BETTERFUND

A MINI-PROJECT REPORT

Submitted by

PRAVEEN PRAKASH TIWARI[RA2011050010003]
MD ANAS HASAN [RA2011050010011]

Studying

B. Tech

Under the Guidance of

Dr. SV Shri Bharathi

Assistant Professor, Department of DSBS

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech mini-project report titled "**BETTERFUND**" is the bonafide work of **PRAVEEN PRAKASH TIWARI** and **MD ANAS HASAN** who carried out the mini-project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. SV Shri Bharathi
SUPERVISOR
Assistant Professor
Department of DSBS

Dr. M. Lakshmi
PROFESSOR & HOD
Department of DSBS

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

Digital Crowdfunding distribution is increasingly powered by automated mechanisms that continuously capture, sort and analyze large amounts of Web-based data. **BETTERFUND** deals with the management of monetary collection features from a statistical point of view.

In particular, it explores the data catching mechanisms enabled by **BETTERFUND** Web API, and suggests statistical tools for the analysis of these data. Special attention is devoted to money popularity and a **Beta model** including random effects is proposed in order to give a first answer to questions like: which are the determinants of popularity? The identification of a model able to describe this relationship, the determination within the set of characteristics of those considered most important in making a song popular is a very interesting topic for those who aim to predict the success of new products.

The **BETTERFUND** mainly consists of three main parts:

1) Two main infrastructures that keep it running:

- **Proliferating data infrastructure**: Following an "end-to-end server and client model", BETTERFUND's proliferating data infrastructure, proposed by Eriksson et al., is the foundation of its service. This infrastructure enables the communication between BETTERFUND's servers and their clients' devices.

- **Social** : The second infrastructure is the audio and streaming infrastructure. BETTERFUND balances the file size and the speed of the internet very well since its streaming service experiences a very low latency, that is the delay, between a user requesting a song and hearing it, is almost imperceptible. BETTERFUND's low latency streaming is owed to Ogg Vorbis format, an open-source loss audio compression method "that offers roughly the same sound quality as mp3, but with a much smaller file size".

2) Three ways **BETTERFUND** works as an interface: interface between websites, between human and devices as well as between human and the large computing system.

3) **BETTERFUND** as a sociotechnical system.

TABLE OF CONTENTS

INTRODUCTION

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.

- **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.

We were highly inspired by the CryptoRelief initiative (www.cryptorelief.in) [2] which raised ~1 billion dollars for Covid Relief in India from the entire global community, in a highly transparent manner.

LITERATURE REVIEW

Feasibility: Technical and Non-Technical

● Technical Feasibility

○ It is to be a ReactJS based application, which will be supported by any web browser.

○ Internet connectivity will be required.

○ Users will require ‘Metamask’ browser extension to sign transactions.

● Social Feasibility

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

● Economic Feasibility

○ Given the Ethereum Blockchain provides us with most of the security features, the development does not require much cost.

○ The only cost would be the server cost of the deployed application.

● Scope

○ With Betterfund we aim to make the crowdfunding process transparent, anti-fraudulent and secure.

This pain point cropped up again and again in our research. money collection platforms would benefit from tweaking algorithms to ensure recommendations are relevant to the user's listening habits.

1. Unwanted ads

A common complaint on most online software platforms is the prevalence of ads. Ads are common on streaming platforms, particularly for consumers who don't pay to subscribe. However, it's important to strike a balance.

Finding natural breaks in a user's listening experience is vital to keep engagement high. For example, an ad in the middle of a key moment in a podcast may cause frustration. Equally, playing too many ads is an easy way to invite complaints.

2. Offline use

For most users, money collection needs to be available everywhere – including offline. A key pain point among listeners is the inability to play money without WIFI or data connection. Whether they're on a plane or running low on monthly data allowances, users want to listen regardless.

money collection streaming services might benefit from making offline listening easier for users. Whether that's by introducing features to make it easier, or by educating users on existing features.

3. Unavailability of money

Users are less likely to have more than one money collection streaming service, compared to TV and movie streaming. The average American subscriber uses 3.4 video streaming services, while only 21% of money collection streaming users have more than one service.

With this, money collection services need to ensure they can act as a one-stop-shop for a user's listening habits.

A common pain point is a lack of availability for certain money. Needing to go to a different platform to listen to an artist isn't appreciated. Maintaining good relationships with artists and content creators is one way to combat this, though that might be easier said than done.

4. The cost of money collection services

Most money collection-lovers understand that streaming services can't be free. However, overpriced services are definitely not appreciated. Often, these complaints are linked to in-app bugs or server errors. When experiencing a bug, a user is less likely to think the platform is worth the cost.

Investing in app infrastructure is an important way to ensure users are getting value for money.

The bottom line

money collection streaming services are only growing in popularity. With more customers opting for on-the-go listening, having a user-friendly platform is important.

