VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A PROJECT REPORT ON

"RISE TOGETHER: A CROWDFUNDING PLATFORM"

A Project report Submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BHAVANA S K   **(1RG20CS019)**

MADHU S   **(1RG20CS033)**

N DEEPAK   **(1RG20CS038)**

SANTHOSH P   **(1RG20CS050)**

Under The Guidance of

Mrs. KAVYASHREE J

Assistant Professor

Dept. of CSE RGIT,

Bengaluru

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R. T. Nagar Post, Bengaluru-560032

2023-2024

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)

Cholanagar, R.T. Nagar Post, Bengaluru-560032

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project report entitled **"RISE TOGETHER: A CROWDFUNDING PLATFORM"** is a bonafide work carried out by **Ms. BHAVANA SK (1RG20CS019), Mr. MADHU S (1RG20CS033), Mr. N DEEPAK (1RG20CS038), Mr. SANTHOSH P (1RG20CS050) in** partial fulfilment for the award of **Bachelor of Engineering in Computer Science Engineering** under **Visvesvaraya Technological University, Belagavi,** during the year **2023-2024.** It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This project report has been approved as it satisfies the academic requirements.

Signature of guide	Signature of HOD	Signature of Principal
Mrs. Kavyashree J	**Dr. Arudra A**	**Dr. Madhu H Gowda**
Assistant Professor,	Associate Professor & HOD,	Principal,
Dept. of CSE	Dept. of CSE	RGIT
RGIT	RGIT	

External Viva

Name of Examiner   **Signature with date**

1.

2.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We here by declare that the project report entitled "**RISE TOGETHER: A CROWDFUNDING PLATFORM**" submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2023-2024**, is record of an original work done by us under the guidance of **Mrs. KAVYASHREE J, Assistant Professor, Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru** and this project work is submitted in the partial fulfilment of requirements for the award of the degree of **Bachelor of Engineering in Computer Science & Engineering.** The results embodied in this thesis have not been submitted to any other University or Institute for award of any degree or diploma.

BHAVANA SK   **(1RG20CS019)**

MADHU S   **(1RG20CS033)**

N DEEPAK   **(1RG20CS038)**

SANTHOSH P   **(1RG20CS050)**

ACKNOWLEDGEMENT

BHAVANA S K   **(1RG20CS019)**

MADHU S   **(1RG20CS033)**

N DEEPAK   **(1RG20CS038)**

SANTHOSH P   **(1RG20CS050)**

ABSTRACT

The project titled **"Rise Together: A Crowdfunding platform"** introduces a groundbreaking Crowdfunding Platform on the Ethereum blockchain, revolutionizing traditional fundraising models. Leveraging smart contracts for seamless automation and execution of campaigns, the platform ensures transparency, security, and efficiency. Ethereum's decentralization eliminates intermediaries, reducing costs, and enhancing accessibility. Key features include a user-friendly interface, contribution tracking, and real-time campaign monitoring. Utilizing Ether as the native cryptocurrency streamlines contributions, offering a secure and standardized means of support. Positioned at the intersection of blockchain and fundraising, this project contributes to the evolution of decentralized finance, showcasing Ethereum's potential for secure, inclusive, and innovative crowdfunding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Overview

Blockchain is a decentralized and distributed ledger technology that securely records all the transactions across a network of computers. It relies on a chain of blocks, where each block contains cryptographic hash of the previous one, ensuring data integrity. The decentralized network of nodes reaches consensus on transaction validity through mechanisms like proof of work or proof of stake. Cryptographic hash functions and immutability make a recorded data resistant to tampering. Smart contracts, self-executing code, automate and enforce the predefined contract terms. Transactions are transparent, visible to all participants, while the participant identities remain pseudonymous. Blockchain applications extend beyond finance to supply chain, healthcare, and voting systems. Its security, transparency, and resistance to censorship contribute to its widespread interest and adoption.

Business runs on information. The faster it's received and the more accurate it is, the better. Blockchain is an ideal for delivering that information because it provides immediate, shared and completely transparent information stored on an immutable of ledger that can be accessed only by permissioned network members. A blockchain network can track an order, payments, accounts, production and much more. And because members share a single view of the truth, you can see all details of transaction end to end, giving you greater confidence, as well as new efficiencies and opportunities.

1.2 Motivation

Raise Funds : Creators are motivated to use crowdfunding platforms because it provides an easy, efficient, organized way to solicit and collect financial from many people through the distributed network. By using web-based technologies, such as online payment systems and social media, creators are able to market and solicit resources safely and easily through the crowdfunding platforms.

Expand Awareness of Work : In addition to raising financial resources, creators are to be motivated to expand awareness of their work by publicizing their crowdfunding project. And unlike traditional fundraising methods in which only the application reviewers read about the project, crowdfunding provides an avenue for anyone on the internet to view one's of project through a brief video and written description. Creators expand awareness by posting links to their project in social media and sending emails about their campaign t friends, family, and a news media outlet.

Form Connections : In addition to raising funds and expanding awareness of work, our data suggests that creators are motivated to engage crowdfunding to connect with people through a long-term interaction that extends well beyond a single financial transaction. Because the crowdfunding platforms store supporter contacts and provide online messaging services, and creators are able to easily communicate with supporters in answering questions and giving a project updates.

Learn New Fundraising Skills : Having control over a crowdfunding campaign forces the creators to gain experience in areas outside their professional expertise. Although creators did not initially report being motivated to be learn, those who had completed campaigns, both the successes and failures, were motivated to participate again to improve skills to fundraise in an effective manner, such as marketing, communication, management, risk taking, and financial planning.

1.3 Problem Statement

Crowdfunding is a financing method that involves funding a project with relatively modest of contributions from the large group of individuals, rather than seeking substantial sums from a small number of investors.

This crowdfunding platform powered by the blockchain technology eliminates the need for a middleman third party in a number of ways, enhancing and supporting the practice. Increased security in hostile environments just one of the many advantages that blockchain technology offers in a wide range of industries.

Blockchain is a linked list that employs hash pointers rather than regular pointers. This allows each blockchain node to not only locate the next node but also verify whether the data in that node has changed. This blockchain-based crowd-funding system can receive funding. Anyone with an internet connection can use this system.

1.4 Proposed System

Blockchain-based Platform : A decentralized platform, open to all, that allows creators to fundraise from anywhere in the world without the need for intermediaries.

Peer-to-Peer : A peer-to-peer network that connects creators with funders, allows for secure and transparent fundraising and enables transactions to be publicly audited.

Ease of Use : A simple, user-friendly interface that simplifies the crowdfunding process and reduces the time it takes to bring your project to life.

Customizable Features : The platform offers customizable features, including rewards and an incentive that allow creators to incentivize funders and increase their chances of success.

1.5 Report Organization

Chapter 1 :

This chapter gives the overall description about the project. It gives the overview of a proposed project work. It tries to answer why is this project is needed in current scenario and what are various motivation factors that motivated to implement this project. This chapter also points out the limitations in the existing systems and tells how these limitations can be help to overcome by using this project.

Chapter 2 :

This chapter gives us details about various base papers that are related to the proposed project work. It shows how various activities related to the project we carried out at different point of time. It gives a short introduction to each base paper, talks about their Shortcomings and tells how this project can overcome those shortcomings.

Chapter 3 :

This chapter introduces the system analysis process. It gives us brief idea whether this project should be done or not based on various feasibility study. It gives the summary of the various feasibility studies that were carried out and shows the advantages of doing a project. At the same time, it also gives the over.

Chapter 4 :

This chapter talks about various hardware and software tools that are necessary in the order to implement this project. It provides details of software and languages that will be used and also lists the minimum requirements needed to run the project.

Chapter 5 :

This chapter shows the detailed design of the architecture, components, modules, and interfaces, and data for the proposed system to satisfy specified requirements. It shows us the various standard UML diagrams that are needed to design the system. It provides visualization of how the data will flow among various components of the system.

Chapter 6 :

This chapter shows the implementation of the structure created during architectural of design and the results of system analysis so construct system elements that meet a stakeholder

requirements and system requirements developed in an early life cycle phase. It shows us the segment of programming code that is used in order to implement this project.

Chapter 7 :

This chapter shows the various test results produced by the system. Various kinds of a tests are performed foe each part of the system and as well as the whole system. It shows us a various pre-defined test cases and result of running these test cases on the system.

Chapter 8 :

This chapter shows the various screenshots of the system. It also shows how a data is processing happens at various stages of the system and the final output is also displayed. And it also shows the outer interface design of the system.

1.6 Introduction Summary

This introduction chapter gives the overview about the project and gives a short kind of description of the proposed project work. It tries to answer why this project is needed in a current scenario and what are various scopes and advantages of this project.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

A literature survey is a critical component of any research project or paper. It is a comprehensive review of existing literature on a specific research topic, which helps the researcher toidentify knowledge gaps, current research trends, and potential avenues for future research. The purpose of a literature survey is to gather information and evidence from various sources, such as academic journals, books, reports, and other reliable sources, to establish a solid foundation for the research.

A literature survey is an essential part of any research project because it helps to define the scope of the project, provide context, and demonstrate the relevance of the research question. It also helps the researcher to understand the existing research landscape and identify the key concepts and theories related to the research topic. A well-conducted literature survey can provide valuable insights and ideas for the research and help the researcher to develop a more comprehensive and accurate understanding of the topic

There are several steps involved in conducting a literature survey for a research project. The first step is to identify the research question or problem and formulate the research objectives. This will help to narrow down the search and identify the relevant literature to be reviewed. The second step is to search for relevant literature using various search engines and databases, such as Google Scholar, Web of Science, Scopus, and PubMed. The search terms should be chosen carefully to ensurethat the literature reviewed is relevant to the research question.

Once the literature has been identified, it should be critically evaluated to determine its quality,relevance, and reliability. This involves reading the abstract, introduction, methodology, results, and conclusion of each paper to assess its contribution to the field and its potential relevance to the research. The literature should be summarized and synthesized, highlighting the key findings, themes,and trends in the field. The literature survey should also identify any gaps or inconsistencies in the existing literature and suggest potential areas for further research.

In conclusion, a literature survey is a critical component of any research project, and it shouldbe conducted carefully and systematically to ensure its accuracy and reliability. It helps the researcherto identify knowledge gaps, current research trends, and potential avenues for future research. It alsoprovides a foundation for the research and helps to establish the relevance and significance of the research question. Therefore, it is important for researchers to invest sufficient time and effort in conducting a literature survey and to ensure that it is well-written and presented.

2.1 Literature Surveys of All Base Papers

Slno	Title	Authors	Publisher & year	Related work	Drawback
1	Crowd-Funding Using Blockchain	Dr. Sumathi VP, Harish Krishna.S, Lakshya Jain, Hisham Ahmed	IEEE 2023	Leveraging Ethereum blockchain, our crowdfunding platform utilizes smart contracts for enhanced reliability and efficiency.	Potential challenges include user complexity,gas price volatility, regulatory hurdles, and scalability concerns.
2	Literature Survey on "Crowdfunding Using Blockchain	Abhinav R.B,Ahmed Mohtesham,Akash,Basavesh M,Farhan Ashraf	IRJET 2023	Survey explores blockchain integration in crowdfunding, addressing asymmetry and transaction costs.	potential for fraudulent activities in crowdfunding.
3	Smart Contracts Security	Harry Virani, Manthan Kyada	ARXIV 2022	Exploring smart contract studies, researchers delve into supply chain efficiency, code analysis, privacy, and challenges in cloud integration.	security vulnerabilities, lack agency, potential transplant for fraudulent activities, and challenges in dispute resolution.
4	Crowdfund Platform Recommenddation Algorithm Based on Collaborative Filtering	Yaming Li, Jiahao Liu , Yuying Jin, Xinxin Fan , Bixi Wang	ICCSCT 2022	The research endorses collaborative filtering, customizing crowdfunding suggestions based on user affinities. Top of Form	The research endorses collaborative filtering, customizing crowdfunding suggestions based on user affinities. Top of Form
5	Blockchain-Based Crowdfundng Application	Viren Patil,Vasvi Gupta, Rohini Sarode	IEEE 2021	Examined crowdfunding methods, ensuring security, transparency, and smart contract utilization.	lacks real- world cryptocurrency integration and practical deployment information.

6	A Survey of Security Vulnerabilities in Ethereum Smart Contracts	Noama Fatima Samreen, Manar H. Alalf	ARXIV 2021	Prior studies lack depth in Ethereum Smart Contract vulnerabilities and prevention.	Current tools focus on specific vulnerabilities without comprehensive analysis.
7	A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses	Huashan Chen,Marcus Pendleton, Laurent Njilla, Shouhuai xu	ACM Compute 2020	Ethereum security insights against prior blockchain surveys.	Lack of coverage on other blockchain implementations beyond Ethereum in the related work.
8	Crowd Funding using Blockchain	Ms. S. Benila,V. Ajay, K. Hrishikesh,R. Karthick	GRD Journals 2019	Implementing blockchain in crowdfunding for improved reliability & efficiency.	Challenge of high gas fees associated with blockchain transactions.
9	A Decentralized Application on Ethereum Blockchain	R. Vishnu Prasad,Ram Dantu Aditya Pau,Paula Mears Kirill Morozov	IEEE 2018	solves eBay issues, enhances privacy, and reduces fees.	Navigating the intricate decentralized framework poses a significant onboarding challenge.
10	Crowdfunding: principles, trends and issues	Stéphane Onnée , Sophie Renault	ResearchGate 2016	Crowdfunding thrives on diverse models, including gifts, rewards, loans, creating multifaceted platforms for collaborative financial support	Intellectual property risks, exposure to replication, a crowdfunding drawback.

