

* 1. **Objectives**

1. **Development of a Decentralized Crowdfunding Platform**: The primary aim of the project is to develop a decentralized crowdfunding platform using blockchain technology and smart contracts. This involves the creation of a secure and transparent platform that eliminates the need for intermediaries and ensures the integrity of fundraising activities.
2. **Designing User-Centric Interface**: A key focus of the project is to design an intuitive and user-friendly interface for the crowdfunding platform. This interface aims to enhance the user experience by prioritizing ease of use and accessibility for individuals of varying technical backgrounds.
3. **Global Engagement and Collaboration**: The project aims to foster global engagement and collaboration by welcoming users from diverse geographical locations and backgrounds. By creating a platform that facilitates collaboration and idea exchange, the project seeks to harness the collective creativity and expertise of a global community.
4. **Ensuring Accessibility for All Users**: An important objective of the project is to ensure accessibility for users with varying levels of technical proficiency. This involves designing features and functionalities that are easy to understand and navigate, thereby enabling broad participation.
5. **Compliance with Legal and Regulatory Standards**: The project is committed to complying with relevant legal and regulatory standards governing crowdfunding activities. This includes implementing measures to protect user data and ensure ethical and lawful practices throughout the platform.

**CHAPTER 2**

**TECHNOLOGY USED**

**2.1 Blockchain**

SathiSahyogi is an innovative online platform designed to support individuals and communities in need by facilitating crowdfunding for important causes. Developed using advanced technologies such as Tailwind.css, React.js and Next.js. SathiSahyogi ensures a seamless user experience across various devices, including computers and mobile phones. Behind the scenes, Solidity, a specialized programming language, is utilized to establish secure and transparent crowdfunding protocols on the internet. These protocols enable users to create, manage, and contribute funds to diverse projects with confidence, free from concerns about fraud or manipulation. Additionally, JavaScript plays a crucial role in integrating the platform with these specialized protocols, enabling smooth communication and data sharing. Furthermore, SathiSahyogi integrates MetaMask, a secure digital wallet solution, to facilitate safe connections to users' financial accounts, enabling hassle-free participation in crowdfunding initiatives. By harnessing the power of these technologies, SathiSahyogi empowers individuals to support meaningful projects and causes in a straightforward, professional, and trustworthy manner.[2][5]

**2.2 Smart Contracts and Languages Used**

* Smart Contracts(Solidity): Crowdfunding.sol
* Components(React.js):
  + Card.jsx
  + Footer.jsx
  + Hero.jsx
  + Logo.jsx
  + Menu.jsx
  + NavBar.jsx
  + PopUp.jsx
* Javascripts(Javascript):
  + Crowdfunding.js
  + Constant.js
  + Index.js
  + \_app.js
  + Deploy.js
* Tailwind Css(Css):
  + Globals.css

**CHAPTER 3**

**RELATED REVIEW**

**3.1 Related Work**

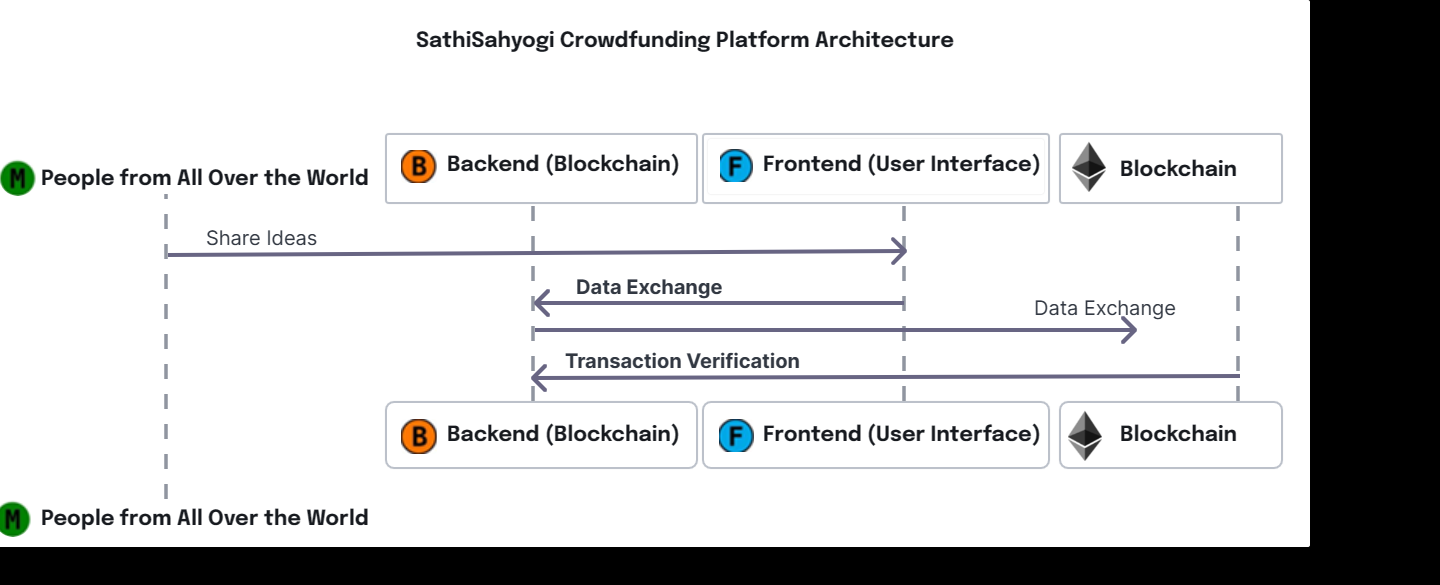
1. **Kickstarter** 
   1. **Overview**: By integrating blockchain technology, Kickstarter, the first project of its kind, sought to transform crowdfunding. Its main goal was to empower backers and creators by using decentralized technology to guarantee safe and open fundraising procedures.
   2. **Technology**: To create irreversible contracts between project creators and backers and automate fund allocation, the project made use of Ethereum-based smart contracts. It also included tokenization for the platform's incentive system and fractional ownership.
   3. **Results**: Backers are now able to track the allocation of funds in real time thanks to the implementation, which significantly increased transparency. By providing governance rights through tokens, it also promoted active community involvement, which improved project support and ownership.
   4. **Problems and Takeaways**: Despite the achievements, scalability issues arose as a result of network congestion during periods of high usage. To tackle this, it was necessary to investigate layer-2 scaling options and optimize the execution of smart contracts. The project also discovered that, to achieve greater user adoption, strong community management and blockchain education are essential.[12][13]
2. **Crypto Cause** 
   1. **Overview**: GoFundMe and Crypto Cause are similar in that they both want to use blockchain technology to raise money for charity. Crypto Cause prioritizes safe and transparent donation procedures for a range of purposes and individual requirements, drawing inspiration from the values outlined in my project SathiSahyogi.
   2. **Technology**: By employing a hybrid blockchain solution based on the Ethereum and Stellar networks, Crypto Cause makes sure that donations are transparently tracked via smart contracts. To track and distribute money effectively, it combines tokenized donations with distinct digital identities for beneficiaries.
   3. **Results**: By using a hybrid blockchain, Crypto Cause was able to track donations with more transparency and give donors the ability to see how their money moves from their contribution to its use. The immutability of blockchain records contributed to a rise in donor confidence on the platform.
   4. **Challenges and Lessons Learned**: The intricacy of blockchain presented Crypto Cause with obstacles like those we faced in the early phases of our project: user adoption. In order to address this, the donation process needed to be made simpler with user-friendly interfaces and extensive user education.
   5. **Reference to Project**: By combining several blockchain networks, improving donation tracking, and cultivating donor trust, Crypto Cause aimed to further innovate on the decentralized and transparent fundraising concepts that our project SathiSahyogi introduced.[14]
3. **Ethereum ICO (Initial Coin Offering)**
   1. **Overview**: Ethereum conducted a groundbreaking ICO in 2014, raising funds to develop a decentralized platform for smart contracts and decentralized applications (DApps).
   2. **Technology**: Utilized blockchain technology to enable the creation and execution of smart contracts, providing a decentralized computing platform.
   3. **Results**: Ethereum's ICO raised over $18 million, establishing it as a major player in the blockchain space. The project has since become a cornerstone for the development of various decentralized applications.
   4. **Challenges and Lessons Learned**: The success of Ethereum's ICO highlighted the potential of crowdfunding for blockchain projects. However, it also emphasized the importance of robust security measures, as demonstrated by subsequent ICO-related security incidents.
   5. **Reference to Project**: The Ethereum ICO serves as a reference for successful crowdfunding strategies and the transformative impact of blockchain on decentralized computing. [15]
4. **Filecoin**:
   1. **Overview**: Filecoin conducted an ICO in 2017 to develop a decentralized storage network, allowing users to rent and monetize their unused storage space.
   2. **Technology**: Leveraged blockchain technology to create a decentralized and secure network for distributed data storage, incentivizing participants with Filecoin tokens.
   3. **Results**: Filecoin's ICO raised over $200 million, making it one of the most successful ICOs at the time. The project aimed to revolutionize data storage and retrieval systems.
   4. **Challenges and Lessons Learned**: Filecoin faced challenges related to technical complexities and delays in launching its mainnet. Lessons learned include the importance of managing community expectations during development.
   5. **Reference to Project**: Filecoin's ICO demonstrates the potential of blockchain in incentivizing decentralized storage solutions, offering valuable insights for projects in a similar space. [16]
5. **Tezos**:
   1. **Overview**: Tezos conducted an ICO in 2017, raising funds for the development of a self amending blockchain platform with a focus on on-chain governance.
   2. **Technology**: Introduced on-chain governance mechanisms to allow token holders to participate in decision-making for protocol upgrades and improvements.
   3. **Results**: Tezos raised over $230 million in its ICO, emphasizing the importance of community involvement in blockchain governance.
   4. **Challenges and Lessons Learned**: Tezos faced legal challenges and internal disputes during its early development, leading to lessons on the significance of clear governance structures and legal compliance.
   5. **Reference to Project**: Tezos serves as a reference for projects exploring on-chain governance and the integration of community-driven decision-making processes. [17]