With this in mind, it's vital for money collection streaming services to be on the pulse of their users' needs. Whether it's in-app bugs, value for money, or inadequate AI, understanding your customers is necessary when making an app which caters to everyone.

CHAPTER 3
SYSTEM ANALYSIS

3.1 PROBLEM STATEMENT:

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 [1] — which has never been compromised yet — can provide that level of security.

- **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.

3.2 PROPOSED 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.

We 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.

3.3 SOFTWARE and HARDWARES

1. Software Requirements

Operating System:

iOS	iOS 13 or above
Android	Android OS 5.0 or above
Mac	OS X 10.13 or above
Windows	Windows 7 or above

Tools:

Figma is a modern, streamlined design tool that makes the developer handoff Process smooth and seamless. Developers now have better access to the designs and have the ability to extract information about typography, redlines, colors, measurement and more.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

SYSTEM ARCHITECTURE:

A Software architecture refers to the blueprint/approach used to build software. Different architectures rely on different standards for building, integrating, and deploying components. There are two common architectures, Monolithic Architectures and Micro services Architectures, with micro services being the most recent architecture.

UML Diagram

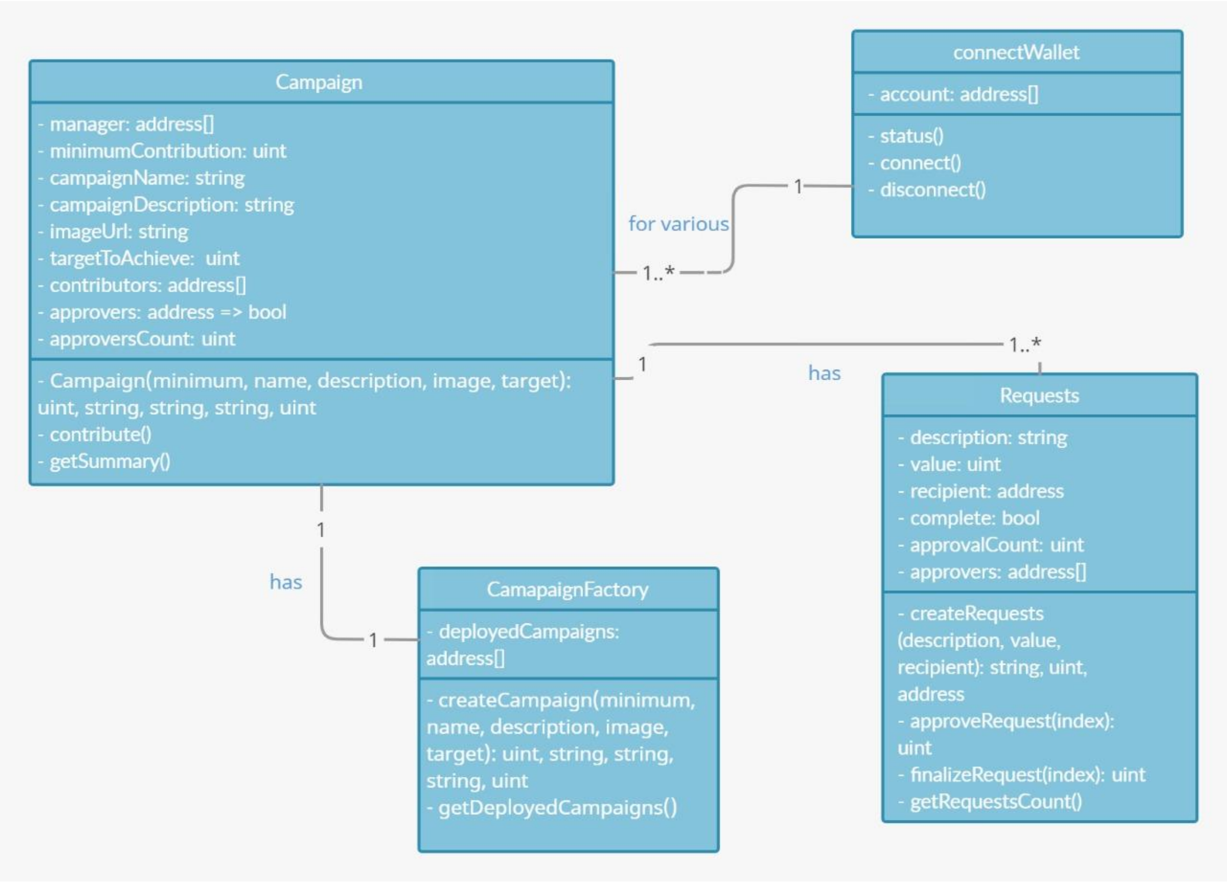
Class Diagram

The **Classes** defined are:

- ➔ Campaign
- ➔ Campaign Factory
- ➔ Requests
- ➔ connect Wallet

The **Relationships** defined are:

- ➔ A User connects his wallet to support various campaigns; one to many.
- ➔ A campaign Factory has its Campaign; one to one.
- ➔ A Campaign has multiple Requests; one to many.