Table 2.1 Literature Survey

- Our project leverages smart contracts on the Ethereum blockchain, as inspired by "Crowd-Funding Using Blockchain" (IEEE 2023), to enhance reliability and efficiency in our crowdfunding platform. We address potential challenges such as user complexity, gas price volatility, regulatory hurdles, and scalability concerns.

- Drawing from "Literature Survey on Crowdfunding Using Blockchain" (IRJET 2023), we integrate blockchain technology into our crowdfunding platform to address asymmetry and transaction costs, while being mindful of potentially fraudulent activities.

- We enhance the security of our smart contracts by incorporating insights from "Smart Contracts Security" (ARXIV 2022), focusing on supply chain efficiency, code analysis, privacy, and mitigating security vulnerabilities, lack of transparency, potential for fraudulent activities, and challenges in dispute resolution.

- Our crowdfunding platform incorporates collaborative filtering algorithms inspired by "Crowdfunding Platform Recommendation Algorithm Based on Collaborative Filtering" (ICCSCT 2022) to customize crowdfunding suggestions based on user affinities, while considering the drawbacks of lacking real-world cryptocurrency integration and practical deployment information.

- Our project aims to mitigate security vulnerabilities in Ethereum smart contracts by addressing insights from "A Survey of Security Vulnerabilities in Ethereum Smart Contracts" (ARXIV 2021), focusing on comprehensive analysis and prevention strategies for specific vulnerabilities.

- By drawing insights from "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defences" (ACM Compute 2020), we enhance the security of our Ethereum-based systems by understanding Ethereum security insights against prior blockchain surveys and exploring defences against vulnerabilities and attacks.

- We implement blockchain in our crowdfunding platform based on "Crowdfunding Using Blockchain" (GRD Journals 2019), aiming to improve reliability and efficiency, while also acknowledging the challenge of high gas fees associated with blockchain transactions.

- Our decentralized application on the Ethereum blockchain is inspired by "A Decentralized Application on Ethereum Blockchain" (IEEE 2018), aiming to solve eBay issues, enhance privacy, and reduce fees, while addressing challenges related to navigating the intricate decentralized framework.

- Informed by "Crowdfunding: Principles, Trends, and Issues" (ResearchGate 2016), our project explores diverse crowdfunding models, such as gifts, rewards, and loans, creating multifaceted platforms for collaborative financial support, while considering risks like intellectual property risks and exposure to replication.

2.2 Literature Survey Summary

This chapter gives details about various base papers that are related to the proposed project work. It shows how various activities related to the project were carried out at different point of time.It gives a short introduction to each base paper, talks about their shortcomings and tells how this project can overcome those shortcomings.

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 Introduction to System Analysis

- **System**: A system is an orderly group of interdependent components linked together according to a plan to achieve a specific objective. Its main characteristics are organization, interaction, interdependence, integration and a central objective.

- **Analysis:** Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis data are collected on the available files decision points and transactions handled by the present system. This involves gathering information and using structured tools for analysis.

- **System Analysis:** System analysis and design are the application of the system approach to problem solving generally using computers. To reconstruct a system the analyst must consider its elements output and inputs, processors, controls, feedback and environment.

3.2 Feasibility Study

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility Study. This type of study if a project can and should be taken. In the conduct of the feasibility study, the analyst will usually consider seven distinct, but inter-related types of feasibility.

3.2.1 Technical Feasibility

This is considered with specifying equipment and software that will successfully satisfy the user requirement the technical needs of the system may vary considerably but might include

- The facility to produce outputs in a given time.
- Response time under certain conditions.

3.2.2 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost / benefit analysis. The procedure is to determine the benefits and savings are expected form a proposed system and a compare them with costs. It benefits outweigh costs; a decision is taken to design and implement the system

will have to be made if it is to have a chance of being approved. There is an ongoing effort that improves in accuracy at each phase of the system life cycle.

3.2.3 Operational Feasibility

It is mainly related to human organization and political aspects. These points are considered are

- What changes will be brought with the system?
- What organizational structures are distributed?
- What new skills will be required?
- Do the existing system staff members have these skills?
- If not, can they be trained in the course of time?

3.3 Functional Requirements

- **User Roles and Login:** The system must support Admins, Creators, and Users with distinct responsibilities. Admins manage core functionality, Creators propose projects, and Users contribute funds.

- **Campaign Management:** The system should approve campaigns before listing, enable Creators to manage details, and allow Users to browse and contribute. It supports two-stage funding goals for campaign.

- **Funding and Transactions:** The system should ensure secure transactions on Ethereum, store user funds securely, and optionally include an admin-managed voting system for fund releases.

- **Content Management:** The system should provide a responsibility to creators provide project details, displayed clearly on campaign pages for user understanding and engagement.

- **Notification and Alert Management:** The system should include robust notification and alert mechanisms to promptly inform users about errors or mismatches in field details, ensuring data accuracy and system reliability.

- **User Interface:** The system must offer an intuitive, user-friendly interface for easy access and interpretation of displayed information, enhancing user experience and efficiency.

- **Transparency and Security:** The system ensures transparency by recording transactions on Ethereum's blockchain, using smart contracts for campaign logic automation, and implementing robust security measures for user data and funds protection.

- **Campaign Funding and Distribution:** The system provides an admin-managed voting system for fund release based on contributor voting. Contributors can vote "approve" or "reject" based on predefined criteria, ensuring transparent and fair fund distribution.

3.4 Non-Functional Requirements

- **Security:** The system must ensure a high level of security for user data, funds, and smart contracts, utilizing strong encryption methods and access control mechanisms compliant with blockchain security standards.

- **Reliability:** The system must be highly reliable and available at all times, with redundancy and fault-tolerant mechanisms to ensure continuous operation.

- **Scalability:** The system must scale up or down to accommodate a growing user base and transaction volume, handling real-time transactions efficiently.

- **Performance:** The system must meet performance requirements, including fast response times, processing speeds, and throughput, without slowing down during peak loads.

- **Interoperability:** The system should integrate seamlessly with blockchain technologies like web3, Geth Go and MetaMask, as well as with other relevant technologies and APIs, following industry standards

- **Maintainability:** The system must be easy to maintain and update, with minimal disruption to operations, through proper documentation and debugging tools to ensure smooth operation and minimal downtime.

3.5 System Analysis Summary

This chapter introduces the system analysis process. It gives brief idea whether this project should be done or not based on various feasibility study. It gives the summary of various feasibility studies that were carried out and shows the advantages of doing this project. At the same time, it also gives the overview of various functional and non- functional requirements of the system.

SYSTEM

REQUIREMENTS

CHAPTER 4

SYSTEM REQUIREMENTS

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer these prerequisites are known as system requirements. System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Failure to meet these all requirements can result in installation problems or performance problems. The former may be preventing a device or application from getting installed, whereas the latter may cause a product to malfunction or perform below expectation or even to hang or a crash. System requirements are also known as minimum system requirements. System requirements only tell what system must have and what it must allow users to do. The system requirements are of two types:

- **Hardware Requirements**
- **Software Requirements**

4.1 Hardware Requirements

- Processor : Multi-core Processor
- Hard disk : 500GB or Higher
- Ram   : 8GB or Higher

4.2 Software Requirements

- Operating System   : Windows
- Front-End   : React
- Back-End   : Node JS
- Smart Contract Language : Solidity
- Frame Work   : Next JS (for front end)
- Library   : Web3
- Code Editor   : VS Code, Remix IDE

4.3 Modules Involved

Preprocessing Module:

Purpose : Enhancing crowdfunding data accuracy.

Functions : Noise removal, format standardization.

Blockchain Integration Module:

Purpose : Ensuring secure, transparent, and decentralized crowdfunding transactions.

Functions : Integration of blockchain technology.

Transaction Segmentation Module:

Purpose : Improving analysis and transparency in crowdfunding transactions.

Functions : Integration of blockchain technology.

Smart Contract Execution Module:

Purpose : Automating fund disbursement based on project milestones.

Functions : Implementation of smart contracts for secure and automated transactions.

Verification and Validation Module:

Purpose : Ensuring secure and validated transactions through blockchain consensus mechanisms.

Functions : Implementation of blockchain consensus mechanisms.

Global Participation Module:

Purpose : Enabling global crowdfunding with reduced barriers and costs through the blockchain.

Functions : Implementation of features to facilitate global participation.

Cost Efficiency Module:

Purpose : Minimizing fees and streamlining operations through blockchain.

Functions : Leveraging blockchain's cost-effective features.

Transparent Fund Utilization Module:

Purpose : Real-time tracking and transparency in fund allocation.

Functions : Utilizing blockchain for transparent and accountable fund utilization.

4.4 Technologies Used

Next JS:

Next.js is an open-source react front-end development web framework that enables the functionality such as server-side rendering and generating static websites for react based of web applications. Utilizing Next.js for web development, I implement server-side rendering and efficient client-side navigation for seamless user experiences. Leveraging its dynamic of routing

and API routes, I ensure scalable and performant applications. My proficiency in the Next.js includes optimizing for SEO and building robust react applications with ease.

Solidity:

Solidity is a high-level programming language designed for writing smart contracts on the blockchain platforms, with Ethereum being use case. It facilitates the creation of secure and a decentralized applications by defining the rules for executing transactions on a blockchain. A solidity supports the development of complex, self-executing contracts, enabling the way of a programmable and trustless interactions. Its syntax is similar to JavaScript, making it accessible for developers to create decentralized applications (DApps) and deploy them on blockchain networks.

Web3:

Web3.js isa collection of libraries that allows you to interact with local or remote Ethereum node using HTTP, IPC or WebSocket. Web3 refers to the third generation of the internet, that emphasizing decentralized and blockchain technology. It enables peer-to-peer interactions, as decentralized applications (DApps), and smart contracts transparent and trustless transactions. With a focus on user control and privacy, web3 aims to reshape the digital landscape by it providing a more autonomous and secure online experience. Cryptocurrencies and the most decentralized identity are integral components of the web3 ecosystem.

Ethereum Smart Contract:

It is the collection of functions and data that reside at a specific address on the Ethereum of blockchain. Ethereum smart contracts are self-executing code running on the Ethereum of the blockchain, automating and enforcing predefined rules in a trustless manner. Programmed in solidity, these contracts power decentralized applications (DApps) and various of blockchain based functionalities.

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.

5.1 Logical Design

Design for WebApps encompasses technical and non-technical activities. The look and feel of content are developed as part of graphic design; the aesthetic layout of the user interface is created as part of interface design; and the technical structure of the WebApp is modeled as part of architectural and navigational design.

This argues that a Web engineer must design an interface so that it answers three primary questions for the end-user:

1. Where am I? – The interface should provide an indication of the WebApp has been accessed and inform the user of her location in the content.

2. What can I do now? – The interface should always help the user understand his current options-what functions are available, what links are live, what content is relevant.

3. Where have I been; where am I going? – The interface must facilitate navigation. Hence it must provide a ―map‖ of where the user has been and what paths may be taken to move elsewhere in the WebApp.

5.2 Design Goal

The following are the design goals that are applicable to virtually every WebApp regardless of application domain, size, or complexity.

1. Simplicity

2. Consistency

3. Identity

4. Visual appeal

5. Compatibility

Design leads to a model that contains the appropriate mix of aesthetics, content, and

technology. The mix will vary depending upon the nature of the WebApp, and as a consequence the design activities that are emphasized will also vary.

5.3 The Activity of the Design Process

1. Interface design-describes the structure and organization of the user interface. Includes a representation of screen layout, a definition of the modes of interaction, and a description of navigation mechanisms. Interface Control mechanisms- to implement navigation options, the designer selects form one of a number of interaction mechanism.

a. Navigation menus

b. Graphic icons

c. Graphic images

Interface Design work flow- the work flow begins with the identification of user, task, and environmental requirements. Once user tasks have been identified, user scenarios are created and analyzed to define a set of interface objects and actions.

2. Aesthetic design-also called graphic design, describes the ―look and feel‖ of the WebApp. Includes colour schemes, geometric layout. Text size, font and placement, the use of graphics, and related aesthetic decisions.

3. Content design-defines the layout, structure, and outline for all content that is presented as part of the WebApp. Establishes the relationships between content objects.

4. Navigation design-represents the navigational flow between contents objects and for all WebApp functions.

a. Content architecture, focuses on the manner in which content objects and structured for presentation and navigation.

b. WebApp architecture, addresses the manner in which the application is structure to manage user interaction, handle internal processing tasks, effect navigation, and present content. WebApp architecture is defined within the context of the development environment in which the application is to be implemented.