**CHAPTER 4**

**PROPOSED MODEL**

**4.1 Proposed Model**

* **SathiSahyogi Crowdfunding Platform Architecture**

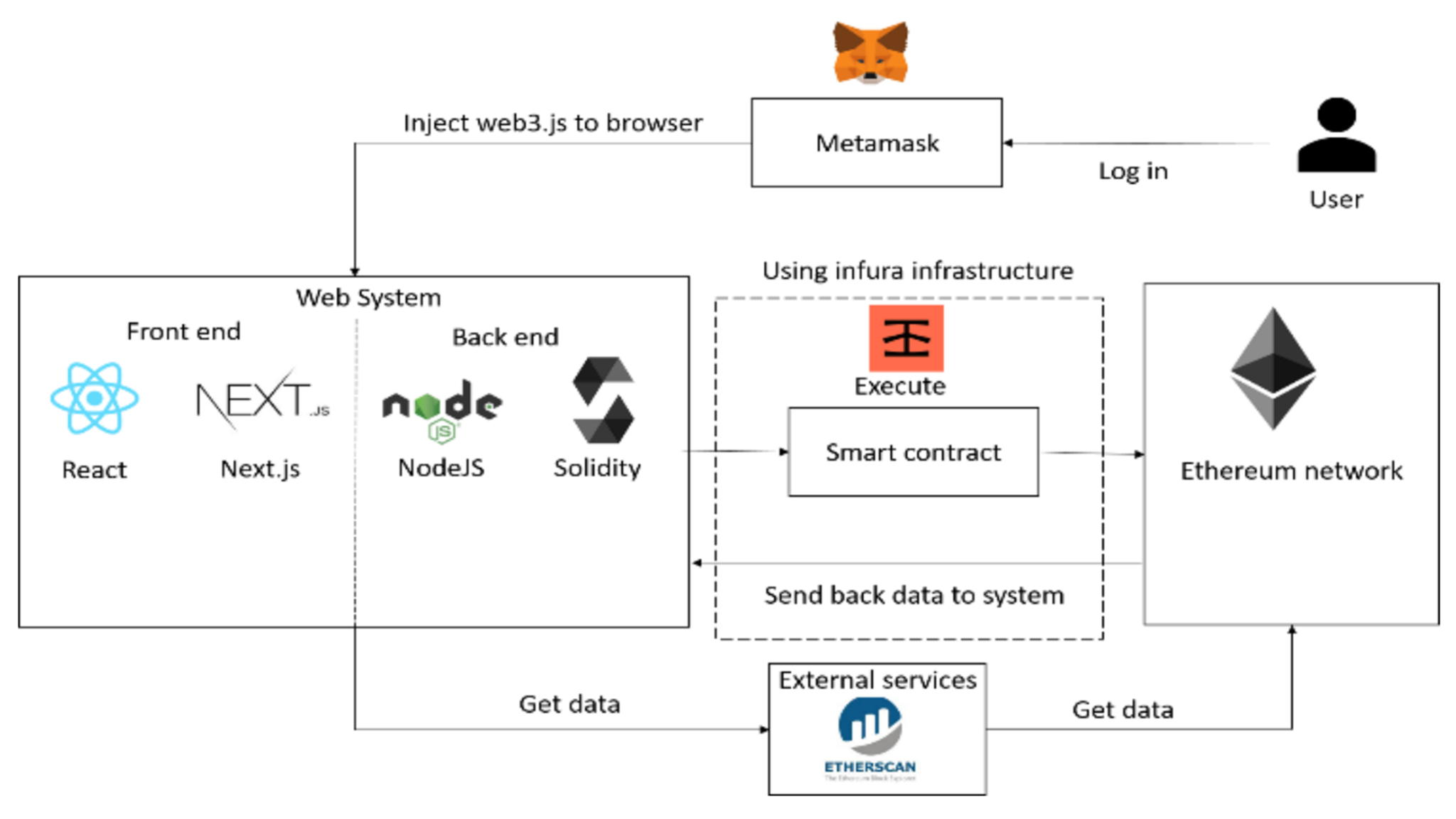
****

**Figure 1**

**Explanation for the figure 1:**

1. **People from All Over the World:** This represents users or participants in the blockchain network. They could be individuals or organizations that want to share ideas or data.
2. **Share Ideas:** This is the initial step where participants share their ideas or data. This could be in the form of transactions or digital assets.
3. **Backend (Blockchain):** This is the underlying technology where the shared ideas are stored. The blockchain is a decentralized and distributed digital ledger that records transactions across multiple computers.
4. **Data Exchange**: This step involves the transfer of data between the participants and the blockchain. The data of the shared ideas is exchanged in this step.
5. **Transaction Verification:** This is a crucial step in the blockchain process. Here, the transactions or shared ideas are verified for their authenticity.
6. **Frontend (User Interface):** This represents the application or platform that participants use to interact with the blockchain. It’s where users can view and manage their transactions.
7. **Blockchain:** This is the final output where the verified transactions are added to the blockchain.

* **Backend Working of SathiSahyogi**

****

**Figure 2**

**Explanation for the figure 2:**

The figure is a diagram of a system that uses a blockchain network, specifically Ethereum, to facilitate crowdfunding. Here’s a breakdown of the image:

* Web System
  + Front End: This refers to the user interface that users interact with. In the image, it shows React and Next.js being used.
  + Back End: This refers to the server-side code that processes information and interacts with the database. In the image, it shows Node.js being used.
* User: This refers to the person interacting with the system.
* Metamask: This is a browser extension that allows users to interact with Ethereum blockchain applications.
* Web3.js: This is a library that allows developers to interact with the Ethereum blockchain from a web browser.
* Infura Infrastructure: Infura is a company that provides blockchain infrastructure services, such as node access. A node is a computer that stores and validates data on a blockchain network.
* Smart Contract: This is a program that runs on the blockchain and automates the execution of an agreement. In this case, the smart contract would likely be used to manage the crowdfunding process, such as collecting donations and distributing funds to the project.
* Solidity: This is a programming language used to develop smart contracts for the Ethereum blockchain.
* External Services: This refers to any other services that the system interacts with, such as Etherscan, which is a block explorer for the Ethereum blockchain.

**4.2 Smart Contracts Working with Flow Diagrams**

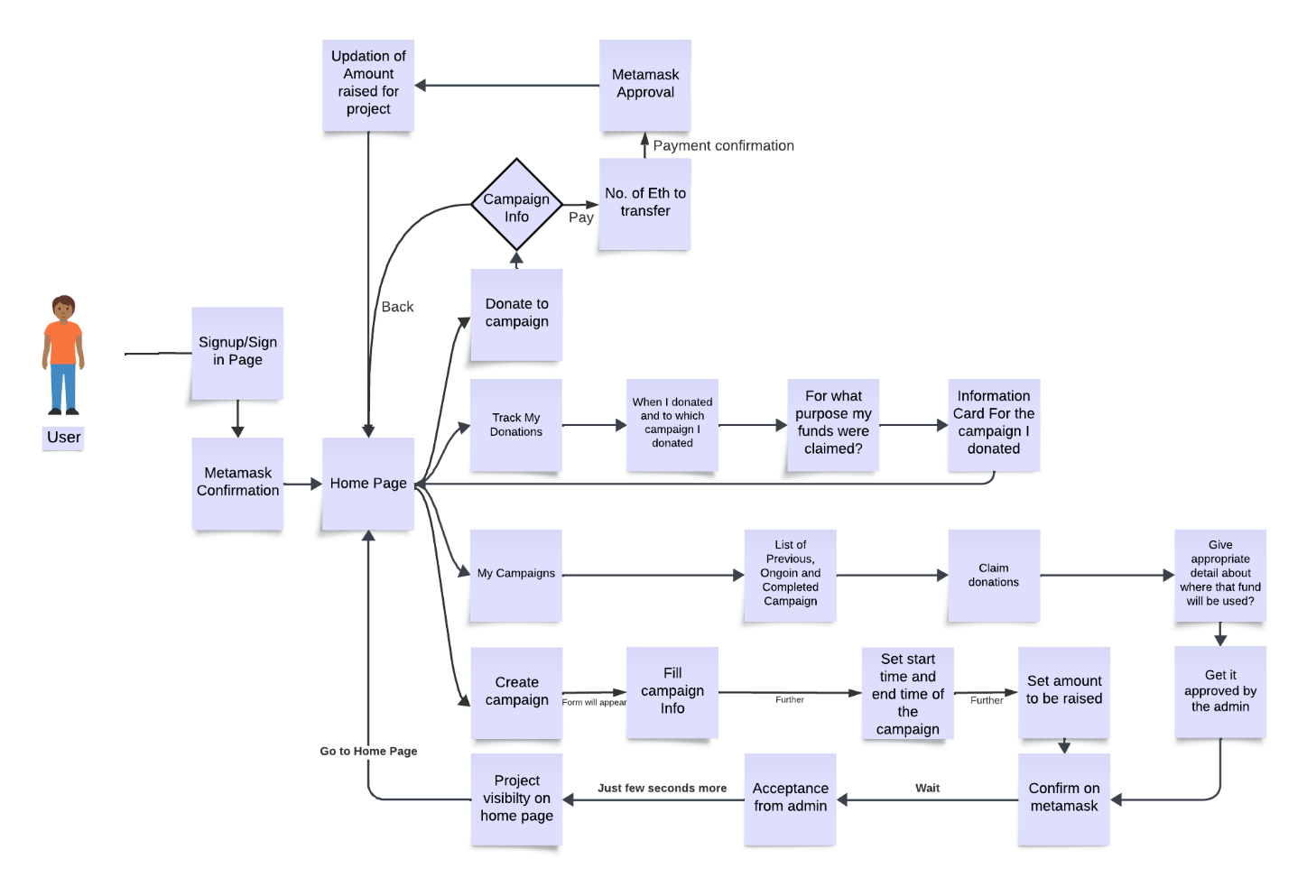


Figure 3

**Explanation for the figure 3:**

1. **Signup/Sign in Page:** The user begins by signing up for an account or signing in to their existing account on the crowdfunding platform.
2. **Campaign Info:** The user browses the platform to find a campaign that they are interested in supporting. Once they have found a campaign, they can review information about the campaign, such as the campaign goals and how the funds will be used.
3. **Donate to Campaign:** If the user decides to donate to the campaign, they can click on the "Donate" button.
4. **Metamask Confirmation:** The user will then be prompted to connect their Metamask wallet to the crowdfunding platform. Metamask is a browser extension that allows users to store and manage their cryptocurrency.
5. **Payment Confirmation:** Once the user's Metamask wallet is connected, they will be shown a confirmation screen that details the amount of cryptocurrency they are donating to the campaign. The user will also be able to review the transaction fee.
6. **Pay:** If the user agrees to the transaction fee and the amount of cryptocurrency they are donating, they can click on the "Pay" button to submit their donation.
7. **Project Visibility on Homepage:** Once the user's donation has been processed, the campaign will be displayed on the homepage of the crowdfunding platform.
8. **Track My Donations:** The user can track their donations by clicking on the "Track My Donations" button. This will show them a list of all the campaigns that they have donated to, as well as the amount of cryptocurrency they have donated to each campaign.
9. **Claim Donations (Optional):** In some cases, the user may be able to claim rewards or benefits for donating to a campaign. This will be indicated on the campaign information page. If a claim option is available, the user can click on the "Claim" button to claim their rewards.
10. **Further Information:** For more information about the donation process, the user can click on the "Give me appropriate detail about where that fund will be used" button. This will provide the user with more information about how the funds raised by the campaign will be used.

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Tool and Technology Used (frontend and backend)**

***GITHUB LINK:*** <https://github.com/viditkulsh/SathiSahyogi>

SathiSahyogi is an innovative online platform designed to support individuals and communities in need by facilitating crowdfunding for important causes. Developed using advanced technologies such as Tailwind.css, React.js and Next.js. SathiSahyogi ensures a seamless user experience across various devices, including computers and mobile phones. Behind the scenes, Solidity, a specialized programming language, is utilized to establish secure and transparent crowdfunding protocols on the internet. These protocols enable users to create, manage, and contribute funds to diverse projects with confidence, free from concerns about fraud or manipulation. Additionally, JavaScript plays a crucial role in integrating the platform with these specialized protocols, enabling smooth communication and data sharing. Furthermore, SathiSahyogi integrates MetaMask, a secure digital wallet solution, to facilitate safe connections to users' financial accounts, enabling hassle-free participation in crowdfunding initiatives. By harnessing the power of these technologies, SathiSahyogi empowers individuals to support meaningful projects and causes in a straightforward, professional, and trustworthy manner.

