University of Mumbai

Smart Crowdfunding using Blockchain

Submitted at the end of semester VII in partial fulfillment of requirements

For the degree of

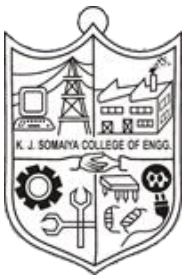
Bachelors in Technology

by

Urmil Chandarana (181108)
Dhairya Mehta (1811023)
Harshavardhan Talele (1921002)
Niha Shaikh (1921006)

Guide

Prof. Swapnil Pawar



Department of Computer Engineering
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)
Batch 2018 -2022

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that the dissertation report entitled **Smart Crowdfunding using Blockchain** submitted by

- Urmil Chandarana
- Dhairya Mehta
- Harshavardhan Talele
- Niha Shaikh

at the end of semester VII of LY B. Tech is a bonafide record for partial fulfillment of requirements for the degree of Bachelors in Technology in Electronics and Telecommunication Engineering of University of Mumbai.

_____ _____

Guide Head of the Department

_____

Principal

Date:

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Smart Crowdfunding using Blockchain** is bonafide record of project work done by

- Urmil Chandarana
- Dhairya Mehta
- Harshavardhan Talele
- Niha Shaikh

during semester VII. This project work is submitted at the end of semester VII in partial fulfillment of requirements for the degree of Bachelors in Technology in Electronics and Telecommunication Engineering of University of Mumbai.

_____

Guide/Examiner1

_____

Expert/Examiner2

Date:

Place: Mumbai-77

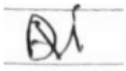
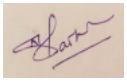
K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written report submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

 **Signature of the Student** **Urmil Chandarana (1811008)**	 **Signature of the Student** **Dhairya Mehta (1811023)**
 **Signature of the Student** **Harshavardhan Talele (1921002)**	 **Signature of the Student** **Niha Shaikh (1921006)**

Date:

Place: Mumbai-77

Abstract

Crowdfunding is the use of small amounts of capital from a large number of individuals to finance a new business venture. Crowdfunding makes use of the easy accessibility of vast networks of people through social media and crowdfunding websites to bring investors and entrepreneurs together, with the potential to increase entrepreneurship by expanding the pool of investors beyond the traditional circle of owners, relatives, and venture capitalists.

In most jurisdictions, restrictions apply to who can fund a new business and how much they are allowed to contribute. Similar to the restrictions on hedge fund investing, these regulations are supposed to protect unsophisticated or non-wealthy investors from putting too much of their savings at risk. Because so many new businesses fail, their investors face a high risk of losing their principal.

When you use a crowdfunding site you will need to register and create a profile, then you can create your own personal campaign or fundraising page where you can tell the story of why you are raising money and what you need it for, set up a fundraising goal, and start raising money. Depending on the type of crowdfunding site you decide to use, you can start collecting the money within days. Funds are deposited directly to the financial institution account you link to your fundraising account upon registration

keywords: blockchain, smart contract, decentralized, ethereum

Table of Contents

List of Images

List of Tables

Chapter 1 Introduction

This chapter presents an overview about the motivation, need and background of the project and topic chosen for final year project.

1.1 Background

Crowdfunding is a way to raise money from a large number of individual contributors or companies. Crowdfunding using blockchain changes the traditional way to deal with philanthropic funding. Generally, when people need to raise a cash to begin a charity event, they have to design strategy, statistical surveying, and models, and afterward present the thoughts around to attract people or organizations. These subsidizing sources included banks, angel contributors, venture capital firms. The present-day crowdfunding model depends on three kinds of on-screen characters: the task initiator who proposes the thought or venture to be funded, people or contributors who invests in the thought, and a platform which puts these two characters together to make the venture successful. Hence this can be used to finance a wide scope of start-ups, pioneering ideas, such as, innovative activities, medical advances, travel and social business enterprise ventures as well.

1.2 Motivation

As a revolutionary technology for recordkeeping, Blockchain is poised to change the future of finance - in accounting, asset registers, payments, trading, collateral management, and more. Hence, our curiosity to learn and work on this technology has really driven us as we wanted to explore more in this subject.

As of today, crowdfunding platforms have accountability and trust problems. In many cases, money from contributors/philanthropists has gone into wrong campaigns and has been misused and implementing a blockchain-based platform can bring in a change has been the main reason.

1.3 Scope

With this project and blockchain smart-contracts, contributors would be informed about the payments to be processed by the Fundraiser through request money forms. A smart contract helps to block the funds within blockchain until the campaign organizer makes progress in the campaign. With the help of our application, people would be able to create campaigns to raise funds for natural calamities, start-ups or any other social/personal causes. Similarly, on the other end the general public can donate funds for these campaigns.

Our application goal is to create transparency between contributors and the campaign organizers in the respect of how the money is spent and where it is spent. The organizers would not be able to spend the money without informing the contributors about the details of spending.

The project is limited for Relief Fund Raising. The project aims to notify all the contributors where the payment for a particular amount is to be made by the Fundraiser but the project does not provide post-payment tracking of the funds for now. We aim to accomplish decentralized crowdfunding applications using blockchain and smart contracts.

Chapter 2 Literature Survey

This chapter presents an overview of the literature survey for the problem statement chosen. Literature survey for this project was an iterative process where continuous assessment and distilling information was required. It helped in investigating the problem and coming to the best possible solution.

Venturing Crowdfunding using Smart Contracts in Blockchain[1]

Referring to this paper, this paper proposes blockchain based crowd funding by using which the platform can give a private, secure and decentralized path for crowdfunding. The main objective of this paper is to let investors contribute to any project effectively by creating smart contracts through which the contributors can have a control over the invested money and also both the project creators and investors can effectively make and reserve funding for the project.

Blockchain in crowdfunding allows decentralization which means that no individual platform or group of platforms control the smart contracts which makes it transparent to everyone in the blockchain. It's a peer to peer network which collectively follows to a protocol for inter-node communication and validate new block, so no one can alter any block without approval of more than 50 percent nodes in the blockchain which makes it secure and safe. Any one can create the project in the website with blockchain and any one who has internet connectivity can donate to the project. Contributors do not have to worry about the empty promises like the traditional crowdfunding. The smart contracts will handle all the transactions so all the money will be stored in smart contracts rather than sending to the third party. Blockchain gives more freedom to project managers and the contributors so that contributors can have fractional contribution to the project.

The smart contract is compiled using the solidity compiler. This gives bytecode and application binary interface as output. Bytecode is then deployed to ethereum blockchain and application binary interface is used to interact with smart contract. Bytecode is hexadecimal representation of the compiled contract which can only be understood by Ethereum Virtual Machine (EVM). The bytecode obtained from the compilation can be

deployed to either rinkeby test network, robsten test network or ethereum live network. After deploying they return the address where the smart contract is deployed using which user can do make the transactions.

The paper also proposes a solution where they have used smart contracts in a way where the campaign organizer has to request for approval to spend the money from the donors and the decision is done by a voting system. The outcome of voting decides whether the money can be used for the required task. Our plan is to deploy a similar voting based smart contract based on our requirements.

After deploying the project, a decentralized web app was created with a frontend for creating a new project, contributing to a project, creating a new request, approving a request and finalizing a request. With the evolution of blockchain, this proposed work has a bright future and a large scope for improvement and evolution. We aim to progress further in an easier and safer way for all ideas that are achieved through the proposed crowdfunding application.

Proposed Solution for Trackable Donations using Blockchain [2]

The blockchain provides a means to obtain a decentralized transaction ledger that can be used to generate, validate and send transactions to other nodes present in the same network. Various cryptographic hash functions of specific cryptocurrencies also increase the security that is needed during financial transactions. The blockchain can be applied to financial services, healthcare services and business and industry. A charity application today needs a system that validates itself without depending on any other system or application. Blockchains are being used as they are not restricted to a particular system and because they can independently verify the integrity and consistency of transactions. Ethereum is chosen as a platform because it is a public platform and has better scalability. It can run 7-20 transactions per second. Through blockchain, the charity system will no longer be monopolised and restricted to one authority. The public will have easy access to the transactions and can verify if their money is being used like they expected. Blockchain is being used by financial institutions to increase cyber security. The advantages of blockchain are that it is fast, cheaper, has a decentralized registry and provides secure payment information. In India, an Aadhar number is issued to all Indian citizens that asserts

their biometric data along with their location and other details. The Aadhar can be utilized along with Blockchain technology for many applications like healthcare and voting. Data loss due to single point failure and privacy disclosure can be eliminated through Blockchain. Consensus protocol is of a large significance as it decides the parameters on which the new node is validated. An inappropriate consensus protocol may lead to undesirable results while using the application. The challenges faced by a blockchain application are the need of resources and scalability.