5.4 Architectural Design

System architecture is a conceptual model that defines the structure and behavior of the system. It comprises of the system components and the relationships describing how they work

together to implement the overall system
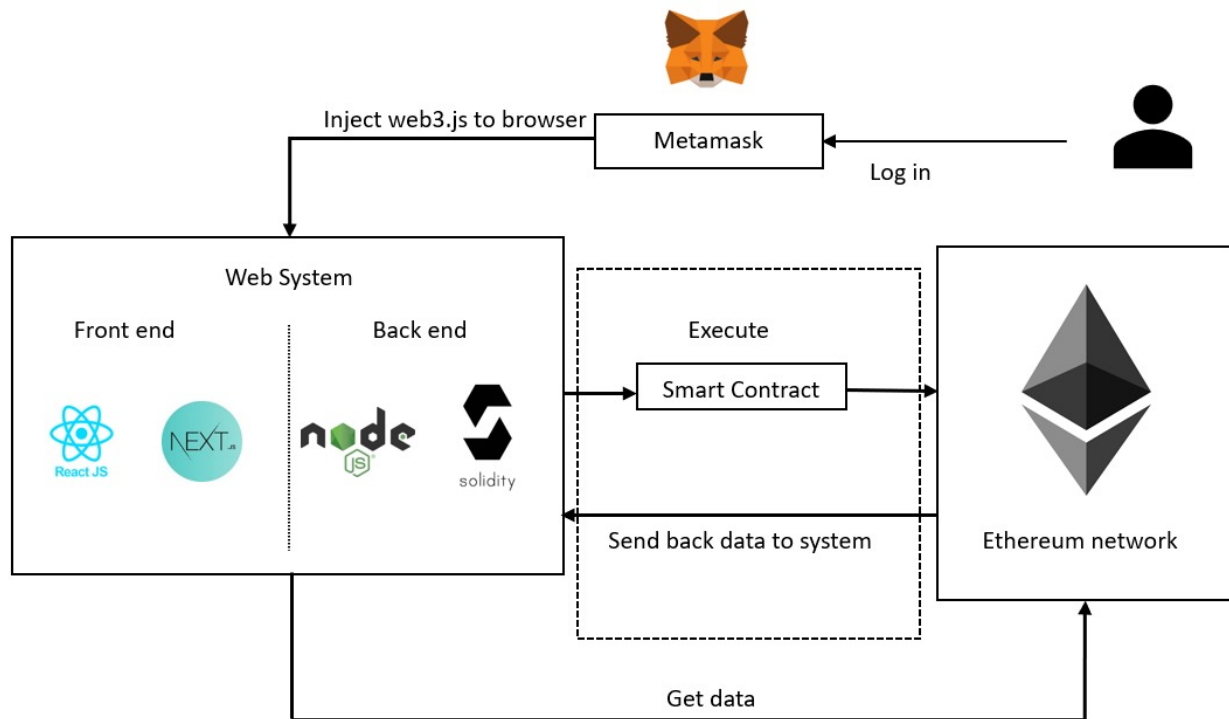
Fig 5.1 System Architecture

5.5 Flowchart
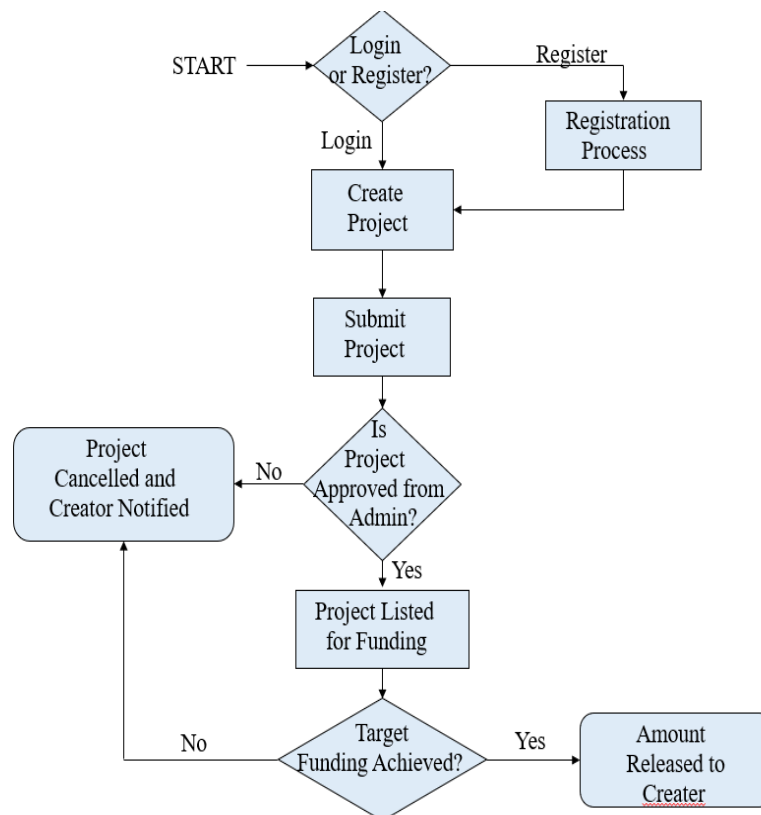
Fig 5.2 Flowchart

For the crowdfunding project utilizing blockchain and Ethereum, the flowchart provides a comprehensive depiction of the operational process. It begins with the creator initializing the

crowdfunding process by creating a project on the platform. Upon submission, the admin reviews the project details and grants approval for the initial funding stage. Backers, upon accessing the platform, are presented with the approved projects and can contribute funds to those they support. The platform diligently records all contributions and updates the project's funding status accordingly. Subsequently, the admin evaluates the project's progress and determines whether it advances to the second funding stage. If endorsed, backers are prompted to vote on the project's viability for continued funding. The platform records these voting outcomes and forwards them to the admin for review. Upon admin approval, the project enters the fund withdrawal phase, where smart contracts executed on the Ethereum network facilitate the transfer of funds. All transactions and project activities are securely recorded on the blockchain, ensuring transparency and accountability throughout the crowdfunding process.

5.6 Sequence Diagram
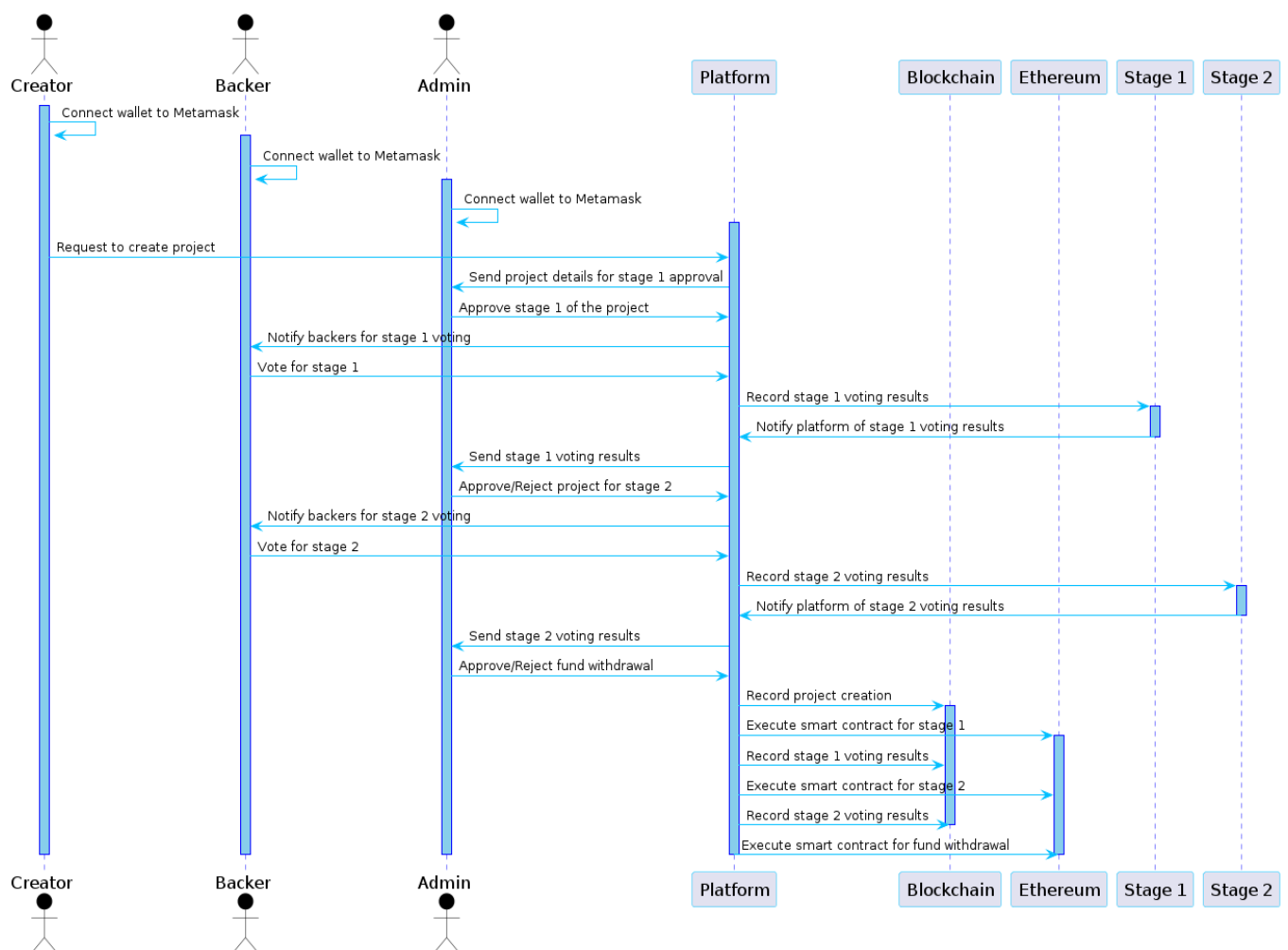
Fig 5.3 Sequence diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between

the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

5.7 Gannt Chart

The Gantt chart has been prepared for 1,2 phases of the project. The single date tasks have been depicted as milestones and are shown along the horizontal line. Similarly, the tasks which take multiple days to accomplish are shown in the vertical line. The Gantt chart has been shown below:

5.7.1 Gannt Chart For Phase-1
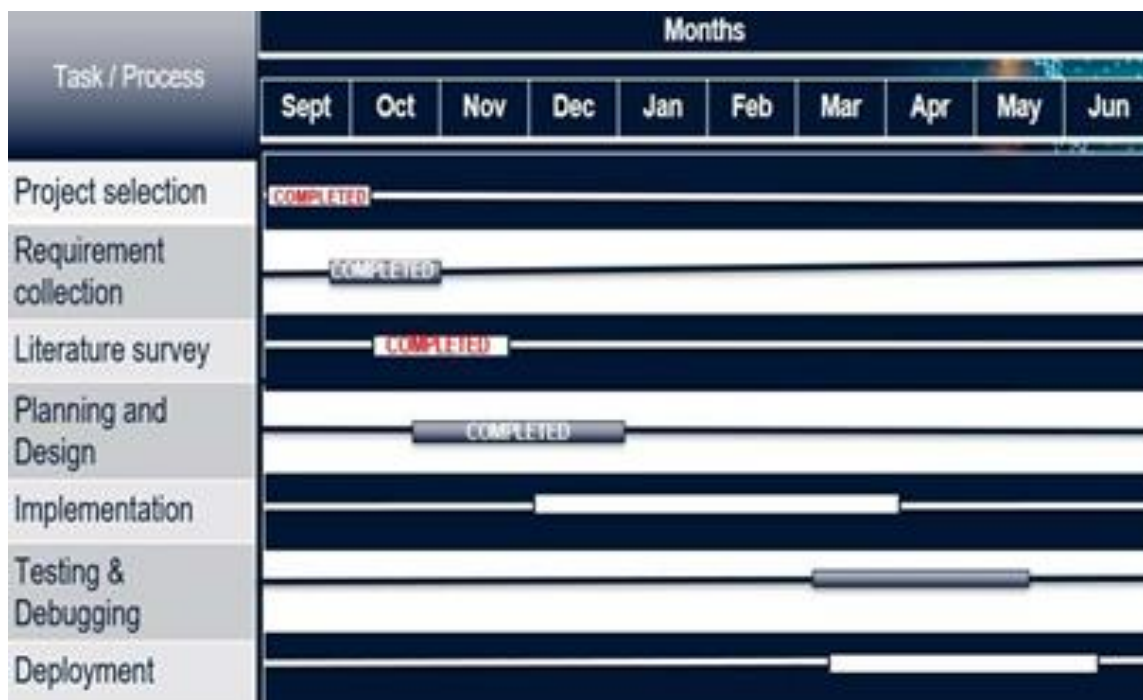
Fig 5.4 Gannt Chart Phase-1

5.7.2   Gannt Chart For Phase-2
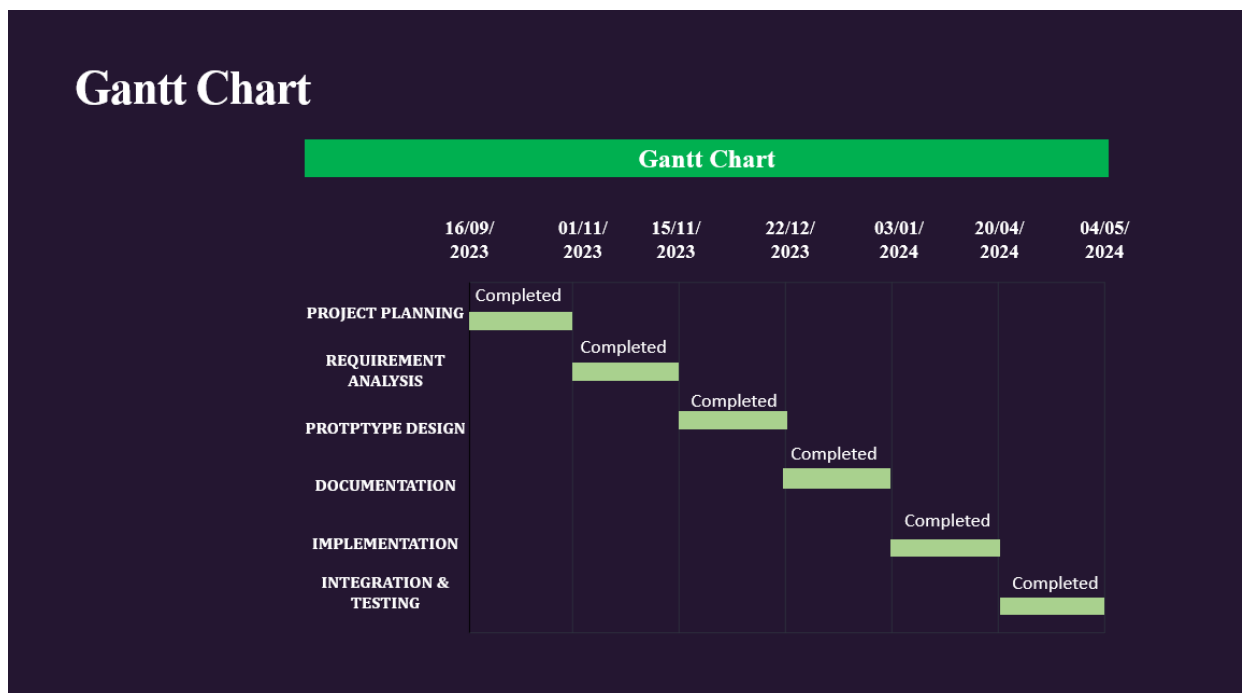
Fig 5.5 Gannt Chart Phase-2

5.8 Life Cycle Model

The agile model is applied for the software development process in our project. Agile means relating to or denoting a method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans. Phases involved in our project:

1. Recognizing hardware and software requirements

2. Developing and working on architecture diagram, class diagram, use case design

3. developing the software required for execution and working on GUI

4. Testing with various media file and checking if model classifies them correctly

5. Deployment on Windows OS and using VS Code.

6.Review and test of whether accuracy desired has been achieved.Repetitively, this

   process iterates to improve accuracy of classification.

Fig 5.6 Agile Model

5.9 Class Design
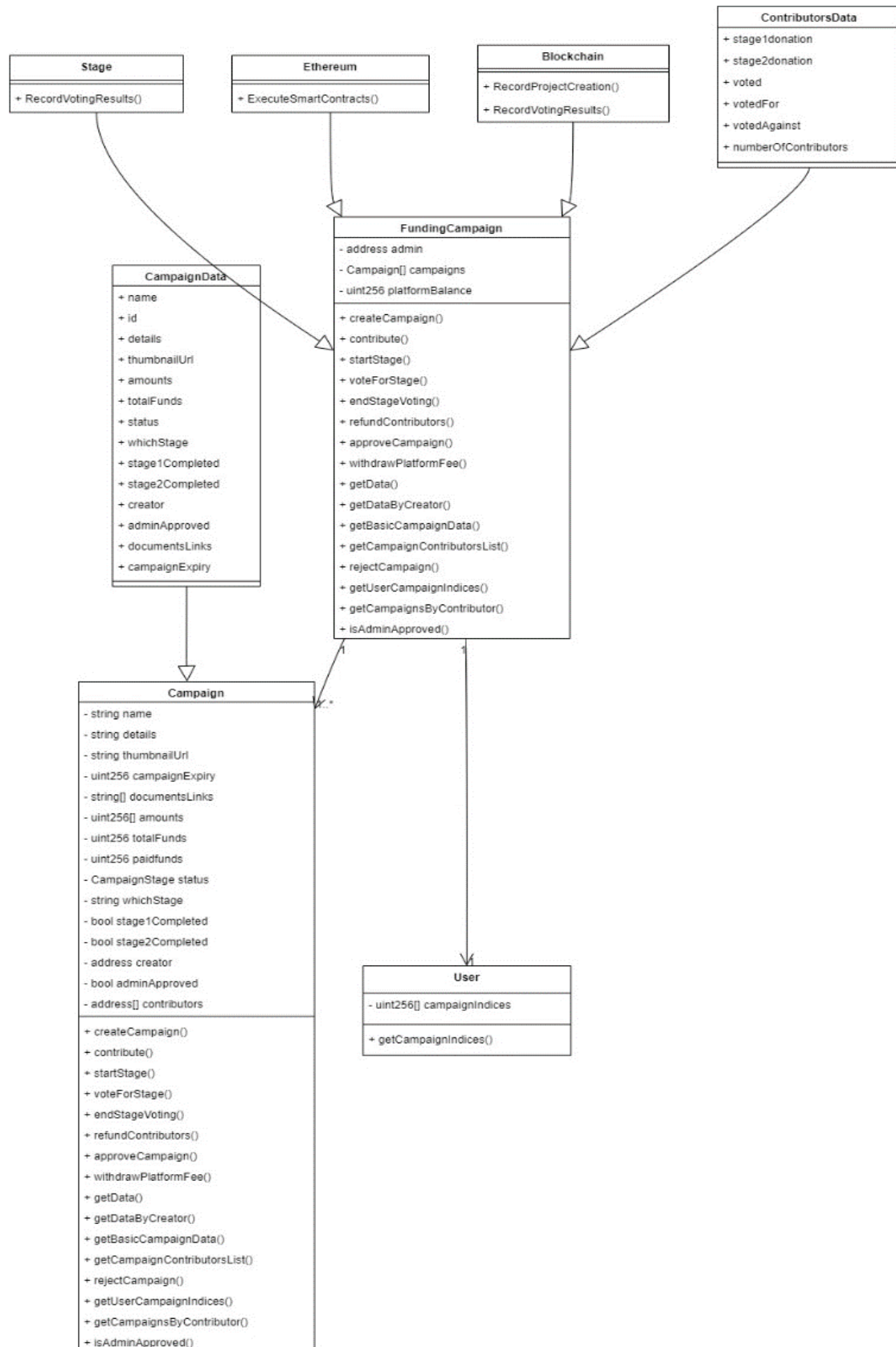
Fig 5.7 Class Design

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type

of static structure diagram that describes the structure of a system by showing the system's classes,

their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagram

1. Shows static structure of classifiers in a system

2. Diagram provides basic notation for other structure diagrams prescribed by UML.

3. Helpful for developers and other team members too

4. Business Analysts can use class diagrams to model systems from business perspective.

5.10 System Design Summary

This chapter shows the detailed design of the architecture, components, modules, interfaces, and data for the proposed system to satisfy specified requirements. It shows various standard UML diagrams that are needed to design the system. It provides a visualization of how the data will flow among various components of the system.

SYSTEM

IMPLEMENTATION

CHAPTER 6

SYSTEM IMPLEMENTATION

System Implementation uses the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developed in the early life cycle phases. These system elements are then integrated to form intermediate aggregates and finally the complete system-of-interest (SoI). Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). System elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing, the software realization processes of coding and testing, or the operational procedures development processes for operators' roles.

6.1 Modular Description

Modular design, or "modularity in design", is a design approach that subdivides a system into smaller parts called modules, that can be independently created and then used in different systems. A modular system can be characterized by functional partitioning into discrete scalable, reusable modules; rigorous use of well-defined modular interfaces; and making use of industry standards for interfaces.

Besides reduction in cost (due to less customization, and shorter learning time), and flexibility in design, modularity offers other benefits such as augmentation (adding new solution by merely plugging in a new module), and exclusion.

Our project consists of five main modules that can be further subdivided and programmed individually.

6.1.1 Data Collection Module

The data collection module for Crowdfunding using Blockchain involves gathering a dataset comprising project information, funding details, and user contributions. Each project entry includes images, descriptions, funding goals, and timelines. The dataset ensures transparency and accountability in crowdfunding campaigns.

6.1.2 Blockchain Integration Module

The core of our system lies in integrating blockchain technology, specifically Ethereum, for secure and transparent crowdfunding operations. Ethereum was chosen for its robust transaction processing capabilities, relatively low transaction fees, active development ecosystem, and support for smart contracts.

6.1.3 Smart Contract Module

Solidity, the programming language for Ethereum smart contracts, is utilized to develop and manage the smart contracts governing various aspects of crowdfunding. These smart contracts handle fund collection, distribution, withdrawal, rejection, and funder reimbursements, ensuring automated and secure fund management.

6.1.4 Crypto Wallet Integration Module

For secure and convenient transactions, we integrate popular crypto wallets such as MetaMask and WalletConnect. These wallets enable users to interact with the blockchain, manage their funds, and participate in crowdfunding campaigns directly from their preferred wallets.

6.1.5 Transparent Transaction Module

Our Transaction Tracking Module ensures transparent and secure fund management within the crowdfunding platform. It records all financial activities in real-time and provides transparent reporting on fund utilization. Integrated with a voting system, it enables contributors to participate in fund release decisions, enhancing transparency and accountability.

6.2 Program Code

Code for Campaign Creation

```
import { useState, useEffect } from 'react';
import { useRouter } from 'next/router';
import { useContract } from '@/components/ContractProvider';
import { useWeb3Modal } from '@web3modal/wagmi/react';
import { parseEther } from 'viem';
import { Bounce, toast } from 'react-toastify';
import { Box, Typography, Button } from '@mui/material';
import { useAccount } from 'wagmi';

const CreateCampaignForm = () => {
  const [formData, setFormData] = useState({
    name: '',
    details: '',
    thumbnailUrl: '',
    targetAmount: 0,
    stage1Amount: 0,
    stage2Amount: 0,
    documentsLinks: [''],
    campaignExpiryDate: '',
  });
  const [loading, setLoading] = useState(false);
  const { crowdFundindContract, executeContractWrite } = useContract();
  const { isConnected } = useAccount();
  const { open } = useWeb3Modal();
  const router = useRouter();

  useEffect(() => {
    if (!isConnected) router.push('/');
```

```
  }, [isConnected]);

  try {
    if (!isConnected) return open();
    setLoading(true);

    const [result, hash] = await executeContractWrite({
      address: crowdFundindContract.address,
      abi: crowdFundindContract.abi,
      functionName: 'createCampaign',
      args: [formData.name, formData.details, formData.thumbnailUrl, [parseEther(`${formData.targetAmount}`),
parseEther(`${formData.stage1Amount}`), parseEther(`${formData.stage2Amount}`)], epoch_time,
formData.documentsLinks],
    });

    setLoading(false);
  } catch (err) {
    setLoading(false);
    console.log(err.message.details);
  }
};

return (
  <>
    {!loading ? (
      <Box mx="auto">
        {/* Form UI */}
      </Box>
    ) : (
      <FullScreenLoading />
    )}
  </>
);
};

export default CreateCampaignForm;
```