* **Solidity:**
  + **Crowdfunding.sol**

| // SPDX-License-Identifier: MIT // The above comment specifies that the code is released under the MIT license.  pragma solidity ^0.8.9; // This line specifies that the code should be compiled using Solidity version 0.8.9 or higher.  contract CrowdFunding {  // Struct to define the structure of a crowdfunding campaign  struct Campaign {  address owner; // Address of the campaign owner  string title; // Title of the campaign  string description; // Description of the campaign  uint256 target; // Target amount to be raised  uint256 deadline; // Deadline for the campaign  uint256 amountCollected; // Amount collected so far  address[] donators; // Array to store addresses of donators  uint256[] donations; // Array to store donation amounts corresponding to donators  }    // Mapping to store Campaign structs with unique IDs  mapping(uint256 => Campaign) public campaigns;   // Variable to keep track of the total number of campaigns created  uint256 public numberOfCampaigns = 0;   // Function to create a new crowdfunding campaign  function createCampaign(address \_owner, string memory \_title, string memory \_description, uint256 \_target, uint256 \_deadline) public returns (uint256) {  // Create a new Campaign struct  Campaign storage campaign = campaigns[numberOfCampaigns];   // Ensure that the deadline is in the future  require(\_deadline > block.timestamp, "The deadline should be a date in the future.");   // Initialize the campaign with provided details  campaign.owner = \_owner;  campaign.title = \_title;  campaign.description = \_description;  campaign.target = \_target;  campaign.deadline = \_deadline;  campaign.amountCollected = 0;   // Increment the number of campaigns  numberOfCampaigns++;   // Return the ID of the newly created campaign  return numberOfCampaigns - 1;  }   // Function to donate to a campaign  function donateToCampaign(uint256 \_id) public payable {  uint256 amount = msg.value; // Get the amount of Ether sent with the transaction  Campaign storage campaign = campaigns[\_id]; // Get the campaign with the provided ID   // Add the donator's address and donation amount to respective arrays  campaign.donators.push(msg.sender);  campaign.donations.push(amount);   // Transfer the donated Ether to the campaign owner  (bool sent,) = payable(campaign.owner).call{value: amount}("");  if (sent) {  // Update the amount collected for the campaign  campaign.amountCollected += amount;  }  }   // Function to get the list of donators and their donation amounts for a campaign  function getDonators(uint256 \_id) view public returns (address[] memory, uint256[] memory) {  return (campaigns[\_id].donators, campaigns[\_id].donations);  }   // Function to get details of all existing campaigns  function getCampaigns() public view returns (Campaign[] memory) {  // Create an array to store all campaigns  Campaign[] memory allCampaigns = new Campaign[](numberOfCampaigns);   // Iterate through the mapping of campaigns and populate the array  for (uint256 i = 0; i < numberOfCampaigns; i++) {  Campaign storage item = campaigns[i];  allCampaigns[i] = item;  }   // Return the array of campaigns  return allCampaigns;  } } |
| --- |

**Explanation:** The provided Solidity contract, CrowdFunding, facilitates the creation and management of crowdfunding campaigns on the Ethereum blockchain. Users can create new campaigns specifying details such as title, description, target amount, and deadline. They can then donate Ether to these campaigns, tracked by the contract, and view the list of donators and their contributions. The contract ensures that campaign deadlines are set in the future and securely handles the transfer of donated Ether to campaign owners. With functions for campaign creation, donation, and retrieval of campaign information, it provides a robust platform for crowdfunding initiatives on the Ethereum network.

* **Components:** 
  + **Card.jsx**

| **// Importing React library for creating components import React from 'react';  // Card component for displaying campaign details const Card = ({ allcampaign, setOpenModel, setDonate, title }) => {   // Function to calculate days left until campaign deadline  const daysLeft = (deadline) => {  const difference = new Date(deadline).getTime() - Date.now();  const remainingDays = difference / (1000 \* 3600 \* 24);  return remainingDays.toFixed(0);  };   return (  // Container for the card component  <div className="px-4 py-16 mx-auto sm:max-w-xl md:max-w-full lg:max-w-screen-xl md:px-24 lg:px-8 lg:py-20">  {/\* Title of the card \*/}  <p className='py-16 text-2xl font-bold leading-5'> {title} </p>  {/\* Grid layout for displaying campaign cards \*/}  <div className="grid gap-5 lg:grid-cols-3 sm:max-w-sm sm:mx-auto lg:max-w-full">  {/\* Mapping through each campaign to display its details \*/}  {allcampaign?.map((campaign, i) => (  // Individual campaign card  <div  onClick={() => (setDonate(campaign), setOpenModel(true))}  key={i + 1}  className="cursor-pointer border overflow-hidden transition-shadow duration-300 bg-white rounded"  >  {/\* Image of the campaign \*/}  <img  src="https://images.pexels.com/photos/932638/pexels-photo-932638.jpeg?auto=compress&amp;cs=tinysrgb&amp;dpr=3&amp;h=750&amp;w=1260"  className="object-cover w-full h-64 rounded"  alt=""  />  {/\* Campaign details \*/}  <div className="py-5 pl-2">  {/\* Days left until campaign deadline \*/}  <p className="mb-2 text-xs font-semibold text-gray-600 uppercase">  Days Left: {daysLeft(campaign.deadline)}  </p>  {/\* Campaign title \*/}  <a  href="/"  aria-label="Article"  className="inLine-block mb-3 text-black transition-colors duration-200 hover:text-deep-purple-accent-700"  >  <p className="text-2xl font-bold leading-5">{campaign.title}</p>  </a>  {/\* Campaign description \*/}  <p className="mb-4 text-gray-700">{campaign.description}</p>  {/\* Campaign target and amount collected \*/}  <div className="flex space-x-4">  <p className="font-semibold"> Target: {campaign.target} Eth</p>  <p className="font-semibold">  Raised: {campaign.amountCollected} Eth  </p>  </div>  </div>  </div>  ))}  </div>  </div>  ); };  // Exporting Card component export default Card;** |
| --- |

**Explanation:** This code defines a React component called Card that displays campaign cards. It receives props such as allcampaign, setOpenModel, setDonate, and title. The daysLeft function calculates the remaining days until a campaign deadline. The component renders a grid layout with individual cards for each campaign. Each card contains campaign details such as title, description, days left, target amount, and amount collected. Clicking on a card opens a modal to donate.

* + **Footer.jsx**

| import React from "react";  // Footer component displaying footer section with product information, useful links, and contact details  const Footer = () => {  // Lists of products, contact details, and useful links  const productList = [  "ERC20 Token",  "Donation",  "IPFS",  ];  const contactList = [  "viditkul08@gmail.com",  "certainlysohail@gmail.com ",  ];  const usefullLink = [  "Home",  "About Us",  <a href="https://github.com/viditkulsh/SathiSahyogi" target="\_blank" rel="noopener noreferrer">  SathiSahyogi Github  </a>,  ];  // Function to generate mailto links  function generateMailtoLink(email) {  if (email.includes("@")) {  return `mailto:${email}`;  } else {  return null; // Return null for non-email entries  }  }  // JSX for the Footer component  return (  <footer className="text-center text-white backgroundMain lg:text-left">  <div className="mx-6 py-10 text-center md:text-left">  {/\* Grid layout for footer sections \*/}  <div className="grid-1 grid gap-8 md:grid-cols-2 lg:grid-cols-4">  {/\* SathiSahyogi information \*/}  <div className="">  <h6 className="mb-4 flex items-center justify-center font-semibold uppercase md:justify-start">  SathiSahyogi  </h6>  <p>  SathiSahyogi is a decentralized crowdfunding platform which eliminates the need for third parties. It also provides transparency on where your money is going.  </p>  </div>  {/\* Product list \*/}  <div className="">  <h6 className="mb-4 flex justify-center font-semibold uppercase md:justify-start">  Products  </h6>  {productList.map((el, i) => (  <p className="mb-4" key={i + 1}>  <a href="#!">{el}</a>  </p>  ))}  </div>  {/\* Useful links \*/}  <div className="">  <h6 className="mb-4 flex justify-center font-semibold uppercase md:justify-start">  Useful links  </h6>  {usefullLink.map((el, i) => (  <p className="mb-4" key={i + 1}>  <a href={el.startsWith("http") ? el : "#!"}>{el}</a>  </p>  ))}  </div>  {/\* Contact details \*/}  <div>  <h6 className="mb-4 flex justify-center font-semibold uppercase md:justify-start">  Contact  </h6>  {contactList.map((el, i) => (  <p className="mb-4" key={i + 1}>  {generateMailtoLink(el) ? (  <a href={generateMailtoLink(el)}>{el}</a>  ) : (  el  )}  </p>  ))}  </div>  </div>  </div>  {/\* Copyright information \*/}  <div className="backgroundMain p-6 text-center">  <span>(c)2024 Copyright: </span>  <a href="https://docs.google.com/document/d/1ubrUqHefL2Fd9ivFEaevwrchO3LCwwT4HSQSYmIdy5E/edit?usp=sharing" className="font-semibold" target="\_blank" rel="noopener noreferrer">  SathiSahyogi  </a>  </div>  </footer>  );  };  export default Footer; // Exporting Footer component |
| --- |

**Explanation:** The Footer component displays the footer section of the webpage, containing information about products, useful links, and contact details. It uses lists to store product names, contact emails, and useful links. The generateMailtoLink function generates mailto links for email addresses. The component renders these lists and links in a structured layout using JSX, providing users with easy access to relevant information and navigation options. Additionally, it includes copyright information at the bottom.

