Department of Computer Science & Engineering

School of Engineering and Technology

HEMWATI NANDAN BAHUGUNA GARHWAL UNIVERSITY

SRINAGAR, UTTRAKHAND, 246174

SESSION: 2022-23

MINI PROJECT REPORT
ON

"Help2gether: Crowdfunding Platform based on Blockchain"

SUBMITTED TO

DR .PREM NATH

Associate Professor

Dept. Of CSE

SUBMITTED BY

ADITYA KUMAR SINGH

B. TECH (CSE)

20134501012

IVth Semester

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We wish to thank the Institute and the Department of Computer Science and Engineering, for giving us the opportunity to work on this project.
We would like to thank Dr. Premnath Sir, who mentored us and guided us through the entire process.
Secondly, I would like to thank my parents who patiently helped me as I went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.
Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.
Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.
Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time
This project has been a learning experience through which we got to know about Blockchain and the processes regarding crowdfunding, and allowed us to get hands-on experience with building a real product.

Aditya Kumar Singh
Computer Science and Engineering (4th Sem)

DECLARATION

I, Aditya Kumar Singh bearing the roll no 20134501012 , student of Computer Science and Engineering Department at Hemwati Nandan Bahuguna Garhwal University (A Central University), Srinagar (Garhwal), Uttarakhand, submit this project report entitled "Help2gether: Crowdfunding Platform based on Blockchain" to Computer Science and Engineering Department, Hemvati Nandan Bahuguna Garhwal University, for the award of the Bachelors of Technology degree in Computer Science & Engineering and declaring that the work done is genuine and produced under the guidance of Dr. Prem Nath Department of Computer Science and Engineering, Hemwati Nandan Bahuguna Garhwal University. I further declare that the reported work in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or university.

Date:05-06-2022
Place: Srinagar

Aditya Kumar Singh
Roll No.- 20134501012

CERTIFICATE

This is to certify that, this project report titled "Help2gether: Crowdfunding Platform based on Blockchain" submitted by Aditya Kumar Singh bearing roll no 20134501012 is bonafide record of the work carried out by him in partial fulfilment for the requirement of the award of Bachelor of Technology in Computer Science and Engineering degree from Hemwati Nandan Bahuguna Garhwal University (A Central University) at Srinagar (Garhwal), Uttarakhand.

Dr. Prem Nath
Department of Computer Science & Engineering
Hemvati Nandan Bahuguna Garhwal University
(A Central University) Srinagar (Garhwal),
Uttarakhand

ABSTRACT

There are lots of developments in the fields of science and technology but even in these days very important agreements related to properties, Crowdfunding are made on paper which is not at all safe as they can be duplicated or forgery can be done. Even if the agreements are digital, they can be hacked if no security surrounds them. But if agreements, applications are made using smart contracts the security will be increased by a large scale. In normal crowd funding the entire amount is directly in the hands of manager and also the amount used by manager doesn't involve investor's opinion. So, there is no security ensured in this case. The proposed methodology creates a smart contract in such a way that the entire funding amount will be inside the smart contract and when the manager wants to make use of funding amount and buy something from a vendor, he/she has to make a spending request and then investors have to vote on that request. If the contract gets majority amount of votes the amount will be automatically sent to the vendor. In this way process of crowd funding can be made better and trustworthy.

Keywords: Blockchain, Smart Contracts, Crowdfunding

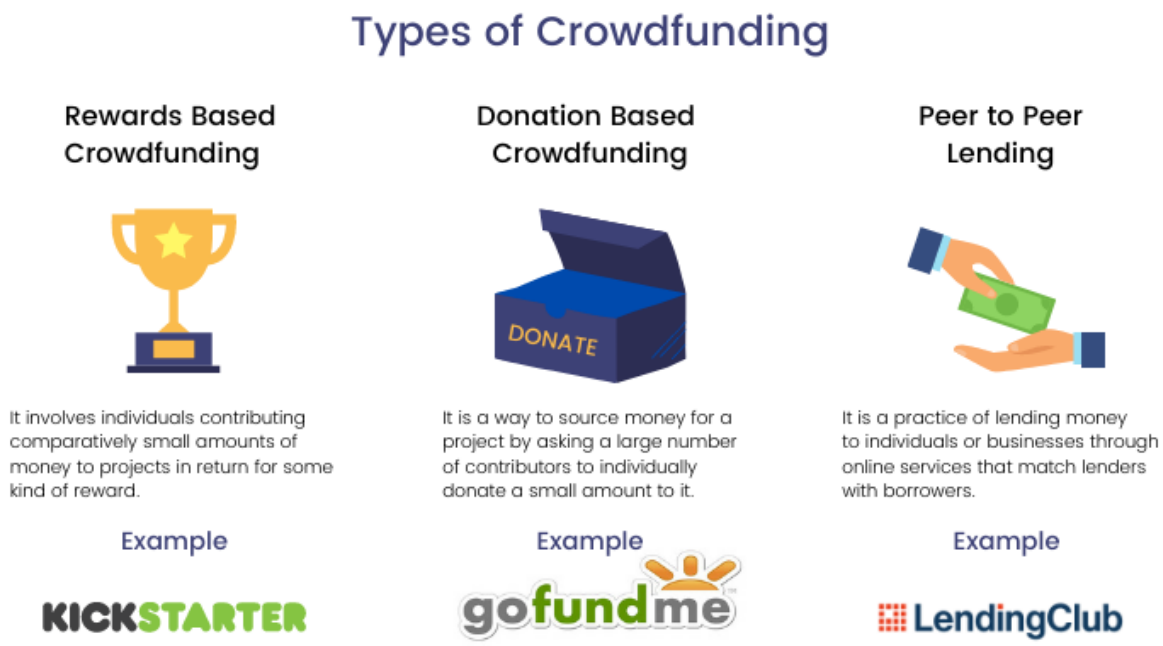
INTRODUCTION OF CROWD FUNDING

Crowdfunding is a way to raise funds for a specific cause or project by asking a large number of people to donate money, usually in small amounts, and usually during a relatively short period of time, such as a few months. Crowdfunding is done online, often with social networks, which make it easy for supporters to share a cause or project cause with their social networks.

Organizations, businesses, and individuals alike can use crowdfunding for any type of project, for example: charitable cause, creative project, business startup, school tuition, or personal expenses.

There are 2 main models or types of crowdfunding:

- **Donation-based funding**, where donors contribute to a total amount for a new project. Often promised return is the product or service that will be developed with the revenue brought in by the crowdfunding campaign. For charitable projects whose ultimate beneficiary is not the donor, there may be some other perk or reward for funders.
- **Investment crowdfunding**, where businesses seeking capital sell ownership stakes online in the form of equity or debt. In this model, individuals who fund become owners or shareholders and have a potential for financial return, unlike in the donation model. This became possible when Title II of the JOBS Act went into effect in September 2013 for accredited investors. Nonprofits generally cannot utilize equity markets.



Some of the best-known crowdfunding sites are GoFundMe, Kiva, Kickstarter, and Indiegogo, but hundreds, if not thousands, of sites exist. Most sites have these characteristics:

- Let you set up a page to describe, promote, and post updates about your project, especially with video
- Accept online donations
- Are easily and highly shareable on social networks
- Specify the kinds of projects or campaigns that can be on the site (e.g., creative/artistic; entrepreneurial; personal needs)
- Charge a percentage of funds raised plus fee per transaction

Some sites have an "all-or-nothing" policy, meaning you have to reach your monetary goal to get any of the funds. Thus, if you use a crowdfunding site, read and understand its FAQs (frequently asked questions), guidelines, terms, and conditions before signing up.

INTRODUCTION Of BLOCKCHAIN

In the simplest terms, Blockchain can be described as a data structure that holds transactional records and while ensuring **security, transparency, and decentralization**. You can also think of it as a chain or records stored in the forms of blocks which are controlled by no single authority. A blockchain is a distributed ledger that is completely open to any and everyone on the network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.

Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, **the data stored on the blockchain is tamper-proof and cannot be changed.**

Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

In order to understand blockchain better, consider an example where you are looking for an option to send some money to your friend who lives in a different location. A general option that you can normally use can be a bank or via a payment transfer application like **PayPal or Paytm**. This option involves third parties in order to process the transaction due to which an extra amount of your money is deducted as transferring fee. Moreover, in cases like these, you cannot ensure the security of your money as it is highly possible that a hacker might disrupt the network and steal your money. In both the cases, it is the customer who suffers. This is where Blockchain comes in.

Instead of using a bank for transferring money, if we use a blockchain in such cases, the process becomes much easier and secure. There is no extra fee involved as the funds are directly processed by you thus, eliminating the need for a third party. Moreover, the **blockchain database is decentralized** and is not limited to any single location meaning that all the information and records kept on **the blockchain are public and decentralized**. Since the information is not stored in a single place, there's no chance of corruption of the information by any hacker.

PROBLEM STATEMENT

Crowdfunding is one of the most popular ways to raise funds for any project, cause or for helping any individual in need. With the onset of Covid we have seen a rise in Crowdfunding activities across the globe which includes small campaigns to help people get oxygen and medical help to large funds such as PM Cares.

The major problems with the Current Crowdfunding Platforms that we wanted to solve were:

- **Security**

As the funds become larger, they need to be heavily secure, although stringent measures such as symmetric encryption are in place to make e-payment safe and secure, it is still vulnerable to hacking. Blockchain — which has never been compromised yet — can provide that level of security.

- **Crowdfunding Platform Face the High Risks of Fraud**

Usually, crowdfunding apps have limited security measures, so investors risk their money by supporting startups. Fraudulent actions may include false claims

about reached milestones or even released products, and in some cases, lead to substantial monetary losses. As soon as a startup gets needed financing, it can disappear from the platform with.

- **Intellectual Property Isn't Protected on Crowdfunding Platforms**

Crowdfunding apps don't protect intellectual property at a necessary level. It's possible that a bigger company noticeshan idea of a small startup and, using its own resources, can implement it faster and even in better quality. It puts creators in a rush, forcing them to either spend more money to patent their idea or to be the first one who brings it to life.

- **Transparency and Anti-Fraud**

We have seen, and continue to see a lot of crowdfunding scams happening around. There is no way to see where the funds are being used. We wanted to make the entire flow of funds transparent at every stage, so that there is no possibility of the money being misused.

- **Global contribution**

With some of the platforms being country specific, it becomes hard for people from other countries to contribute to various campaigns. Using blockchain anyone in the world can contribute to the campaign. Transactions are quick and convenient.

- **The burden of marketing and advertising**

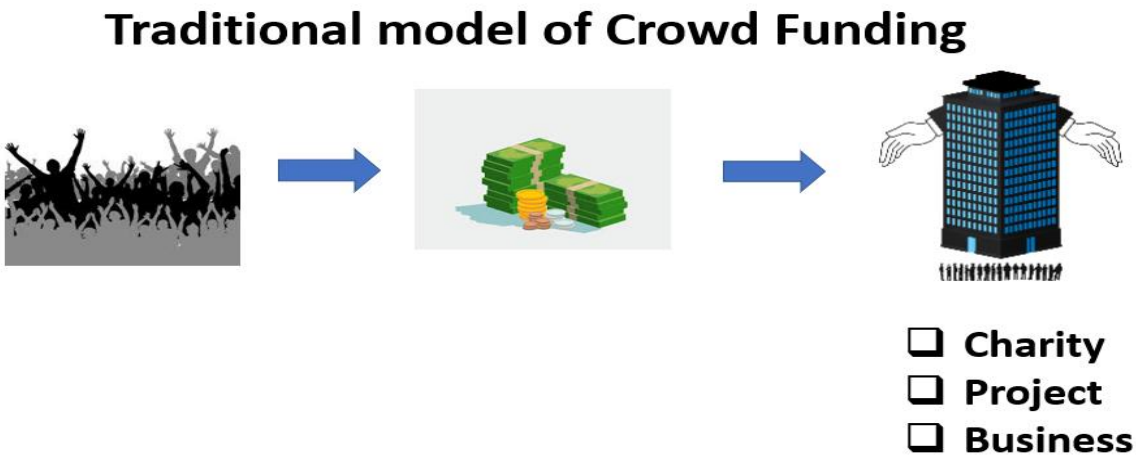
Making good ideas for the project under the crowdfunding platform does not provide a guarantee of success. To attract new people to the project, you should know how to make the crowdfunding page more visible. For advertising, marketing, branding, and strategies like search engine optimization for the projects is a practical requirement. However, it takes time from the actual invention or business but it is helpful to make projects more popular.

❖ The platform cash-grab

Every crowdfunding platform is dependent on one or more devoted crowdfunding platforms. These platforms do not provide free opportunities for people to join the projects. They take enough fees from people for joining the projects under the crowdfunding. However, the inventors who have a critical level of funding of projects can be a major point on this side.

Crowdfunding platform in a centralized system

In this type of crowdfunding platform, there is a need for a trusty relationship between the company and the crowd. The company expects that their fund gets paid on the basis of project progress and whenever some event completed then, they will get paid by the crowd. Similarly, the supporters expect that their fund is going to the right project or if the project gets crashed in between then, they will get back their fund from the company.



traditional model of crowd funding is very simple any company or NGO or any organization is start collecting fund from crowd for specific purpose like charity, project, business.