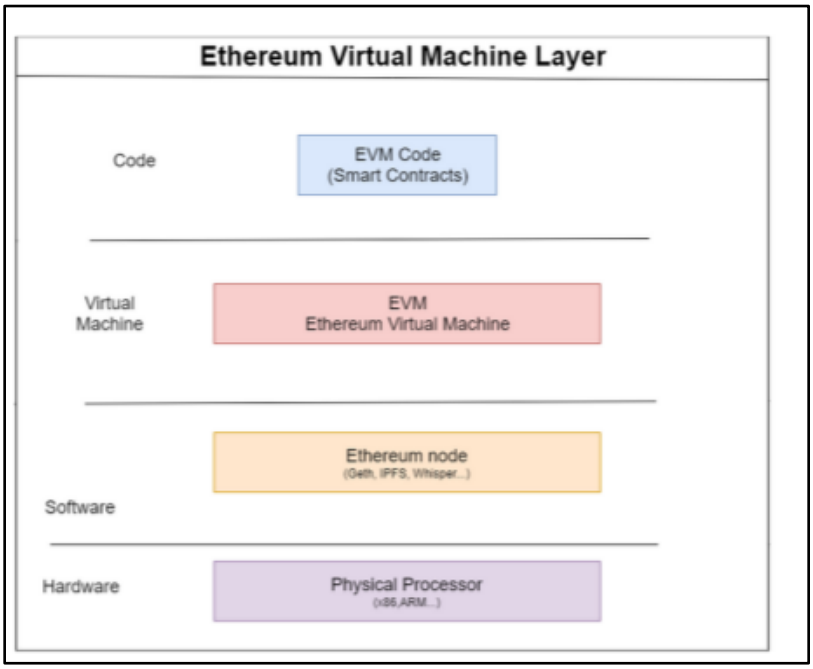


Figure 1: Ethereum Architecture

The paper establishes the idea behind the need for a solid smart contract and the reason for providing the transparency. Our aim is to implement a tracking system similar to the one mentioned in the paper where the contributor not only knows about whether the money has been used or not but also gets a say and can see the reasons and the use of money and be assured that the money is being used for the required purposes. The paper also helps us in creating a structure for the deployment and the contents that would entail on the dashboard. The paper also provides with the differences between different consensus protocols, the need for selecting the correct mechanism and how to decide between different consensus mechanisms for our particular use case. We also learnt about the importance of EVM I.e. Ethereum Virtual Machine and how to go about using EVM IDEs

such as Remix to help write smart contracts. We also used Ganache for setting up Ethereum Blockchain and to verify the working of Solidity contracts.

KYC in Tecra.space[5]

The process of KYC followed by Tecra.space is as follows:

- For KYC procedure a user submits documents to one of the administrators where he wants to take a loan or use another service.
- Individual participants are responsible for collecting personal data (administrator, government agencies, companies, or users themselves) and stored in a decentralized network.
- The administrator checks and confirms the passage of KYC if everything is normal.
- The administrator is responsible for entering the data about the user into the blockchain platform, to which other administrator, organizations and state structures have access. All parties can control and regulate the KYC process. The system will monitor changes and updating of the user data, and if someone breaks the rules, it will become known to all parties.
- When a user wants to use the services of another administrator, this second administrator accesses the system and thus confirms the user's identity.
- The access to user data will be based solely on its consent. The user must log in with cryptocurrency transactions i.e. use the private key to initiate the information exchange operation.

Chapter 3 Project Design

This chapter presents an overview of system architecture used in the project, The project management plan and Software Requirement Specification of the project. It also consists of several design diagrams to understand process involved in different functionalities of the application.

3.1 Proposed System Model

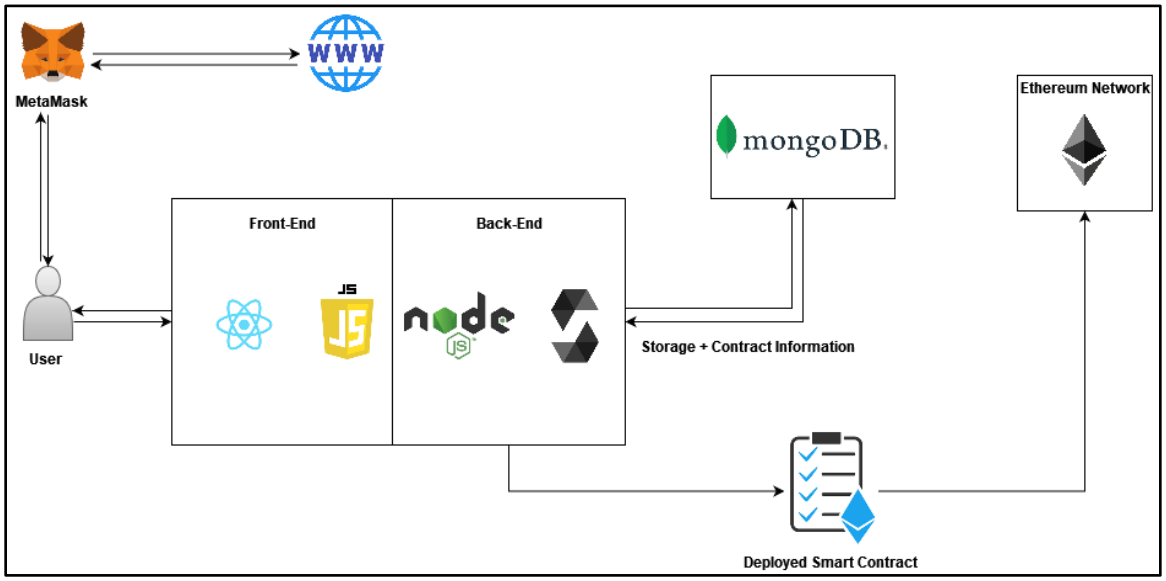


Figure 2: System Architecture

Below are the system components used:

- React.js - an open source Javascript library that we have used for building user interfaces
- MongoDB - an open source NoSQL database management program
- Node.js - running environment for a JavaScript programming language that holds many excesses; it requires libraries that can easily be accessed from JavaScript programming for better use
- MetaMask – to allow users to store and manage account keys, broadcast transactions, send and receive Ethereum-based tokens, and securely connect to decentralized applications through a compatible web browser

The backend data is stored in MongoDB with Node.js to manipulate and write queries to access or store the data. The users being campaign organizers with the Meta Mask account would deploy the smart contracts onto the Ethereum network to be public to contributors. The users being contributors play a vital role in the funding by -

- Being keen to be able to approve the campaign for the cause
- Funding for the campaign

3.2 Software Project Management Plan

3.2.1 Work Breakdown Structure

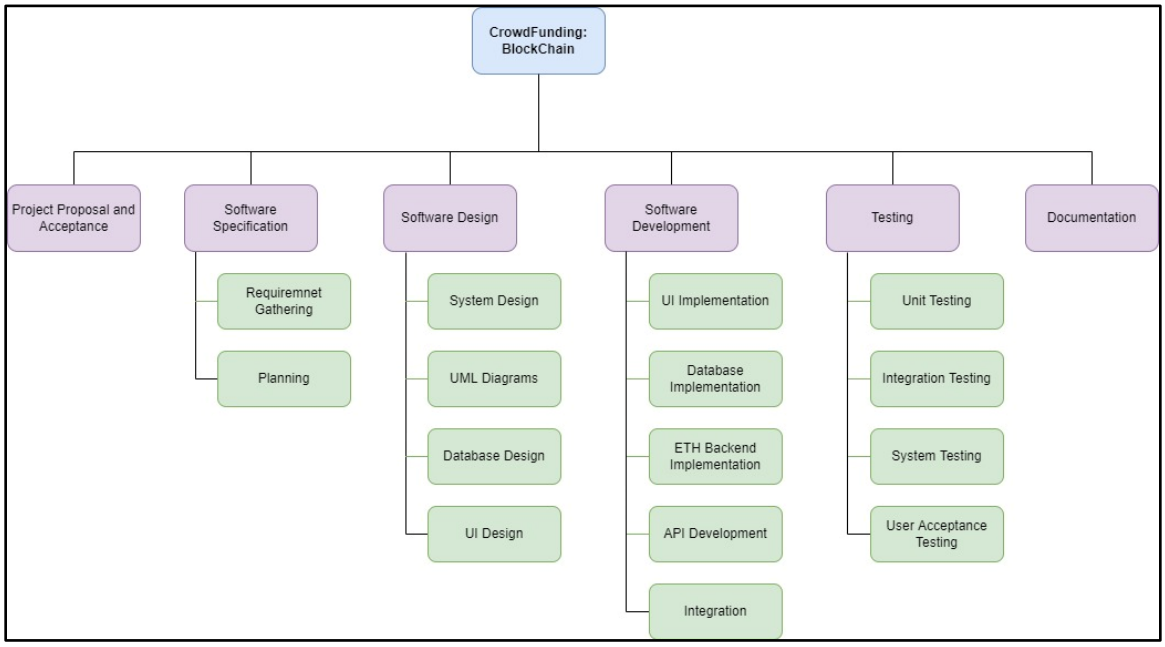


Figure 3: Work Breakdown Structure

3.2.2 GANTT Chart

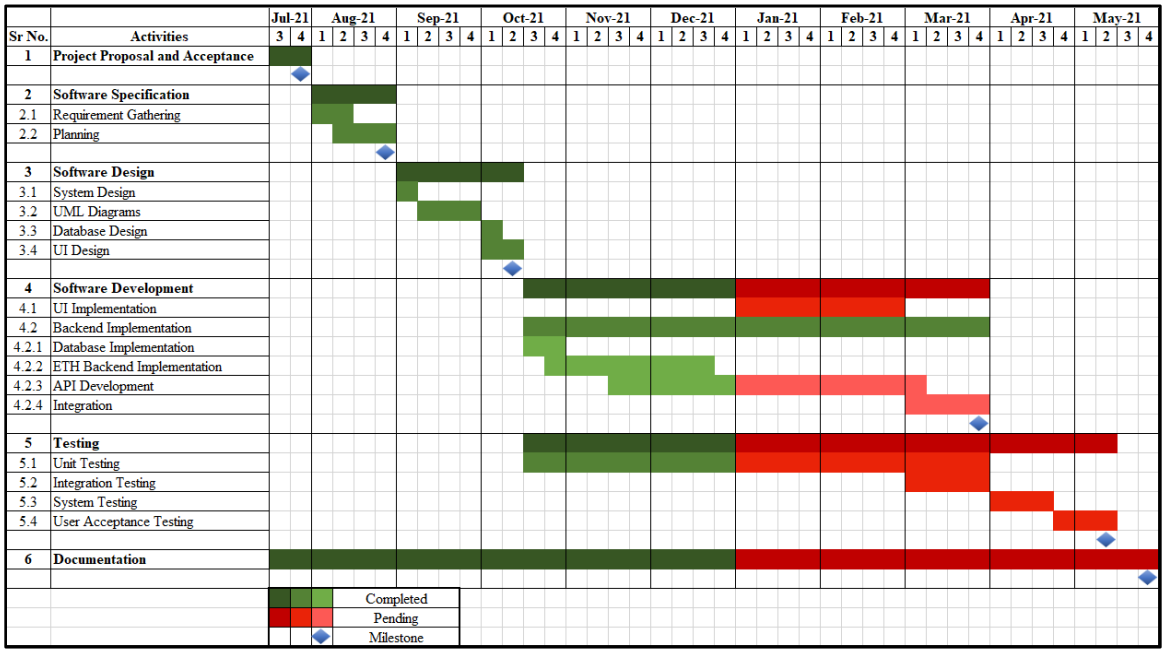


Figure 4: GANTT Chart

3.2.3 Project Timeline and Milestones

Table 1: Project Timeline

Duration	Task
July Second fortnight	Submit Initial Project Draft and Proposal
August First fortnight	Perform Literature Survey and other market requirements.
August Second fortnight	Perform detailed research on required resources and start courses related to the tech-stack involved.
Sept. First fortnight	Create Design Specifications and System designs
Sept. Second fortnight	Start Interface Designing alongside with implementation of Front-end development