* + **Hero.jsx**

| import React, { useState } from "react";  // Hero component displaying the main section of the landing page with a campaign creation form  const Hero = ({ titleData, createCampaign }) => {  // State variables to manage form input fields  const [campaign, setCampaign] = useState({  title: "",  description: "",  amount: "",  deadline: ""  });  // Function to create a new campaign  const createNewCampaign = async (e) => {  e.preventDefault(); // Prevents default form submission behavior  try {  const data = await createCampaign(campaign); // Calls the createCampaign function passed as a prop  console.log("Campaign created successfully:", data);  } catch (error) {  console.error("Error creating campaign:", error);  }  };  // JSX for the Hero component  return (  <div className="relative">  <span className="coverLine"></span>  <img  src="https://images.pexels.com/photos/3228766/pexels-photo-3228766.jpeg?auto-compress&amp;cs=tinys rgb&amp; dpr=2&amp;h=750&amp;w=1260"  className="absolute inset-0 object-cover w-full h-full"  alt=""  />  <div className="relative bg-opacity-75 backgroundMain">  {/\* SVG wave decoration \*/}  <svg  className="absolute inset-x-0 bottom-0 text-white"  viewBox="0 0 1160 163"  >  <path  fill="currentColor"  d="M-164 13L-104 39.7C-44 66 76 120 196 141C316 162 436 152 556 119.7C676 88 796 34 916 13C1036 -8 1156 2 1216 7.7L1276 13V162.5H1216C1156 162.5 1036 162.5 916 162.5C796 162.5 676 162.5 556 162.5C436 162.5 316 162.5 196 162.5C76 162.5 -44 162.5 -104 162.5H-164V13Z"  />  </svg>  <div className="relative px-4 py-16 mx-auto overflow-hidden sm:max-w-xl md:max-w-full lg:max-w-screen-xl md:px-24 lg:px-8 lg:py-20">  <div className="flex flex-col items-center justify-between xl:flex-row">  {/\* Text content \*/}  <div className="w-full max-w-xl mb-12 xl:mb-0 xl:pr-16 xl:w-7/12">  <h2 className="max-w-lg mb-6 font-sans text-3xl font-bold tracking-tight text-white sm:text-5xl sm:leading-none">  SathiSahyogi <br className="hidden md:block" />  A Decentralized Crowdfunding Marketplace  </h2>  <p className="max-w-xl mb-4 text-base text-gray-200 md:text-lg">  This decentralized supply chain marketplace is an online platform that connects buyers and sellers within the supply chain ecosystem. It serves as a centralized hub where businesses can procure raw materials, components, finished products, or services from suppliers, manufacturers, distributors, and other partners involved in the production and distribution process.  </p>  <a  href="/"  aria-label=""  className="inline-flex items-center font-semibold tracking-wider transition-colors duration-200 text-teal-accent-400 hover:text-teal-accent-700 text-gray-200"  >  Learn More about SathiSahyogi  <svg  className="inline-block w-3 ml-2"  fill="currentColor"  viewBox="0 0 12 12"  >  <path d="M9.707,5.2931-5-5A1,1,0,0,0,3.293, 1.707L7.586,6,3.293,10.2" />  </svg>{" "}  </a>  </div>  {/\* Campaign creation form \*/}  <div className="w-full max-w-xl xl:px-8 xl:w-5/12">  <div className="bg-white rounded shadow-2xl p-7 sm:p-10">  <h3 className="mb-4 text-xl font-semibold sm:text-center sm:mb-6 sm:text-2xl">  Campaign  </h3>  <form>  {/\* Form inputs \*/}  {/\* Title \*/}  <div className="mb-1 sm:mb-2">  <label  htmlFor="firstName"  className="inline-block mb-1 font-medium"  >  Title  </label>  <input  onChange={(e) =>  setCampaign({  ...campaign,  title: e.target.value  })  }  placeholder="Title"  required  type="text"  className="flex-grow w-full h-12 px-4 mb-2 transition duration-200 bg-white border border-gray-300 rounded shadow-sm appearance-none focus:border-deep-purple-accent-400 focus:outline-none focus:shadow-outline"  id="firstName"  name="firstName"  />  </div>  {/\* Description \*/}  <div className="mb-1 sm:mb-2">  <label  htmlFor="lastName"  className="inline-block mb-1 font-medium"  >  Description  </label>  <input  onChange={(e) =>  setCampaign({  ...campaign,  description: e.target.value,  })  }  placeholder="Description"  required  type="text"  className="flex-grow w-full h-12 px-4 mb-2 transition duration-200 bg-white border border-gray-300 rounded shadow-sm appearance-none focus:border-deep-purple-accent-400 focus:outline-none focus:shadow-outline"  id="lastName"  name="lastName"  />  </div>  {/\* Target Amount \*/}  <div className="mb-1 sm:mb-2">  <label  htmlFor="email"  className="inline-block mb-1 font-medium"  >  Target Amount  </label>  <input  onChange={(e) =>  setCampaign({  ...campaign,  amount: e.target.value,  })  }  placeholder="Amount"  required  type="text"  className="flex-grow w-full h-12 px-4 mb-2 transition duration-200 bg-white border border-gray-300 rounded shadow-sm appearance-none focus:border-deep-purple-accent-400 focus:outline-none focus:shadow-outline"  id="email"  name="email"  />  </div>  {/\* Deadline \*/}  <div className="mb-1 sm:mb-2">  <label  htmlFor="deadline"  className="inline-block mb-1 font-medium"  >  Deadline  </label>  <input  onChange={(e) =>  setCampaign({  ...campaign,  deadline: e.target.value,  })  }  placeholder="Date"  required  type="date"  className="flex-grow w-full h-12 px-4 mb-2 transition duration-200 bg-white border border-gray-300 rounded shadow-sm focus:border-deep-purple-accent-400 appearance-none focus:outline-none focus:shadow-outline"  id="deadline"  name="deadline"  />  </div>  {/\* Submit button \*/}  <div className="mt-4 mb-2 sm:mb-4">  <button  onClick={(e) => createNewCampaign(e)}  type="submit"  className="inline-flex items-center justify-center w-full h-12 px-6 font-medium tracking-wide text-white transition duration-200 rounded shadow-md bg-deep-purple-accent-400 hover:bg-deep-purple-accent-700 focus:shadow-outline focus:outline-none newColor"  >  Create Campaign  </button>  </div>  {/\* Form instruction \*/}  <p className="text-xs text-gray-600 sm:text-sm">  Create your Campaign for raise funds  </p>  </form>  </div>  </div>  </div>  </div>  </div>  </div>  );  };  export default Hero; // Exporting Hero component |
| --- |

**Explanation:** The Hero component represents the main section of the landing page, featuring a form to create a new campaign. It utilizes state hooks to manage form input values. When the form is submitted, it triggers the createNewCampaign function, which calls the createCampaign function passed as a prop to initiate the campaign creation process. The component includes text content describing the platform, a form with input fields for campaign details, and a button to submit the form. It provides visual appeal with background images and SVG decorations.

* + **Logo.jsx**

| import React from 'react';  // Logo component displaying the SathiSahyogi logo  const Logo = ({ color }) => {  return (  <img  className={`w-8 ${color} text-teal-accent-400`} // Dynamically sets the width and color of the logo  src="https://github.com/viditkulsh/SathiSahyogi/assets/91754462/efe9fa2d-ddd8-4116-880c-ee6f483ded14" // Image source URL  alt="SathiSahyogi Logo" // Alternative text for accessibility  />  );  };  export default Logo; // Exporting Logo component |
| --- |

**Explanation:** The Logo component renders an image element displaying the SathiSahyogi logo. It accepts a prop color to dynamically set the color of the logo. The className attribute applies Tailwind CSS classes to control the width and color of the logo. The src attribute specifies the URL of the logo image. The alt attribute provides alternative text for the logo for accessibility purposes. The component exports the Logo component for use in other parts of the application.

* + **Menu.jsx**

| import React from 'react';  // Menu component displaying an SVG icon  const Menu = () => {  return (  <svg className="w-5 h-5 text-white" viewBox="0 0 24 24">  {/\* Three horizontal lines representing the menu icon \*/}  <path  fill="currentColor"  d="M23, 13H1c-0.6,0-1-0.4-1-1s0.4-1,1-1h22c0.6,0,1,0.4,1,1S23.6,13,23,13z"  />  <path  fill="currentColor"  d="M23,6H1C0.4,6,0,5.6,0,5c0-0.6,0.4-1,1-1h22c0.6,0,1,0.4,1,1C24,5.6,23.6,6,23,6z"  />  <path  fill="currentColor"  d="M23, 20H1c-0.6,0-1-0.4-1-1s0.4-1,1-1h22c0.6,0,1,0.4,1,1S23.6,20,23,20z"  />  </svg>  );  };  export default Menu; // Exporting Menu component |
| --- |

**Explanation:** The Menu component renders an SVG icon representing a menu. It consists of three horizontal lines stacked vertically. Each <path> element within the SVG defines a line segment of the icon. The fill attribute sets the color of the icon to the current text color. The viewBox attribute defines the coordinate system and aspect ratio of the SVG. The component exports the Menu component for use in other parts of the application

* + **NavBar.jsx**

| import React, { useState, useContext } from "react";  // Internal Imports  import { CrowdFundingContext } from "../Context/CrowdFunding"; // Importing CrowdFundingContext  import { Logo, Menu } from "../Components/index"; // Importing Logo and Menu components  // Navigation bar component  const NavBar = () => {  // State variables and context initialization  const { currentAccount, connectWallet } = useContext(CrowdFundingContext); // Using context for currentAccount and connectWallet  const [isMenuOpen, setIsMenuOpen] = useState(false); // State variable for menu toggle  const menuList = ["White Paper", "Project", "Donation"]; // List of menu items  // JSX rendering for navigation bar  return (  <div className="backgroundMain ">  <div className="px-4 py-5 mx-auto sm:max-w-xl md:max-w-full lg:max-w-screen-xl md:px-24 lg:px-8">  <div className="relative flex items-center justify-between">  <div className="flex items-center">  {/\* Logo and company name \*/}  <a  href="/"  aria-label="Company"  title="Company"  className="inline-flex items-center mr-8"  >  <Logo color="text-white" />  <span className="ml-2 text-xl font-bold tracking-wide text-gray-100 uppercase">  SathiSahyogi  </span>  </a>  {/\* Navigation links \*/}  <ul className="flex items-center hidden space-x-8 lg:flex">  {menuList.map((el, i) => (  <li key={i + 1}>  <a  href="/"  aria-label="Our product"  title="Our Product"  className="font-medium tracking-wide text-gray-100 transition-colors duration-200 hover:text-teal-accent-400"  >  {el}  </a>  </li>  ))}  </ul>  </div>  {/\* Wallet connection button \*/}  {!currentAccount && (  <ul className="flex items-center hidden space-x-8 lg:flex">  <li>  <button  onClick={() => connectWallet()}  className="inline-flex items-center justify-center h-12 px-6 font-medium tracking-wide text-green transition duration-200 rounded shadow-md bg-deep-purple-accent-400 hover:bg-deep-purple-accent-700 focus:shadow-outline focus:outline-none background"  aria-label="Connect Wallet"  title="Connect Wallet"  >  Connect Wallet  </button>  </li>  </ul>  )}  {/\* Menu button for mobile view \*/}  <div className="lg:hidden z-40">  <button  aria-label={isMenuOpen ? "Close Menu" : "Open Menu"}  title={isMenuOpen ? "Close Menu" : "Open Menu"}  className="p-2 -mt-2 -mr-2 transition duration-200 rounded hover:bg-gray-200 focus:bg-gray-200 focus:outline-none focus:shadow-outline"  onClick={() => setIsMenuOpen(!isMenuOpen)}  >  <Menu />  </button>  {/\* Mobile menu \*/}  {isMenuOpen && (  <div className= "absolute top-0 left-0 w-full">  <div className="p-5 bg-green-900 border rounded shadow-sm">  <div className="flex items-center justify-between mb-4">  <div>  <a  href="/"  area-label="SathiSahyogi"  title = "SathiSahyogi"  className="inline-flex items-center"  >  <Logo color="text-green-50" />  <span className="ml-2 text-xl font-bold tracking-wide text-green-50 uppercase">  SathiSahyogi  </span>  </a>  </div>  <div>  <button  aria-label="Close Menu"  title="Close Menu"  className="p-2 -mt-2 -mr-2 transition duration-200 rounded hover:bg-gray-200 focus:bg-gray-200 focus:outline-none focus:shadow-outline"  onClick={() => setIsMenuOpen(false)}  >  <Menu />  </button>  </div>  </div>  {/\* Mobile menu links \*/}  <nav>  <ul className="space-y-4">  {menuList.map((el, i) => (  <li key={i + 1}>  <a  href="/"  aria-label="Out Product"  title="Out Product"  className="font-medium tracking-wide text-green-100 transition-colors duration-200 hover:text-deep-purple-accent-400"  >  {el}  </a>  </li>  ))}  {/\* Wallet connection button for mobile view \*/}  <li>  <a  onClick={() => connectWallet()}  className="inline-flex items-center background justify-center w-full h-12 px-6 font-medium tracking-wide text-green-450 transition duration-200 rounded shawdow-md bg-deep-purple-accent-400 hover:bg-deep-purple-accent-700 focus:shawdow-outline focus:outline-none"  aria-label="Sign up"  title="Sign up"  >  Connect Wallet  </a>  </li>  </ul>  </nav>  </div>  </div>  )}  </div>  </div>  </div>  </div>  );  };  export default NavBar; // Exporting NavBar component |
| --- |

**Explanation:** This file defines the NavBar component, responsible for rendering the navigation bar of the application. It imports React and necessary internal components and context. Within the component, it utilizes state variables for managing the menu's open/close state and fetches the currentAccount and connectWallet function from the CrowdFundingContext. The JSX structure represents the navigation bar layout, including the company logo, navigation links, and a wallet connection button. The component also provides a responsive design for mobile view, toggling the menu's visibility. Lastly, it exports the NavBar component for use in other parts of the application.