The major problems with this Current Crowdfunding model are: -

- **Security**
- **Crowdfunding Platform Face the High Risks of Fraud**
- **Intellectual Property Isn't Protected on Crowdfunding Platforms**
- **Transparency and Anti-Fraud**
- **Global contribution**

OUR OBJECTIVE

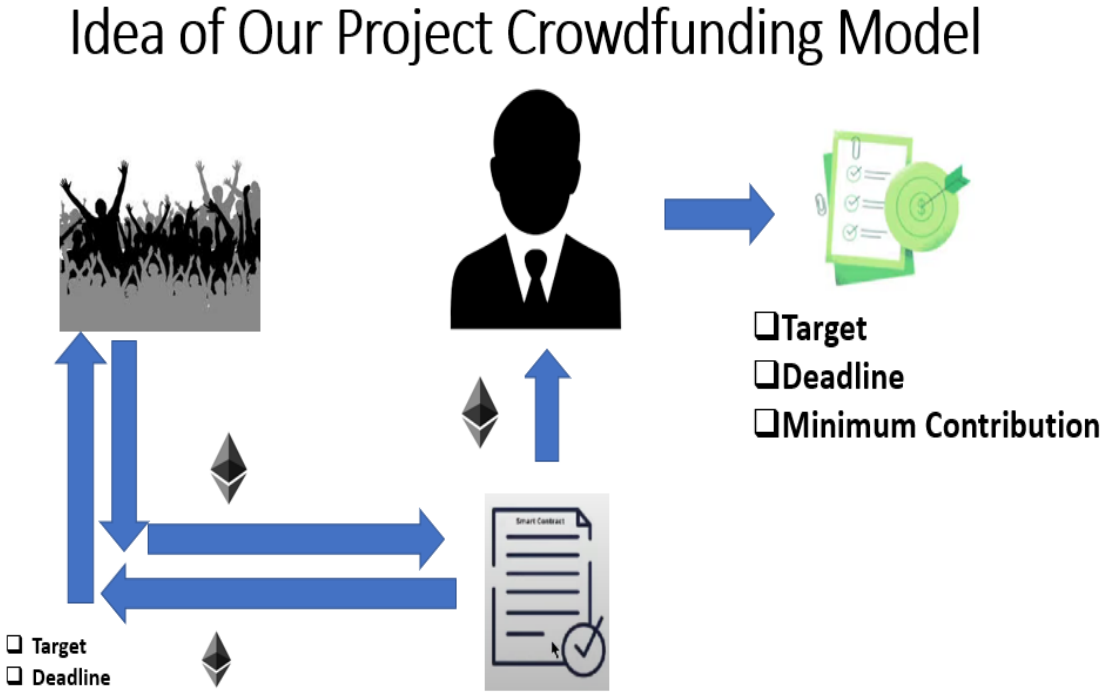
Crowdfunding over the years have helped people but has also seen heavy frauds in the name of Crowdfunding. With **Help2gether** we want to bring transparency to the process of crowdfunding and build trust among people to contribute to all the causes.

SCOPE OF THIS PROJECT

With **Help2gether** I aim to make the crowdfunding process transparent, anti-fraudulent and secure.

OUR SOLUTION

- Any web-based application is a centralized application which means that anything we do on the platform is managed by a server which is owned by a single company.
- I propose a Decentralized Application powered by Ethereum Blockchain, where all the information about campaigns, contributions, withdrawal requests and funds are kept on a Blockchain Network, visible to all and decentralized. This means the funds and transactions are visible to and stored at every node on the blockchain, and prevents the data from being stored in a centralized server, single location.
- Hence not letting the money get into the hands of anyone and eliminating every possibility of it getting misused — an elegant and logical solution to the problem in hand.



Idea of project is very simple and effective you know blockchain is disruptive technology and disruptive technology means completely change the ecosystem of traditional model so our idea is making web app using blockchain. because Blockchain is a distributed immutable ledger which is completely transparent. So my idea is crowd not give money direct to company. company create one smart contract and **set the target, deadline, minimum** contribution now crowd donate ether to the smart contract then company want to transfer ether from own account that time this possible when user give the permission

Smart Contract

One of the biggest problems of modern business is trust. With each passing day, we lose faith in big companies and even other humans. Without the trust factor and transparency, we are always advised to dread slowly. We take much more precaution, and this comes at the expense of flexibility that modern business requires.

To solve this problem, we depend on third-party intermediaries. But that also demands a certain level of trust. What if the intermediary embezzles our funds or goes bankrupt?

Here, blockchain can provide the ultimate solution with its feature of Smart Contracts.

Well, smart contracts are just simple computer codes that are run on the blockchain, and they are self-executable when we meet certain conditions. They are immutable, transparent, and, most importantly, they are decentralized. Clearly, this gives them an edge over traditional contract.

Nick Szabo introduced the term smart contract in 1990. He described a smart contract as a digital form of a set of promises, and when these promises are met, the users are guaranteed a specific outcome.

Additionally, all of these premises of smart contract are possible because of blockchain. Smart contract found recognition due to its induction in the Ethereum blockchain, where anyone can code a smart contract using the Ethereum Virtual Machine or Solidity.

Smart contracts are one of the safest modes of transaction today. Though a crude version of the Smart contract existed in the Bitcoin blockchain, but Ethereum with Turing Complete language introduced a superior version of the smart contract, which has even more incredible computation prowess.

The flow of the Smart Contract:

First the smart contract is written with the functionalities that are required in **remix solidity** open-source tool. Then smart contract functionalities are checked and verified before compiling it. Once the functionalities are verified smart contracts are compiled to get the Abi and byte code. As the browser cannot understand the **byte code, Abi** is used as an interface for browser and the bytecode to connect to any network like Polygon Mumbai. Fig1. represents the smart contract compilation process. Then to add user interface to this smart contract web3 and react is ate used with semantic-ui. For react to get access to the contract functions web3 is used. Then tests are performed using mocha test runner to check if the functionalities are properly working. All the code is written in node using node js as java script is being run outside the browser. This also helps project to use third party libraries. Nextjs is used for server-side rendering and routing the of the web pages. Finally, solidity contract is deployed to the Ethereum test network using Infura and gets the address where it is deployed and gives it to the application. Then server is started and at localhost: 3000 web application can be accessed.

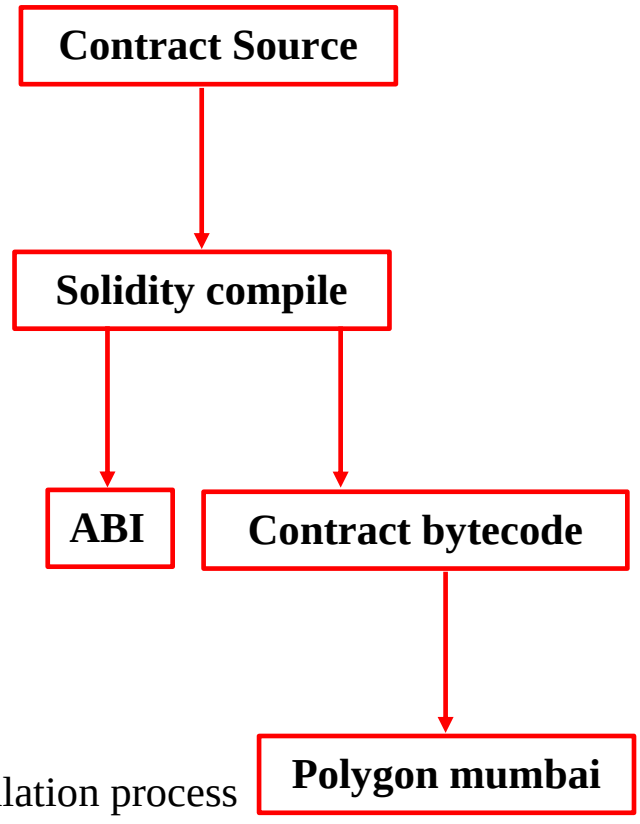
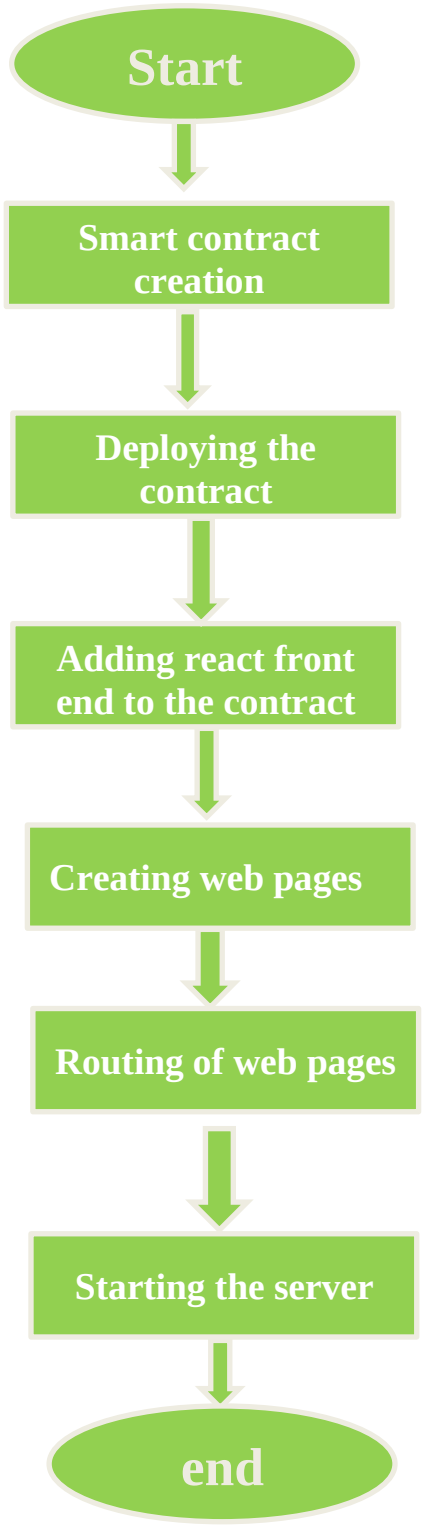


fig: smart contract compilation process

SYSTEM DESIGN

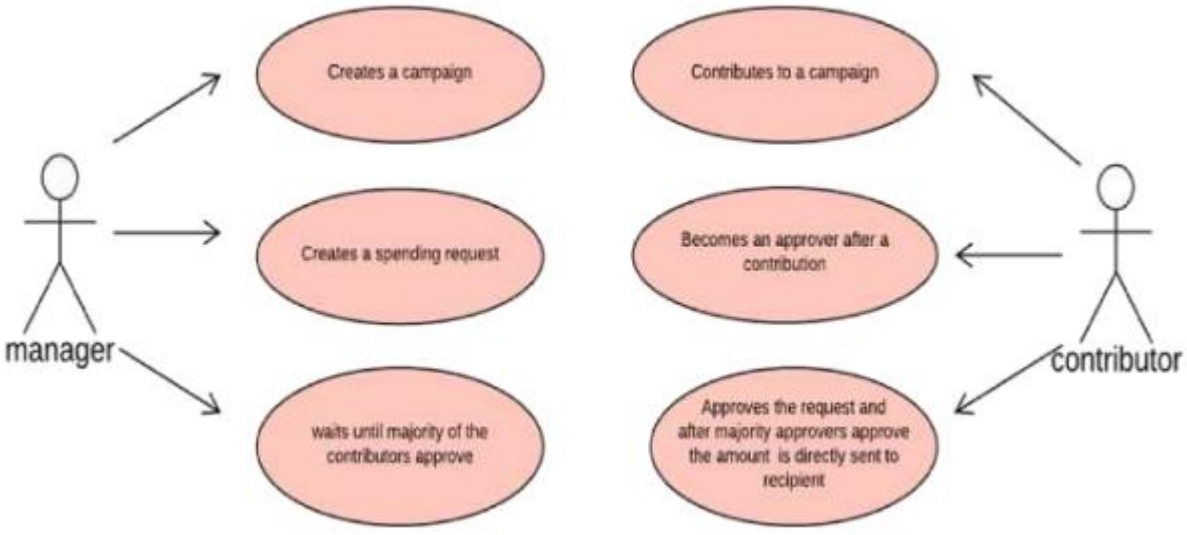
Flow chart



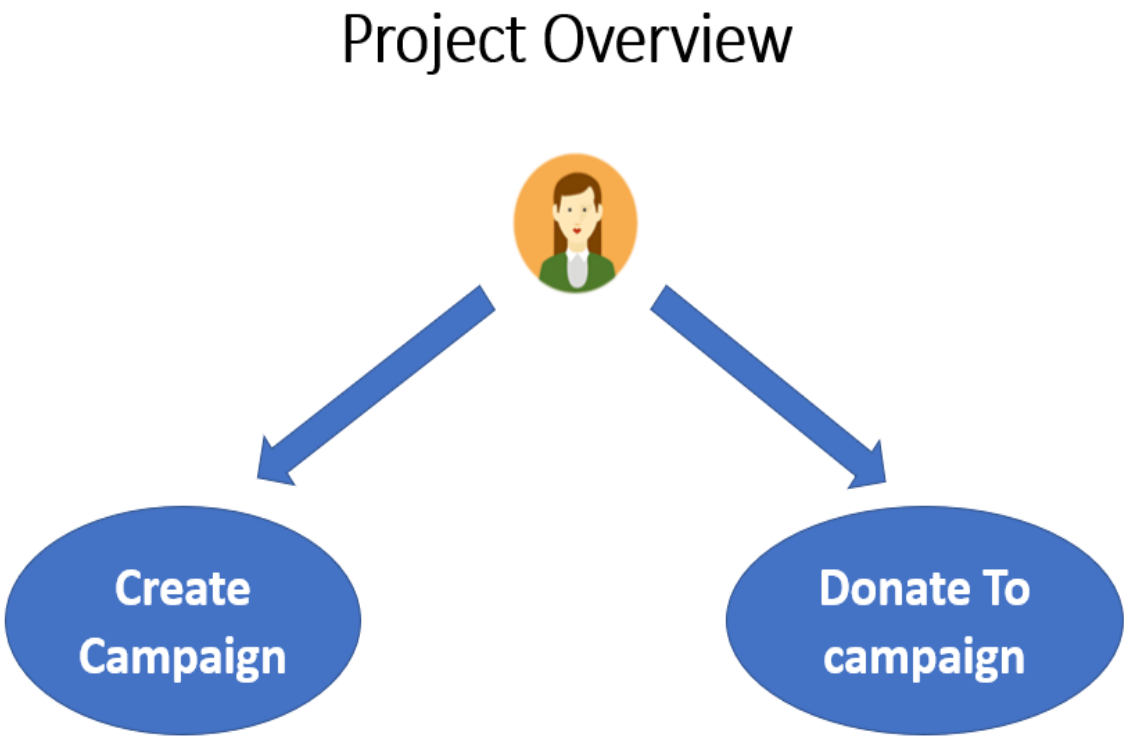
represents the complete flow of creating the project. First smart contract is created according to our needs. Then the ABI (Application Binary Interface) code and interface of the contract are used to interact with it and then deploy the contract. Then frontend is added to the contract using react js. In this way all web pages are created and nextjs is used as routing mechanism to route these pages and provide them a hyperlink. Then finally server is started. At localhost: 3000 we will find our web application running.