Oct First fortnight	Continuing with basic UI/UX and Front-End Scripting. Implement basic backend (Smart-Contracts, ETH-chain API calls) and related Unit Tests
Oct Second fortnight	Continuation of Backend Development. Integrating Backend with Front-end and ETH Blockchain
Nov First fortnight	Buffer period to complete any pending work and performing all unit, modular and integration testing along with documentation 7th Semester Deliverable: Basic working prototype model
The objectives which will be achieved before VII semester examination: • Perform Comprehensive Literature Review. • Basic System and Interface designing. • Backend Development along with Testing and related documentation. • 4. Integration of Front-End with Backend resulting in a basic working prototype model.	
Jan. Second fortnight	Finalization of system model and all functionalities (All UI and backend functionalities and improvements)
Feb First fortnight	Implementing final UI/UX development with improvement Adding newly decided backend functionalities and improvements as suggested.

Feb Second fortnight	Continuing with the final system model development as above. (UI and backend) along with Unit, modular testing and documentations
March First fortnight	Integration of the final system Front-End with Backend and Test-ETH Blockchain Performing Integration and System testing with documentation
March Second fortnight	Writing research paper on the work done Performing remaining system tests along with other boundary checks.
April First fortnight	Buffer period to complete remaining work Focus on improving system performances and adding any suggestions. Correcting any issues found during testing
April Second fortnight	Buffer period to complete the work Correcting any issues found during testing and improvement phase. Finalizations of the deliverables.

The objectives which will be achieved before VIII semester final defense:

- Finalized System model with all documented functionalities.
- Development of final UI/UX and Backend along with final Integration.
- Performing different types of Tests and their documentation**.**

- Correcting any issues found during testing and focus on any improvements either suggested or thought of.
- Preparing research paper of the Project.

3.3 Software Requirement Specification

3.3.1 Product Perspective

As a revolutionary technology for recordkeeping, Blockchain is poised to change the future of finance - in accounting, asset registers, payments, trading, collateral management, and more. Hence, our curiosity to learn and work on this technology has really driven us as we wanted to explore more in this subject. As of today, crowdfunding platforms have accountability and trust problems. In many cases, money from contributors/philanthropists has gone into wrong campaigns and has been misused and implementing a blockchain-based platform can bring in a change has been the main reason. With this project and blockchain smart-contracts, contributors would be informed about the payments to be processed by the Fundraiser through request money forms. A smart contract helps to block the funds within blockchain until the campaign organizer makes progress in the campaign. With the help of our application, people would be able to create campaigns to raise funds for natural calamities, start-ups or any other social/personal causes. Similarly, on the other end the general public can donate funds for these campaigns.

3.3.2 Product Functions

The system provides 4 main functionalities to the user i.e. Organizing, Donating, Voting and Post-payment tracking. The system allows organizing a campaign while also allowing for donations from various sources. It uses a voting mechanism for transfer of small funds to the campaign organizer and in some time, will also provide the post-payment tracking of the funds disbursed to the organization.

3.3.3 User Classes and Characteristics

- Users here can be anyone who wants to get funding for any personal or social cause. He/She can create a campaign through our application to collect funds for the same.
- User can be anyone who wants to donate for any social or personal cause

- Users can be anyone who has a crypto wallet and is willing to accept and donate funds in the form of crypto currency and has appropriate knowledge of tools.

3.3.4 Operating Environment

3.3.4.1 Hardware Requirements

- Intel i3 processor
- 4-8 GB RAM
- Internet/Modem

3.3.4.2 Software Requirements

- Operating System: Windows 7+ OS or Mac OS High Sierra or any Linux Debian Distro
- Web Browser: Chrome, Mozilla Firefox, Opera, Edge
- Metamask
- Crypto Wallet

3.3.5 Design and Implementation Constraints

Creating a smart contract requires a lot of knowledge of how gas consumption works. Gas is an important element for transactions to take place, it is a kind of transaction fee applied on crypto transactions. Gas consumption increases with increase in the complexity of functions in smart contracts. Hence it is a challenge to implement functions in smart contracts with least complexity possible without compromising on functionalities.

3.3.6 User Documentation

- User Manual, documentation and Instructions.
- Description Document.
- Demonstration and Explanation Videos.

3.3.7 Assumptions and Dependencies

- User must be using crypto wallets for performing crypto transactions
- Web 3.0 is a main dependency for our system to work and interact with Ethereum smart contracts.
- Government must not put a ban on private cryptocurrencies, else we may need to change the blockchain platform to an accepted one.

3.3.8 External Interface Requirements

3.3.8.1 User Interface Requirements

3.3.8.2 Hardware Interface Requirements

- Device with internet connection.
- A computer/laptop for coding.
- Intel i5 processor
- 8 GB RAM

3.3.8.3 Software Interface Requirements

- Operating System: Windows 7+ OS or Mac OS High Sierra or any Linux Debian Distro.
- IDE, Node JS, React JS, MongoDB, Solidity, Ganache, Metamask

3.3.8.4 Communication Interface Requirements

- The web page based communication will take place using HTTP get and post for sending data to and from the server.

3.3.9 Functional Requirements

Users:

- Creating, Updating, Deleting user
- Fetch user details
- Login User
- Logout User
- View list of all campaigns created by this particular user only

Campaign Creator:

- Create, Update and Delete a campaign
- Get campaign details
- Create, Update and Delete campaign request
- Fetch all requests associated with a campaign

Campaign-Contributor:

- Vote for a campaign
- Donate for a campaign (Transfer from user to smart contract)

- Withdraw from campaign (Transfer from smart contract to contributor)

3.3.10 Non-Functional Requirements

Performance Requirements

The render time for display of the webpage should be in 95% of the cases, less than 2 seconds except in cases where the user has a low speed network. The login-in time for any user either the organizer or the user should be not more than 3.5 seconds. The time for creation of any request or campaigns should be less than 5 sec or an error message should be shown.

Security Requirements

System will use a secured database. Normal users can just read and edit those Information (including personal information) to which they have access to. System will have a number of users and every user has access constraints. All the passwords are hashed while storing in the database so that no one can directly access them remotely.

Software Quality Attributes

- **Maintainable**: Different versions of the system should be easy to maintain. For development it should be easy to add code to the existing system, and should be easy to upgrade for new features and new technologies from time to time.
- **Ease of Use**: This can be measured in terms of ease of use. The application should be user-friendly. Should be easy to learn. Navigation should be simple. The system must be easy to use for input preparation, operation, and interpretation of the output.
- **Flexible**: Should be Adaptable with other screen resolutions when interacted with. Should be easy to interface with other standard 3rd party components.

Other Requirements

- Database requirements: The storage of data in the database should be in a way that it must incorporate faster retrievals of data and information. The Database should be flexible enough so as to get adjusted fastly and easily when the system gets updated or goes for maintenance.