MONOLITHIC:

This is the industry standard of software development, where software is designed to be a single executable unit. This architecture is ideal for applications where requirements are fixed.

In a monolithic architecture, we divide an application into layers, with each layer providing specific functionality:

Presentation Layer: This layer implements the application UI elements and client-side API requests. It is what the client sees and interacts with.
Controller Layer: All software integrations through HTTP or other communication methods happen here.
Service Layer: The business logic of the application is present in this layer.
Database Access Layer: All the database accesses, including both SQL and NoSQL, of the applications, happens in this layer.
We often group layers together, with the Presentation Layer being called the frontend and Controller, Service, and Data Access Layer being grouped into the backend*. This simplifies software as communication between two parties. Any application can be described as a frontend (client) talking to a backend (server).

Dividing applications into these layers led to design patterns like MVC, MVVC, MVP, as well as frameworks that implement them like Spring for Java, .NET for C#, Qt for C++, Django for Python, and Node.js for JavaScript.

MICROSERVICES:

Micro services build off monolithic architectures. Instead of defining software as a single executable unit, it divides software into multiple executable units that interrupt with one another. Rather than having one complex client and one complex server communicating with one another, micro services split clients and servers into smaller units, with many simple clients communicating with many simple servers.

In even simpler terms, micro services split a large application into many small applications.

The tradeoff between the two is summarized below:

Monolithic Architecture: Complex Services, Simple Relationships. Better for apps with Fixed Requirements (like a Calculator)

Microservices Architectures: Simple Services, Complex Relationships. Better for apps with Variable/Scaling Requirements (like a Social Media application)
Microservices borrows the exact same design patterns and layer methodology as Monolithic architectures, it only implements them with different tools.

MODULES:-

Frontend:

The frontend is the graphical UI of an application or site that the client interacts with. Webpage frontends have the option of being pre rendered on a server and sent to a browser (server-side rendering aka SSR) or rendered directly in a browser (client-side rendering aka CSR). Application front ends are usually downloaded (as seen with Desktop and Mobile applications). CSR is more commonplace as UI components can be dynamically rendered and updated with lower latencies. Many frontend UI frameworks exist with many being in JS, but CSR can also be done through other languages like C#, C++, Java, and more using Web Assembly.

HTML5:

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is an abbreviation of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text document within the tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model(DOM).

CSS5:

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate.css file, and reduce complexity and repetition in the structural content. CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading. One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

JAVASCRIPT:

JavaScript is a high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype- based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web.
JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs forworking with text, arrays, dates, regular expressions, and the DOM, but the language itself doesnot include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features. Initially only implemented client- side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such asword processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets. The terms Vanilla JavaScript and Vanilla JS refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code. Chrome extensions, Opera's extensions, Apple's Safari 5 extensions, Apple's Dashboard Widgets, Microsoft's Gadgets,Yahoo! Widgets, Google Desktop Gadgets, and Serene Klipfolio are implemented using JavaScript.

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:

1. **NextJS :** Next.js is an open-source React front-end development web framework that enables functionality such as server-side rendering and generating static websites for React based web applications.

2. **Chakra UI :** Chakra UI is a simple, modular and accessible component library that gives the building blocks one needs to build React applications.

3. **Solidity :** It is the programming language for implementing Ethereum based Smart Contracts.

4. **Web3** : web3.js is a collection of libraries that allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket.

5. **Ethereum Smart Contract** : It is the collection of functions and data that reside at a specific address on the Ethereum Blockchain.