Code for Explore Campaign

```
import React, { useEffect, useState } from 'react';
import { Card, CardContent, Typography, CardMedia, LinearProgress, Box, FormControl, InputLabel, Select,
MenuItem, TextField } from '@mui/material';
import { useContract } from '@/components/ContractProvider';
import { useWeb3Modal } from '@web3modal/wagmi/react';
import { useAccount } from 'wagmi';
import { useRouter } from 'next/navigation';
import Link from 'next/link';
import Web3 from 'web3';

const CampaignCard = () => {
  const [filter, setFilter] = useState('all');
  const [searchQuery, setSearchQuery] = useState('');
  const [campaigns, setCampaigns] = useState([]);
  const { crowdFundindContract, executeContractRead } = useContract();
  const { isConnected } = useAccount();
  const { open } = useWeb3Modal();
  const router = useRouter();

  useEffect(() => {
    async function fetchData() {
      try {
        if (!isConnected) {
          await open();
          router.push(isConnected ? '/explore' : '/');
        } else {
          await handleGetData();
```

```
          }
        } catch (e) {
          console.error(e);
        }
      }
      fetchData();
  }, []);

  const handleGetData = async () => {
    try {
      const result = await executeContractRead({
        address: crowdFundindContract.address,
        abi: crowdFundindContract.abi,
        functionName: 'getData',
        args: [],
      });

      const filteredCampaigns = result.filter((campaign) => {
        return campaign.whichStage || campaign.adminApproved;
      });

      const adminRejectedCampaigns = filteredCampaigns.filter((campaign) => {
        return campaign.whichStage !== 'Admin Rejected';
      });

      setCampaigns(adminRejectedCampaigns.reverse());
    } catch (e) {
      console.error(e);
    }
  };

  const handleChange = (event) => {
    setFilter(event.target.value);
  };

  const handleSearchChange = (event) => {
    setSearchQuery(event.target.value);
  };

  const formatDateString = (dateString) => {
    // Date formatting logic
  };

  const CheckStatus = (campaign) => {
    // Check campaign status logic
  };

  const filteredCampaigns = campaigns.filter((campaign) => {
    // Filter logic
  });

  return (
    <div>
      {/* Filter and search section */}
      <div>
        <TextField
          id="search"
          label="Search Campaigns"
          variant="outlined"
          value={searchQuery}
          onChange={handleSearchChange}
          style={{ margin: '8px', width: '100%' }}
        />
      </div>
      <div style={{ display: 'flex', gap: '20px' }}>
```

```jsx
            <div style={{ width: "30%" }}>
              {isConnected ? (
                <>
                  <FormControl style={{ margin: '8px', minWidth: '120px', width: '100%' }}>
                    <InputLabel
                      id="filter-label"
                      style={{ marginBottom: '4px', backgroundColor: '#fff', paddingLeft: '4px', paddingRight: '4px' }}
                    >
                      Filter
                    </InputLabel>
                    <Select
                      labelId="filter-label"
                      id="filter-select"
                      value={filter}
                      onChange={handleChange}
                      style={{ width: '100%' }}
                    >
                      {/* Menu items */}
                    </Select>
                  </FormControl>
                </>
              ) : (
                <Typography variant="body1">Please connect your wallet</Typography>
              )}
            </div>

            {/* Campaign cards section */}
            {/* Display filtered campaigns */}
          </div>

      </div>
    );
};

export default CampaignCard;
```

Code for ContractAddress

```ts
import { Abi, Address, getAddress } from 'viem'

import { crowdFundingAbi } from '../../abis/CrowdFunding'
import { localhost } from 'viem/chains'

export type ContractABIPair = {
    ADDRESS: Address
    ABI: Abi
}

// TODO: Add in contract deployments and their ABIs for each network supported
type ContractDeployments = {
    CROWD_FUNDING: ContractABIPair
}

const LOCALHOST: ContractDeployments = {
    // SimpleNFT: https://sepolia.etherscan.io/address/0x1cfD246a218b35e359584979dDBeAD1f567d9C88
    CROWD_FUNDING: {
        ADDRESS: getAddress('0x1E528898Dd0cFf9242ba2fb3A27c445EeD0EA6dF', localhost.id),
        ABI: crowdFundingAbi,
    },
}

const CONTRACTS = {
    LOCALHOST,
}
```

export default CONTRACTS

Code for Smart Contract

```solidity
pragma solidity ^0.8.1;

contract FundingCampaign {
    address public admin;

    enum CampaignStage {
        NotStarted,
        Stage1,
        Stage2,
        Completed
    }

    struct Campaign {
        string name;
        string details;
        uint256[] amounts;
        uint256 totalFunds;
        CampaignStage status;
        bool stage1Completed;
        bool stage2Completed;
        address creator;
        bool adminApproved;
        address[] contributors;
        mapping(address => uint256) contributions;
        mapping(address => uint256) stageContributions;
        mapping(address => bool) votedForStage;
        mapping(CampaignStage => uint256) stageVotesFor;
        mapping(CampaignStage => uint256) stageVotesAgainst;
    }

    struct User {
        uint256[] campaignIndices;
    }

    Campaign[] public campaigns;
    uint256 public platformBalance = 0;
    uint256 constant PLATFORM_FEE_PERCENTAGE = 1;

    event Contribution(
        address indexed contributor,
        uint256 amount,
        uint256 campaignIndex
    );
    event Withdrawal(uint256 amount, address recipient, uint256 campaignIndex);
    event StageCompletion(uint256 stage, uint256 campaignIndex);

    constructor() {
        admin = msg.sender;
    }

    modifier onlyAdmin() {
        require(msg.sender == admin, "Only admin can call this function");
        _;
    }

    function createCampaign(
        string memory _name,
        string memory _details,
        uint256[] memory _amounts,
        uint256 _expiryTimestamp,
        string[] memory _documentsLinks
    ) external {
```

```solidity
        require(msg.sender != address(0), "Creator address cannot be zero");
        require(bytes(_name).length > 0, "Campaign name cannot be empty");
        require(_amounts.length == 3, "Invalid amounts array length");
        require(
            _amounts[0] > 0 && _amounts[1] > 0 && _amounts[2] > 0,
            "Amounts must be greater than 0"
        );
        require(
            _amounts[1] + _amounts[2] == _amounts[0],
            "Stage 2 amount + Stage 1 amount must be equal to target amount"
        );
        Campaign storage newCampaign = campaigns.push();
        newCampaign.name = _name;
        newCampaign.details = _details;
        newCampaign.amounts = _amounts;
        newCampaign.creator = msg.sender;
        newCampaign.adminApproved = false;
        newCampaign.status = CampaignStage.NotStarted;
        newCampaign.campaignExpiry = _expiryTimestamp; // Assigning campaign expiry from user input
        newCampaign.documentsLinks = _documentsLinks; // Assigning document links from user input
    }

    function contribute(uint256 _campaignIndex) external payable {
        require(_campaignIndex < campaigns.length, "Invalid campaign index");
        require(msg.value > 0, "Contribution amount must be greater than 0");

        Campaign storage campaign = campaigns[_campaignIndex];

        require(
            campaign.status != CampaignStage.Completed,
            "Contribution not allowed in completed campaigns"
        );

        // Check if the campaign is expired
        require(
            block.timestamp < campaign.campaignExpiry,
            "Contribution not allowed, campaign has expired"
        );

        // Check if the campaign is in Stage 1 or Stage 2
        if (
            keccak256(bytes(campaign.whichStage)) == keccak256("stage1") ||
            keccak256(bytes(campaign.whichStage)) == keccak256("stage2")
        ) {
            uint256 stageIndex = getIndex(campaign.whichStage); // 2
            uint256 stageAmount;
            if (keccak256(bytes(campaign.whichStage)) == keccak256("stage1")) {
                stageAmount = campaign.amounts[stageIndex];
            } else {
                stageAmount = campaign.amounts[0]; // 5000000000000000000
            }
            require(
                campaign.totalFunds + msg.value <= stageAmount,
                "Contribution exceeds stage amount"
            );
            campaign.totalFunds = campaign.totalFunds + msg.value;
            // Update the contribution mapping based on the current stage
            if (
                keccak256(abi.encodePacked(campaign.whichStage)) ==
                keccak256("stage1")
            ) {
                if (campaign.stageContributions[msg.sender] == 0) {
                    User storage user = users[msg.sender];
                    user.campaignIndices.push(_campaignIndex);
                    campaign.contributors.push(msg.sender);
```

```
            }
            campaign.stageContributions[msg.sender] += msg.value;
        } else if (
            keccak256(abi.encodePacked(campaign.whichStage)) ==
            keccak256("stage2")
        ) {
            if (campaign.contributions[msg.sender] == 0) {
                User storage user = users[msg.sender];
                user.campaignIndices.push(_campaignIndex);
                bool senderExists = false;
                for (uint256 i = 0; i < campaign.contributors.length; i++) {
                    if (campaign.contributors[i] == msg.sender) {
                        senderExists = true;
                        break;
                    }
                }
                if (!senderExists) {
                    campaign.contributors.push(msg.sender);
                }
            }
            campaign.contributions[msg.sender] += msg.value;
        }
    } else {
        revert("Contribution not allowed in this stage");
    }
    emit Contribution(msg.sender, msg.value, _campaignIndex);
}

function startStage(uint256 _campaignIndex) external onlyAdmin {
    require(_campaignIndex < campaigns.length, "Invalid campaign index");

    Campaign storage campaign = campaigns[_campaignIndex];

    require(
        block.timestamp < campaign.campaignExpiry,
        "Stage Change not allowed, campaign has expired"
    );

    if (!campaign.stage1Completed || !campaign.stage2Completed) {
        if (
            keccak256(abi.encodePacked(campaign.whichStage)) ==
            keccak256(abi.encodePacked(""))
        ) {
            campaign.status = CampaignStage.Stage1;
            campaign.whichStage = "stage1";
        } else if (
            keccak256(abi.encodePacked(campaign.whichStage)) ==
            keccak256(abi.encodePacked("stage1")) &&
            campaign.stage1Completed
        ) {
            campaign.status = CampaignStage.Stage2;
            campaign.whichStage = "stage2";

            for (uint256 i = 0; i < campaign.contributors.length; i++) {
                address contributor = campaign.contributors[i];
                campaign.votedForStage[contributor] = false;
            }
        }
    } else {
        revert("Campaign already completed");
    }
}
```

```solidity
    }

    // Function to vote for a stage in a campaign
    function voteForStage(uint256 _campaignIndex, bool _vote) external {
        require(_campaignIndex < campaigns.length, "Invalid campaign index");

        Campaign storage campaign = campaigns[_campaignIndex];

        require(
            campaign.status != CampaignStage.NotStarted &&
            campaign.status != CampaignStage.Completed,
            "Voting not allowed in this stage"
        );

        // Check if the caller is eligible to vote based on the campaign stage
        if (campaign.whichStage == "stage1") {
            require(
                campaign.stageContributions[msg.sender] > 0,
                "Only contributors to Stage 1 can vote"
            );
        } else if (campaign.whichStage == "stage2") {
            require(
                campaign.contributions[msg.sender] > 0,
                "Only contributors to Stage 2 can vote"
            );
        }

        // Check if the caller has not already voted
        require(
            !campaign.votedForStage[msg.sender],
            "You have already voted for this stage"
        );

        // Update voting status and counts based on the vote
        campaign.votedForStage[msg.sender] = true;
        if (_vote) {
            campaign.stageVotesFor[campaign.status]++;
        } else {
            campaign.stageVotesAgainst[campaign.status]++;
        }
    }

    // Function to get the length of stage contributions in a campaign
    function getStageContributionsLength(uint256 _campaignIndex) external view returns (uint256) {
        require(_campaignIndex < campaigns.length, "Invalid campaign index");

        Campaign storage campaign = campaigns[_campaignIndex];

        uint256 count = 0;
        string memory currentStage = campaign.whichStage;

        // Count contributions based on the current stage
        if (keccak256(bytes(currentStage)) == keccak256(bytes("stage1"))) {
            for (uint256 i = 0; i < campaign.contributors.length; i++) {
                address contributor = campaign.contributors[i];
                if (campaign.stageContributions[contributor] > 0) {
                    count++;
                }
            }
        } else if (keccak256(bytes(currentStage)) == keccak256(bytes("stage2"))) {
            for (uint256 i = 0; i < campaign.contributors.length; i++) {
                address contributor = campaign.contributors[i];
                if (campaign.contributions[contributor] > 0) {
                    count++;
                }
            }
```

```
        }
      }

      return count;
    }

    // Function to end voting for a campaign stage
    function endStageVoting(uint256 _campaignIndex) external onlyAdmin {
        require(_campaignIndex < campaigns.length, "Invalid campaign index");

        Campaign storage campaign = campaigns[_campaignIndex];

        // Check if the campaign has reached the required votes for stage completion
        uint256 votesForStage = campaign.stageVotesFor[campaign.status];
        uint256 votesAgainstStage = campaign.stageVotesAgainst[campaign.status];
        uint256 requiredVotes = getStageContributionsLength(_campaignIndex) / 2;

        if (votesForStage > votesAgainstStage && votesForStage >= requiredVotes) {
            // Admin approves the current stage
            // Implement stage completion logic here
        } else {
            // Refund all contributors to the current stage
            refundContributors(_campaignIndex);
        }
    }
```