3.4 Software Design Document

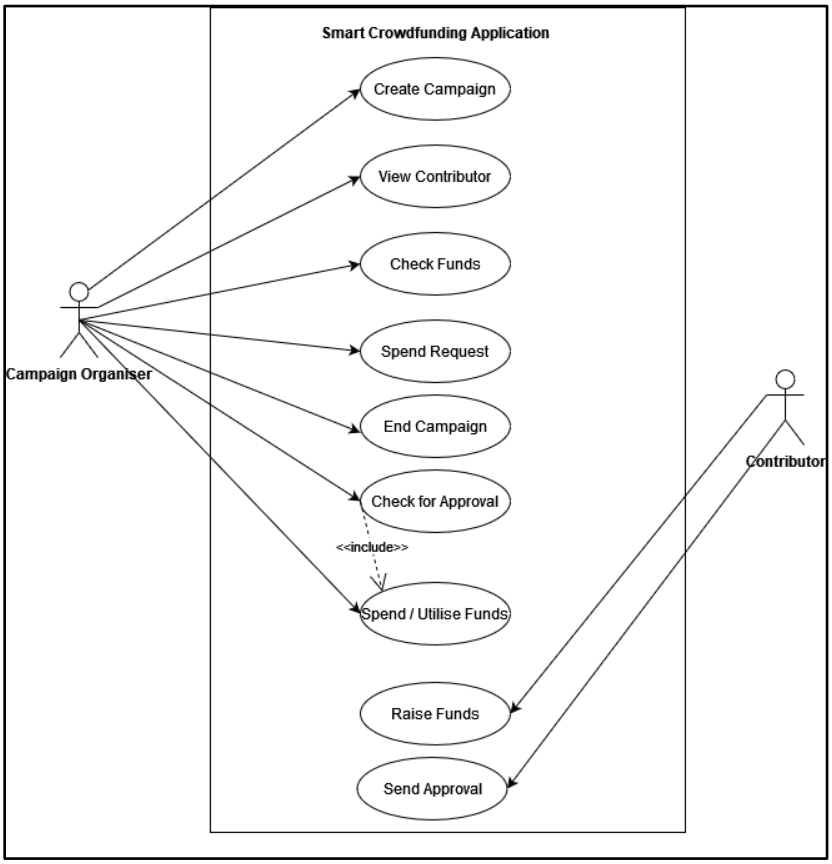


Figure 5: Use Case Diagram

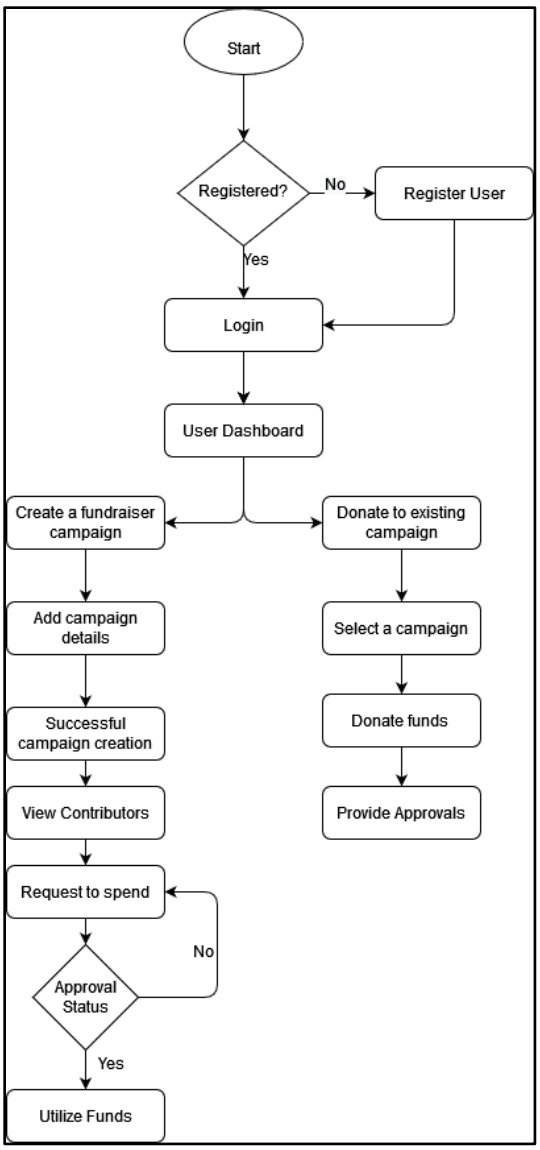


Figure 6: Application Flow Diagram

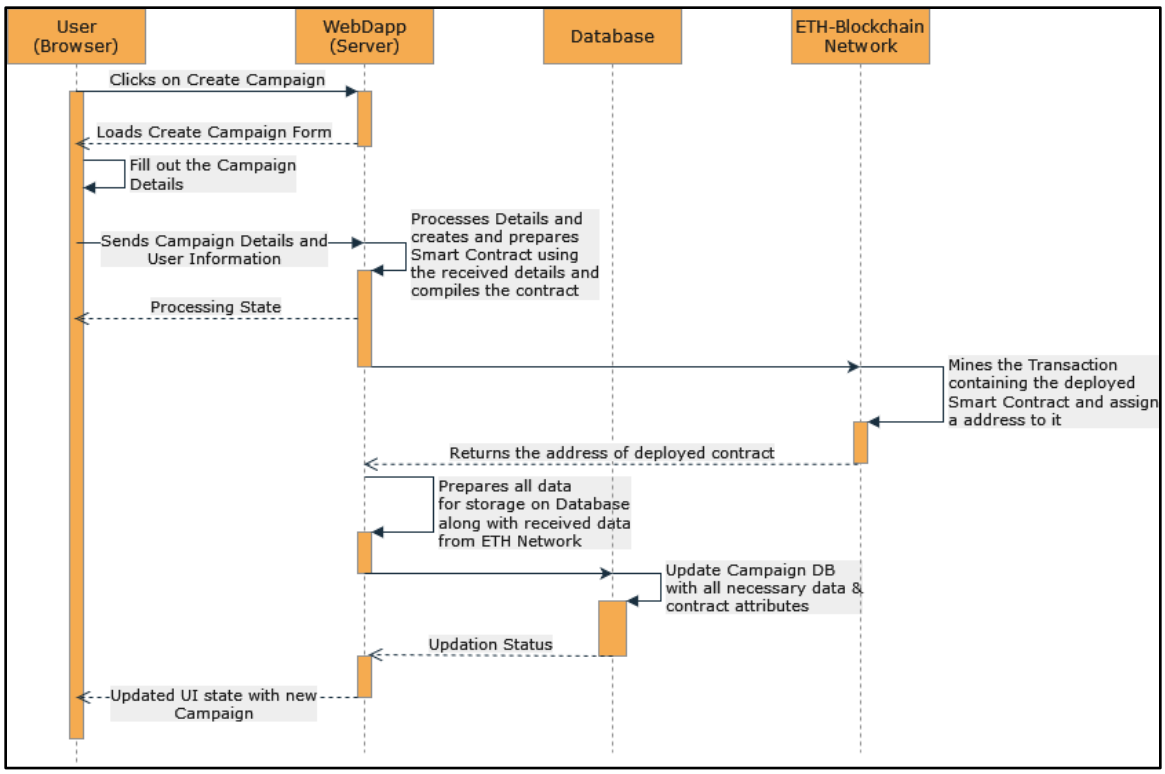


Figure 7: Sequence Diagram for Creation of Campaign Contract

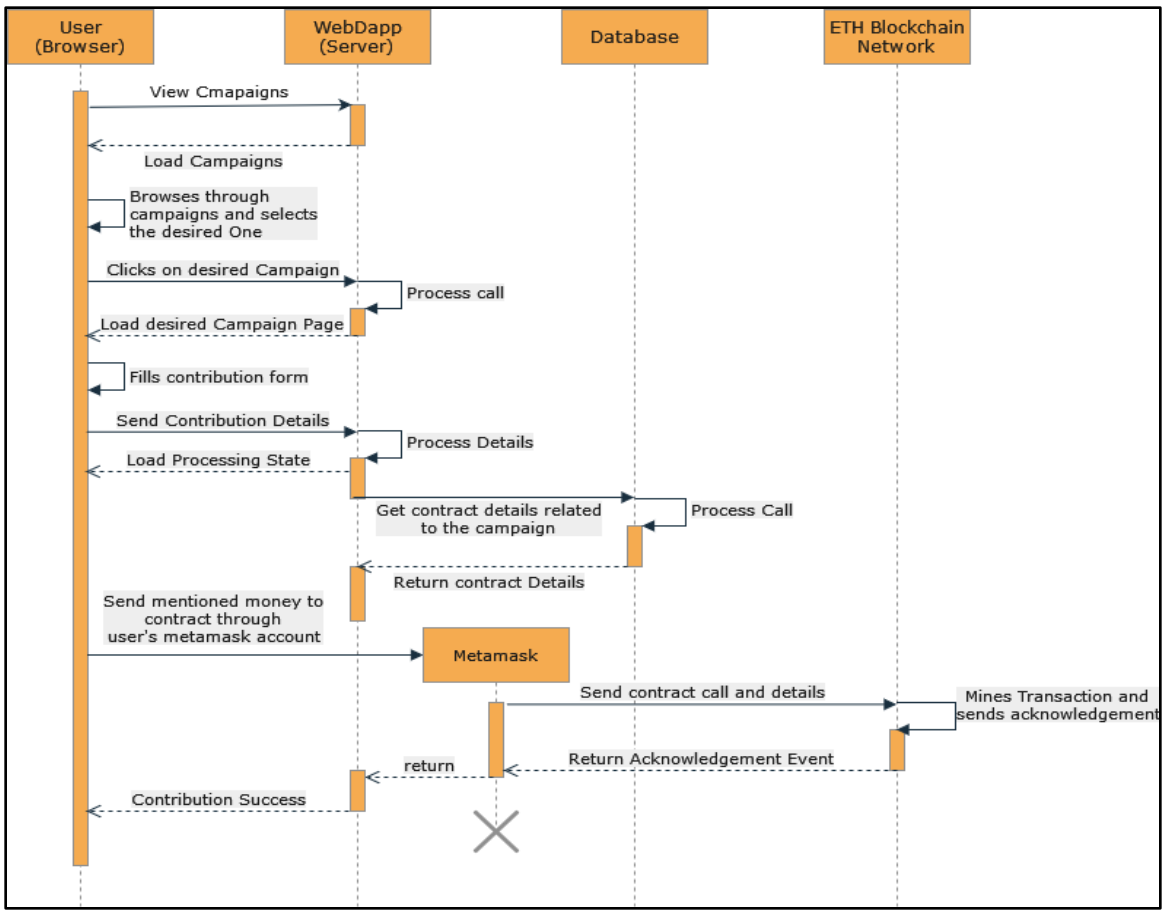


Figure 8: Sequence Diagram for Funding a Campaign

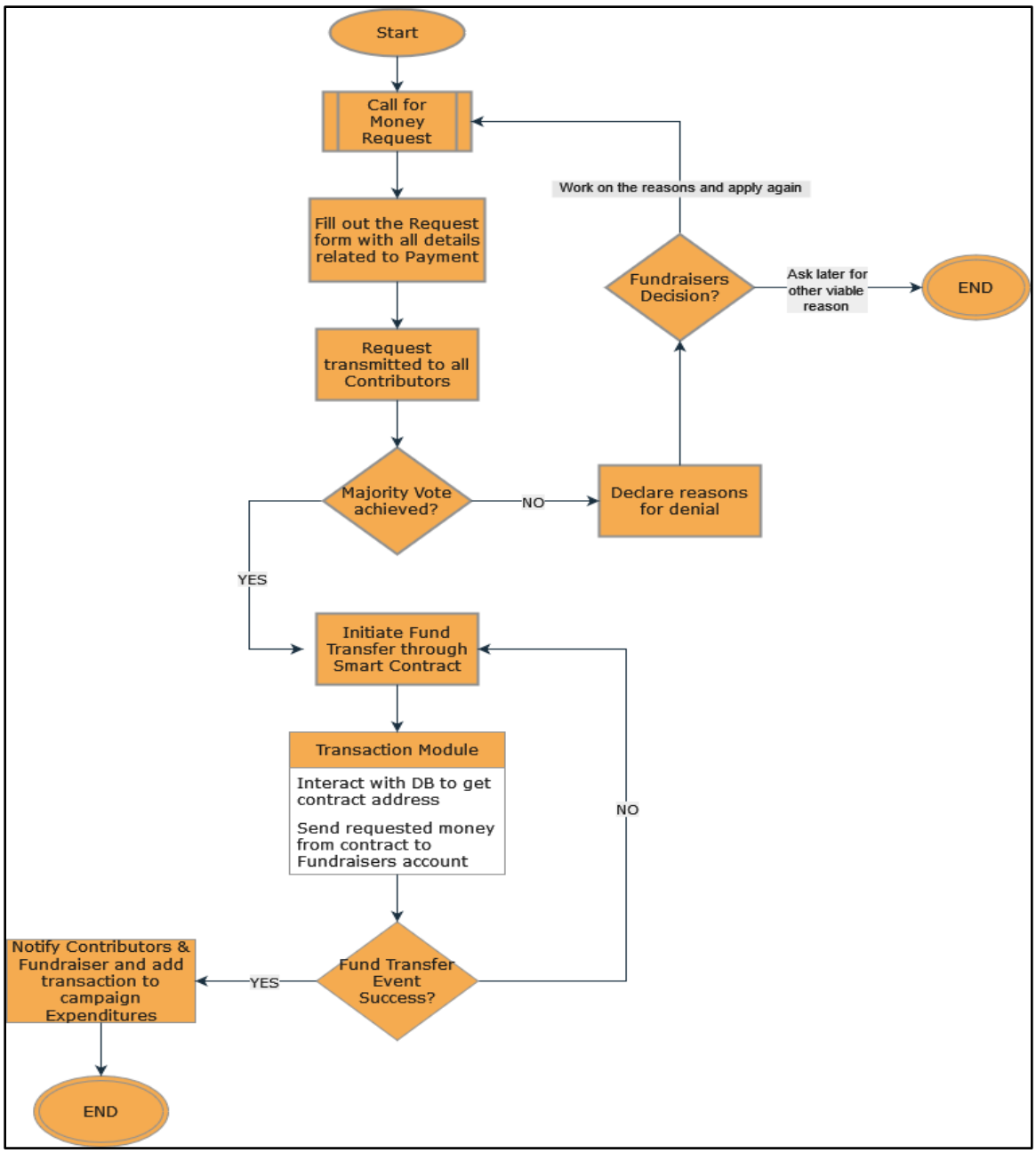


Figure 9: Flowchart for Crediting Money to Fund Raiser

Chapter 4 Implementation

This chapter presents an overview of how the tools studied in thesis and literature review were put to great effect in executing the project implementation and experimentation.

4.1 Project Structure

- SmartCrowdfunding/
  - CampaignDocuments/
  - config/
  - controller/
  - ETHBackend/
    - build/
    - contract/
    - compile-contract.js
    - deploy-contract.js
  - frontend/
    - build/
    - src/
    - components/
    - resources/
    - public/
  - middlewares/
    - auth.js
  - models/
    - Campaign.js
    - User.js
  - routes/
    - campaign.js
    - user.js
  - .env
  - server.js

Table 2: Description of Project Structure

File or Folder	Description
CampaignDocuments/	Contains all the documents related to a particular campaign for ex: Campaign Resources, Request Documents, Cover Photo etc.. Every campaign has a different folder and each campaign has one folder

	related to each request created during the entire campaign duration.
config/	Contains JS files consisting of all the configuration codes for the application development and environment
controller/	Consists of code for each model in the system eg: backend functions for user, campaigns etc.
ETHBackend/ **ETHBackend/contracts** **ETHBackend/build**	Contains all the files related to the contract functionalities i.e. creation, compilation and deployment. This module also contains the contract code and is responsible for contract execution, deployment and Ethereum wallet provider. Contains the smart contract and its dependencies created for the campaign. Compromises of JSON files consisting of compiled contract ABI data and bytecode.
frontend/ **frontend/build/** **frontend/src/**	Contains the components, scripts, code files related to the React frontend. This module is responsible for the UI and their functionalities consisting of functions related to each module/component and communication with the backend services and API's. Contains the build for the frontend. Contains all the components and their functionality files along with static files or images or resources.
middlewares/	Consists of authentication and other different middleware function files required for the application's routing.

models/	This folder includes code file for database models and Schemas. This is where the schema for user and campaign models are exported from for creation of documents
routes/	This folder consists of all the files related to each models in the system specifically for handling the different routes and protocols. This module handles different routes through the website and API responsible for execution for model specific functionality
.env	Contains all the environment variables that are sensitive like API keys and passwords for databases and accounts/wallets which is not supposed to be exposed to the user
server.js	This is the main backend server file responsible for running backend services and API's. It is also responsible for database services & connections and passing or handling different service calls to the respective routing modules

4.2 UI Implementation

1. Homepage:

This is the landing Page of the website. Here, the user can get to know about the website, featured campaigns, motto and can browse to different pages from here. A particular user can login or signup to access more functionality related to his accounts and other related stuffs.

2. Campaigns Page:

This is a broad page where user can view all of the campaigns that are currently in progress or have been completed. He/She can further explore a particular campaign on the *'Campaign Page'*. This page further allows a user to find a particular campaign through the search functionality or filter out campaigns belonging to a particular campaign.

3. Campaign Page:

This page is a sub-page of the above-mentioned page. It contains all the idea and structure of a particular campaign. It also contains all the other necessary fields such as its Request History, documents for that campaign, donor list, etc. This page is particular to a campaign and the user can browse through and donate (only if he/she is logged in) for the same.

Here, the campaign organizer can create a request and also view its voting. But this functionality is only for campaign organizer.

4. User Account Page:

This is where the User can view his/her data and security settings. Along with this, he/she can also view their donation history in different campaigns and also update various other user related settings.

5. About Us:

This Page contains the details about the goals, motto and the profiles of all the developers and persons involved in the development of the website.

4.3 Backend Implementation

4.3.1 Database Schema

User

- _id: ObjectId
- fullName: string → Required
- dob: date → Required
- userName: string → Required
- emailID: string → Required
- password: string:Hash → Required
- currentCity: String → Not Required
- state: String → Not Required
- donatedCampaigns: default : [{
    - campaignId: ObjectId(Campaign)
    - donationAmount: float
    - donatedOn: Date

  }] → Not Required

Campaign

- _id: ObjectId
- campaignName: string → Required
- campaignDescription: string → Required
- campaignCoverMedia: string (Path)
- campaignResources: [string] (ImagePath)
- campaignCategory: string
- campaignCreatedOn: date
- campaignLastEditedOn: date
- campaignOrganiser: ObjectId(User) → Required
- requiredFunding: Number → Required
- amountCollected: Number
- smartContractAddress: string
- campaignRequest: {
    - requestNumber: integer
    - requestTitle: string
    - requestDescription: string
    - requestResources: [string] (ImagePath)
    - requestAmount: float
    - requestCreatedOn: date
    - requestLastEditedOn: date
    - upVotePercentage: float
    - deadline: date

    }
- currentVote: {
    - yes:[{
        - userId: ObjectId(User

      }]
    - no:[{
        - userId: ObjectId(User)

      }]

  }
- requestVotingHistory: [{
- requestNumber: integer
- requestTitle: string
    - requestDescription: string
    - requestResources: [string] (ImagePath)
    - requestAmount: float
    - requestCreatedOn: date
    - requestLastEditedOn: date
    - upVotePercentage: float
    - deadline: date
    - requestStatus: {Fund Disbursed, Cancelled, Funds Denied}

  }]
- donors: [{

- userId: ObjectId(User)
- donationAmount: float
- donatedOn: Date

}]

4.3.2 API Details

Table 3: User APIs

HTTP Method	Route	Description
POST	/api/user	This request will add a user to the Users Database and handle any errors
GET	/api/user/:id	This request will return user details
PUT	/api/user/:id	This request will update the user data corresponding to that particular id
DELETE	/api/user/:id	This request deletes the user Data for that particular id.
POST	/api/user/login	This request verifies the User using JWT tokens and logs the user in.
POST	/api/user/logout	This request logs out the user by expiring the JWT token.

Table 4: Campaign Creator APIs

HTTP Method	Route	Description
POST	/api/campaign	This request creates a new campaign object

PUT	/api/campaign/:id	This request will update the campaign data corresponding to the passed particular id
GET	/api/campaign/:id	This request will return campaign details corresponding to the passed particular id
DELETE	/api/campaign/:id	This request deletes the campaign Data for that particular id.
GET	/api/campaign/:id/request	This request will return all the requests from requestVOtingHistory of Campaign
POST	/api/campaign/:id/request	This request creates a new campaign request for a campaign with id=id
PUT	/api/campaign/:id/request/current	This request will update a particular campaign request corresponding to a campaign with id=id
POST	/api/campaign/:id/request/current/:status	This request deletes a particular campaign request corresponding to a campaign with id=id

Table 5: Campaign Contributor APIs

HTTP Method	**Route**	**Description**
POST	api/campaign/:id/vote	This request adds a contributor's vote for a particular campaign
POST	api/campaign/:id/donate	This request adds the donor's amount to a campaign's(with id=id) donor list. Also transfers the amount from the user's Metamask to smart contract. If transactionStatus fails, do not perform anything.