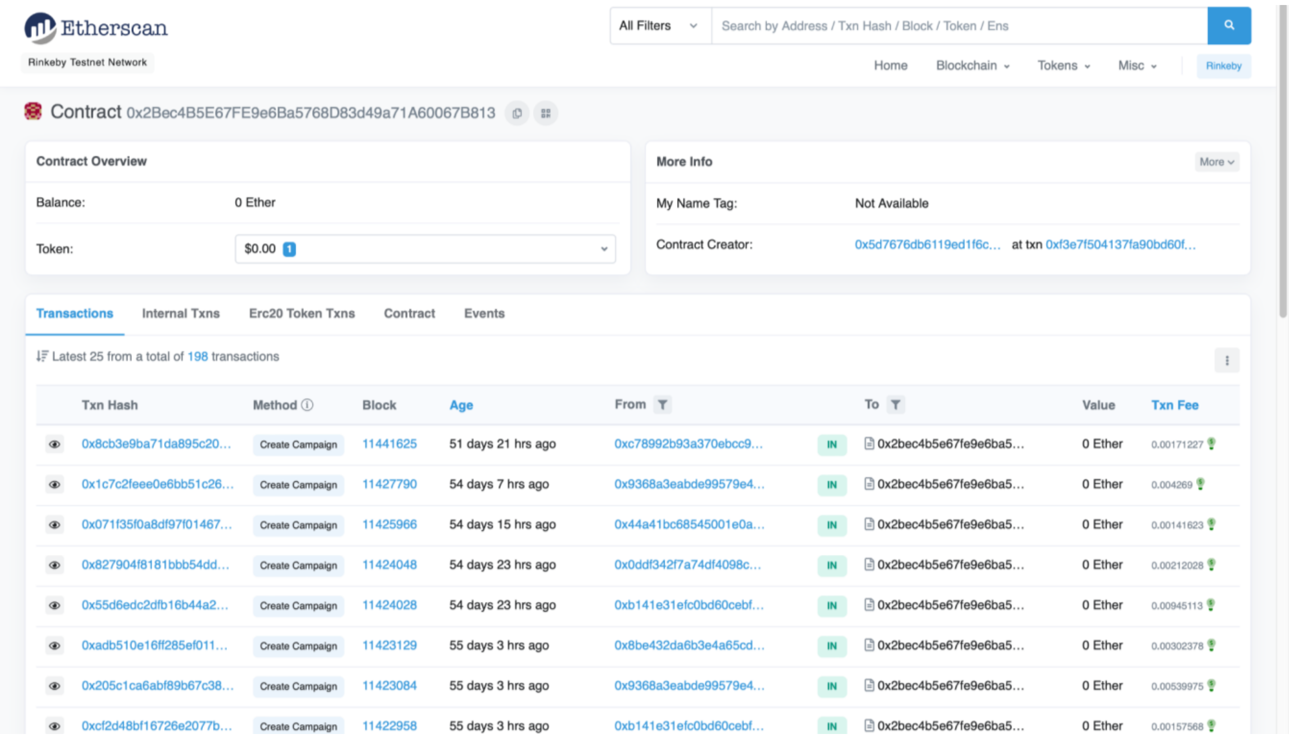
SOURCE CODE:-

```
pragma solidity ^0.4.17;

contract CampaignFactory {
    address[] public deployedCampaigns;

    function createCampaign(uint minimum,string name,string description,string image,uint target) public {
        address newCampaign = new Campaign(minimum, msg.sender,name,description,image,target);
        deployedCampaigns.push(newCampaign);
    }

    function getDeployedCampaigns() public view returns (address[]) {
        return deployedCampaigns;
    }
}


contract Campaign {
  struct Request {
      string description;
      uint value;
      address recipient;
      bool complete;
      uint approvalCount;
      mapping(address => bool) approvals;
  }

  Request[] public requests;
  address public manager;
  uint public minimunContribution;
  string public CampaignName;
  string public CampaignDescription;
  string public imageUrl;
  uint public targetToAchieve;
  address[] public contributers;
  mapping(address => bool) public approvers;
  uint public approversCount;


  modifier restricted() {
      require(msg.sender == manager);
      _;
```

```
      require(msg.sender == manager);
      _;
  }

  function Campaign(uint minimun, address creator,string name,string description,string image,uint target) public {
      manager = creator;
      minimunContribution = minimun;
      CampaignName=name;
      CampaignDescription=description;
      imageUrl=image;
      targetToAchieve=target;
  }

  function contibute() public payable {
      require(msg.value > minimunContribution );

      contributers.push(msg.sender);
      approvers[msg.sender] = true;
      approversCount++;
  }

  function createRequest(string description, uint value, address recipient) public restricted {
      Request memory newRequest = Request({
         description: description,
         value: value,
         recipient: recipient,
         complete: false,
         approvalCount: 0
      });

      requests.push(newRequest);
  }

  function approveRequest(uint index) public {
      require(approvers[msg.sender]);
      require(!requests[index].approvals[msg.sender]);

      requests[index].approvals[msg.sender] = true;
      requests[index].approvalCount++;
  }