Use case diagram

the use case diagram of the project. This represents how the application actually works by showing the tasks done buy manager and contributor. First the manager creates the campaign. Then contributor will contribute to the campaign. Then after contributors contribute manager creates a spending request. Every contributor becomes a approver after they donate some money. Then contributors vote to this spending request created by the manager. After at least 51% of the contributors approve the spending request then the amount will be automatically transfer to the vendor specified in the spending request by the manager.



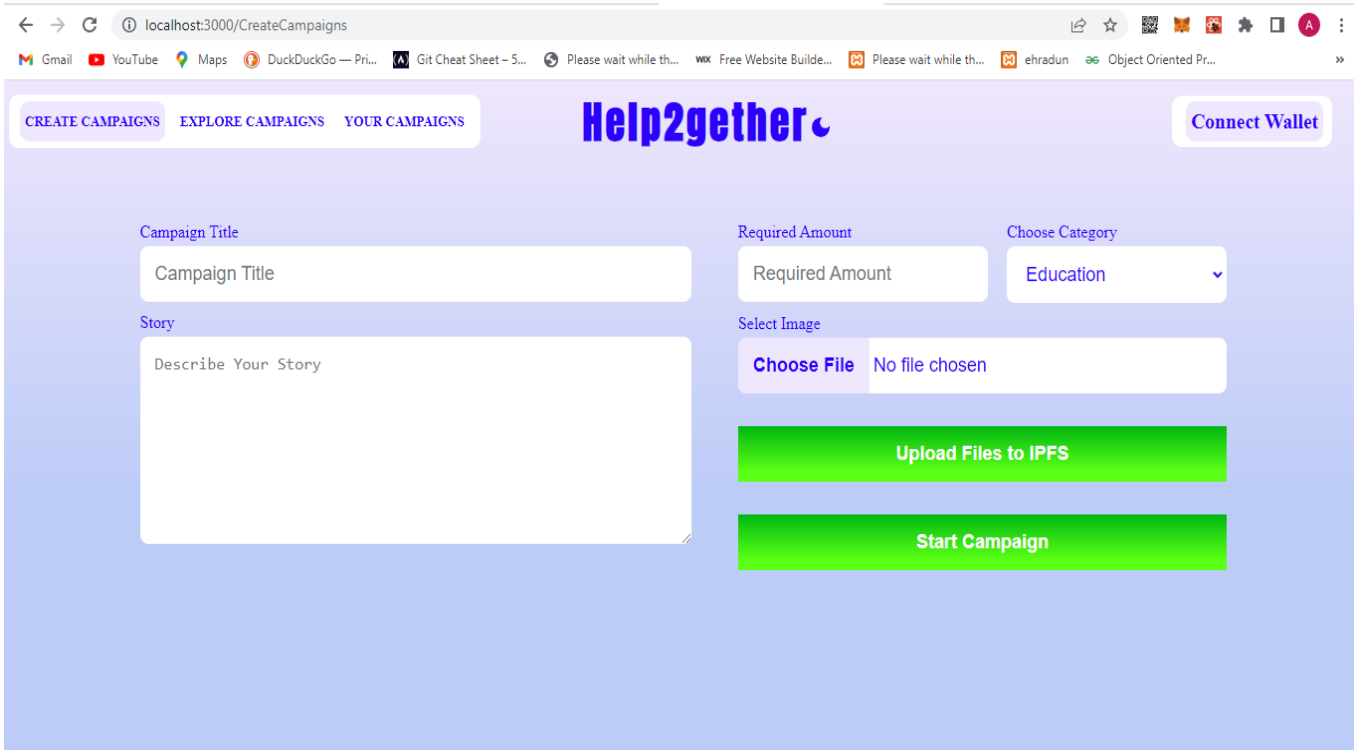
Use case diagram for this project



mainly our web app gives two functionalities for user: -
1) create campaign
2) donate to campaign

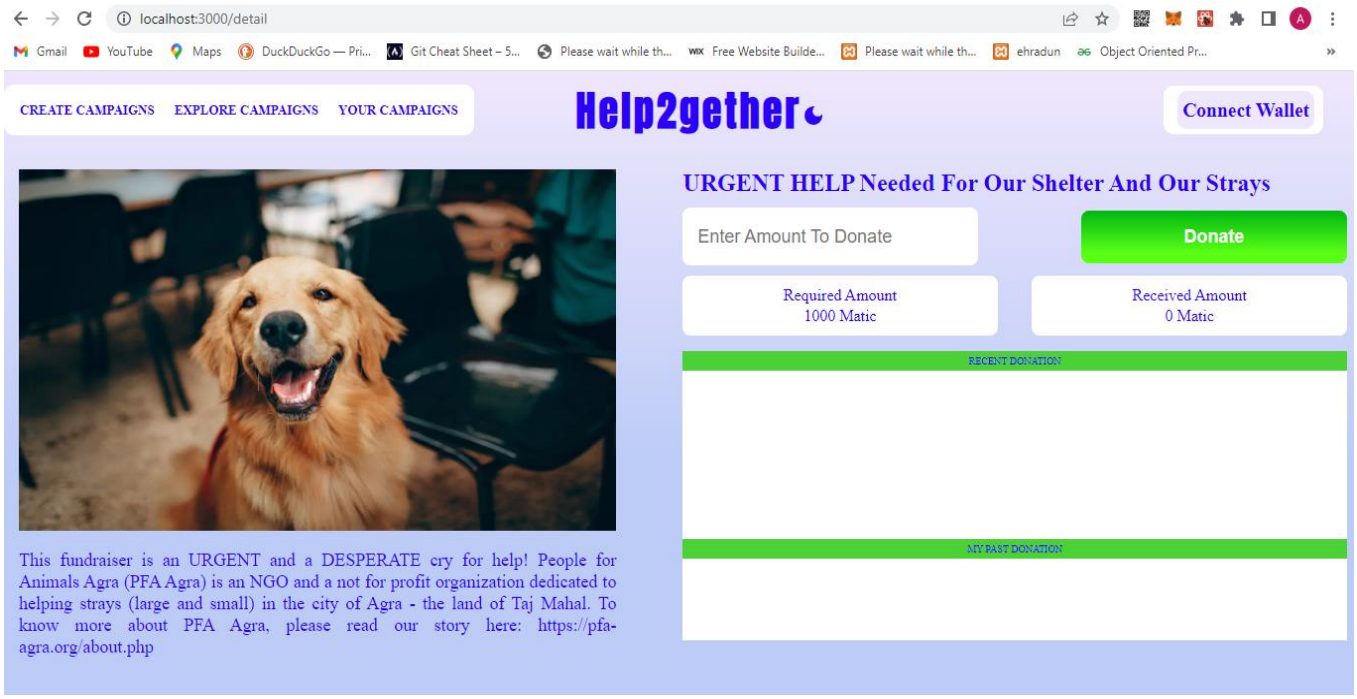
Feature No:1 (Create Campaign)

Just like Crowdfunding in the real world as well as on other crowdfunding platforms, anyone can create a campaign in a few minutes. The campaign information will be managed by the Ethereum-based smart contract and thus cannot be tampered with.



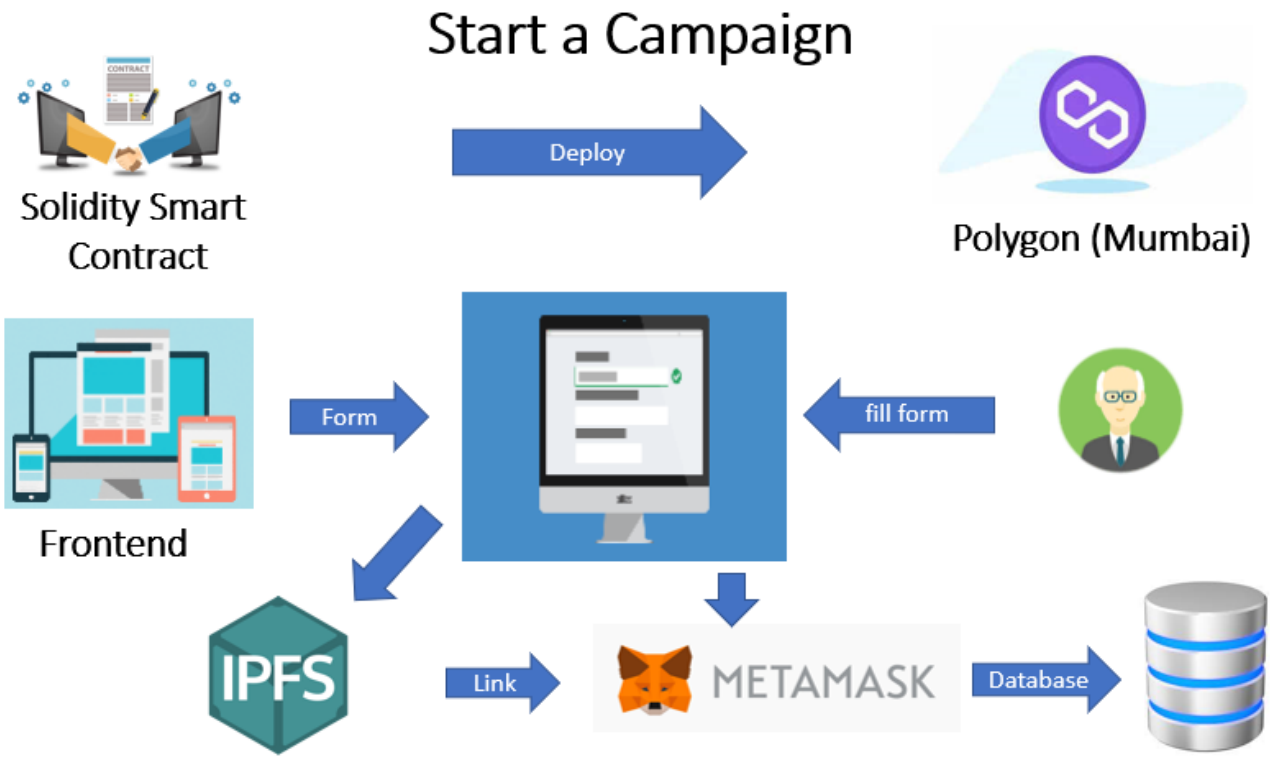
Feature No:2 (Donate to campaign)

Once a campaign has been created, users can share the campaign and anybody can contribute to the campaign. **The funds will go to the address of the campaign and not to the creator of the campaign,** thus making the process more efficient and anti-fraudulent.



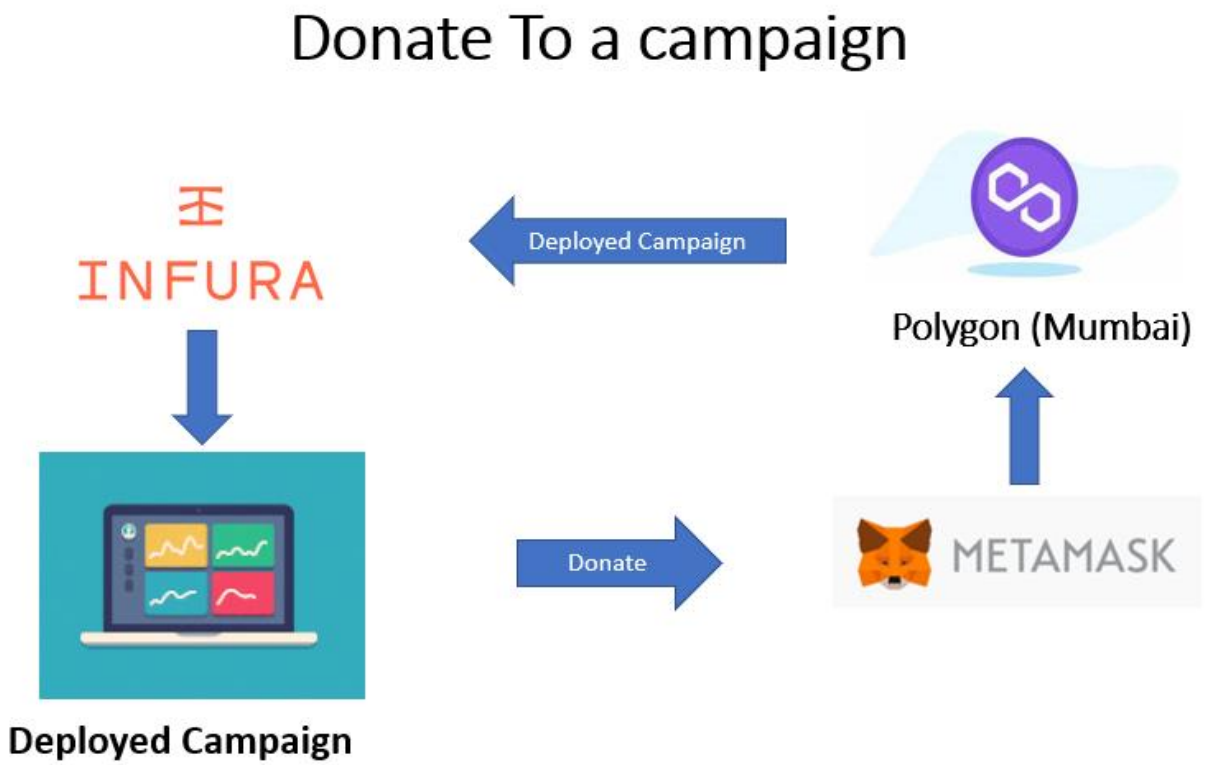
How This Feature Work: -

Feature No:1 (Create Campaign)



Firstly, written a smart contract using solidity programming language and then this contract deploy on polygon (Mumbai)
Now come on frontend user come and fill the form then image and story of the form is upload in IPFS and generate the link then data of form and link upload in the database (polygon) using meta mask.
Why generate link of image and story using IPFS because: -
Storing images to blockchain is very expensive
Then our campaign is successfully created.