4.3.3 ETH Backend

Table 6: ETH Backend Variables

Variables	Description
address private _contractOwner	Deployer of the Contract i.e. the Decentralized Application.
address private _campaignCreator	Campaign Creator who is organizing the funding.
uint32 public _nContributors	No. of Contributors

Table 7: ETH Backend Functions

Functions	Description
mapping(uint32 => address) public _addresses	Index Mapping with Contributors
mapping(address => uint256) private _addressETHmap	Address mapping with contribution amount
constructor(address campaignCreator)	Constructor defines the contract Owner and the campaign organizer for onlyOwner and onlyOrganizer specific modifiers and function.
event _logReceiveMoneyEvent (address _sendersAddress, uint256 _amount)	Event & Modifiers- Logs Received Amount
modifier onlyOwner	Only Owner modifier. This enables a function to be called only by the Owner of the contract i.e. _contractOwner.
function _contractBalance() view public returns (uint256 amount)	View Function - Returns the current contract Balance
function _sendRequestedMoney(uint256 _requestedAmount) public onlyOwner	Owner specific function. Sends Requested Money to campaignCreator after approval from the campaign contibutors.

function _dissolveCampaign() public onlyOwner	Dissolves Campaign and returns the remaining money back to the contributors on a percentage system.
function _rollbackFunds() internal onlyOwner	Owner Specific Function. RollBack Contributor's Remaining Funds.
receive() external payable	Receives Money from contributors.

4.3.4 Implementation of Suggestions by Project Guide

We are planning to perform KYC in the following process to verify the users of our application

- For KYC procedure a user submits documents to one of the administrators where he wants to take a loan or use another service.
- Individual participants are responsible for collecting personal data (administrator, government agencies, companies, or users themselves) and stored in a decentralized network.
- The administrator checks and confirms the passage of KYC if everything is normal.
- The administrator is responsible for entering the data about the user into the blockchain platform, to which other administrator, organizations and state structures have access. All parties can control and regulate the KYC process. The system will monitor changes and updating of the user data, and if someone breaks the rules, it will become known to all parties.
- When a user wants to use the services of another administrator, this second administrator accesses the system and thus confirms the user's identity.
- The access to user data will be based solely on its consent. The user must log in with cryptocurrency transactions i.e. use the private key to initiate the information exchange operation.