* + **PopUp.jsx**

| import React, { useState, useEffect } from "react";  // Popup component for displaying donation modal  const PupUp = ({ setOpenModel, donate, donateFunction, getDonations }) => {  // State variables for amount and donation data  const [amount, setAmount] = useState("");  const [allDonationData, setAllDonationData] = useState([]);  // Function to create a donation  const createDonation = async () => {  try {  const data = await donateFunction(donate.pId, amount);  console.log(data);  } catch (error) {  console.log(error);  }  };  // useEffect hook to fetch donations when component mounts or when campaign ID changes  useEffect(() => {  const fetchDonations = async () => {  const donationData = await getDonations(donate.pId);  setAllDonationData(donationData);  };  fetchDonations();  }, [donate.pId, getDonations]);  // JSX for the donation modal  return (  <>  <div className="justify-center items-center flex overflow-x-hidden overflow-y-auto fixed inset-0 z-50 outline-none focus:outline-none">  <div className="relative w-auto my-6 mx-auto max-w-3xl">  {/\* Popup Content \*/}  <div className="border-0 rounded-lg shadow-lg relative flex flex-col w-full bg-white outline-none focus:outline-none">  {/\* Popup Header \*/}  <div className="flex items-start justify-between p-5 border-b border-solid border-slate-200 rounded-t">  <h3 className="text-3xl font-semibold">{donate.title}</h3>  <button  className="p-1 ml-auto bg-transparent border-0 text-black opacity-5 float-right text-3xl leading-none font-semibold outline-none focus:outline-none"  onClick={() => setOpenModel(false)}  >  <span className="bg-transparent text-black opacity-5 h-6 w-6 text-2xl block outline-none focus:outline-none">X</span>  </button>  </div>  {/\* Popup Body \*/}  <div className="relative p-6 flex-auto">  <p className="text-lg leading-relaxed">{donate.description}</p>  {/\* Input for donation amount \*/}  <input  onChange={(e) => setAmount(e.target.value)}  placeholder="Amount"  required  type="text"  className="flex-grow w-full h-12 px-4 mb-2 transition duration-200 bg-white border border-gray-300 rounded shadow-sm appearance-none focus:border-deep-purple-accent-400 focus:outline-none focus:shadow-outline"  id="amount"  name="amount"  />  {/\* Displaying donation data \*/}  {allDonationData.map((donation, i) => (  <p key={i} className="my-4 text-slate-500 text-lg leading-relaxed">  {i + 1}: {donation.donation} - {donation.donator.slice(0, 35)}  </p>  ))}  </div>  {/\* Popup Footer \*/}  <div className="flex items-center justify-end p-6 border-t border-solid border-slate-200 rounded-b">  {/\* Close button \*/}  <button  className="text-red-500 background-transparent font-bold uppercase px-6 py-2 text-sm outline-none focus:outline-none mr-1 mb-1 ease-linear transition-all duration-150"  type="button"  onClick={() => setOpenModel(false)}  >  Close  </button>  {/\* Donate button \*/}  <button  className="background text-white active:bg-emerald-600 font-bold uppercase text-sm px-6 py-3 rounded shadow hover:shadow-lg outline-none focus:outline-none mr-1 mb-1 ease-linear transition-all duration-150"  type="button"  onClick={() => createDonation()}  >  Donate  </button>  </div>  </div>  </div>  </div>  {/\* Background overlay \*/}  <div className="opacity-25 fixed inset-0 z-40 bg-black"></div>  </>  );  }; |
| --- |

**Explanation:** This component renders a popup modal for donating to a campaign. It consists of input fields for the donation amount, displays donation data, and provides buttons for donation submission and modal closure. useEffect fetches donations when the component mounts or when the campaign ID changes. The createDonation function triggers donation creation upon button click, utilizing the provided donation function.

* **Javascripts:**
  + **Crowdfunding.js**

| import React, { useState, useEffect } from "react";  import Web3Modal from "web3modal";  import { ethers } from "ethers";  // Internal Import  import { CrowdFundingABI, CrowdFundingAddress } from "./constants";  // Function to fetch Smart Contract instance  const fetchContract = (signerOrProvider) =>  new ethers.Contract(CrowdFundingAddress, CrowdFundingABI, signerOrProvider);  // Creating context for CrowdFunding functionality  export const CrowdFundingContext = React.createContext();  // Provider component for CrowdFunding context  export const CrowdFundingProvider = ({ children }) => {  // State variables  const titleData = "Crowd Funding Contract";  const [currentAccount, setCurrentAccount] = useState("");  const [openError, setOpenError] = useState(false);  const [error, setError] = useState("");  // Function to check if wallet is connected  const checkIfWalletConnected = async () => {  try {  if (!window.ethereum) {  setOpenError(true);  setError("Install MetaMask");  return;  }  const accounts = await window.ethereum.request({  method: "eth\_accounts",  });  if (accounts.length) {  setCurrentAccount(accounts[0]);  } else {  console.log("No Account Found");  }  } catch (error) {  console.log("Something went wrong while connecting to wallet");  }  };  // useEffect hook to check wallet connection on component mount  useEffect(() => {  checkIfWalletConnected();  }, []);  // Function to connect wallet  const connectWallet = async () => {  try {  if (!window.ethereum) {  console.log("Install MetaMask");  return;  }  const accounts = await window.ethereum.request({  method: "eth\_requestAccounts",  });    setCurrentAccount(accounts[0]);  } catch (error) {  console.log("Error while connecting to wallet");  }  };  // Function to create a new campaign  const createCampaign = async (campaign) => {  // Extracting campaign details  const { title, description, amount, deadline } = campaign;  try {  // Initializing Web3Modal to connect to wallet  const web3Modal = new Web3Modal();  const connection = await web3Modal.connect();  const provider = new ethers.providers.Web3Provider(connection);  const signer = provider.getSigner();  const contract = fetchContract(signer);  // Creating campaign transaction  const transaction = await contract.createCampaign(  currentAccount,  title,  description,  ethers.utils.parseUnits(amount, 18),  new Date(deadline).getTime()  );  await transaction.wait();  console.log("Contract call success", transaction);  } catch (error) {  console.log("Contract call failure", error);  }  };  // Other functions for getting campaigns, user campaigns, donations, and donating    return (  // Providing values to the context  <CrowdFundingContext.Provider  value={{  titleData,  currentAccount,  createCampaign,  // Other functions...  connectWallet,  }}  >  {children}  </CrowdFundingContext.Provider>  );  }; |
| --- |

**Explanation:** This file establishes a context and provider for CrowdFunding functionality in the React application. It utilizes ethers.js for Ethereum interactions. checkIfWalletConnected checks if MetaMask is installed and connected. connectWallet connects the wallet. createCampaign creates a new campaign. These functions use Web3Modal for wallet connection and interact with the CrowdFunding smart contract. The provider component wraps its children with the context, providing access to functions and state variables throughout the application.

* + **Constant.js**

| // Importing JSON data representing the CrowdFunding contract  import crowdFunding from "./CrowdFunding.json";  // Exporting constants for CrowdFunding contract address and ABI  export const CrowdFundingAddress = "0x5FbDB2315678afecb367f032d93F642f64180aa3"; // Address of the deployed CrowdFunding contract  export const CrowdFundingABI = crowdFunding.abi; // ABI (Application Binary Interface) of the CrowdFunding contract |
| --- |

**Explanation:** This JavaScript file imports JSON data representing the CrowdFunding contract and exports two constants: CrowdFundingAddress containing the address of the deployed CrowdFunding contract and CrowdFundingABI containing the ABI (Application Binary Interface) of the contract. These constants can be imported and used in other parts of the application, such as for interacting with the deployed contract on the Ethereum blockchain.

* + **Index.js**

| // Importing components for export  import Card from "./Card"; // Component for displaying campaign cards  import Footer from "./Footer"; // Footer component for the application  import Hero from "./Hero"; // Hero component for showcasing main content or image  import NavBar from "./NavBar"; // Navigation bar component for navigation  import Menu from "./Menu"; // Menu component for displaying menu options  import Logo from "./Logo"; // Logo component for displaying the logo of the application  import PopUp from "./PopUp"; // Pop-up component for displaying notifications or messages  // Exporting all components for use in other parts of the application  export { Footer, NavBar, Logo, Menu, Card, PopUp, Hero }; |
| --- |

**Explanation:** This JavaScript file exports various components used in the application, including Card, Footer, Hero, NavBar, Menu, Logo, and PopUp. These components serve different purposes such as displaying campaign cards, rendering the footer and navigation bar, showcasing main content or images, displaying menu options, showing the logo, and handling pop-up notifications or messages. Exporting them allows other parts of the application to import and utilize these components easily.

* + **\_app.js**

| /\* Importing global styles \*/  import "@/styles/globals.css";  /\* Internal imports for NavBar, Footer, and CrowdFundingProvider \*/  import { NavBar, Footer } from "../Components";  import { CrowdFundingProvider } from '../Context/CrowdFunding';  /\* Default function for the main App component \*/  export default function App({ Component, pageProps }) {  return (  <>  {/\* Wrapping the main component with CrowdFundingProvider \*/}  <CrowdFundingProvider>  {/\* Rendering NavBar component \*/}  <NavBar />  {/\* Rendering the main component with its props \*/}  <Component {...pageProps} />  {/\* Rendering Footer component \*/}  <Footer />  </CrowdFundingProvider>  </>  );  } |
| --- |

**Explanation:** This code is the main entry point of the React application. It imports global styles, NavBar, Footer, and CrowdFundingProvider components. The App function renders these components, with the main component passed as a prop along with its pageProps. It ensures that the entire application is wrapped within the CrowdFundingProvider, enabling access to crowdfunding context throughout the application.

* + **Deploy.js**

| // Importing Hardhat Runtime Environment (hre) for Ethereum development  const hre = require("hardhat");  // Asynchronous function for deploying CrowdFunding contract  async function main() {  // Get the ContractFactory of the CrowdFunding contract  const CrowdFunding = await hre.ethers.getContractFactory("CrowdFunding");    // Deploying the CrowdFunding contract  const crowdFunding = await CrowdFunding.deploy();  // Waiting for the deployment transaction to be mined  await crowdFunding.waitForDeployment();  // Logging the address where CrowdFunding contract is deployed  console.log(`CrowdFunding deployed to ${crowdFunding.target}`);  }  // Executing the main function and handling errors  main().catch((error) => {  console.error(error);  process.exitCode = 1;  }); |
| --- |