```

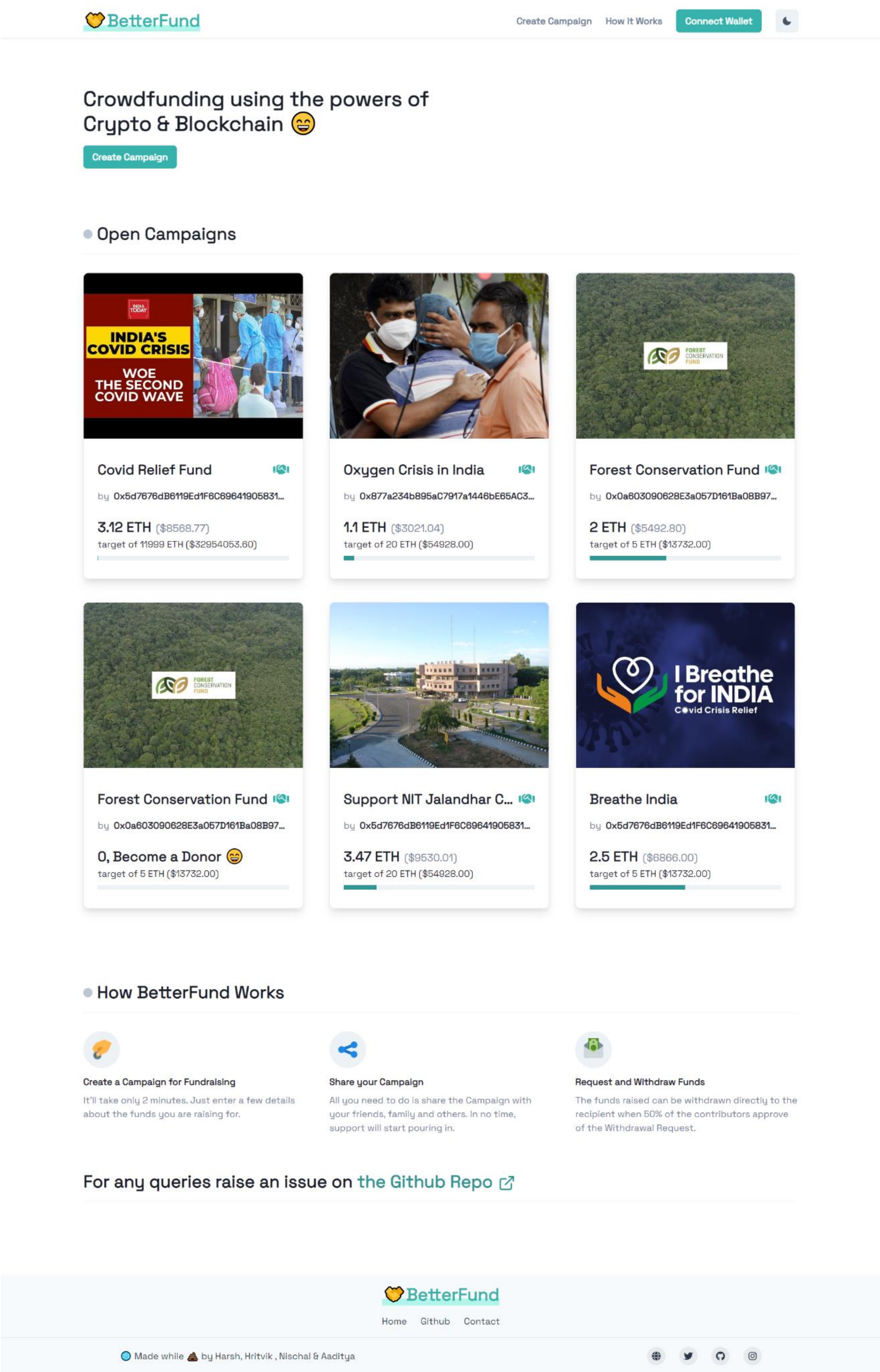
```
        });

        requests.push(newRequest);
    }

    function approveRequest(uint index) public {
        require(approvers[msg.sender]);
        require(!requests[index].approvals[msg.sender]);

        requests[index].approvals[msg.sender] = true;
        requests[index].approvalCount++;
    }

    function finalizeRequest(uint index) public restricted{
        require(requests[index].approvalCount > (approversCount / 2));
        require(!requests[index].complete);

        requests[index].recipient.transfer(requests[index].value);
        requests[index].complete = true;

    }


     function getSummary() public view returns (uint,uint,uint,uint,address,string,string,string,uint) {
         return(
             minimunContribution,
             this.balance,
             requests.length,
             approversCount,
             manager,
             CampaignName,
             CampaignDescription,
             imageUrl,
             targetToAchieve
           );
     }

     function getRequestsCount() public view returns (uint){
         return requests.length;
     }
}
```

Experiment Results :