Code for Contract Provider

```
import { createContext, useCallback, useContext, useEffect, useState } from 'react'
import { Abi, Address, getContract, GetContractReturnType, parseEther } from 'viem'
import { useAccount, usePublicClient, useWalletClient } from 'wagmi'

// Types
type TxHash = Address | undefined
type ContractReadArgs = { address: Address; abi: Abi; functionName: string; args?: unknown[] }
type ContractWriteArgs = { address: Address; abi: Abi; functionName: string; args: unknown[]; value?: number }
type ContractContextValues = {
        executeContractRead: (args: ContractReadArgs) => Promise<unknown>
        executeContractWrite: (args: ContractWriteArgs) => Promise<[unknown, TxHash]>
        txSuccess: boolean
        txError: string | null
        resetTxNotifications: () => void
        crowdFundindContract: GetContractReturnType
}
type ContractProviderProps = {
        children: React.ReactNode
}

// Create context with initial values
const ContractContext = createContext<ContractContextValues>({
        executeContractRead: () => Promise.resolve(undefined),
        executeContractWrite: () => Promise.resolve([undefined, undefined]),
        txSuccess: false,
        txError: null,
        resetTxNotifications: () => { },
        crowdFundindContract: {} as GetContractReturnType,
})

// Context provider component
export const ContractProvider: React.FC<ContractProviderProps> = ({ children }: ContractProviderProps) => {
        const [txSuccess, setTxSuccess] = useState<boolean>(false)
        const [txError, setTxError] = useState<string | null>(null)
        const [crowdFundindContract, setCrowdFundindContract] = useState<GetContractReturnType>({} as
GetContractReturnType)

        const publicClient = usePublicClient()
```

```
            const { data: walletClient } = useWalletClient()
            const { address: account } = useAccount()

            const resetTxNotifications = () => {
                    setTxSuccess(false)
                    setTxError(null)
            }

            const executeContractRead = useCallback(async ({ address, abi, functionName, args }:
ContractReadArgs): Promise<unknown> => {
                    // Contract read logic here
            }, [publicClient])

            const executeContractWrite = useCallback(async ({ address, abi, functionName, args, value }:
ContractWriteArgs): Promise<[unknown, TxHash]> => {
                    // Contract write logic here
            }, [publicClient, walletClient, account])

            useEffect(() => {
                    if (walletClient && publicClient) {
                            setCrowdFundindContract(
                                    getContract({
                                            // Contract configuration
                                    }),
                            )
                    }
            }, [walletClient, publicClient])

            return (
                    <ContractContext.Provider
                            value={{
                                    crowdFundindContract,
                                    executeContractRead,
                                    executeContractWrite,
                                    txSuccess,
                                    txError,
                                    resetTxNotifications,
                            }}
                    >
                            {children}
                    </ContractContext.Provider>
            )
}

export const useContract = () => {
        const context: ContractContextValues = useContext(ContractContext)
        if (context === undefined) {
                throw new Error('useContract must be used within a ContractProvider component.')
        }
        return context
}
```

6.3 System Implementation Summary

This chapter shows the implementation of the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developed in the early life cycle phases. It shows the segment of programming code that is used in order to implement this project.

TESTING

CHAPTER 7

TESTING

7.1 Unit Testing:

Unit tests are a form of software testing that focuses on verifying the individual functionality of its smallest components, called units. These can be functions, methods, or classes of our application. The main purpose of unit tests is to validate the correct behavior and functionality of these units.

7.2 Integration Testing:

Integration testing involves testing the integration of various modules and components of the system to ensure they work together as expected. In a case of crowdfunding platform project integration testing would involve testing blockchain, smart contract and transaction segment.

7.3 System Testing:

System testing involves testing the entire system as a whole to ensure that it meets a specified requirements and performs as expected.

7.4 Performance Testing:

Performance testing is involves testing the system's performance under various loads and the conditions to ensure that it's meet performance requirements. In the case of the crowdfunding platform project performance testing would involve testing the system's responsive time, and throughput, reliability under various loads and conditions.

7.5 Security Testing:

Security testing involves a testing the system's security measures to ensure that they are very effective and protect the system from unauthorized access and attacks. In the case of our crowdfunding platform project security testing would have involve testing the blockchain and smart contract and the transaction segments.

7.6 User Acceptance Testing:

User acceptance testing involves testing the system from the user's perspective to ensure that it meets their requirements and expectations. In the case of the crowdfunding platform project user acceptance testing would involve testing the system's usability, and ease of use, and the user interface.

7.7 Test Cases:

Testcase ID	Testcase Type	Step Name	Results	Status
TC001	Functional	Connecting to wallet functionality	Wallet provider selection without errors. Authentication process smooth. Successful connection. Wallet details visible.	Successful
TC002	Usability Testing	Evaluate Home Page Layout and Navigation	Clear layout with accessible navigation. Consistent menus. Readable text. Responsive design. Functional interactive elements.	Successful
TC003	Transaction	Verify Funds Transfer via MetaMask	MetaMask prompts authentication. Transaction details accurate. Successful transfer confirmed. Balance updated.	Successful
TC004	Error Message	Validate Mandatory Fields and Constraints	Proper error messages for missing fields. Constraints enforced. Guidance for correction provided.	Unsuccessful

Table 7.1 Test Cases

OUTPUTS & SCREENSHOTS

CHAPTER 8

OUTPUTS AND SCREENSHOTS

8.1 Outcome of The Project

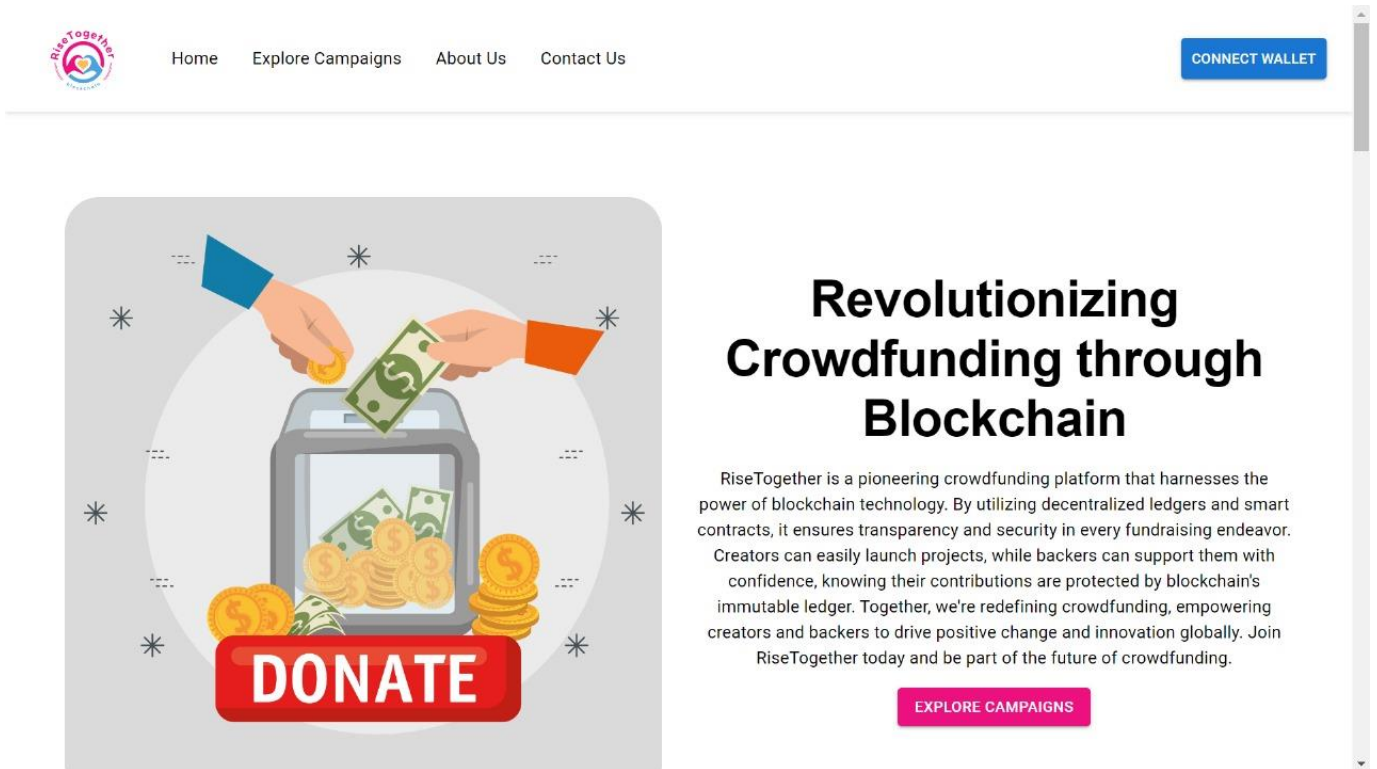**Fig 8.1 Homepage**

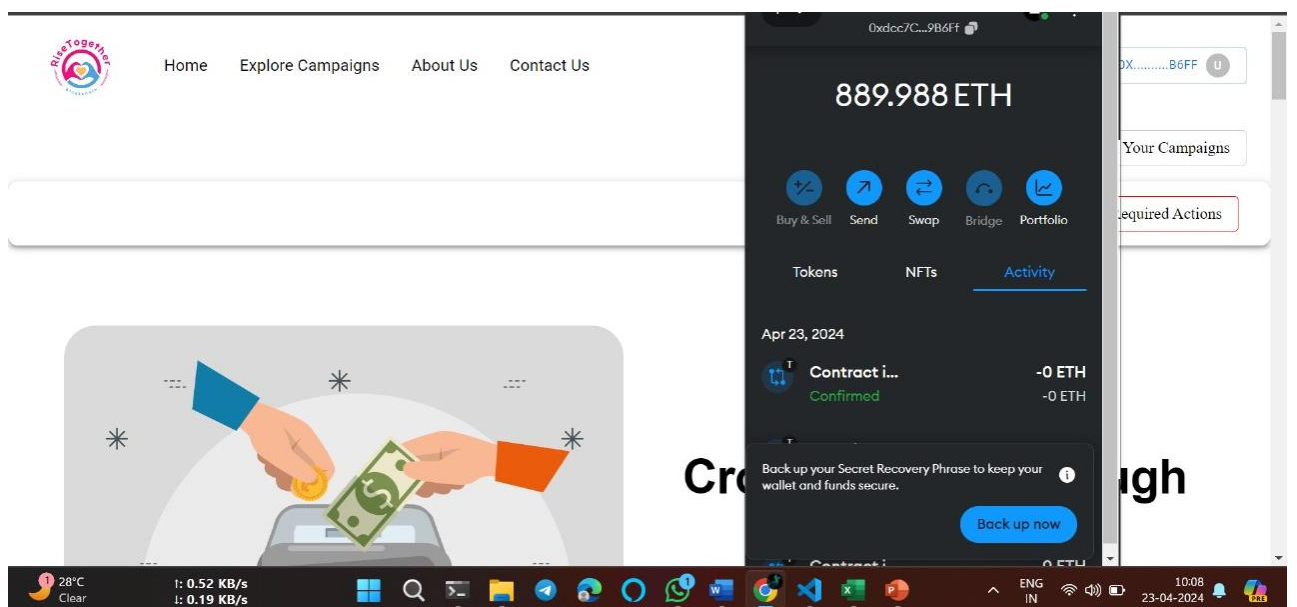**Fig 8.2 Wallet Integration**
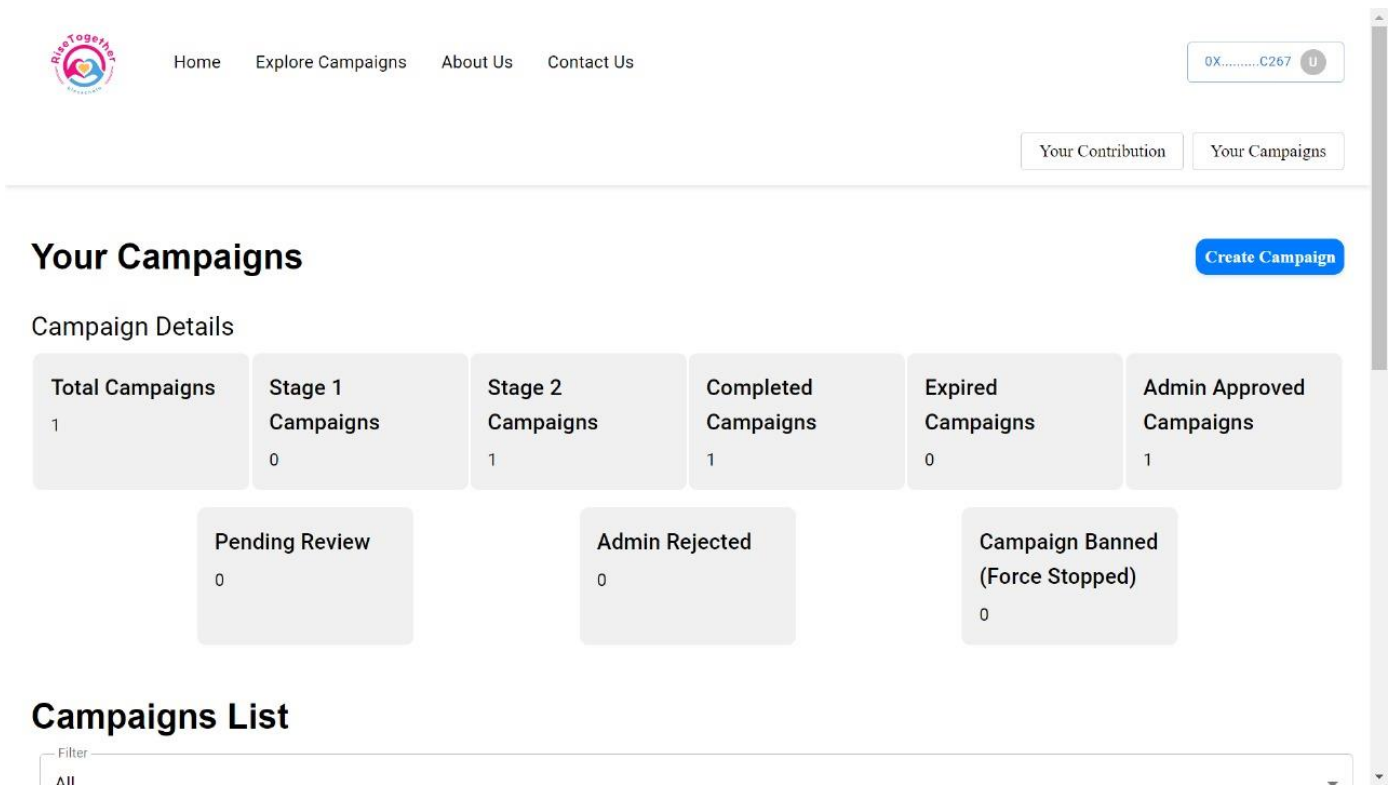
Fig 8.3 UserView-1

Fig 8.4 UserView-2

Fig 8.5 Campaign Creation

8.2 Output Summary

This chapter shows various screenshots of the system. It also shows how data processing happens at various stages of the system and the final output is also displayed. And it also shows the outer interface design of the system.