Feature No:2 (Donate to campaign)



Now see how user donate to campaign firstly we fetch data (deployed campaign) from polygon (database) and using infura show the deployed campaign on frontend now user come and fill donation form and now donate amount then we deployed the data in the polygon (Mumbai)

TECH STACK

In order to achieve the solution, we have chosen a tech stack that is: -

- ❖ Optimized for speed
- ❖ Efficient
- ❖ Secure

The Technologies that have been used are: -

- ❑ **Vs Code**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

- ❑ **Remix IDE**

Remix Ide is used for the entire journey of smart contract development by users at every knowledge level. It requires no setup, fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. The IDE comes in 2 flavors (web app or desktop app) and as a VS Code extension.

- ❑ **Solidity**

Solidity is a contract-oriented, high-level programming language for implementing smart contracts. Solidity is highly influenced by C++, Python and JavaScript and has been designed to target the Ethereum Virtual Machine (EVM).

❑ Node.js

- Node.js is an open-source server environment.
- Node.js allows you to run JavaScript on the server
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

❑ Next.js

Next.js is a lightweight framework for React applications. In web development, it's used to build fast web apps thanks to its static site and server-side rendering.

❑ Polygon

- Polygon is a cryptocurrency, with the symbol MATIC, and also a technology platform that enables blockchain networks to connect and scale.
- Polygon is a "layer two" or "sidechain" scaling solution that runs alongside the Ethereum blockchain — allowing for speedy transactions and low fees.

❑ Hardhat

Hardhat is an environment developers use to test, compile, deploy and debug dApps based on the Ethereum blockchain. As such, it helps coders and developers to manage many of the tasks that are inherent to developing dApps and smart contracts

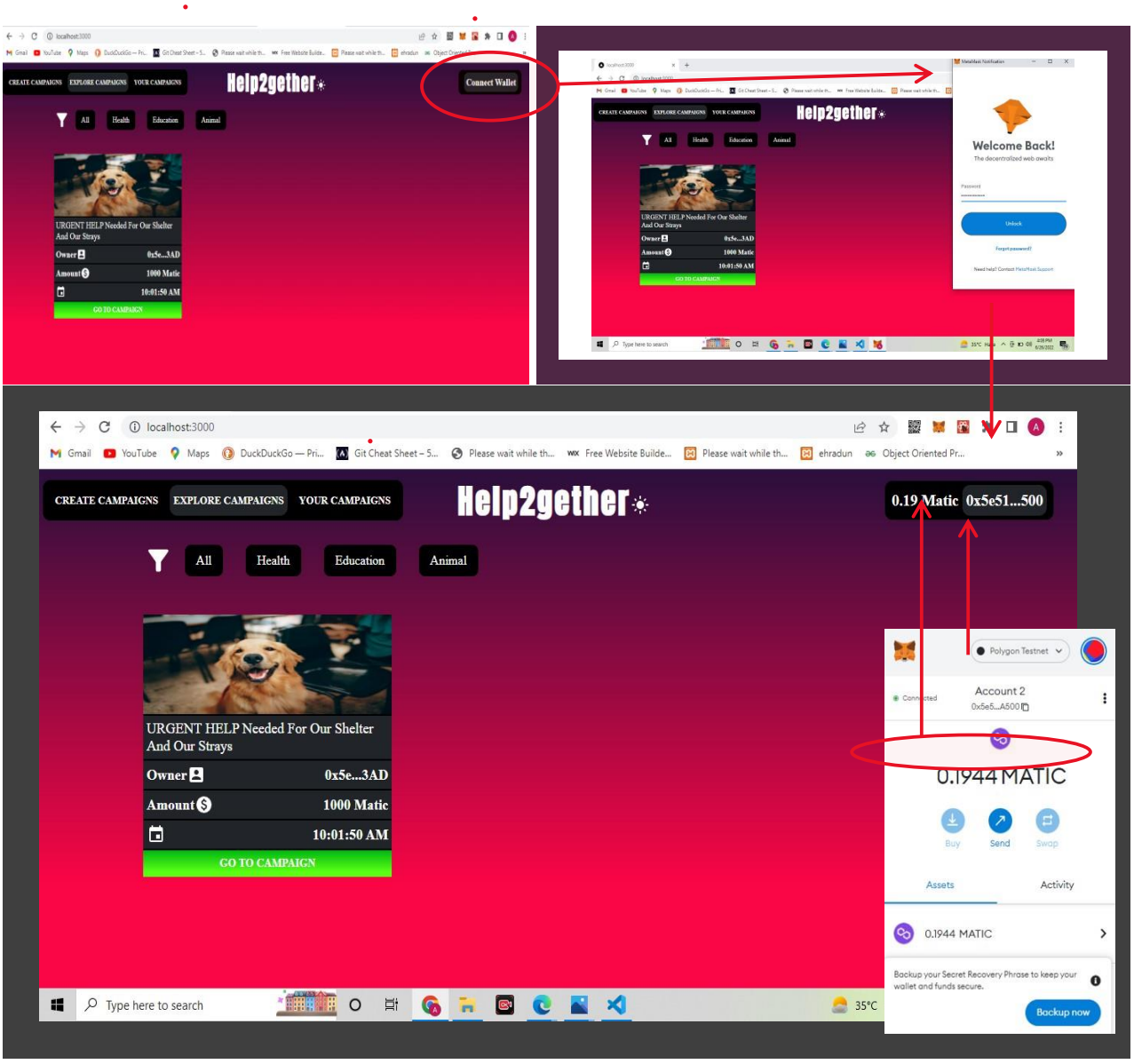
❑ Styled-components

Styled-components is a popular library that is used to style React applications. It allows you to build custom components by writing actual CSS in your JavaScript.

Web App Usage Instructions

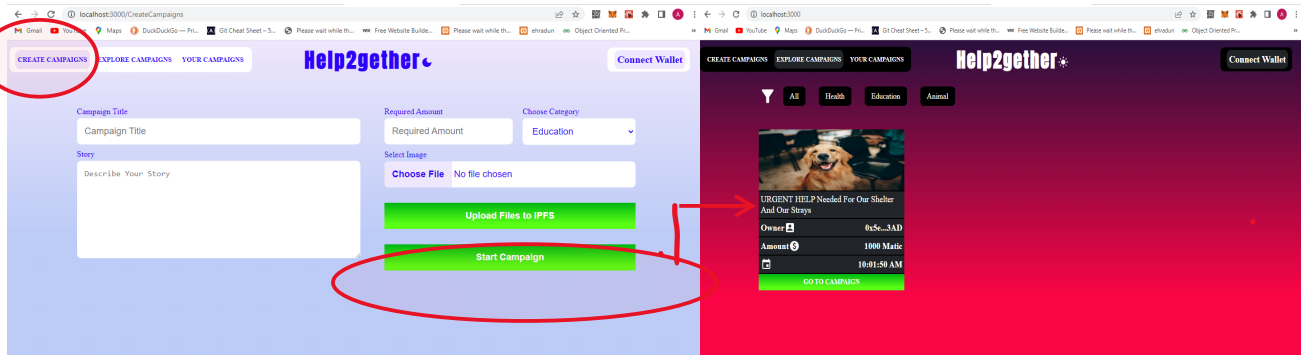
1. Connect Wallet:

In order to perform any transactions, be it creation of a campaign or contributing to one, a user first needs to connect an Ethereum wallet to the site. We have made use of a browser extension called Meta mask to connect the wallet, which can be used to authorize transactions for cryptocurrency.



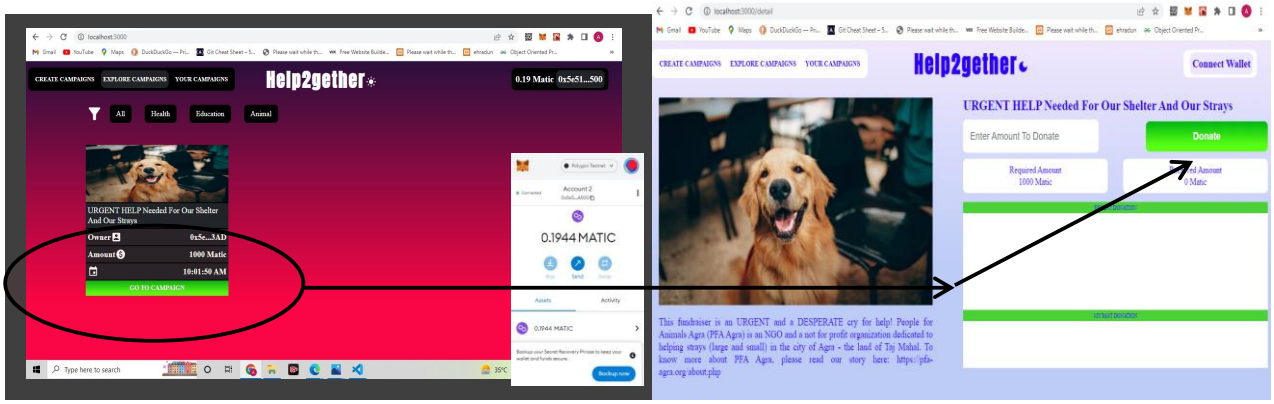
2. Creating a Campaign:

Once a wallet has been connected, anyone can create a crowdfunding campaign. The process is highly intuitive and self-explanatory, and the user only has to supply the data as asked in the forms.



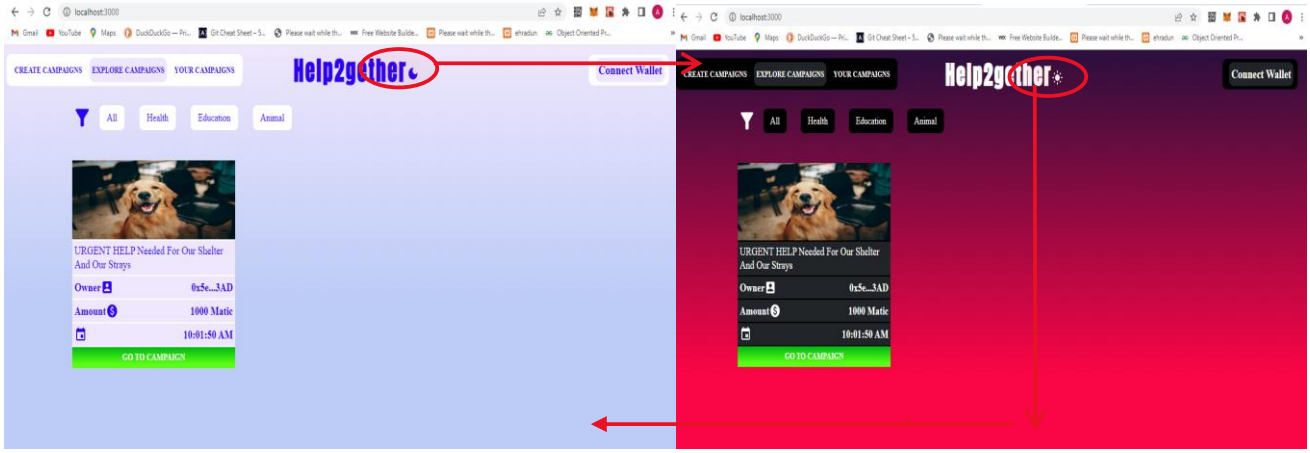
3. Contributing to a Campaign

Any user whose wallet has been connected to the app can contribute to a campaign. The process is simple and detailed in the flow below. The user only needs to select the campaign, enter the amount he wishes to contribute, and then authorize the transaction (in this case, with the Meta mask extension).



3. Change color of theme

After clicking icon of either moon and sun you can change the theme of web app here two type of theme is present dark theme and light theme.



How Does Block Chain Support Crowd Funding?

There are several areas where block-chain supports and improves crowdfunding, crowdfunding platforms powered by blockchain technology removes the need for intermediate third party.

- **Decentralization:**
  Since block-chain is decentralized it doesn't rely on any other platforms to create funds. for starters, no longer to be obliged to any rules and any project can get visibility and funded if the investors think to invest, eliminates fees which makes crowdfunding less expensive for the creators.

- **Access Equity:**
  To provide investors equity or ownership block-chain relies on asset tokenization. For example, a person who plans to create multiple new products with the incoming funds and grant small ownerships stake in the company This could potentially open whole new world of opportunity.

- **Universal Opportunity:**
  Any project using a block-chain-based crowdfunding model can get funded. Any person with an internet connection can contribute projects.

- **Flexible Options:**
  Using block-chain as asset tokenization grants creators and entrepreneurs more liberties. usually asset tokens have their own currency to enable organizations to hire professionals and advertisers.

- **Peer-to-Peer:**
  The crypto currencies are exchangeable on a peer-to-peer network. This usually help the people for their investment which even generates more interest in the entire process.