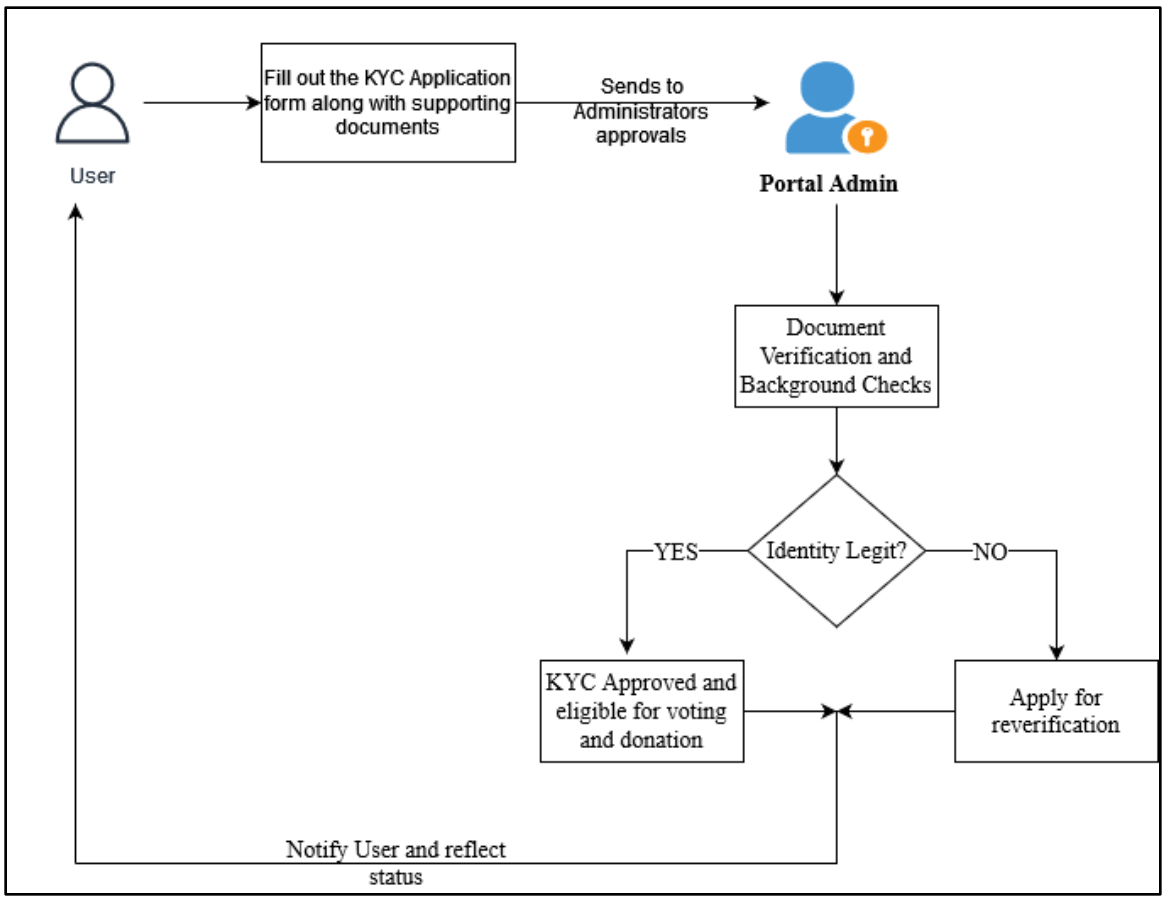


Figure 10: Proposed KYC Verification Process

4.4 Experimental Results

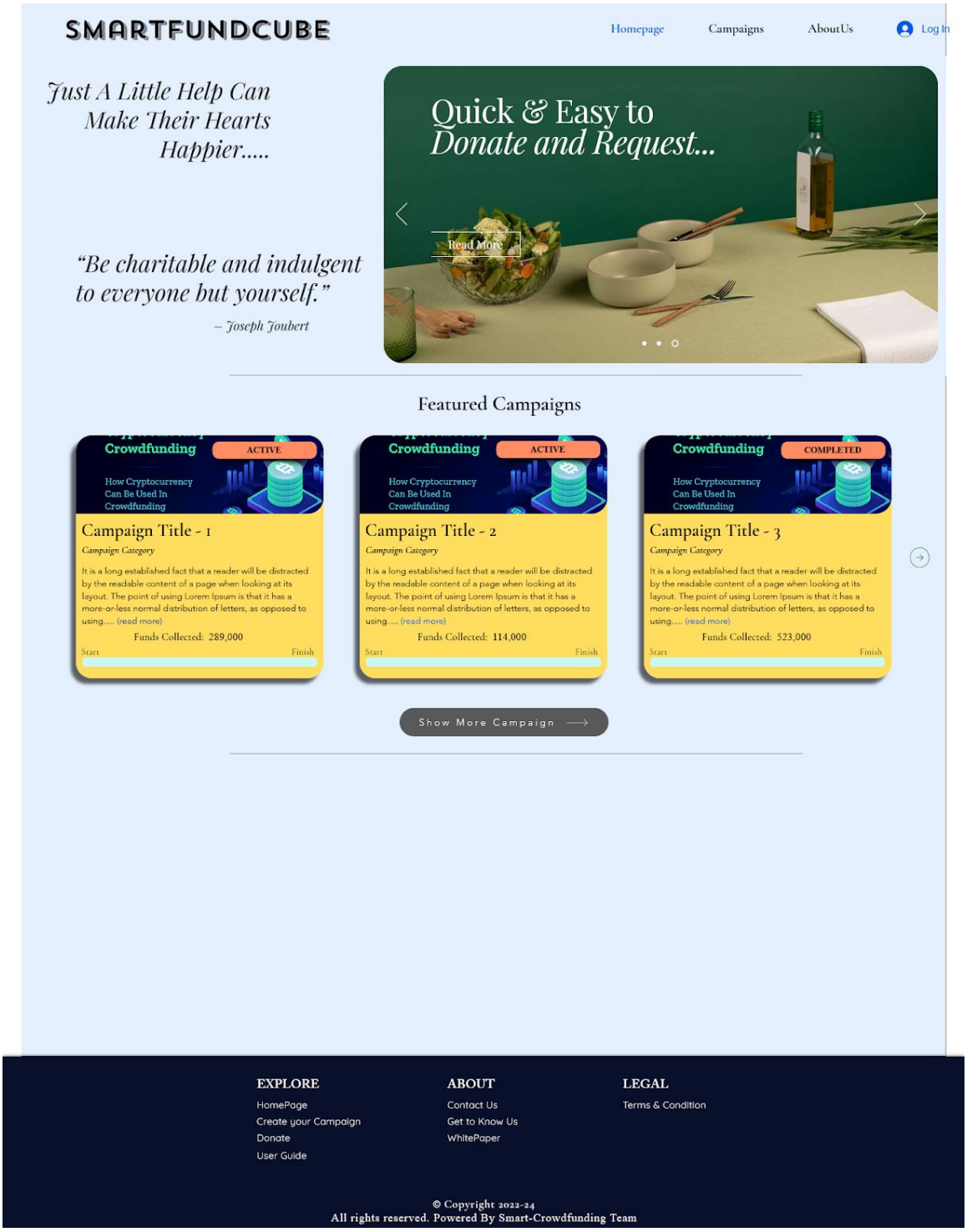


Figure 11: Home Page

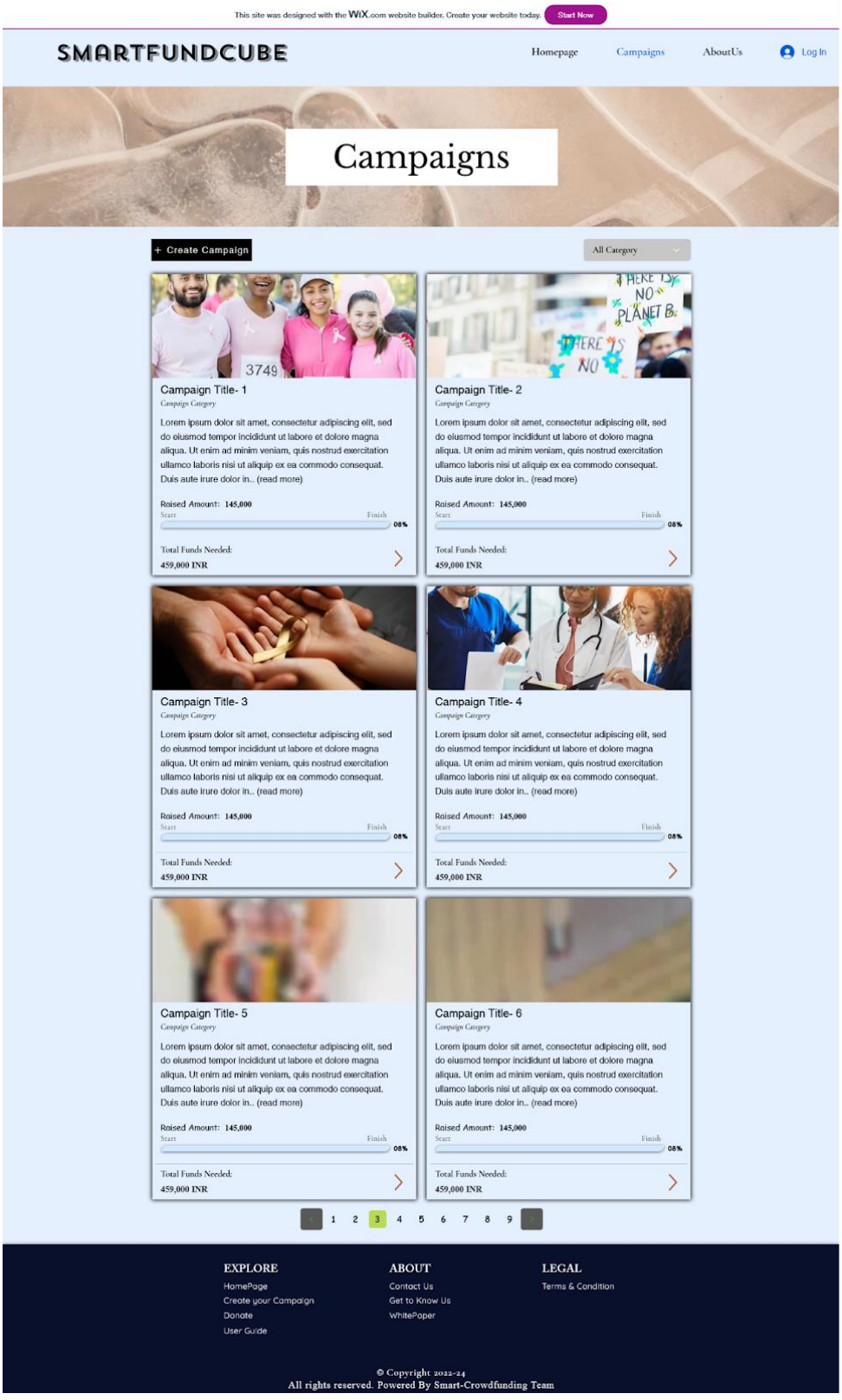


Figure 12: Campaigns Page

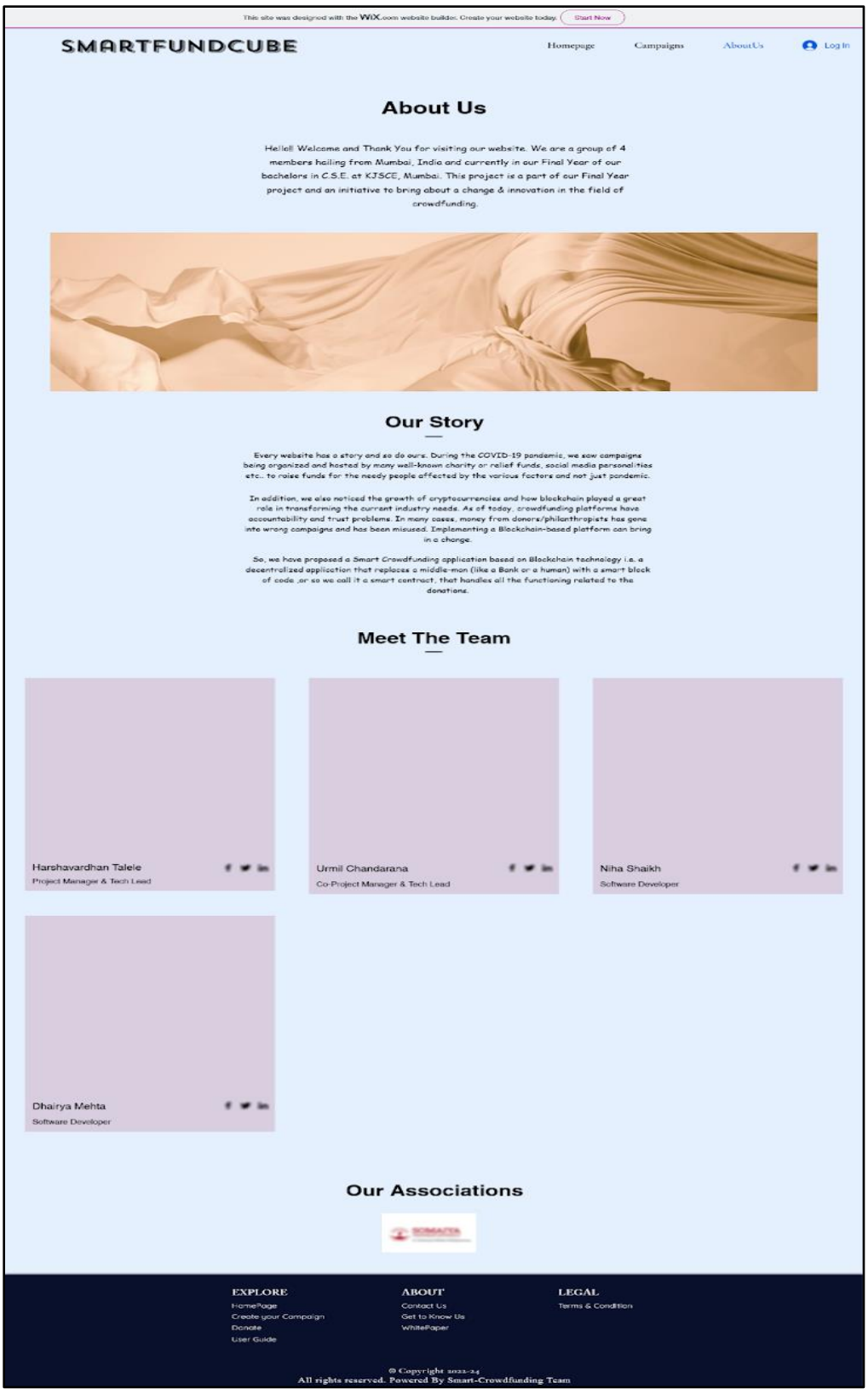


Figure 13: About Us Page

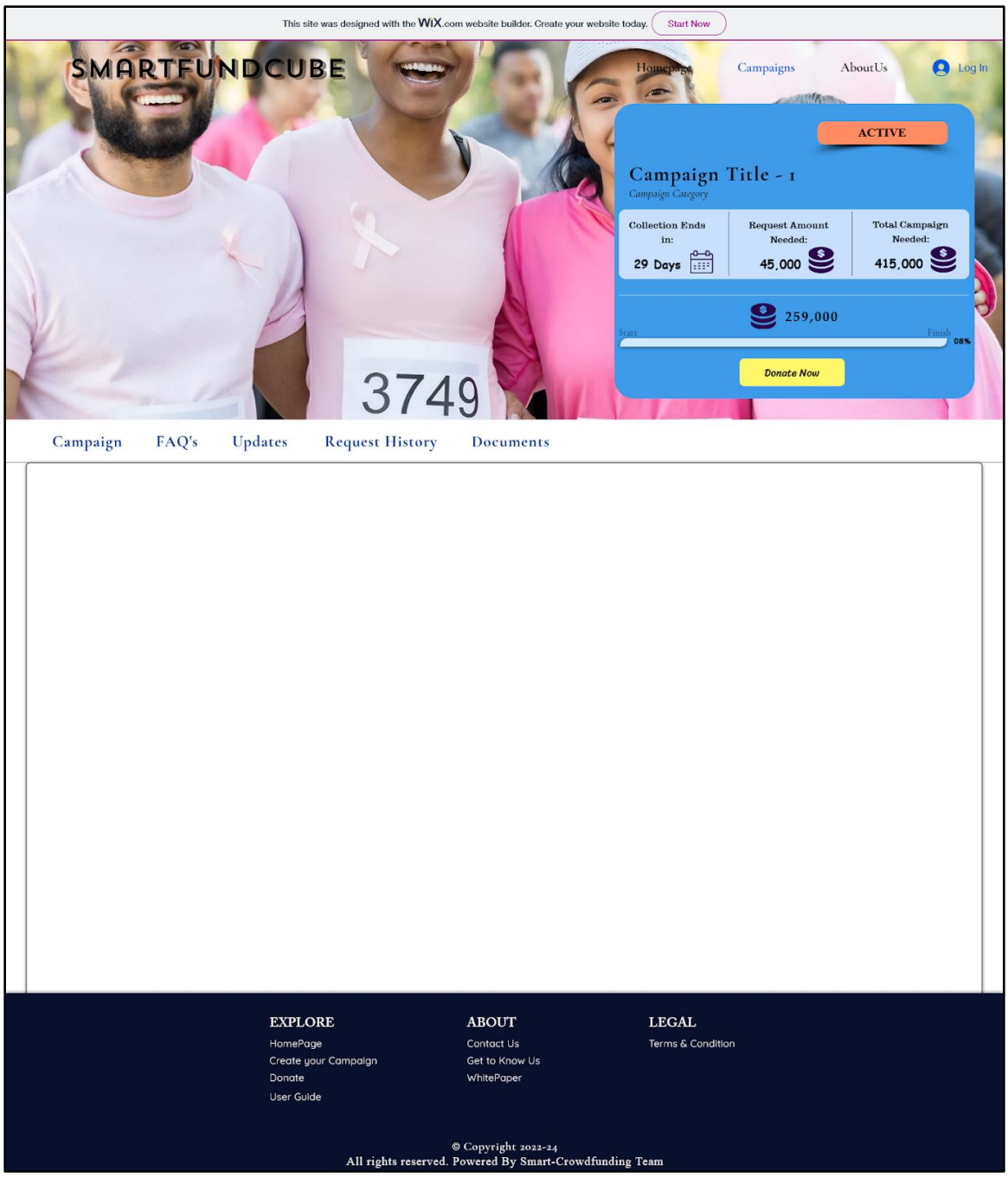


Figure 14: Campaign Page

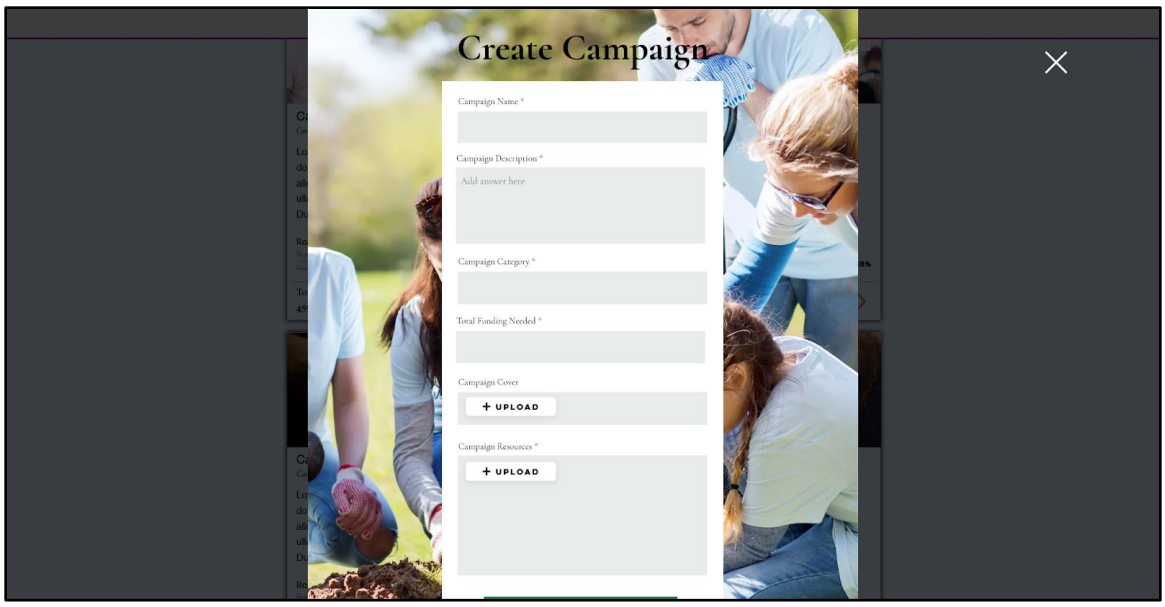


Figure 15: Campaign Creation Form

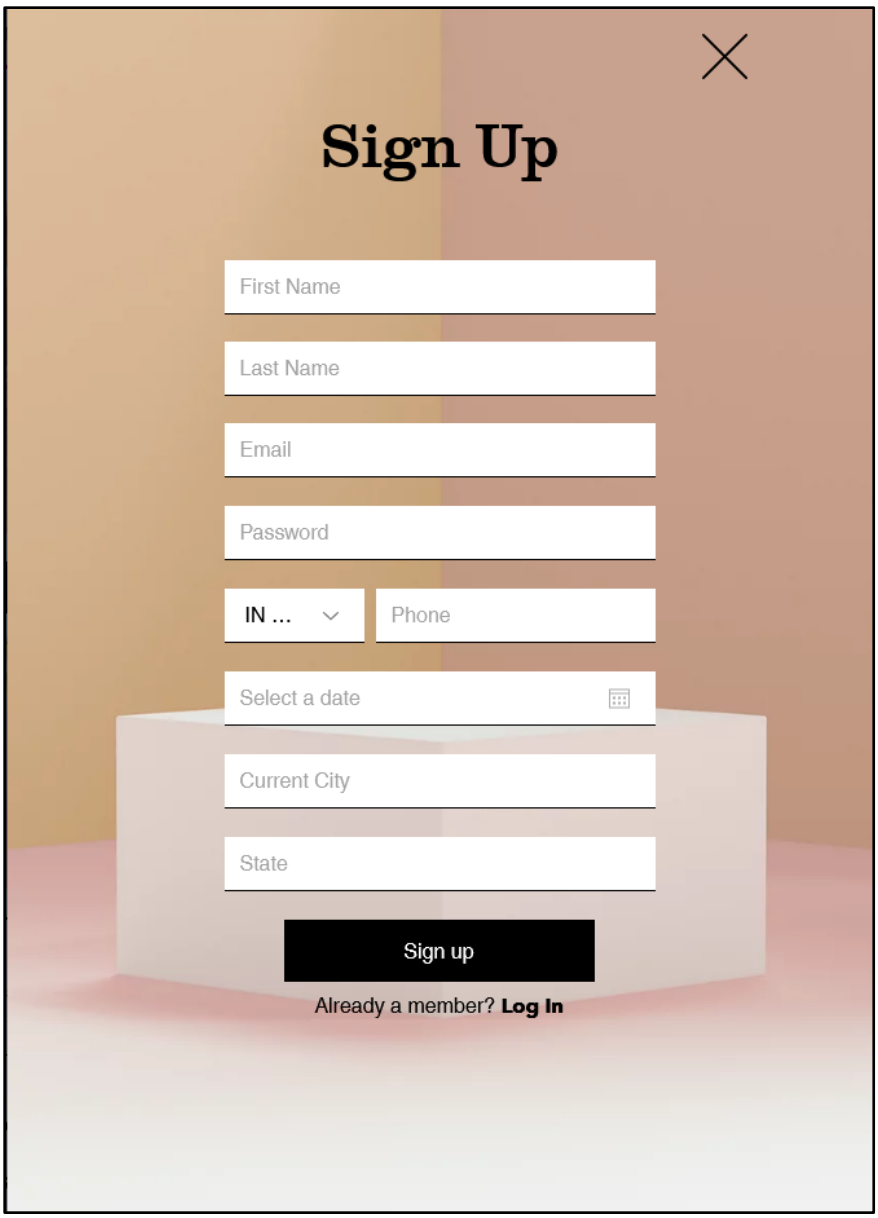


Figure 16: User Registration Form

Chapter 5 Conclusion

This chapter presents conclusion of this report, future work for the system

- Perform Comprehensive Literature Review along with thorough analysis of each for the development and goals for the system.
- Performed different designs for the system such as system architecture, database design, different sequence diagrams for certain activities etc..
- Performed Unit testing along with module testing for all the API's and modules designed.
- Integration of Front-End with Backend resulting in a basic working prototype model to give a brief idea of how the system works and functions.
- Created UI designs for various components of the system to give us an idea of how end-users will interact with the system.
- Learned the blockchain technology and Ethereum blockchain through courses to understand how they work, what decentralized applications are and how smart contracts play a role in building decentralized applications.
- Created modules and smart contracts using solidity for creation, compilation and deployment of smart contracts. Tested the functionalities on remix IDE and successfully tested the same through CLI.