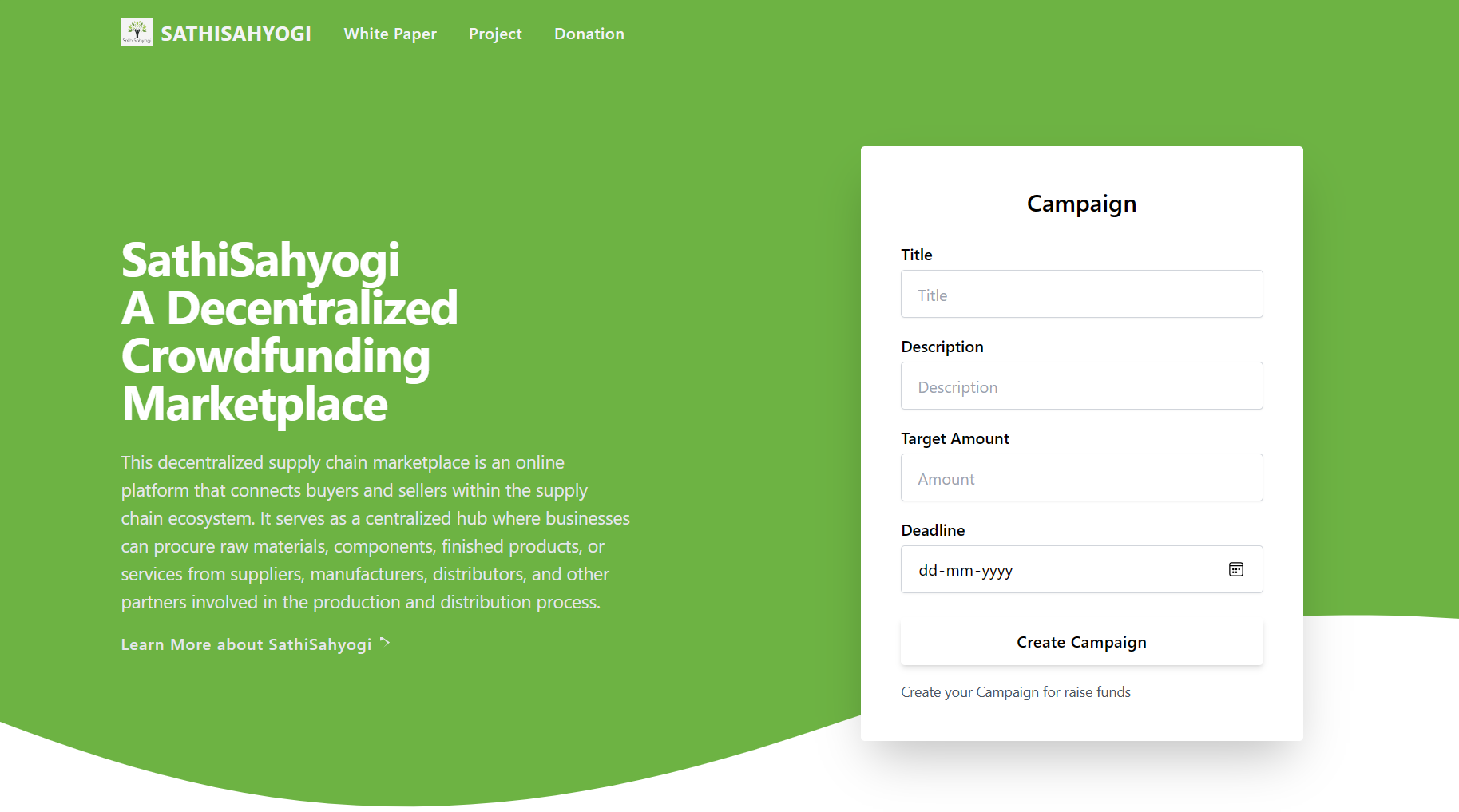
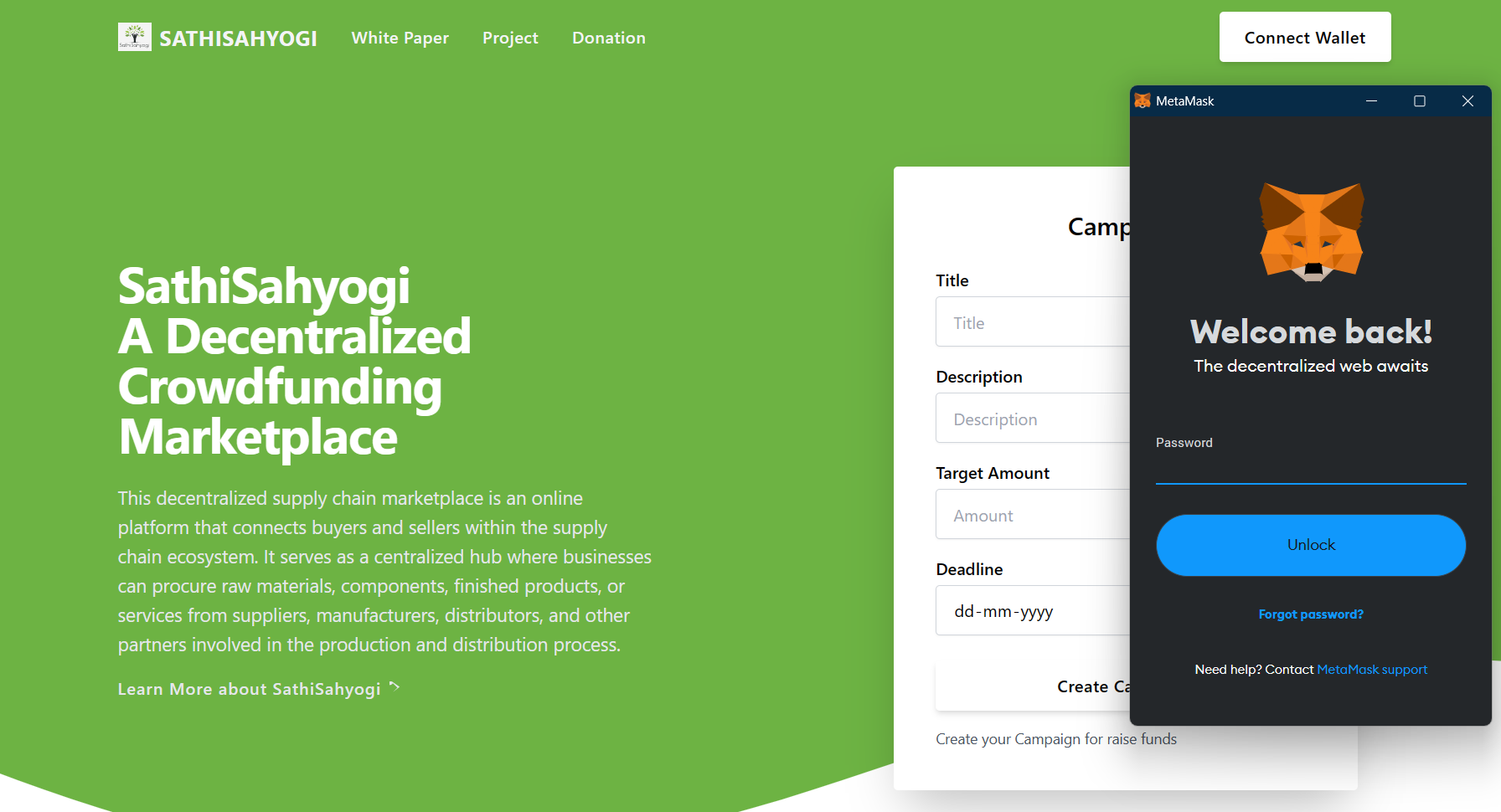
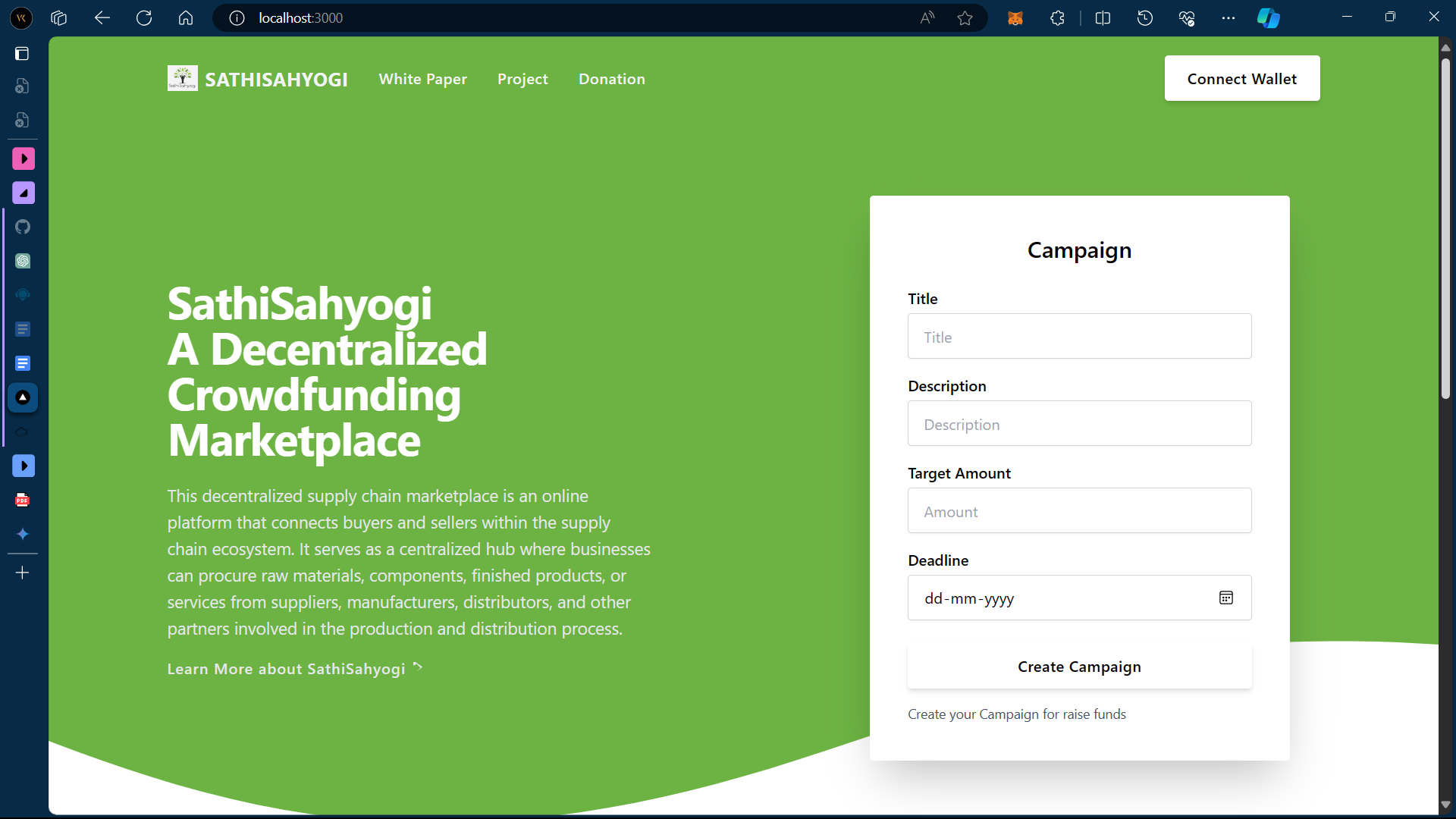
**Explanation:** This script deploys the CrowdFunding smart contract using Hardhat. It first imports the Hardhat Runtime Environment (hre) and then asynchronously deploys the CrowdFunding contract. After deployment, it waits for the deployment transaction to be mined. Finally, it logs the address where the CrowdFunding contract is deployed. Any errors during deployment are caught and logged, and the script exits with a process exit code of 1 in case of an error.

* **Tailwind Css:**
  + **Globals.css**

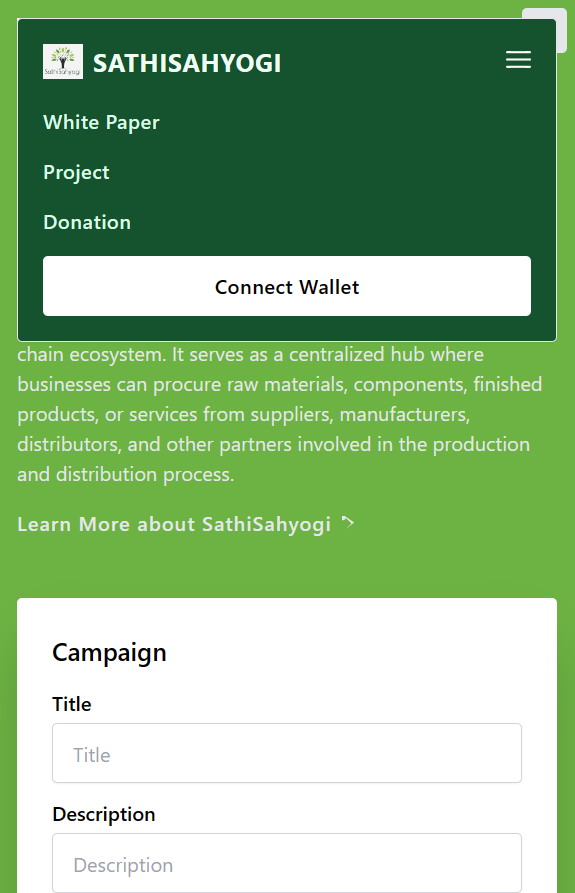
| **/\* Applying Tailwind CSS directives for styling \*/**  **@tailwind base;**  **@tailwind components;**  **@tailwind utilities;**  **/\* Styling class for setting background color to white \*/**  **.background {**  **background-color: #ffffff; /\* Updated to white background color of the image \*/**  **}**  **/\* Styling class for setting background color to green \*/**  **.backgroundMain {**  **background-color: #6db343; /\* Updated to green color of tree leaves in the image \*/**  **}**  **/\* Styling class for setting text color to black \*/**  **.newColor {**  **color: #000000; /\* Updated to black text color in the image \*/**  **}**  **/\* Styling class for a cover line element \*/**  **.coverLine {**  **width: 100%;**  **height: 2rem;**  **background-color: #ffffff; /\* Kept as white or can be changed depending on where it's used \*/**  **position: absolute;**  **bottom: -1.9rem;**  **z-index: 3;**  **}** |
| --- |

**Explanation:** This CSS file applies Tailwind CSS directives for styling. It defines classes to set background colors to white and green, change text color to black, and style a cover line element with specific dimensions and positioning. These classes can be applied to HTML elements to achieve consistent styling throughout the application, adhering to the provided color codes and layout properties.