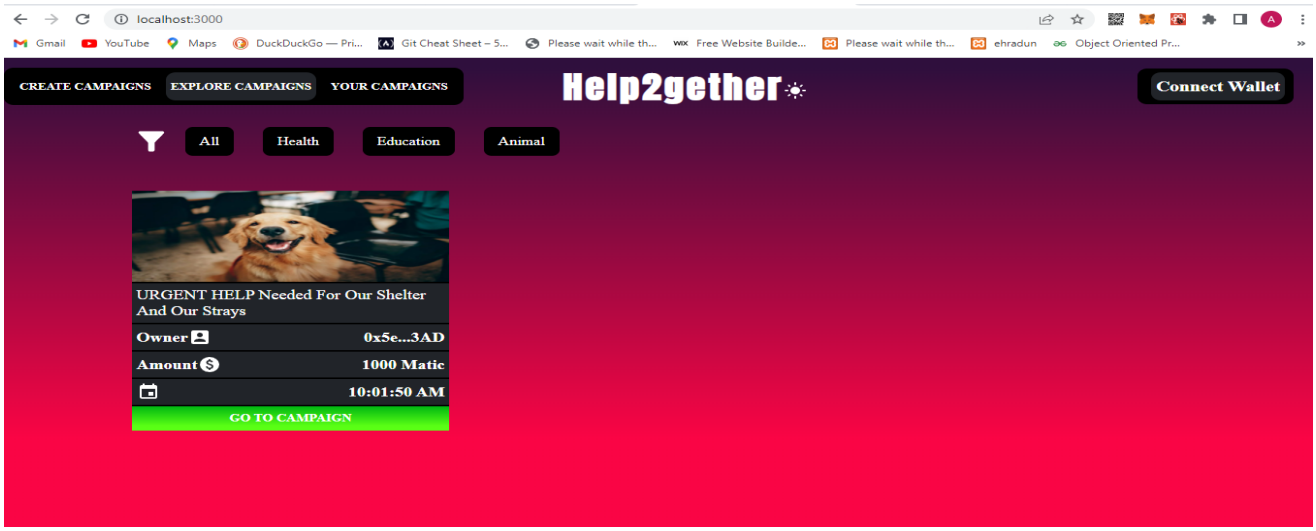
Advantages of Web2gether

Blockchain technology offers a clear advantage for the crowdfunding industry, bringing transparency and security to the space and protecting both creators and donors in these interactions.

- **Enhanced security**
- **Greater transparency**
- **Instant traceability**
- **Increased efficiency and speed**
- **Automation**

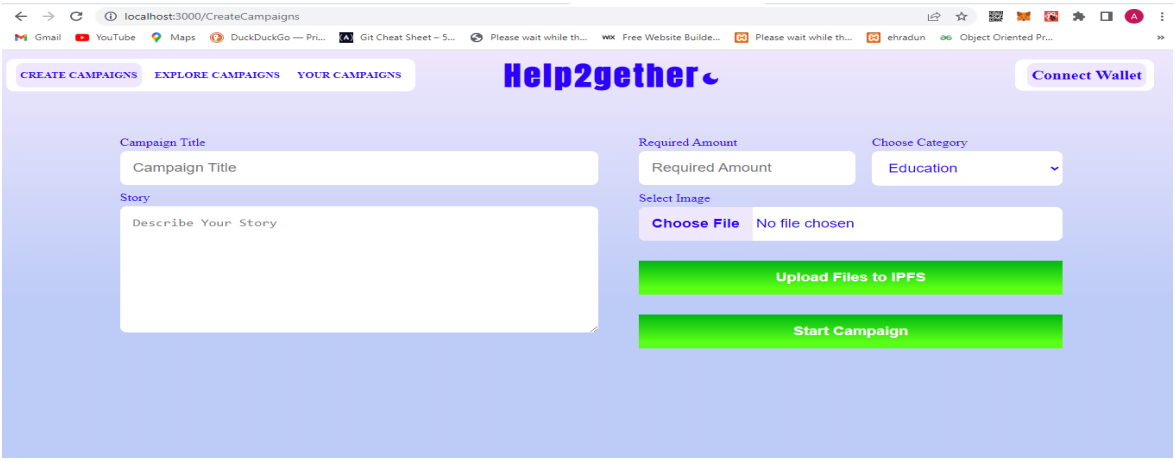
Result

When the web application is started the first page that is seen is campaign page where existing campaigns are displayed and a new campaign can also be created as shown in Fig. All these operations can be performed provided that Metamask is installed in the browser.

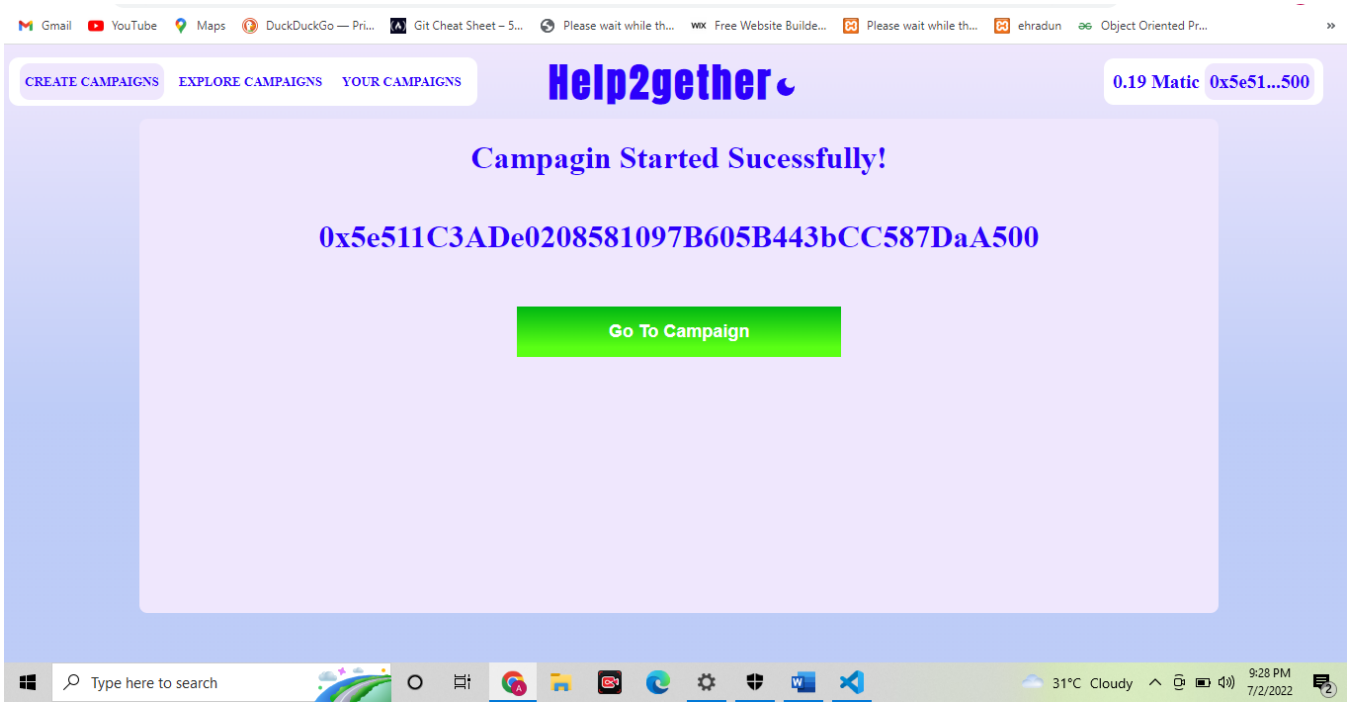


view of campaigns web page

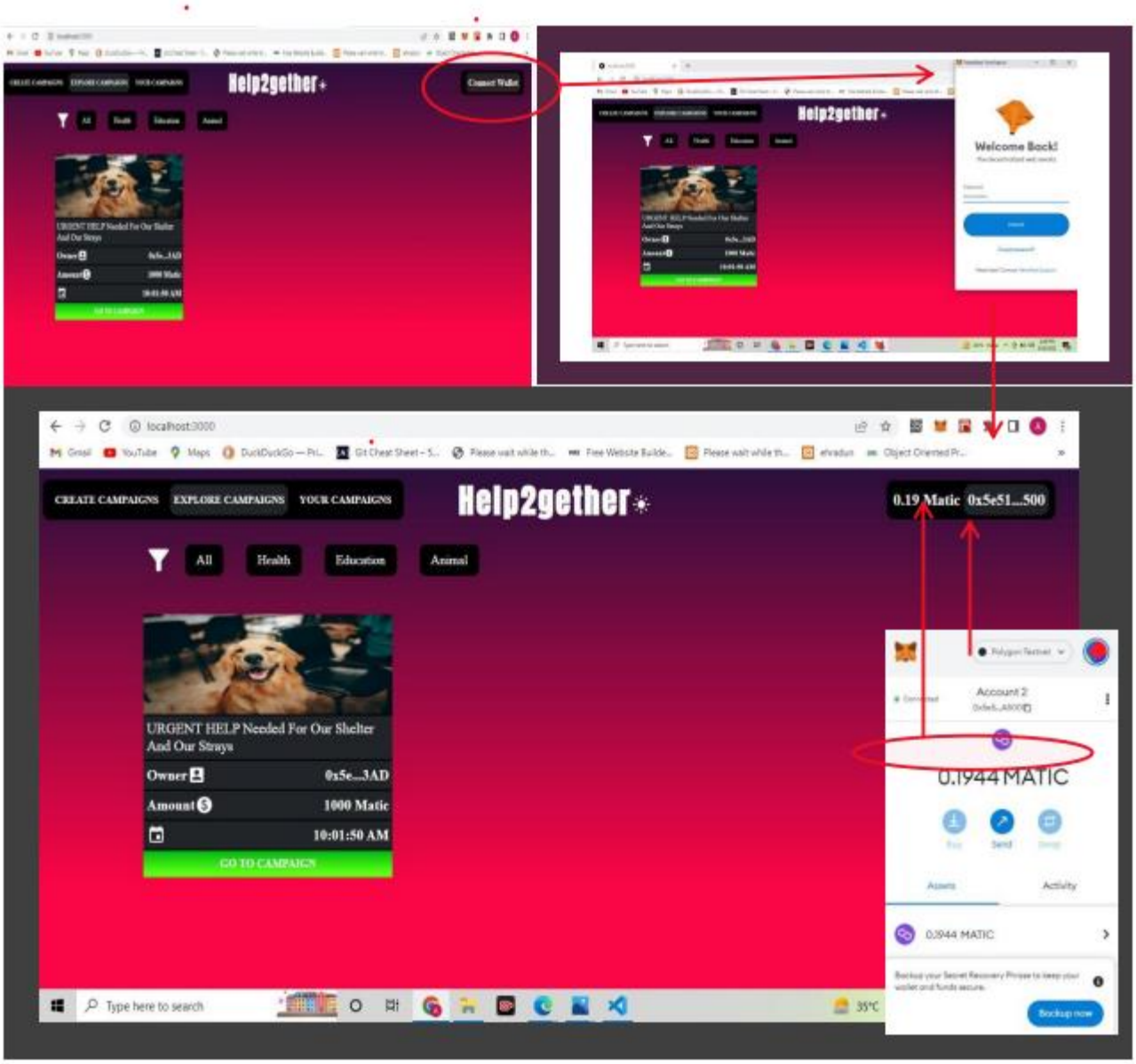
When create campaign is pressed it will redirect to the page where a transaction needs to be performed in order to create a new campaign as shown in Fig.



Campaigns creation web page

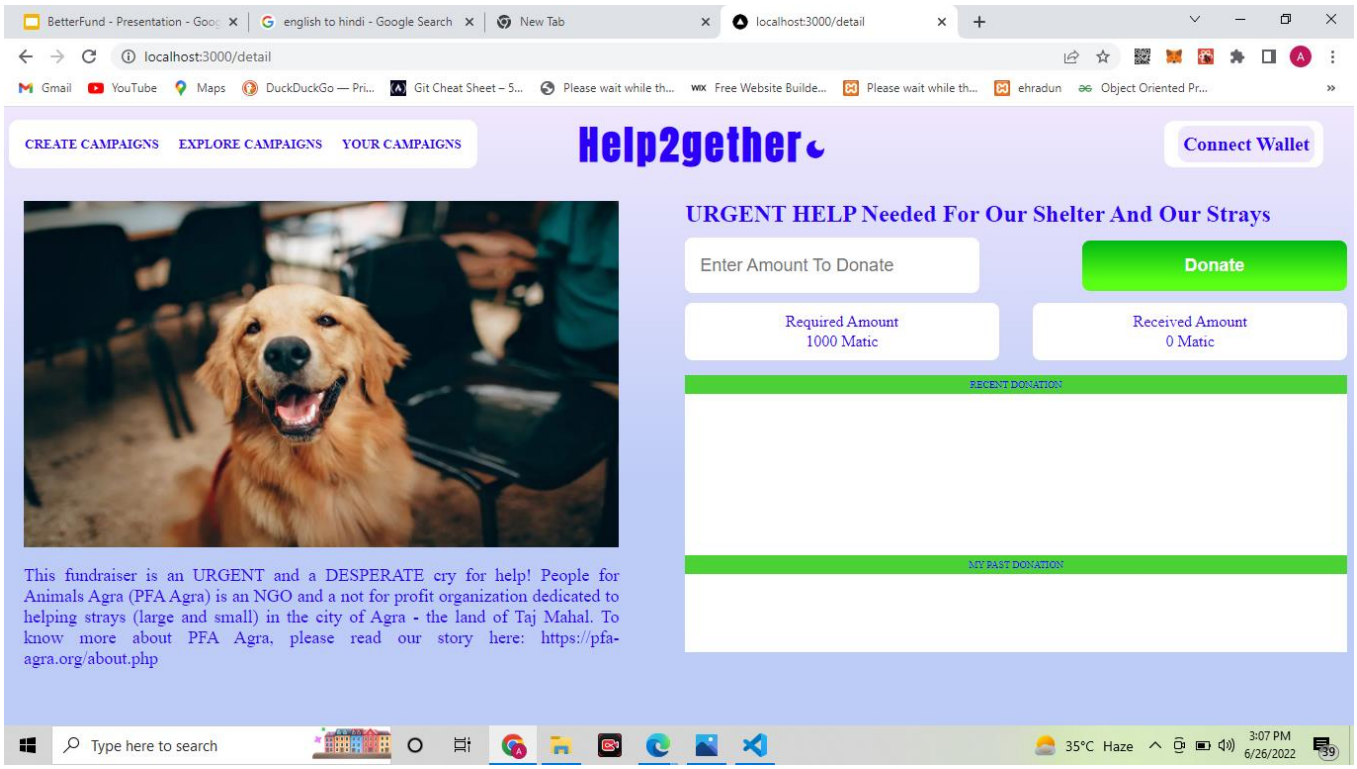


In order to perform any transactions, be it creation of a campaign or contributing to one, a user first needs to connect an Ethereum wallet to the site. We have made use of a browser extension called Meta mask to connect the wallet, which can be used to authorize transactions for cryptocurrency.