- Performed a thorough analysis on different functionalities by breaking up bigger components and listed out the routes, services and API needed for the same. Then performed unit tests to see their working.

5.1 Future Work

- Finalized System model with all documented functionalities.
- Development of final UI/UX and Backend along with final Integration.
- Performing different types of Tests and their documentation.
- Correcting any issues found during testing and focus on any improvements either suggested or thought of.
- Preparing research paper of the Project.

Chapter 6 References

This chapter contains references used for development of the project and other technicalities mentioned.

[1] Venturing Crowdfunding using Smart Contracts in Blockchain | IEEE Conference Publication

[2] Proposed Solution for Trackable Donations using Blockchain | IEEE Conference Publication

[3] Building A Decentralized Application on the Ethereum Blockchain | IEEE Conference Publication | IEEE Xplore

[4] Factors Affecting Crowdfunding Success: A Systematic Analysis of the Empirical Studies | IEEE Conference Publication | IEEE Xplore

[5] Tecra Space - WHITE PAPER

[6] GitHub - rushikesh611/DisReliefFund: Ethereum based Crowdfunding website for disaster relief funds.

[7] Solidity - A Brief Documentation

[8] How to reduce gas cost in Solidity | by tak | LayerX | Medium

[9] Optimizing your Solidity contract's gas usage | by Mark Mathis | Coinmonks | Medium

[10] Using JWT (JSON Web Tokens) to authorize users and protect API routes | by Maison Moa | Medium