CONCLUSION

CONCLUSION

Our project, titled "Rise Together: A Crowdfunding Platform powered by Blockchain," addresses the inherent issues of transparency and fraud prevalent in conventional crowdfunding methods. With the implementation of a smart contract-based solution, our primary objective is to establish a secure environment for investors, ensuring the safety of their donations while actively involving them in the decision-making process regarding the startup's expenditure. By integrating blockchain technology into our crowdfunding platform, we elevate transaction transparency, fostering a sense of trust and confidence among users. Smart contracts play a pivotal role in this initiative, offering contributors real-time insights into the utilization of funds and guaranteeing the security of financial contributions. Our platform meticulously maintains a detailed transaction history, effectively mitigating the risk of fraudulent activities. Our overarching goal is to create a crowdfunding platform that is not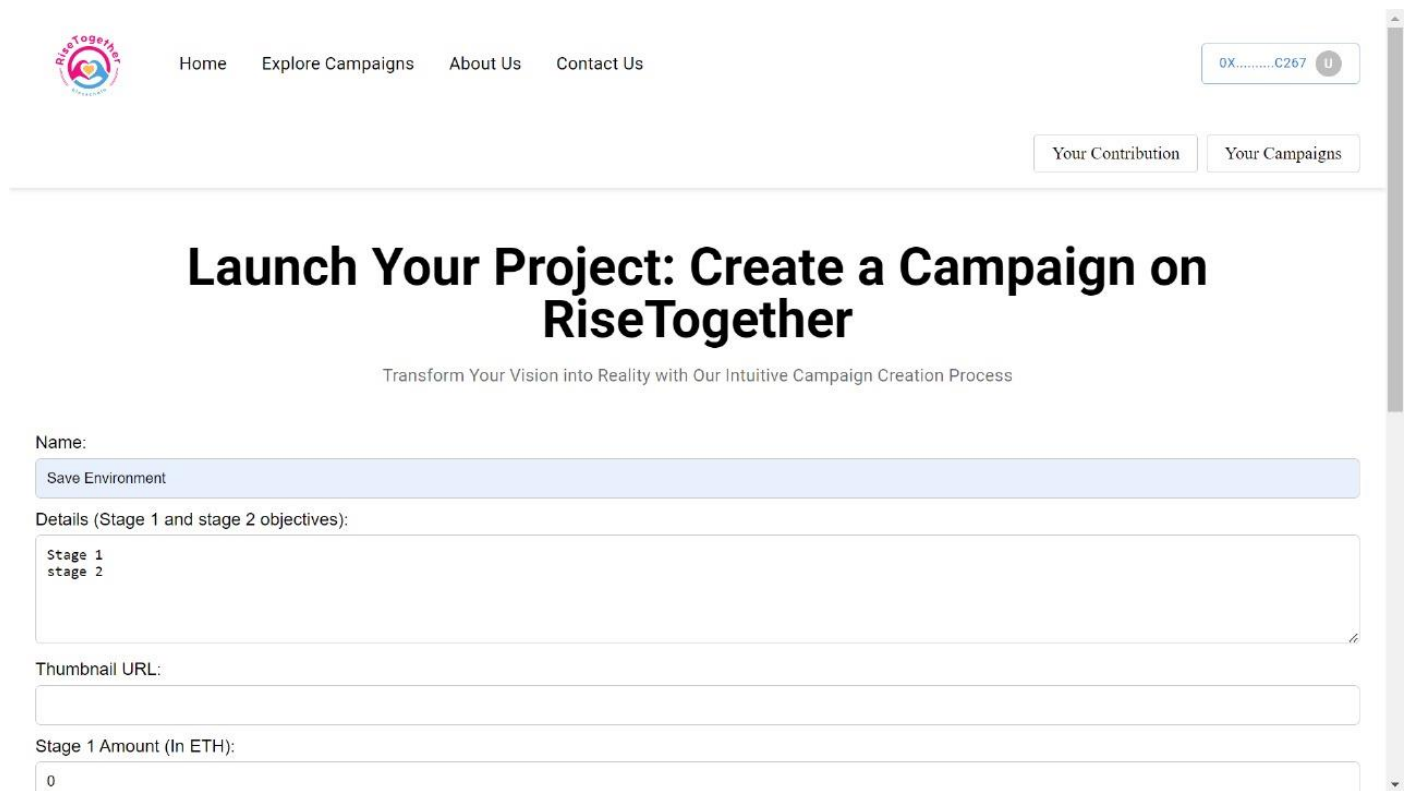 only transparent but also resilient against fraudulent practices, embodying the principles of decentralization. We recognize the current limitations and are committed to addressing them in future iterations, continuously striving to enhance the security and reliability of blockchain-based crowdfunding. Through iterative development, we aim to redefine the landscape of crowdfunding, making it more accountable, secure, and accessible to a global audience.

SCOPE FOR FUTURE WORK

SCOPE FOR FUTURE WORK

In the scope for future work, the main goal is to improve the crowdfunding platform using blockchain technology while overcoming current limitations. This involves exploring advanced blockchain solutions to handle more transactions and reduce network congestion. Additionally, integrating decentralized storage solutions can improve data storage and retrieval capabilities. To address transparency concerns, advanced blockchain consensus methods can be considered, providing a more energy-efficient and secure environment for crowdfunding transactions. Smart contract upgrades focusing on efficiency and cost-effectiveness will streamline fund management processes. Furthermore, exploring interoperability solutions can enable seamless interaction between different blockchain networks, expanding the platform's reach. Implementing decentralized identity solutions can strengthen user authentication and privacy protection, ensuring a secure and trusted environment for participants. Overall, the future work aims to leverage advanced blockchain technologies to create a more efficient, secure, and user-friendly crowdfunding ecosystem.

REFERENCES

REFERENCES

[1]   Menon, Arjun and Kadam, Kaustubh and Kumar, Pranav and Shah, Subash Kumar, Decentralized Crowdfunding Using Blockchain (January 15, 2023). Available at SSRN: https://ssrn.com/abstract=4324640 or DOI: 10.2139/ssrn.4324640.

[2]   Zad, Saniya and Khan, Zishan and Warambhe, Tejas and Jadhav, Rushikesh and Alone, Vinod, Crowdfunding Using Blockchain Technology (December 31, 2022). Available at SSRN: https://ssrn.com/abstract=4330476 or DOI: http://dx.DOI.org/10.2139/ssrn.4330476.

[3]   Laxman Pandhare, Prof. Priyanka Shirbhate (2022). Review On Blockchain Technology. Ijraset Journal for Research in Applied Science and Engineering Technology. DOI Link: https://DOI.org/10.22214/ijraset.2022.39833.

[4]   Ritvik Gupta, Mayank Yadav, Usha Dhankar. Crowdfunding using Ethereum Blockchain. Ijraset Journal for Research in Applied Science and Engineering Technology (2022). DOI: 10.22214/ijraset.2022.43130.

[5]   Dhansri Sudhir Bawankule, Rushikesh Bharatrao Adhau, Shyamli Rajesh Deotale, Kiran Laxman Pandhare, Prof. Priyanka Shirbhate (2022). Review On Blockchain Technology. Ijraset Journal for Research in Applied Science and Engineering Technology. DOI Link: https://DOI.org/10.22214/ijraset.2022.39833.

[6]   Ashari, Firmansyah. Smart Contract and Blockchain for Crowdfunding Platform. International Journal of Advanced Trends in Computer Science and Engineering (2020). 9. 3036-3041. 10.30534/ijatcse/2020/83932020.

[7]   Hassija, Vikas & Chamola, Vinay & Zeadally, Sherali. (2020). BitFund: A Blockchain- based Crowdfunding Platform for Future Smart and Connected Nation. Sustainable Cities and Society. 60. 102145. 10.1016/j.scs.2020.102145.

[8]   Dhiman, Tushar and gulyani, Vidit and Bhushan, Bharat, Application, Classification and System Requirements of Blockchain Technology (May 14, 2020). Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Available at SSRN: https://ssrn.com/abstract=3600745 or http://dx.DOI.org/10.2139/ssrn.3600745.

[9]   Lilani, Siddharth & Modi, Jimit & Malani, Dhrumil & Soni, Fenil. (2019). Securing the Software Development Life Cycle (SDLC) with a Blockchain Oriented Development Approach.

[10]   Sarkar, Abhrajit, Crowd Funding in India: Issues & Challenges Available at Social Science Research Network February 27, 2016.SSRN: https://ssrn.com/abstract=2739008 or http://dx.DOI.org/10.2139/ssrn.2739008

PAPER PUBLISHED

Rise Together: A Crowdfunding Platform

Assistant Professor Mrs. Kavyashree J

Bhavana S K, Madhu S, N Deepak, Santhosh P

Department of Computer Science and Engineering

Rajiv Gandhi Institute of Technology

Bangalore-560032, Karnataka

Abstract - The project titled "Rise Together: A Crowdfunding platform" introduces a groundbreaking Crowdfunding Platform on the Ethereum blockchain, revolutionizing traditional fundraising models. Leveraging smart contracts for seamless automation and execution of campaigns, the platform ensures transparency, security, and efficiency. Ethereum's decentralization eliminates intermediaries, reducing costs, and enhancing accessibility. Key features include a user-friendly interface, contribution tracking, and real time campaign monitoring. Utilizing Ether as the native cryptocurrency streamlines contributions, offering a secure and standardized means of support. Positioned at the intersection of blockchain and fundraising, this project contributes to the evolution of decentralized finance, showcasing Ethereum's potential for secure, inclusive, and innovative crowdfunding.

Key Words: Rise Together, Crowdfunding, Blockchain, Smart Contract, Transparency, Decentralization, Wallet Integration.

1. INTRODUCTION

Blockchain Overview

Blockchain is a decentralized and distributed ledger technology that securely records transactions across a network of computers. It relies on a chain of blocks, where each block contains a cryptographic hash of the previous one, ensuring data integrity. The decentralized network of nodes reaches consensus on transaction validity through mechanisms like Proof of Work or Proof of Stake. Cryptographic hash functions and immutability make the recorded data resistant to tampering. Smart contracts, self-executing code, automate and enforce predefined contract terms. Transactions are transparent, visible to all participants, while participant identities remain pseudonymous. Blockchain applications extend beyond finance to supply chain, healthcare, and voting systems. Its security, transparency, and resistance to censorship contribute to its widespread interest and adoption.

Key elements of a blockchain

*   Distributed ledger technology

All network participants have access to the distributed ledger and its immutable record of transactions. With this shared ledger, transactions are recorded only once, eliminating the duplication of effort that's typical of traditional business networks.

*   Immutable records

No participant can change or tamper with a transaction after it's been recorded to the shared ledger. If a transaction record includes an error, a new transaction must be added to reverse the error, and both transactions are then visible.

*   Smart contracts

To speed transactions, a set of rules called a smart contract is stored on the blockchain and executed automatically. A smart contract can define conditions for corporate bond transfers, include terms for travel insurance to be paid and much more.

How blockchain works

*   As each transaction occurs, it is recorded as a block of data.

*   Each block is connected to the ones before and after it.

*   Transactions are blocked together in an irreversible chain: a blockchain.