**5.2 Results**

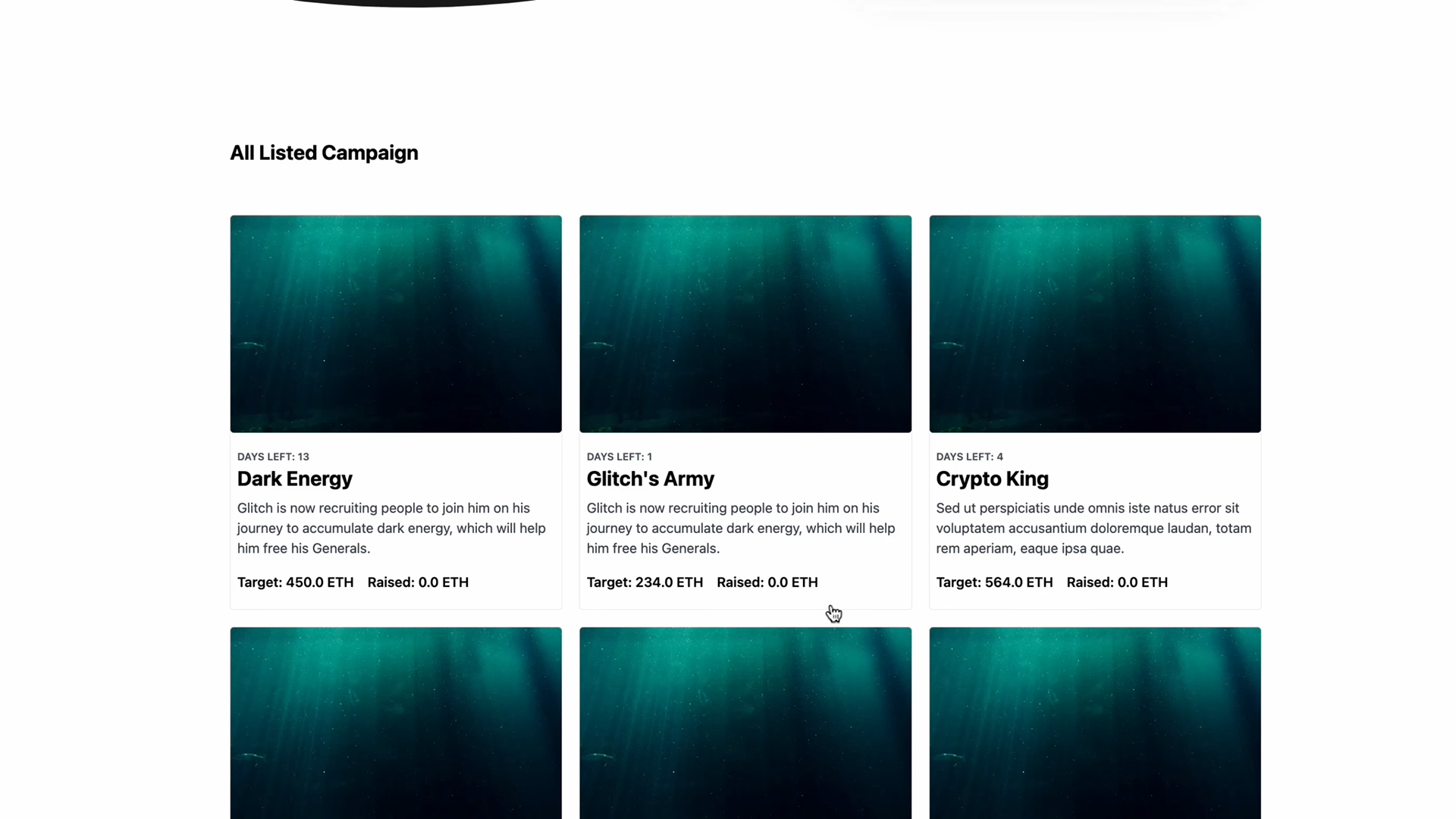
****

**Figure 4**

****

**Figure 5**

Figure 4 shows the homepage of SathiSahyogi on a laptop screen and Figure 5 shows how the homepage SathiSahyogi will look on small screens, Home screen is made using all the files in the components. Connect wallet button has been integrated using crowdfunding.js file.

****Figure 6

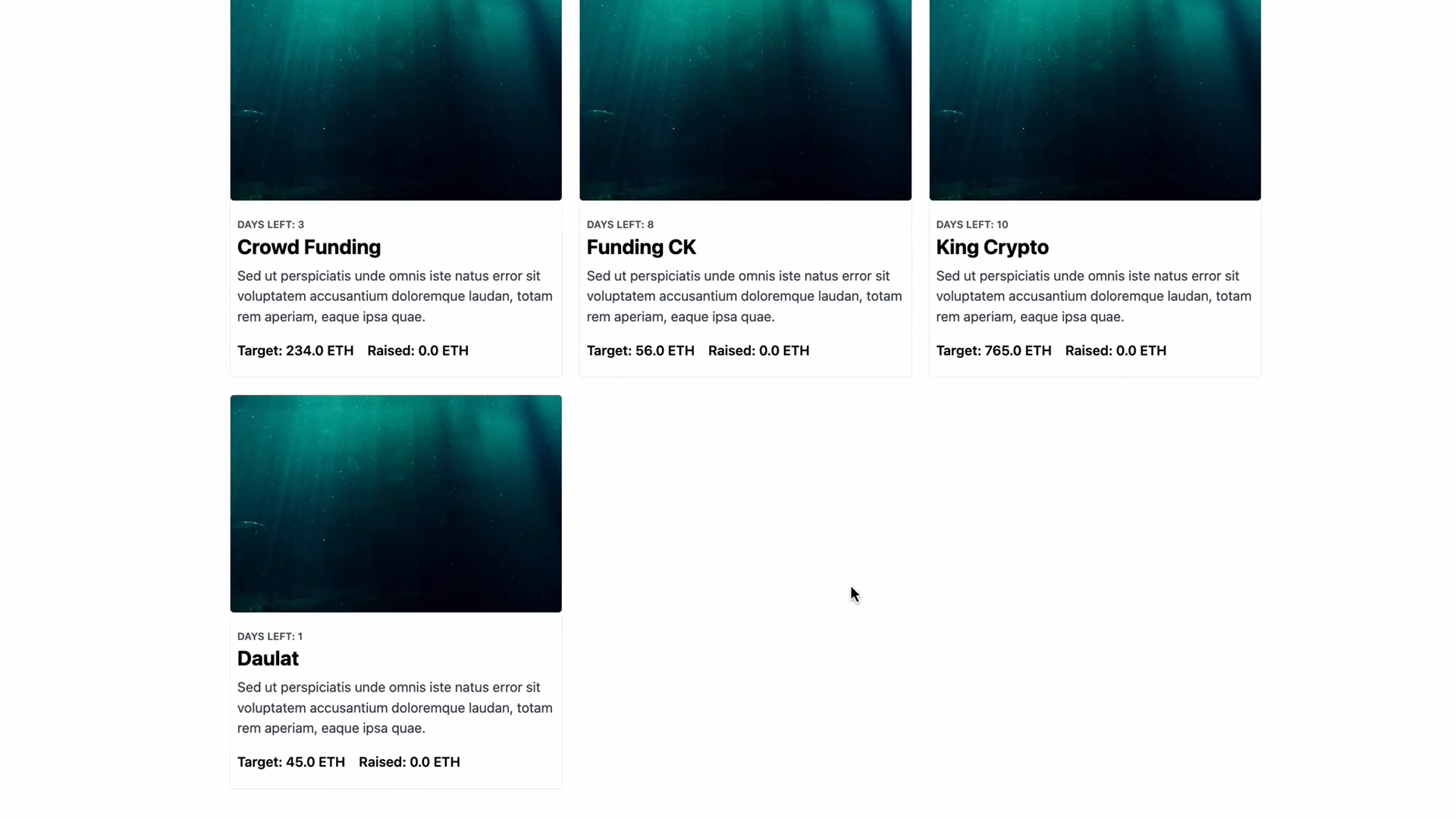
****

Figure 7

Figure 6 and Figure 7 are the listed campaigns of my crowdfunding platform which can be displayed specifically using card.jsx and popup.jsx.