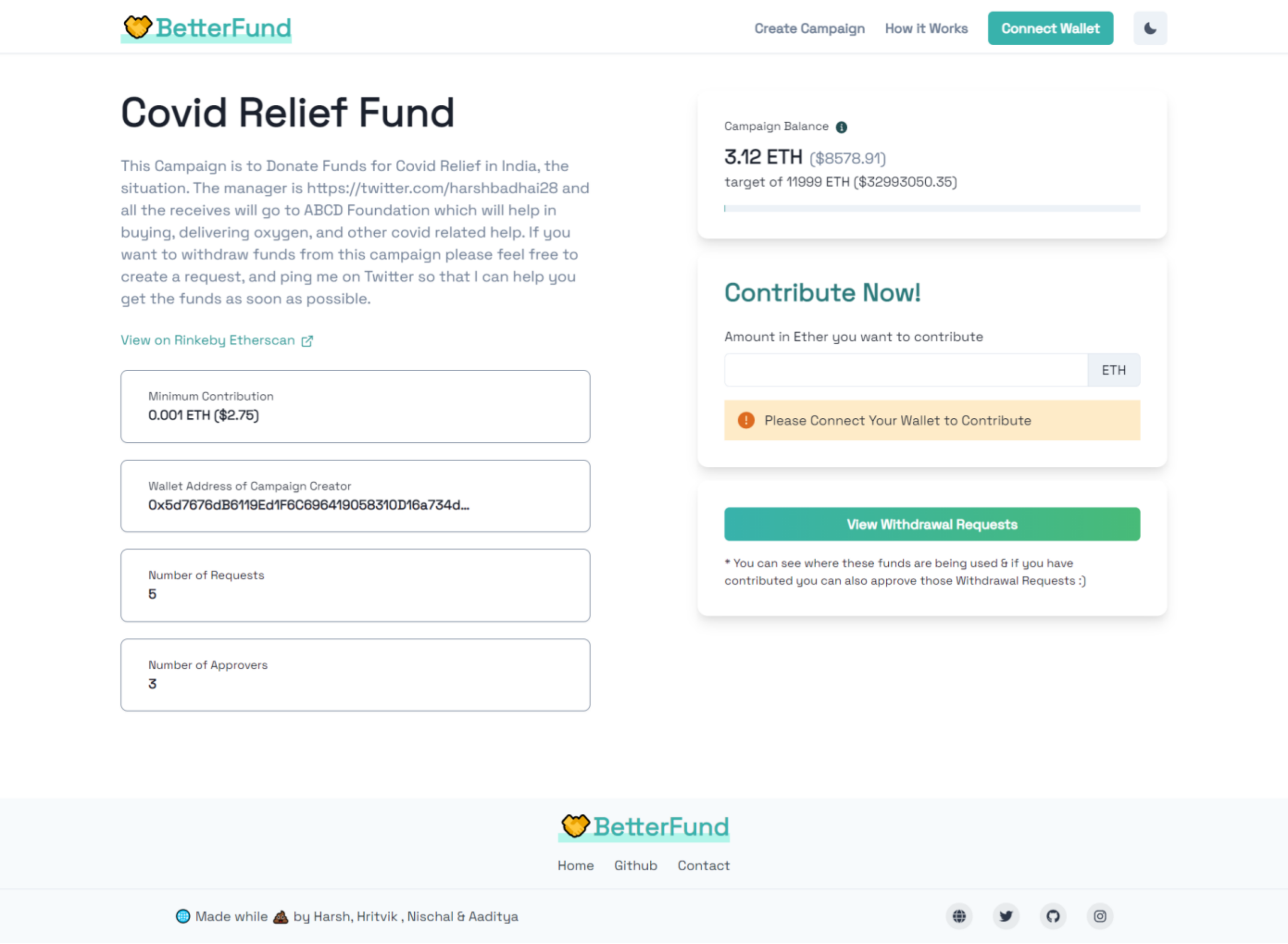


OUTPUT :

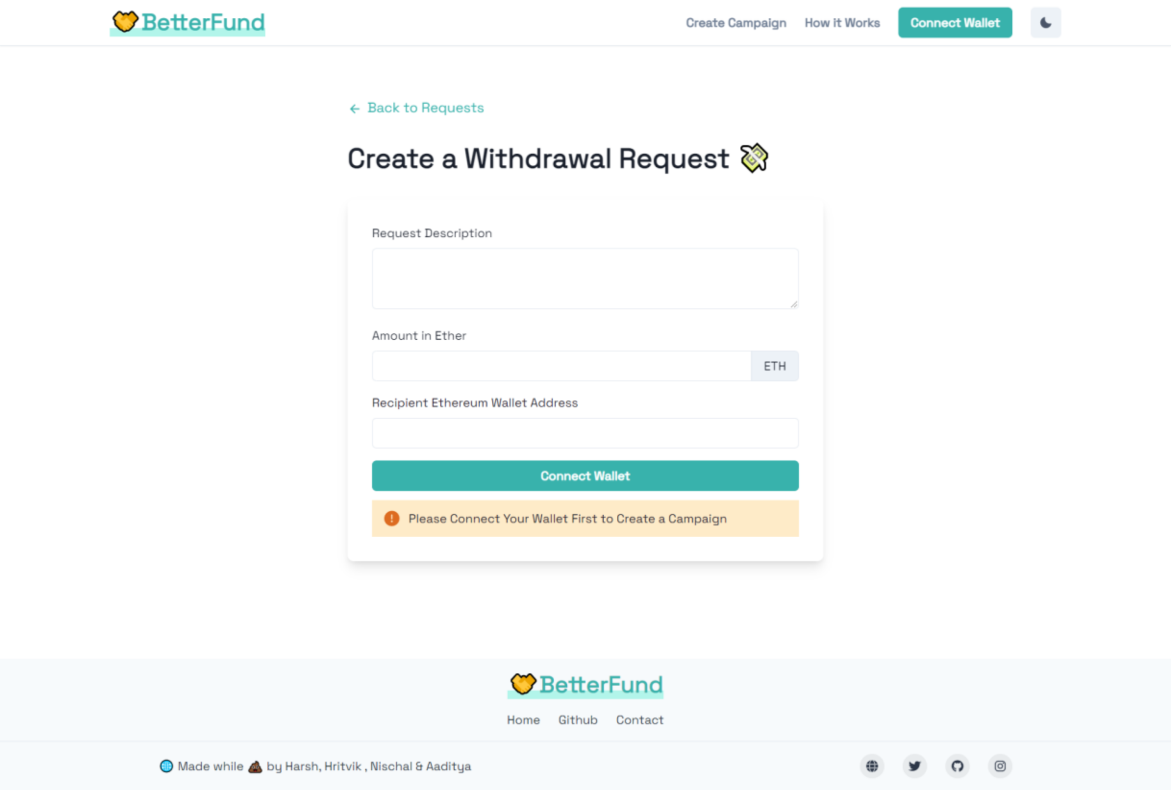
Home Page:



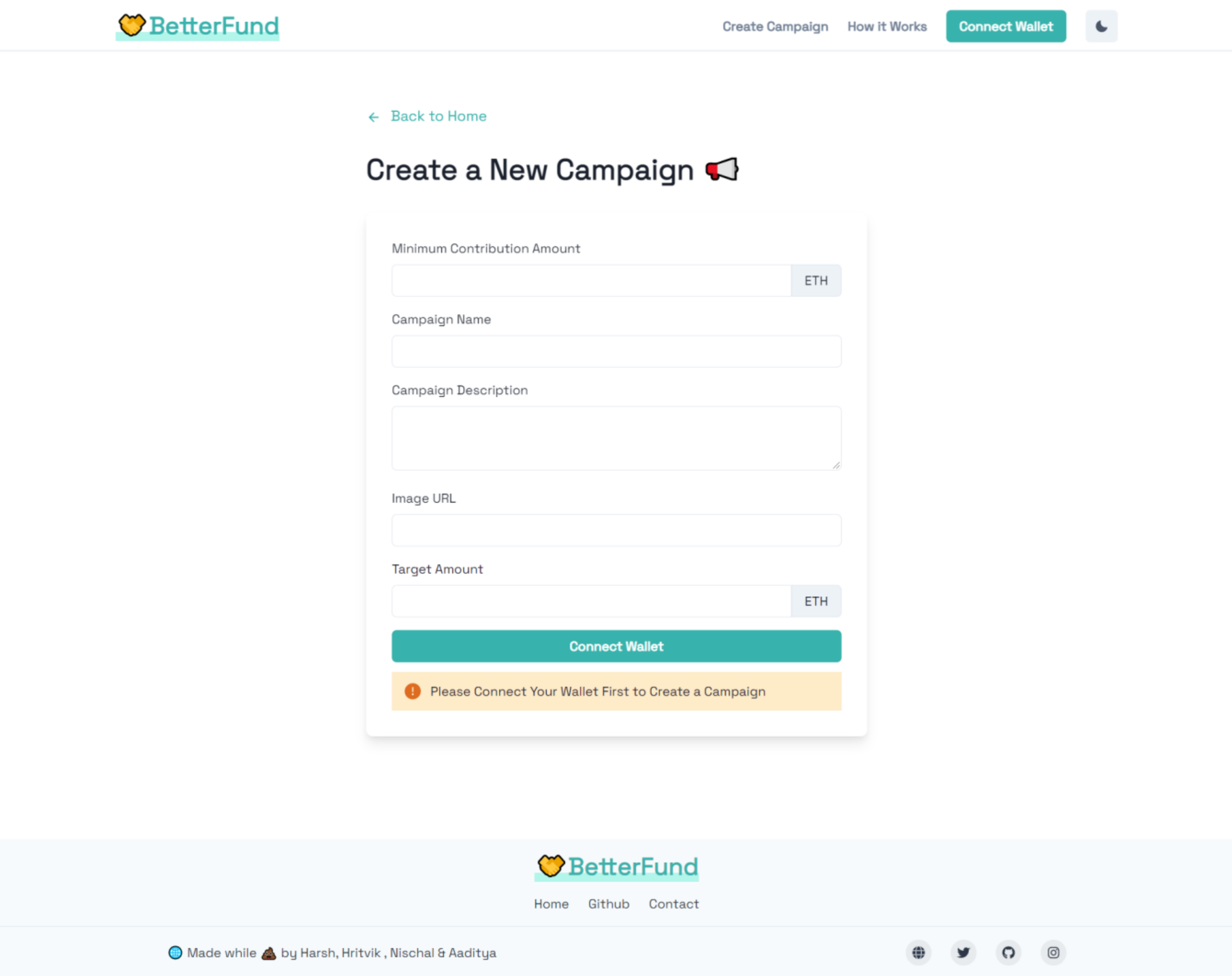
Campaign Page :