Metamask wallet

Any user whose wallet has been connected to the app can contribute to a campaign. The process is simple and detailed in the flow below. The user only needs to select the campaign, enter the amount he wishes to contribute, and then authorize the transaction (in this case, with the Meta mask extension).



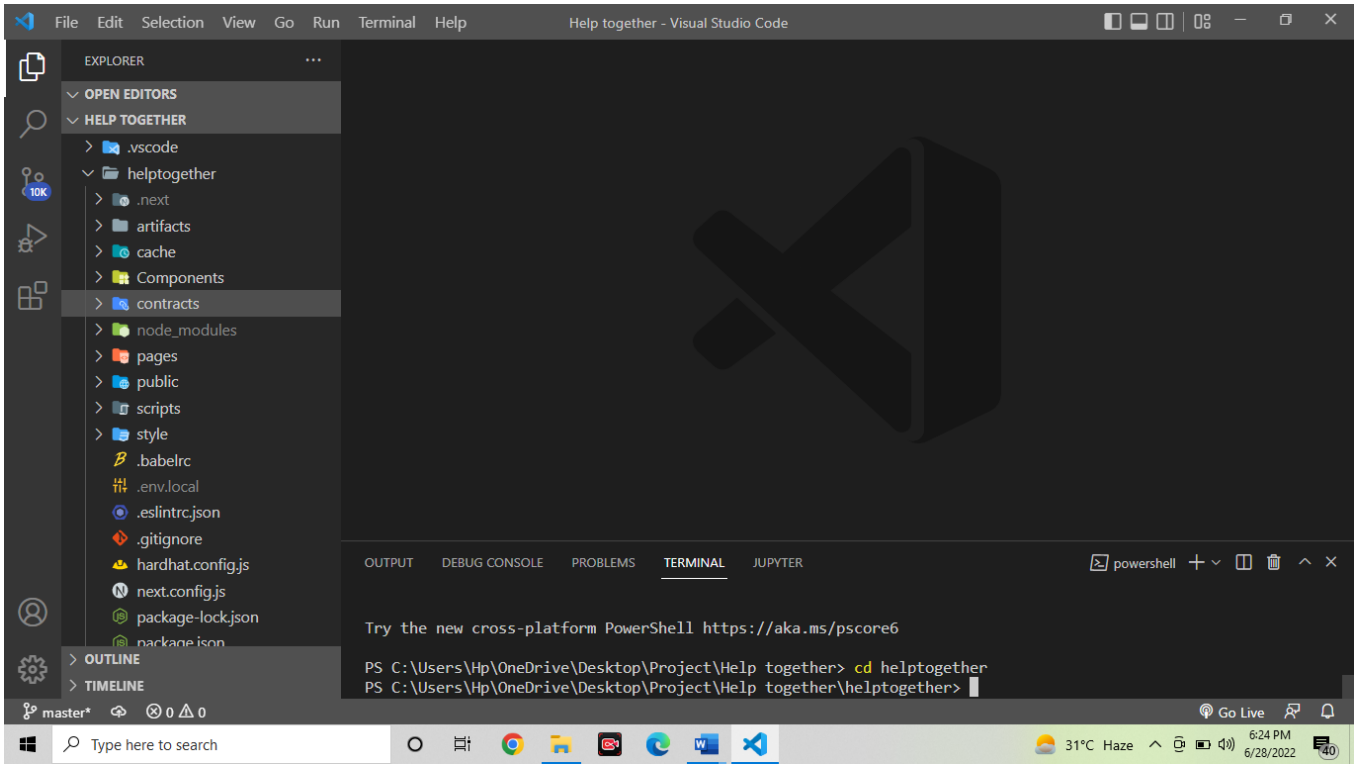
Contributing to a Campaign

Observation

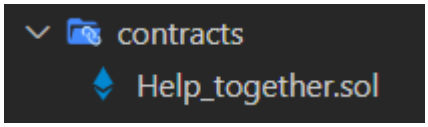
In Crowd Funding using blockchain to raise money for a startup first a campaign has to be created and a unique 64-character public key is assigned to that particular campaign to which investors can use as an address to donate. Investors can contribute the money in the form of ether. The contributed money will remain in the smart contract itself and will not be directly given to the campaign creator. If the campaign creator wants to spend the money, he/she has to make a spending request which will consists of the amount of the ether needed and also the address or the public key of vendor to whom the ether should be sent so that campaign creator can get the required material which ensures that the contributed ether is always securely kept inside the smart contract and contributors have to vote to spending request created. If the percentage of the votes is at least 51 % then the amount will automatically be transferred to the vendor. This method of Crowd Funding is lot faster and more secure as compared to the existing system. In current system the contributed amount is in the direct hands of the campaign creator and he/she can run away with that money, the time taken to transfer of the money raised to the campaign creator or the startup is slow and also there is fee that will be cut from the contributed amount as a third party like Kickstarter is involved or used in raising the required capital. Hence, the Crowd Funding using blockchain is more revolutionary and efficient method for raising capital for the startups.

SOURCE CODE

Folder Structure: -



Contracts: -



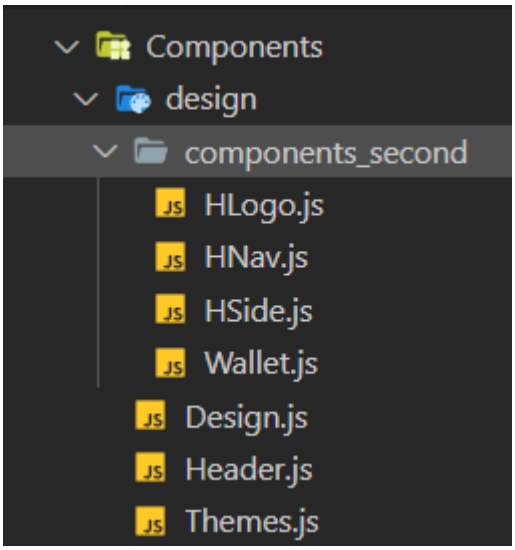
Helptogether.sol

```
//SPDX-License-Identifier: UNLICENSED
pragma solidity >0.7.0 <= 0.9.0;

contract Help_together_store{
    address[] public deployedCampaigns;

    event help_together_event(
        string event_title,
        string indexed category,
        string img_link,
        address indexed creater,
```

```
        address campaignAddress,
        uint indexed timestamp,
        uint event_requiredamount
    );

    function createCampaign(
        string memory store_fund_title ,
        string memory store_story,
        string memory store_image,
        string memory store_category,
        uint store_amount_required)public{
        Help_together fundrise_campaign = new Help_together(store_fund_title,
        store_story, store_image, store_amount_required,msg.sender);
        deployedCampaigns.push(address(fundrise_campaign));

        emit help_together_event(store_fund_title,
        store_category,
        store_image,
        msg.sender ,
        address(fundrise_campaign) ,
        block.timestamp ,
        store_amount_required);
    }
}
contract Help_together{
   string public fund_title;
   string  public story;
   string public image;
   uint public amount_required;
   uint public total_amount_received;
   address payable public creater;

  event fund_donated_event(address indexed donar , uint indexed amount , uint  indexed
timestamp);
   constructor(
       string memory title_fund,
       string memory story_link,
       string memory img_link,
       uint required_amount,
       address campaignOwner ){
        creater = payable(campaignOwner);
        fund_title=title_fund;
        story=story_link;
        image=img_link;
        amount_required=required_amount;
```

```
        }

    function fund_donate()public payable {
        require(amount_required > total_amount_received,
        "total required amount is fullfilled thankyou");
        creater.transfer(msg.value);
        total_amount_received= total_amount_received+msg.value;
        emit fund_donated_event(msg.sender, msg.value , block.timestamp);
    }
}
```

Components: -



HLogo.js: -

```
import styled from "styled-components"
import LightModeIcon from '@mui/icons-material/LightMode';
import DarkModeIcon from '@mui/icons-material/DarkMode';
import { App } from '../Design';
import { useContext } from 'react';
```

```
export const HLogo = () => {
  const change_theme  = useContext(App);
  return (
    <Title>
       Help2gether
      <change_theme>
      {change_theme.theme=='light' ? <DarkModeIcon
onClick={change_theme.function_change_theme}/> : <LightModeIcon
onClick={change_theme.function_change_theme}/> }
      </change_theme>
      </Title>
  )
}

const Title = styled.h1`
 font-weight: bold;
  font-size: 40px;
  margin-left: 23px;
  font-family: 'Anton', sans-serif;
  letter-spacing: 3px;
  cursor: pointer;`

const change_theme = styled.div`
display: flex;
justify-content: center;
align-items: center;
height: 100%;
padding: 5px;
width: 45px;
border-radius: 12px;
cursor: pointer;
`
```

HNav.js

```
import styled from "styled-components"
import {useRouter} from 'next/router';
import Link from 'next/link';
```

```
export const HNav = () => {
  const Router = useRouter();
  return (
    <Navmain>
    <Link passHref href={'/CreateCampaigns'}><HLinks active={Router.pathname ==
"/CreateCampaigns" ? true : false}>
    Create Campaigns
    </HLinks></Link>
    <Link passHref href={'/'}><HLinks active={Router.pathname == "/" ? true : false}>
    Explore Campaigns
    </HLinks></Link>
    <Link passHref href={'/YourCampaigns'}><HLinks active={Router.pathname ==
"/YourCampaigns" ? true: false}>
      Your Campaigns
    </HLinks></Link>
  </Navmain>
  )
}

const Navmain = styled.div`
  display: flex;
  align-items: center;
  justify-content: space-between;
  background-color: ${(props) => props.theme.bgDiv};
  padding: 6px;
  height: 50%;
  border-radius: 10px;
  margin-left: 5px;
  margin-right: -261px;
  `

const HLinks = styled.div`
  display: flex;
  align-items: center;
  justify-content: space-between;
  background-color: ${(props) => props.active ? props.theme.bgSubDiv :
props.theme.bgDiv };
  height: 100%;
  font-family: 'Roboto';
  margin: 5px;
  border-radius: 10px;
  padding: 0 5px 0 5px;
  cursor: pointer;
  text-transform: uppercase;
  font-weight: bold;
  font-size: small;
```

```
`
```

HSide.js

```
import styled from 'styled-components';
import Wallet from './Wallet';
export const HSide = () => {
  return (
    <SideMain>
    <Wallet/>
    </SideMain>
  )
}

const SideMain = styled.div`
  display: flex;
  justify-content: center;
  align-items: center;
  margin-right: 16px;
  height: 50%;
`
```

Wallet.js

```
import styled from "styled-components";
import { useState } from "react";
import { ethers } from "ethers";

   const networks = {
    polygon: {
      chainId: `0x${Number(80001).toString(16)}`,
      chainName: "Polygon Testnet",
      nativeCurrency: {
        name: "MATIC",
        symbol: "MATIC",
        decimals: 18,
      },
      rpcUrls: ["https://rpc-mumbai.maticvigil.com/"],
      blockExplorerUrls: ["https://mumbai.polygonscan.com/"],
    },
    };


  const Wallet = () => {
```

```
    const [balance, setBalance] = useState("");
    const [address, setAddress] = useState("");


    const Wallet_ConFunc = async () => {
        await window.ethereum.request({ method: "eth_requestAccounts" });
        const provider = new ethers.providers.Web3Provider(window.ethereum, "any");
        if (provider.network !== "matic") {
          await window.ethereum.request({
            method: "wallet_addEthereumChain",
            params: [
              {
                ...networks["polygon"],
              },
            ],
          });
        }
          const account = provider.getSigner();
          const Address = await account.getAddress();
          setAddress(Address);
          const Balance = ethers.utils.formatEther(await account.getBalance());
          setBalance(Balance);

      };
  return (
    <WalletMain onClick={Wallet_ConFunc}>
      {balance == '' ? <Balance></Balance> : <Balance>{balance.slice(0,4)}
Matic</Balance> }
      {address == '' ? <Address>Connect Wallet</Address> :
<Address>{address.slice(0,6)}...{address.slice(39)}</Address>}
    </WalletMain>
  )
}


const WalletMain = styled.div`
  display: flex;
  align-items: center;
  justify-content: space-between;
  background-color: ${(props) => props.theme.bgDiv};
  padding: 5px 9px;
  height: 100%;
  color: ${(props) => props.theme.color};
  border-radius: 10px;
  margin-right: 15px;
  font-family: 'Roboto';
```

```
  font-weight: bold;
  font-size: small;
  cursor: pointer;
`;

const Address = styled.h2`
    background-color: ${(props) => props.theme.bgSubDiv};
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 0 5px 0 5px;
    border-radius: 10px;
`

const Balance = styled.h2`
    display: flex;
    height: 100%;
    align-items: center;
    justify-content: center;
    margin-right: 5px;
`


export default Wallet
```