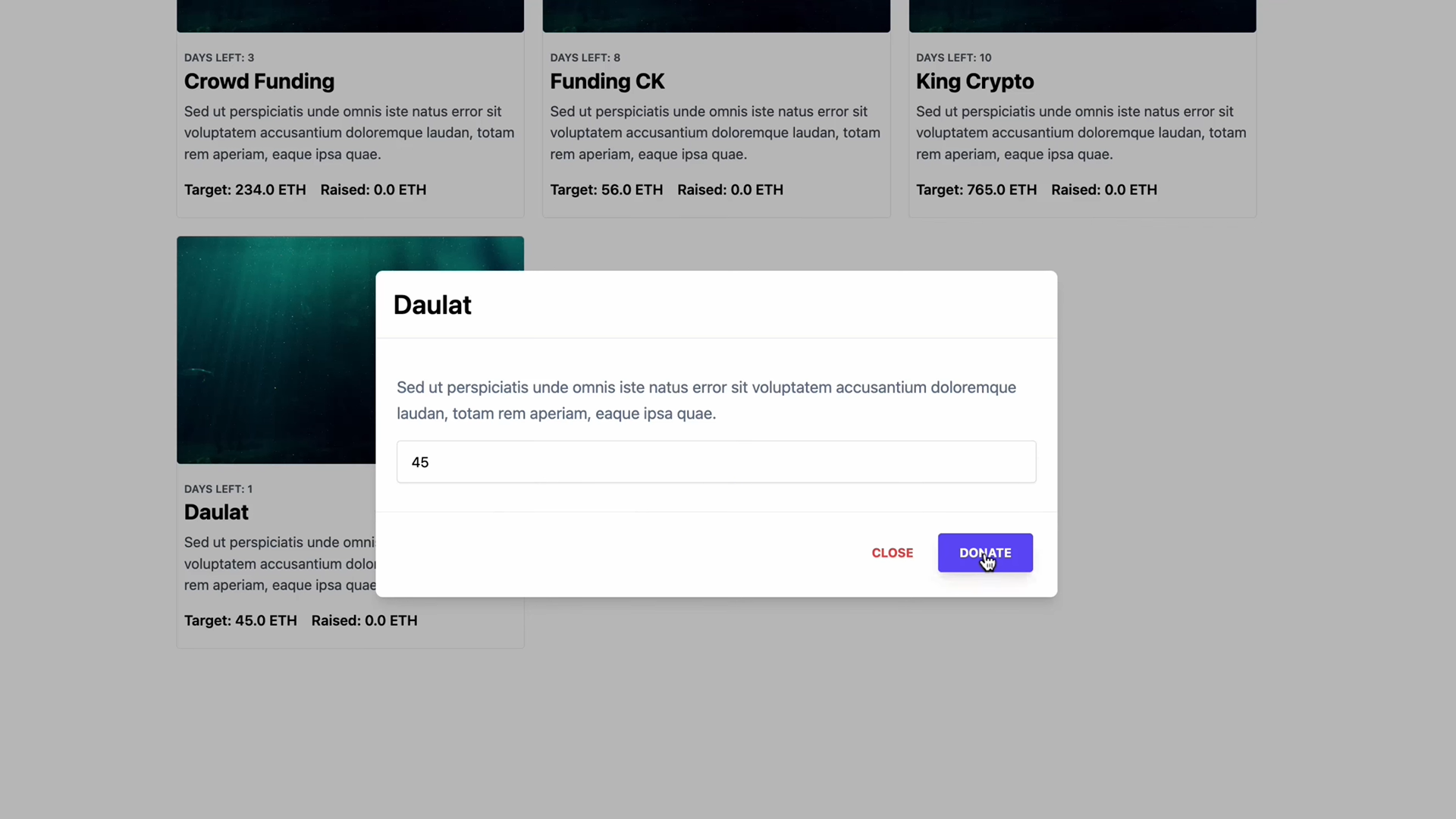


Figure 8

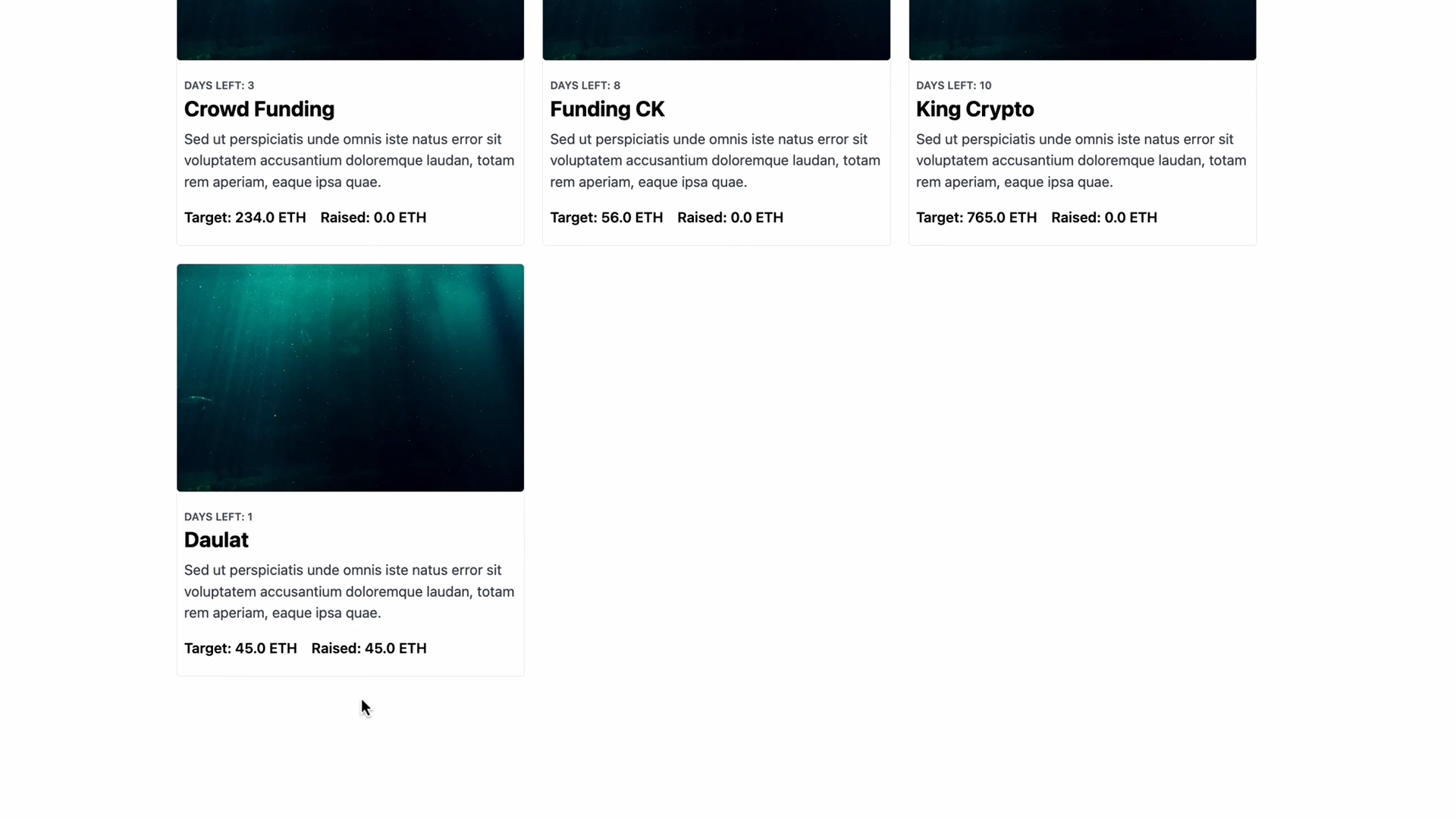


Figure 9

Figure 8 and figure 9 shows how campaigns can get funds and how it will be updated upon the campaign card.

**CONCLUSION**

In conclusion, SathiSahyogi represents a groundbreaking initiative in the realm of charity crowdfunding. By harnessing the power of blockchain technology, this platform addresses critical issues such as transparency, trust, and efficiency that have long plagued traditional fundraising methods. The centralized nature of many existing platforms often leads to a lack of transparency regarding how donations are utilized, which can erode public trust and hinder the success of genuine charitable endeavors. SathiSahyogi seeks to remedy this by providing a decentralized platform where every transaction is recorded and verified on a transparent ledger, thus ensuring complete visibility into the allocation and usage of funds.

Moreover, SathiSahyogi democratizes the process of charitable giving by making it accessible to a diverse range of individuals and organizations. Through its user-friendly interface and streamlined processes, SathiSahyogi empowers users to contribute to causes they believe in with ease and confidence. Whether it's supporting local community projects, humanitarian initiatives, or environmental conservation efforts, SathiSahyogi offers a platform where donors can make a tangible impact on the causes that matter most to them.

Furthermore, the use of smart contracts on the SathiSahyogi platform enhances security and accountability by automating the execution of transactions according to predefined rules. This not only reduces the risk of fraud or mismanagement but also eliminates the need for intermediaries, thereby ensuring that a larger portion of donations reaches the intended beneficiaries.

Looking ahead, SathiSahyogi has the potential to revolutionize the landscape of charity fundraising by fostering a culture of transparency, accountability, and community engagement. As the platform continues to evolve and expand its reach, it has the opportunity to make a significant and lasting impact on society by facilitating meaningful collaborations between donors, charitable organizations, and beneficiaries. In essence, SathiSahyogi represents a beacon of hope in the philanthropic world, where transparency, efficiency, and compassion converge to create a brighter future for all.

**FUTURE SCOPE**

Moving forward, SathiSahyogi holds immense potential for further development and expansion in the realm of charity crowdfunding. Here are some avenues for future growth and enhancement:

1. **Global Outreach**: SathiSahyogi can expand its reach beyond geographical boundaries to cater to charitable causes worldwide. By facilitating international transactions and collaborations, the platform can support a diverse range of initiatives and make a global impact on pressing issues such as poverty alleviation, healthcare access, and environmental sustainability.
2. **Integration of Emerging Technologies**: As technology continues to advance, SathiSahyogi can explore the integration of emerging technologies such as artificial intelligence (AI), Internet of Things (IoT), and decentralized finance (DeFi) to enhance its functionalities and offer innovative solutions for fundraising and impact assessment.
3. **Community Engagement**: The platform can foster greater community engagement by incorporating features such as user forums, live events, and gamification elements. By encouraging active participation and collaboration among donors, beneficiaries, and charitable organizations, SathiSahyogi can build a vibrant and supportive community committed to driving positive change.
4. **Enhanced Transparency**: SathiSahyogi can further enhance transparency by implementing advanced reporting and auditing mechanisms powered by blockchain technology. By providing real-time updates on fundraising progress, project milestones, and fund utilization, the platform can instill greater confidence and trust among donors and stakeholders.
5. **Partnerships and Collaborations**: Collaborating with other organizations, including nonprofits, corporations, and government agencies, can amplify the impact of SathiSahyogi and facilitate synergistic efforts towards common goals. Strategic partnerships can enable access to additional resources, expertise, and networks, thereby expanding the scope and effectiveness of charitable initiatives supported by the platform.
6. **Education and Advocacy**: SathiSahyogi can play a pivotal role in raising awareness about pressing social issues and promoting philanthropic values through educational campaigns, outreach programs, and advocacy initiatives. By empowering individuals with knowledge and inspiring action, the platform can catalyze positive change at both local and global levels.

**REFERENCES**

1. [*Blockchain in Charity & Philanthropy: Benefits and Future (pixelplex.io)*](https://pixelplex.io/blog/blockchain-charity-projects-and-their-philanthropic-benefits/)
2. [*How to build a blockchain charity or crowdsourcing platform - LogRocket Blog*](https://blog.logrocket.com/build-blockchain-charity-crowdsourcing-platform/)
3. *S. Saranya, S. P. Muvvala, V. Chauhan and R. Satwik, "Crowdfunding Charity Platform Using Blockchain," 2022 International Conference on Inventive Computation Technologies (ICICT), Nepal, 2022, pp. 1-7, doi: 10.1109/ICICT54344.2022.9850562.*
4. *Baber, Hasnan. (2020). Blockchain-Based Crowdfunding. 10.1007/978-981-15-1137-0\_6.*
5. [Blockchain Project Crowdfunding | Smart contract](https://www.youtube.com/watch?v=eTSieBUeq9I&pp=ygUtY2hhcml0eSBjcm93ZGZ1bmRpbmcgdXNpbmcgYmxvY2tjaGFpbiBwcm9qZWN0)
6. [Blockchain Charity Application | Ethereum | Blockchain Project](https://www.youtube.com/watch?v=2I5vliaVJxY&pp=ygUtY2hhcml0eSBjcm93ZGZ1bmRpbmcgdXNpbmcgYmxvY2tjaGFpbiBwcm9qZWN0)
7. [*Creating a CrowdFunding Application with the Algorand Blockchain | Algorand Developer Portal*](https://developer.algorand.org/solutions/creating-crowdfunding-application-algorand-blockchain/)
8. *A. Singh, R. Rajak, H. Mistry and P. Raut, "Aid, Charity and Donation Tracking System Using Blockchain," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 457-462, doi: 10.1109/ICOEI48184.2020.9143001.*
9. *Rangone, A., Busolli, L. Managing charity 4.0 with Blockchain: a case study at the time of Covid-19. Int Rev Public Nonprofit Mark* ***18****, 491–521 (2021). https://doi.org/10.1007/s12208-021-00281-8*
10. [*How Blockchain Is Revolutionizing Crowdfunding | OpenMind (bbvaopenmind.com)*](https://www.bbvaopenmind.com/en/economy/business/how-blockchain-is-revolutionizing-crowdfunding/)
11. [*Charity Digital - Topics - Blockchain for charity fundraising: behind the buzzword*](https://charitydigital.org.uk/topics/topics/blockchain-for-charity-fundraising-behind-the-buzzword-5882)
12. [*Kickstarter - Wikipedia*](https://en.wikipedia.org/wiki/Kickstarter)
13. [*Kickstarter*](https://www.kickstarter.com/)
14. [*https://www.investopedia.com/terms/s/smart-contracts.asp*](https://www.investopedia.com/terms/s/smart-contracts.asp)
15. [*Initial Coin Offerings: The Ethereum ICO Boom | Gemini*](https://www.gemini.com/cryptopedia/initial-coin-offering-explained-ethereum-ico)
16. [*Ecosystem · Filecoin Foundation*](https://www.fil.org/ecosystem/)
17. [Home | Tezos](https://tezos.com/)
18. [Evolving Fundraising: 9 Takeaways from the 2023 Philanthropic Landscape Report - Blackbaud Giving Fund](https://blackbaudgivingfund.org/blog/evolving-fundraising-9-takeaways-from-the-2023-philanthropic-landscape-report/)