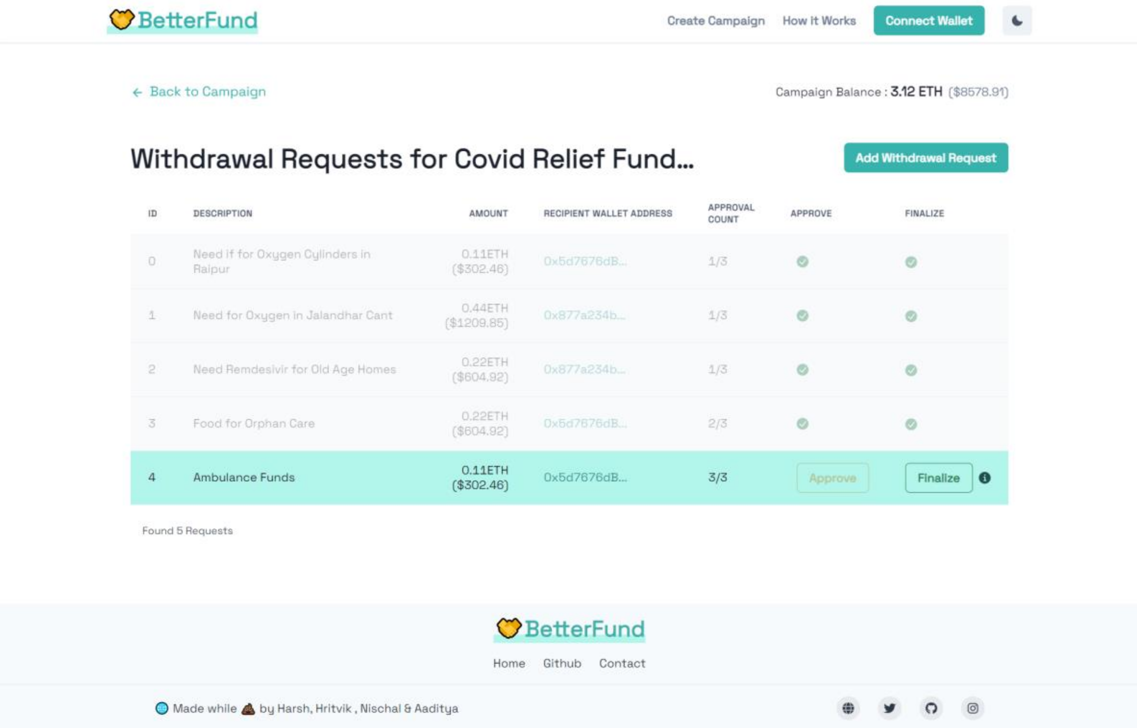
New Withdrawl Request Page :



Create Campaign Page :



Withdrawal request Page :



CHAPTER 5

CONCLUSION

Our Project, “BetterFund: Crowdfunding Platform powered by Blockchain”, is complete, live 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.

CHAPTER 6

REFERENCE

1. Blockchain & Smart Contracts:

   https://www.dappuniversity.com/articles/how-to-build-a-blockchain-app

2. CryptoRelief platform: https://cryptorelief.in

3. Next JS Documentation: https://nextjs.org/

4. Learning Solidity Language: https://cryptozombies.io/

5. web3.js - Ethereum JavaScript API: https://web3js.readthedocs.io/en/v1.3.4/

6. How data is stored in Ethereum Blockchain:

   https://laurentsenta.com/articles/storage-and-dapps-on-ethereum-blockchain/

7. Metamask Ethereum Wallet : https://metamask.io/

8. Rinkeby Ethereum Test Network : https://www.rinkeby.io/#stats

9. SAFEGUARDING THE RECORDS OF HOMES, LANDS, AND PROPERTY FOR DISPLACED PEOPLES USING DECENTRALIZED TRUST - DR. VICTORIA LEMIEUX

10. SEARCHING FOR TRUST: BLOCKCHAIN TECHNOLOGY IN AN AGE OF DISINFORMATION - Victoria L. Lemieux, University of British Columbia, Vancouver

11. A Review on Blockchain Technology and Blockchain Projects Fostering Open Science - Dr. Stephan Leible