Design.js

```
import styled, { ThemeProvider, createGlobalStyle } from "styled-components";
import { useState, createContext } from "react";
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import Header from "./Header";
import themes from "./Themes";


const App = createContext();
const Design = ({ children }) => {
  const [theme, Themechanger] = useState("light");

  const function_change_theme = () => {
    Themechanger(theme == "light" ? "dark" : "light");
  };

  return (
```

```
    <App.Provider value={{function_change_theme, theme}}>
      <ThemeProvider theme={themes[theme]}>
      <ToastContainer />
        <LayoutWrapper>
          <GlobalStyle />
          <Header />
          {children}
        </LayoutWrapper>
      </ThemeProvider>
      </App.Provider>
  );
}

const LayoutWrapper = styled.div`
  min-height: 100vh;
  background-color: ${(props) => props.theme.bgColor};
  background-image: ${(props) => props.theme.bgImage};
  color: ${(props) => props.theme.color};
`;

const GlobalStyle = createGlobalStyle`
    body {
        margin: 0;
        padding: 0;
        overflow-x: hidden;
    }
`;

export default Design;
export {App};
```

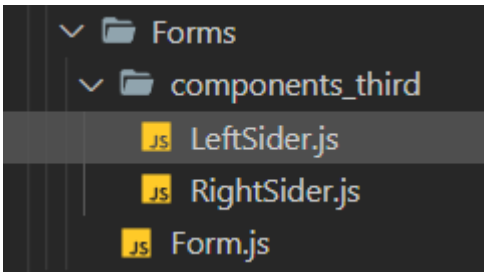
Header.js

```
import React from 'react'
import styled from 'styled-components';
import { HLogo } from './components_second/HLogo';
import { HNav } from './components_second/HNav';
import { HSide } from './components_second/HSide';


export const Header = () => {
  return (
    <MainHeader>
```

```
    <HNav />
    <HLogo/>
    <HSide/>
    </MainHeader>
  )
}

const MainHeader = styled.div`
  width: 100%;
  height: 75px;
  display: flex;
  // in flex bydefult in row
  justify-content: space-between;
  align-items: center;
`

export default Header
```

Themes.js

```
const dark = {
    color: '#fff',
    bgColor: '#923cb5',
    bgImage: 'linear-gradient(180deg, #2f0f3d 2%, #fa0545 80%)',
    bgDiv: 'black',
    bgSubDiv: 'rgb(33, 36, 41)'
}
const light = {
    color: "#2d00fb",
    bgColor: '#efe7fd',
    bgImage: 'linear-gradient(180deg, #efe7fd 0%, #bdccf7 60%)',
    bgDiv: '#fff',
    bgSubDiv: '#efe7fd'
}

const themes = {
    light: light,
    dark: dark,
}

export default themes;
```



LeftSider.js

```
import styled from 'styled-components';
import { FormState } from '../form';
import { useContext } from 'react';
const LeftSider = () => {
  const Handler = useContext(FormState);

    return (
        <FormLeft>
          <FormInput>
            <label>Campaign Title</label>
            <Input onChange={Handler.FormHandler} value={Handler.form.campaignTitle}
placeholder='Campaign Title' name='campaignTitle'>
            </Input>
          </FormInput>
          <FormInput>
            <label>Story</label>
            <TextArea onChange={Handler.FormHandler} value={Handler.form.story}
name="story" placeholder='Describe Your Story'>
            </TextArea>
          </FormInput>
        </FormLeft>
      )
}

const FormLeft = styled.div`
  width:48%;
`

const FormInput = styled.div`
  display:flex ;
  flex-direction:column;
  font-family:'poppins';
  margin-top:10px ;
```

```
`
const Input = styled.input`
  padding:15px;
  background-color:${(props) => props.theme.bgDiv} ;
  color:${(props) => props.theme.color} ;
  margin-top:4px;
  border:none ;
  border-radius:8px ;
  outline:none;
  font-size:large;
  width:100% ;
`

const TextArea = styled.textarea`
  padding:15px;
  background-color:${(props) => props.theme.bgDiv} ;
  color:${(props) => props.theme.color} ;
  margin-top:4px;
  border:none;
  border-radius:8px ;
  outline:none;
  font-size:large;
  max-width:100%;
  min-width:100%;
  overflow-x:hidden;
  min-height:160px ;
`


export default LeftSider
```

RightSider.js

```
import styled from 'styled-components';
import { FormState } from '../form';
import { useState, useContext } from 'react';
import { toast } from 'react-toastify';
import {TailSpin} from 'react-loader-spinner';
import {create as IPFSHTTPClient} from 'ipfs-http-client';

const client = IPFSHTTPClient("https://ipfs.infura.io:5001/api/v0");

const RightSider = () => {
  const Handler = useContext(FormState);
  const [uploadLoading, setUploadLoading] = useState(false);
```

```
  const [uploaded, setUploaded] = useState(false);

  const uploadFiles = async (e) => {
    e.preventDefault();
    setUploadLoading(true);

    if(Handler.form.story !== "") {
      try {
        const added = await client.add(Handler.form.story);
        Handler.setStoryUrl(added.path)
      } catch (error) {
        toast.warn(`Error Uploading Story`);
      }
    }


      if(Handler.image !== null) {
          try {
              const added = await client.add(Handler.image);
              Handler.setImageUrl(added.path)
          } catch (error) {
            toast.warn(`Error Uploading Image`);
          }
      }

      setUploadLoading(false);
      setUploaded(true);
      Handler.setUploaded(true);
      toast.success("Files Uploaded Sucessfully")
}

  return (
    <FormRight>
      <FormInput>
        <FormRow>
          <RowFirstInput>
            <label>Required Amount</label>
            <Input onChange={Handler.FormHandler} value={Handler.form.requiredAmount}
name="requiredAmount" type={'number'} placeholder='Required Amount'></Input>
          </RowFirstInput>
          <RowSecondInput>
            <label>Choose Category</label>
            <Select onChange={Handler.FormHandler} value={Handler.form.category}
name="category">
              <option>Education</option>
              <option>Health</option>
```

```
              <option>Animal</option>
            </Select>
          </RowSecondInput>
        </FormRow>
      </FormInput>
      {/* Image */}
      <FormInput>
        <label>Select Image</label>
        <Image onChange={Handler.ImageHandler} type={'file'} accept='image/*'>
        </Image>
      </FormInput>
      {uploadLoading == true ? <Button><TailSpin color='#fff' height={20} /></Button> :
        uploaded == false ?
        <Button onClick={uploadFiles}>
          Upload Files to IPFS
        </Button>
        : <Button style={{cursor: "no-drop"}}>Files uploaded Sucessfully</Button>
      }
      <Button onClick={Handler.startCampaign}>
        Start Campaign
      </Button>
    </FormRight>
  )
}

const FormRight = styled.div`
  width:45%;
`

const FormInput = styled.div`
  display:flex ;
  flex-direction:column;
  font-family:'poppins';
  margin-top:10px;
`

const FormRow = styled.div`
  display: flex;
  justify-content:space-between;
  width:100% ;
`

const Input = styled.input`
  padding:15px;
  background-color:${(props) => props.theme.bgDiv} ;
  color:${(props) => props.theme.color} ;
```

```
  margin-top:4px;
  border:none ;
  border-radius:8px ;
  outline:none;
  font-size:large;
  width:100% ;
`

const RowFirstInput = styled.div`
  display:flex ;
  flex-direction:column ;
  width:45% ;
`

const RowSecondInput = styled.div`
  display:flex ;
  flex-direction:column ;
  width:45% ;
`

const Select = styled.select`
  padding:15px;
  background-color:${(props) => props.theme.bgDiv} ;
  color:${(props) => props.theme.color} ;
  margin-top:4px;
  border:none ;
  border-radius:8px ;
  outline:none;
  font-size:large;
  width:100% ;
`

const Image = styled.input`
  background-color:${(props) => props.theme.bgDiv} ;
  color:${(props) => props.theme.color} ;
  margin-top:4px;
  border:none ;
  border-radius:8px ;
  outline:none;
  font-size:large;
  width:100% ;
  &::-webkit-file-upload-button {
    padding: 15px ;
    background-color: ${(props) => props.theme.bgSubDiv} ;
    color: ${(props) => props.theme.color} ;
    outline:none ;
```

```
    border:none ;
    font-weight:bold ;
  }
`

const Button = styled.button`
  display: flex;
  justify-content:center;
  width:100% ;
  padding:15px ;
  color:white ;
  background-color:#00b712 ;
  background-image:
      linear-gradient(180deg, #00b712 0%, #5aff15 80%) ;
  border:none;
  margin-top:30px ;
  cursor: pointer;
  font-weight:bold ;
  font-size:large ;
`

export default RightSider
```

Form.js

```
import styled from 'styled-components';
import RightSider from './components_third/RightSider';
import LeftSider from './components_third/LeftSider';
import { createContext, useState } from 'react';
import {TailSpin} from 'react-loader-spinner';
import {ethers} from 'ethers';
import {toast} from 'react-toastify';
import Help_together_store from
'../../artifacts/contracts/Help_together.sol/Help_together_store.json'

const FormState = createContext();

const Form = () => {
    const [form, setForm] = useState({
        campaignTitle: "",
        story: "",
        requiredAmount: "",
        category: "education",
    });
```

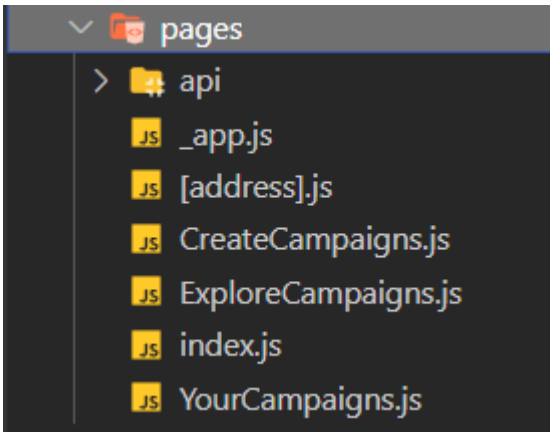
```
    const [loading, setLoading] = useState(false);
    const [address, setAddress] = useState("");
    const [uploaded, setUploaded] = useState(false);

    const [storyUrl, setStoryUrl] = useState();
    const [imageUrl, setImageUrl] = useState();

    const FormHandler = (e) => {
        setForm({
            ...form,
            [e.target.name]: e.target.value
        })
    }

    const [image, setImage] = useState(null);

    const ImageHandler = (e) => {
        setImage(e.target.files[0]);
    }

    const startCampaign = async (e) => {
        e.preventDefault();
        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();

        if(form.campaignTitle === "") {
          toast.warn("Title Field Is Empty");
        } else if(form.story === "" ) {
          toast.warn("Story Field Is Empty");
        } else if(form.requiredAmount === "") {
          toast.warn("Required Amount Field Is Empty");
        } else if(uploaded == false) {
            toast.warn("Files Upload Required")
        }
        else {
          setLoading(true);

          const contract = new ethers.Contract(
            process.env.NEXT_PUBLIC_ADDRESS,
            Help_together_store.abi,
            signer
          );

          // const CampaignAmount =
ethers.utils.parseEther(form.store_amount_required);
```

```
          const campaignData = await contract.createCampaign(
            form.store_fund_title,
            storyUrl,
            imageUrl,
            form.store_category,
            parseInt(form.store_amount_required)
          );

          await campaignData.wait();

          setAddress(campaignData.to);
        }
    }

  return (
    <FormState.Provider value={{form, setForm, image, setImage, ImageHandler,
FormHandler, setImageUrl, setStoryUrl, startCampaign, setUploaded}} >
    <FormWrapper>
    <FormMain>
    {loading == true ?
                address !== " " ?
                    <Spinner>
                        <TailSpin height={60} />
                    </Spinner> :
                <Address>
                    <h1>Campagin Started Sucessfully!</h1>
                    <h1>{address}</h1>
                    <Button>
                        Go To Campaign
                    </Button>
                </Address>
                :
                    <FormInputsWrapper>
    <LeftSider />
    <RightSider />
    </FormInputsWrapper>
    }
    </FormMain>
    </FormWrapper>
    </FormState.Provider>
  )
}

const FormWrapper = styled.div`
    width: 100%;
    display:flex;
```

```
    justify-content:center;
`

const FormMain = styled.div`
    width:80%;
`

const FormInputsWrapper = styled.div`
    display:flex;
    justify-content:space-between ;
    margin-top:45px ;
`

const Spinner = styled.div`
    width:100%;
    height:80vh;
    display:flex ;
    justify-content:center ;
    align-items:center ;
`
const Address = styled.div`
    width:100%;
    height:80vh;
    display:flex ;
    display:flex ;
    flex-direction:column;
    align-items:center ;
    background-color:${(props) => props.theme.bgSubDiv} ;
    border-radius:8px;
`

const Button = styled.button`
    display: flex;
  justify-content:center;
  width:30% ;
  padding:15px ;
  color:white ;
  background-color:#00b712 ;
  background-image:
      linear-gradient(180deg, #00b712 0%, #5aff15 80%) ;
  border:none;
  margin-top:30px ;
  cursor: pointer;
  font-weight:bold ;
  font-size:large ;
`
```

```
export default Form;
export {FormState};
```

Pages:-



_app.js

```
import Design from '../components/design/Design';
import '../style/font.css'
function MyApp({ Component, pageProps }) {
  return (
    <Design>
```

```
      <Component {...pageProps} />
    </Design>
    )
}

export default MyApp
```

[address].js

```
import styled from "styled-components";
import Image from "next/image";
import {ethers} from 'ethers';
import Help_together_store from
'../artifacts/contracts/Help_together.sol/Help_together_store.json';
import Help_together from '../artifacts/contracts/Help_together.sol/Help_together.json'
import { useEffect, useState } from "react";


export default function Detail({Data, DonationsData}) {
  const [mydonations, setMydonations] = useState([]);
  const [story, setStory] = useState('');
  const [amount, setAmount] = useState();
  const [change, setChange] = useState(false);

  useEffect(() => {
    const Request = async () => {
      let storyData;

      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(
        Data.address,
        Help_together.abi,
        provider
      );
```

```
      fetch('https://ipfs.infura.io/ipfs/' + Data.storyUrl)
            .then(res => res.text()).then(data => storyData = data);

      const MyDonations = contract.filters.donated(Address);
      const MyAllDonations = await contract.queryFilter(MyDonations);

      setMydonations(MyAllDonations.map((e) => {
        return {
          donar: e.args.donar,
          amount: ethers.utils.formatEther(e.args.amount),
          timestamp : parseInt(e.args.timestamp)
        }
      }));

      setStory(storyData);
    }

    Request();
  }, [change])


  const DonateFunds = async () => {
    try {
      await window.ethereum.request({ method: 'eth_requestAccounts' });
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = provider.getSigner();

      const contract = new ethers.Contract(Data.address, Help_together.abi, signer);

      const transaction = await contract.fund_donate({value:
ethers.utils.parseEther(amount)});
      await transaction.wait();

      setChange(true);
      setAmount('');

  } catch (error) {
      console.log(error);
  }

  }

  return (
    <DetailWrapper>
      <LeftContainer>
```

```
        <ImageSection>
          <Image
            layout="fill"
            src={
              "https://ipfs.infura.io/ipfs/" + Data.image
            }
          />
        </ImageSection>
        <Text>
          {story}
        </Text>
      </LeftContainer>
      <RightContainer>
        <Title>{Data.title}</Title>
        <DonateSection>
          <Input value={amount} onChange={(e) => setAmount(e.target.value)}
type="number" placeholder="Enter Amount To Donate" />
          <Donate onClick={DonateFunds}>Donate</Donate>
        </DonateSection>
        <FundsData>
          <Funds>
            <FundText>Required Amount</FundText>
            <FundText>{Data.requiredAmount} Matic</FundText>
          </Funds>
          <Funds>
            <FundText>Received Amount</FundText>
            <FundText>{Data.receivedAmount} Matic</FundText>
          </Funds>
        </FundsData>
        <Donated>
          <LiveDonation>
            <DonationTitle>Recent Donation</DonationTitle>
            {DonationsData.map((e) => {
              return (
                <Donation>
                <DonationData>{e.donar.slice(0,6)}...{e.donar.slice(39)}</DonationData>
                <DonationData>{e.amount} Matic</DonationData>
                <DonationData>{new Date(e.timestamp *
1000).toLocaleString()}</DonationData>
              </Donation>
              )
            })
            }
          </LiveDonation>
          <MyDonation>
            <DonationTitle>My Past Donation</DonationTitle>
```

```
            {mydonations.map((e) => {
              return (
                <Donation>
                <DonationData>{e.donar.slice(0,6)}...{e.donar.slice(39)}</DonationData>
                <DonationData>{e.amount} Matic</DonationData>
                <DonationData>{new Date(e.timestamp *
1000).toLocaleString()}</DonationData>
              </Donation>
              )
            })
            }
          </MyDonation>
        </Donated>
      </RightContainer>
    </DetailWrapper>
  );
}


export async function getStaticPaths() {
  const provider = new ethers.providers.JsonRpcProvider(
    process.env.NEXT_PUBLIC_RPC_URL
  );

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

  const getAllCampaigns = contract.filters.help_together_event();
  const AllCampaigns = await contract.queryFilter(getAllCampaigns);

  return {
    paths: AllCampaigns.map((e) => ({
        params: {
          address: e.args.campaignAddress.toString(),
        }
    })),
    fallback: "blocking"
  }
}

export async function getStaticProps(context) {
  const provider = new ethers.providers.JsonRpcProvider(
    process.env.NEXT_PUBLIC_RPC_URL
```

```
  );

  const contract = new ethers.Contract(
    context.params.address,
    Help_together.abi,
    provider
  );

  const title = await contract.fund_title();
  const storyUrl = await contract.story();
  const image = await contract.image();
  const requiredAmount = await contract.amount_required();
  const receivedAmount = await contract.total_amount_received();
  const owner = await contract.owner();

  const Donations = contract.filters.fund_donated_event();
  const AllDonations = await contract.queryFilter(Donations);


  const Data = {
      address: context.params.address,
      title,
      requiredAmount: ethers.utils.formatEther(requiredAmount),
      image,
      receivedAmount: ethers.utils.formatEther(receivedAmount),
      storyUrl,
      owner,
  }

  const DonationsData =  AllDonations.map((e) => {
    return {
      donar: e.args.donar,
      amount: ethers.utils.formatEther(e.args.amount),
      timestamp : parseInt(e.args.timestamp)
  }});

  return {
    props: {
      Data,
      DonationsData
    }
  }


}
```

```
const DetailWrapper = styled.div`
  display: flex;
  justify-content: space-between;
  padding: 20px;
  width: 98%;
`;
const LeftContainer = styled.div`
  width: 45%;
`;
const RightContainer = styled.div`
  width: 50%;
`;
const ImageSection = styled.div`
  width: 100%;
  position: relative;
  height: 350px;
`;
const Text = styled.p`
  font-family: "Roboto";
  font-size: large;
  color: ${(props) => props.theme.color};
  text-align: justify;
`;
const Title = styled.h1`
  padding: 0;
  margin: 0;
  font-family: "Poppins";
  font-size: x-large;
  color: ${(props) => props.theme.color};
`;
const DonateSection = styled.div`
  display: flex;
  width: 100%;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
`;
const Input = styled.input`
  padding: 8px 15px;
  background-color: ${(props) => props.theme.bgDiv};
  color: ${(props) => props.theme.color};
  border: none;
```

```
  border-radius: 8px;
  outline: none;
  font-size: large;
  width: 40%;
  height: 40px;
`;
const Donate = styled.button`
  display: flex;
  justify-content: center;
  width: 40%;
  padding: 15px;
  color: white;
  background-color: #00b712;
  background-image: linear-gradient(180deg, #00b712 0%, #5aff15 80%);
  border: none;
  cursor: pointer;
  font-weight: bold;
  border-radius: 8px;
  font-size: large;
`;
const FundsData = styled.div`
  width: 100%;
  display: flex;
  justify-content: space-between;
  margin-top: 10px;
`;
const Funds = styled.div`
  width: 45%;
  background-color: ${(props) => props.theme.bgDiv};
  padding: 8px;
  border-radius: 8px;
  text-align: center;
`;
const FundText = styled.p`
  margin: 2px;
  padding: 0;
  font-family: "Poppins";
  font-size: normal;
`;
const Donated = styled.div`
  height: 280px;
  margin-top: 15px;
  background-color: ${(props) => props.theme.bgDiv};
`;
const LiveDonation = styled.div`
  height: 65%;
```

```
  overflow-y: auto;
`;
const MyDonation = styled.div`
  height: 35%;
  overflow-y: auto;
`;
const DonationTitle = styled.div`
  font-family: "Roboto";
  font-size: x-small;
  text-transform: uppercase;
  padding: 4px;
  text-align: center;
  background-color: #4cd137;
`;
const Donation = styled.div`
  display: flex;
  justify-content: space-between;
  margin-top: 4px;
  background-color: ${(props) => props.theme.bgSubDiv};
  padding: 4px 8px;
`;
const DonationData = styled.p`
  color: ${(props) => props.theme.color};
  font-family: "Roboto";
  font-size: large;
  margin: 0;
  padding: 0;
`;
```

CreateCampaigns.js

```
import Form from "../Components/Forms/form"
const CreateCampaigns = () => {
  return (
    <div>
    <Form/>
    </div>
  )
}

export default CreateCampaigns
```

ExploreCampaigns.js

```
const ExploreCampaigns = () => {
  return (
    <div>ExploreCampaigns</div>
  )
}

export default ExploreCampaigns
```

index.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 { useState } from 'react';
import Link from 'next/link';
import Help_together_store from
'../artifacts/contracts/Help_together.sol/Help_together_store.json';

export default function Index({AllData, HealthData, EducationData,AnimalData}) {
  const [filter, setFilter] = useState(AllData);

  return (
    <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>

      {/* Card */}
      {filter.map((e) => {
```

```
        return (
          <Card>
          <CardImg>
            <Image
              layout='fill'
              src={"https://ipfs.infura.io/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 href={'/' + e.address}><Button>
            Go to Campaign
          </Button></Link>
        </Card>
        )
      })}
        {/* Card */}

      </CardsWrapper>
    </HomeWrapper>
  )
}



export async function getStaticProps() {
  const provider = new ethers.providers.JsonRpcProvider(
    process.env.NEXT_PUBLIC_RPC_URL
  );

  const contract = new ethers.Contract(
    process.env.NEXT_PUBLIC_ADDRESS,
    Help_together_store.abi,
```

```
    provider
  );

  const getAllCampaigns = contract.filters.help_together_event();
  const AllCampaigns = await contract.queryFilter(getAllCampaigns);
  const AllData = AllCampaigns.map((e) => {
    return {
      title: e.args.event_title,
      image: e.args.img_link,
      owner: e.args.creater,
      timeStamp: parseInt(e.args.timestamp),
      amount: ethers.utils.formatEther(e.args.event_requiredamount),
      address: e.args.campaignAddress
    }
  });

  const getHealthCampaigns =
contract.filters.help_together_event(null,'Health',null,null,null,null,null);
  const HealthCampaigns = await contract.queryFilter(getHealthCampaigns);
  const HealthData = HealthCampaigns.map((e) => {
    return {
      title: e.args.event_title,
      image: e.args.img_link,
      owner: e.args.creater,
      timeStamp: parseInt(e.args.timestamp),
      amount: ethers.utils.formatEther(e.args.event_requiredamount),
      address: e.args.campaignAddress
    }
  });

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

  const getAnimalCampaigns = contract.filters.help_together_event(null, 'Animal' ,
null,null,null,null,null);
```

```
  const AnimalCampaigns = await contract.queryFilter(getAnimalCampaigns);
  const AnimalData = AnimalCampaigns.map((e) => {
    return {
      title: e.args.event_title,
      image: e.args.img_link,
      owner: e.args.creater,
      timeStamp: parseInt(e.args.timestamp),
      amount: ethers.utils.formatEther(e.args.event_requiredamount),
      address: e.args.campaignAddress
    }
  });

  return {
    props: {
      AllData,
      HealthData,
      EducationData,
      AnimalData
    }
  }
}




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: 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;
`
```

YourCampaign.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 Help_together_store from
'../artifacts/contracts/Help_together.sol/Help_together_store.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);
```

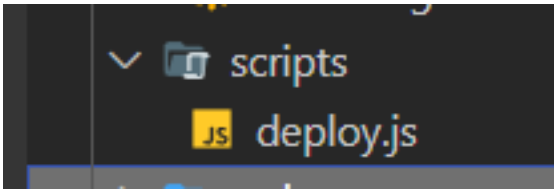
```
      const signer = Web3provider.getSigner();
      const Address = await signer.getAddress();

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

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

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

  return (
    <HomeWrapper>

      {/* Cards Container */}
      <CardsWrapper>

      {/* Card */}
      {campaignsData.map((e) => {
        return (
          <Card>
          <CardImg>
            <Image
              layout='fill'
              src={"https://ipfs.infura.io/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 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;
```

Scripts:-



deploy.js

```
const hre = require('hardhat');

async function main() {
    const CampaignFactory = await hre.ethers.getContractFactory("Help_together_store")
    const campaignFactory = await  CampaignFactory.deploy();
    await campaignFactory.deployed();
    console.log("factory deployed to :" , campaignFactory.address);
  }

  main()
    .then(() => process.exit(0))
    .catch((error) => {
      console.log(error);
      process.exit(1);
    });
```

Test.js

```
const   Help_together_store =
require("./artifacts/contracts/Help_together.sol/Help_together_store.json");
const { ethers } = require("ethers");
require("dotenv").config({ path: "./.env.local" });

const main = async () => {
    const provider = new ethers.providers.JsonRpcProvider(
      process.env.NEXT_PUBLIC_RPC_URL
    );

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

    const getDeployedCampaign = contract.filters.help_together_event(null,
'Health',null,null,null,null,null);
    let events = await contract.queryFilter(getDeployedCampaign);
    let event = events.reverse();
    console.log(event);
};

main();
```

CONCLUSION

Our Project, "Web2gether: Crowdfunding Platform Based on Blockchain", is complete, and fully functional.

Conventional crowdfunding methods have long suffered from lack of transparency and fraud. It is an avoidable problem, and we believe that we have implemented a solid solution that can do away with these long-standing problems.

The aim to have a transparent, anti-fraudulent, decentralized platform has been achieved to a great extent. This project has covered the weak points of general crowdfunding platforms to provide transparency to the process of crowdfunding and build trust among people, so that they may contribute their wealth to good causes without fear of fraud.

The paper proposes a smart contract-based solution to enable secure way of crowd funding by ensuring that the money donated by the investors is safe and also each and every step taken in the startup with help of donated money involves investor's opinion i.e. whenever the campaign creator wants to spend the money he/she has to make a spending request where the purpose of using the money, to whom the money is being sent(vendor) and the amount needed should be mention.

The main advantage of using the smart contract which is a concept of blockchain is that it is resilient against many threats. Also it provides many features like improved reliability, faster and efficient operation. Web

Application has been computed successfully and was also tested by taking “test cases”.

It is user friendly, and has required options, which can be utilized by user to perform the desired operation. The goals achieved by the software are:

- Creating a campaign
- Contributing to campaign
- Approving the spending request
- Finalizing payment

Limitations regarding the project are that the vendor address is not verified by which campaign creator and the vendor can get together and scam the contributors. Also one person can contribute only one time for campaign/startup from a account as of now.

The Ethereum accounts from which the contributors are investing also not verified. Future works will aim to find a way to verify the Ethereum accounts whether it may be Meta mask or MyEtherWallet(Ethereum wallet similar to meta mask) accounts with help of a government id so that one person cannot have more than one account and also make sure that a person is able to contribute as many times he/she wants, so that the Crowdfunding using blockchain becomes more secure and reliable.

REFERENCES

Below given links are the references from where I have gathered some information for making web dapp.

- https://nextjs.org/
- https://www.blockchain.com/
- https://dev.to/ruppysuppy/how-i-created-a-crowdfunding-platform-with-web3-micro-frontends-3pb2
- https://metamask.io/
- https://web3js.readthedocs.io/en/v1.3.4/