ABOUT THE PROJECT

Blockchain crowdfunding platforms provide many ways for startups to raise money to create their own digital currencies. It is a peer-to-peer financial model, some of the most famous cryptocurrencies are Bitcoin, Ethereum, etc. customer. If anyone creates a new cryptocurrency (like Ethereum), anyone can join this fundraiser and contribute as much as they want. Ethereum blockchain provides the basis for crowdfunding decentralized applications. Ethereum is a decentralized blockchain that can be designed to work in a variety of applications thanks to its many features. Ethereum is programmable due to its diversity and functionality. It is important to understand Ethereum before you start using distributed applications. We will use the open source, decentralized Ethereum platform (Decentralized Applications) to build and use decentralized applications. Solidity is an Ethereum programming language that allows us to create smart contracts that run automatically when certain events occur. Learning Ethereum is necessary to create user-friendly implementations of the integrity of the system. Ethereum's decentralized structure, DAO support, and smart contracts make it an excellent platform for creating crowdsourced websites that can operate independently without intermediaries. The main aim of the proposed project is to solve the gap in the crowd by making the business safe, eliminating middlemen/agencies and ensuring transparent budget of the platform.

2. Body of Paper

Ethereum-based crowdfunding decentralized app uses smart contracts to organize crowdfunding fundraising. These smart contracts automatically transfer money to project developers Section is subject to the rules and conditions listed previously. Use intermediaries such as banks or other financial institutions Thanks to this approach, there is no longer a need for organizations. Decentralized autonomous organization (DAO) is an entity managed by smart contracts and completes itself as a smart contract. The code is also supported by Ethereum. DAO can be used to manage revenue was created and distributed through crowdfunding, eliminating the need for a middleman. Ethereum (ETH), a cryptocurrency and virtual Ethereum used to fund online transactions Machines (EVMs) that execute smart contracts are two unique features of Ethereum. Ethereum is great because of its decentralized structure and smart contracts A platform for building crowdsourced applications.
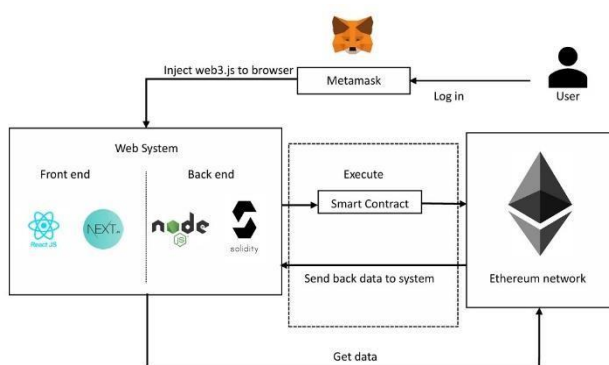
Fig-1: System Architecture
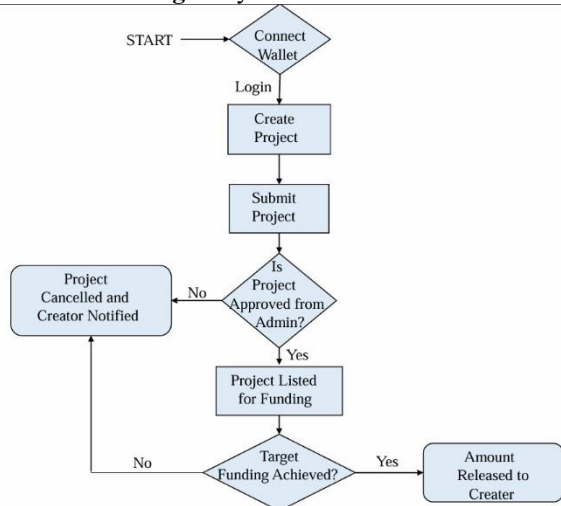
Fig-2: Data Flow

This system empowers users to bring their ideas to life by creating and potentially securing funding for projects through cryptocurrency. The journey begins by connecting a crypto wallet, acting as a source for both project initiation fees and future project payouts. Once connected, users can craft their project, outlining its details and objectives. After submission, a project admin assesses it for approval. If it receives the green light, the project goes live for public funding. This opens the door for interested supporters to explore projects and contribute cryptocurrency to help reach the targeted amount. Upon successful fundraising, the platform releases the funds to the project creator. In essence, this flowchart serves as a roadmap, guiding you through the process of launching your project fuelled by cryptocurrency on this platform.

Fig-3: Sequence Diagram

This platform fosters project creation and potential funding
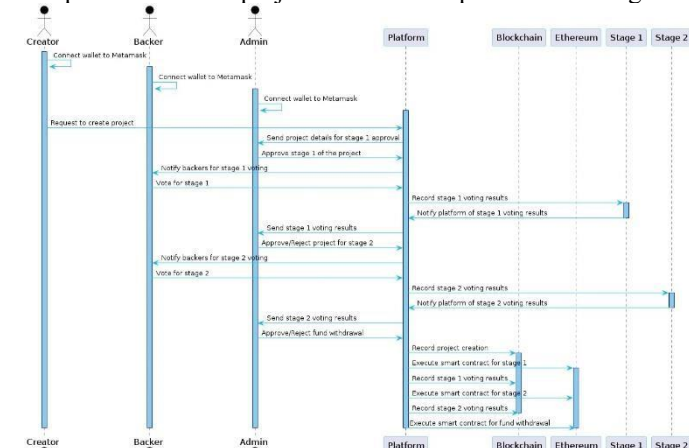
through cryptocurrency. First, creators link their crypto wallets, used for both project fees and receiving raised funds. They then submit their project proposal, including details and goals. An admin reviews it for adherence to platform guidelines. If approved, the project enters Stage 1, where backers are notified to vote. Based on the vote and the admin's decision, the project advances to Stage 2 (if applicable) with potentially more details or funding required. This stage mirrors Stage 1 with admin review, backer voting, and final approval. Finally, upon successful completion of both stages, the platform releases the funds to the project creator. This flowchart essentially navigates you through launching your cryptocurrency-driven project on this platform.

Fig-4: Class Diagram
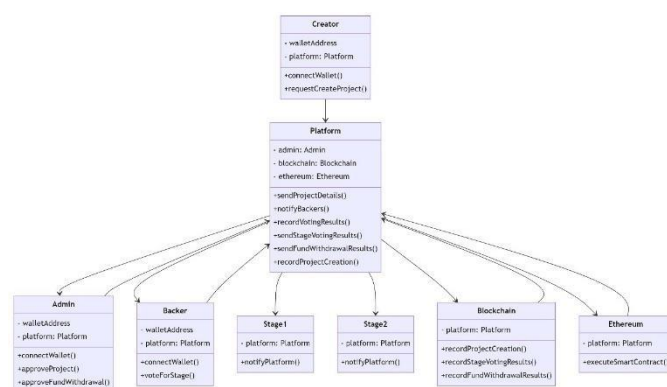
3. RELATED WORK

This paper explores the emergence of blockchain-based crowdfunding platforms and their potential to transform traditional fundraising models. We begin by reviewing relevant academic literature and ongoing projects to identify the advantages and challenges of this innovative approach.

TECHNOLOGIES USED

Web3 is a broad term encompassing the technologies behind decentralized applications (Dapps). It refers to the concept of a

new, more open web built on blockchains, where data and applications are distributed and user-owned.

Solidity is a programming language specifically designed for writing smart contracts. Smart contracts are self-executing contracts with predefined terms stored on a blockchain. Solidity allows developers to create secure and transparent applications that run on blockchains like Ethereum.

Next.js is a React framework that simplifies building server side rendered and statically generated web applications. In the context of dApps, Next.js provides a robust foundation for creating user-friendly interfaces that interact with smart contracts deployed on a blockchain.

Here's a breakdown of how these technologies collaborate:
Solidity Smart Contract: Developers write the logic for the dApp using Solidity. This logic defines the rules and functionalities of the application, such as user authentication, data storage, and transaction processing.

Deployment: The smart contract is then compiled and deployed onto a blockchain network, like Ethereum. Once deployed, the contract becomes immutable and transparent, meaning anyone can view its code and verify its execution.

Next.js Frontend: The user interface of the dApp is built using Next.js. This involves creating components that interact with the deployed smart contract. Tools like web3.js are often used to facilitate this communication between the frontend and the blockchain.

User Interaction: Users interact with the dApp through the Next.js interface. They can trigger functions defined within the smart contract, such as sending and receiving cryptocurrency, voting on proposals, or managing digital assets.

Benefits of using this Tech Stack:

Decentralization: By leveraging blockchain technology, dApps built with Web3, Solidity, and Next.js are censorship resistant and operate without a central authority. Security: Smart contracts are tamper-proof and transparent, ensuring secure and reliable execution of dApp functionalities. **Scalability:** Blockchain networks can handle a high volume of transactions, making dApps built on this stack suitable for largescale applications.

User-friendly Interfaces: Next.js provides a powerful framework for building user-friendly and interactive dApp interfaces.

CURRENT RESEARCH LANDSCAPE

Trust, Transparency, and Accessibility: Examine how blockchain technology, through features like smart contracts and consensus mechanisms, can foster trust, transparency, and accessibility within crowdfunding campaigns. This paves the way for further exploration of blockchain's ability to address limitations in current crowdsourcing methods. Decentralized Finance (DeFi) and Crowdfunding: Buterin (2017) delves into the integration of DeFi and crowdfunding. Their research highlights the evolving role of Ethereum Decentralized Applications (DApps) in fundraising and underscores the significance of smart contracts in automating processes and reducing reliance on intermediaries. **Smart Contract-Based Engine**: Emphasize the critical role of smart contracts in blockchain crowdfunding platforms. Their research demonstrates how these automated administrative processes manage competition policies, budgets, and facilitate more efficient and effective fundraising.

Security and Transparency: Cornerstones of Success: Zhang et al. (2020) investigate how blockchain technology's

immutable data storage can minimize the risk of fraud and forgery, thereby bolstering stakeholder trust. Security and transparency are fundamental for blockchain crowdfunding platforms to thrive.

User Experience: A Critical Factor: Kim et al. (2021) shift the focus to user experience (UX) within blockchain crowdfunding platforms. Their research emphasizes the importance of features like user-friendly interface design, seamless wallet connectivity, and transparency to enhance usability and accessibility for all participants.

Practical Application: Examining real-world applications, such as the "Rising Together: Crowdfunding Platform," offers valuable insights into the practical utilization of blockchain crowdfunding. Additional case studies can further illuminate the diverse applications of these platforms.

Future Considerations and Challenges

Environmental Management: Swan and Sundararajan (2022) highlight the need for ongoing research and innovation to address challenges faced by blockchain crowdfunding platforms. Compliance management, scalability, and robust security remain crucial areas of focus.

Security Research and Best Practices: Research focused on security best practices for blockchain crowdfunding platforms is essential to safeguard public funds. Investigate how blockchain crowdfunding is affecting traditional financial institutions and how they are adapting or integrating this technology.

CORE MODULES

This detailed explanation shows the main structure of a powerful blockchain-based crowdfunding platform. Each mode plays an important role in improving the accuracy of the data used in the system. The preprocessing module serves as the basis for cleaning and processing the data to ensure the reliability of the analysis. Blockchain integration enhances this further by creating a secure and tamper-proof record of transactions. Smart contracts add another layer of trust by automatically distributing money according to a certain priority. Expanding international collaboration through special features can foster knowledge acquisition with a wide range of sponsors. Finally, the transparent spend module uses the power of blockchain to track income in real time, ensuring that spend information is accurate and verifiable. Essentially, each mode works together to create a data-driven and secure ecosystem that benefits everyone involved.

PROJECT OUTCOME

The impact of Rise Together extends far beyond a mere fundraising platform. Since its inception, the platform has sparked a surge in crowdfunding campaigns, attracting a significant influx of cryptocurrency from a diverse array of global contributors. This democratization of fundraising not only empowers creators but also galvanizes supporters, fostering a sense of collective participation. What sets Rise Together apart is its intuitive user interface, which streamlines campaign creation and encourages involvement by accepting various cryptocurrencies. However, its true strength lies in its unwavering commitment to transparency and equity. Through a unique voting system, contributors actively shape the destiny of campaigns, ensuring that only the most deserving causes receive funding. Moreover, rigorous admin verification procedures instil further confidence, guaranteeing the

legitimacy of each endeavour. As funds securely flow to approved campaigns, Rise

Together transcends geographical barriers, uniting disparate communities in pursuit of shared objectives. With a steadfast vision for global positive change, Rise Together serves as a potent catalyst for collective action and generosity, illuminating the path towards a brighter tomorrow.

4. CONCLUSION

In conclusion, "Rise Together" stands as a solution to the limitations of traditional crowdfunding through the transformative power of blockchain technology. Our platform prioritizes transparency and security for both investors and startups. By leveraging smart contracts, we empower investors with real-time control and assurance over their contributions. Furthermore, the immutable transaction history diminishes the potential for fraud. "Rise Together" aspires to not only be transparent but also a bastion against fraudulent activity, all while upholding the core principles of decentralization. We acknowledge there's room for improvement, and future iterations will focus on solidifying the security and dependability of our blockchain-powered crowdfunding platform. Through continuous development, "Rise Together" strives to redefine the crowdfunding landscape, fostering a more accountable, secure, and globally accessible ecosystem.

5. REFERENCES

1. Menon, Arjun and Kadam, Kaustubh and Kumar, Pranav and Shah, Subash Kumar, Decentralized Crowdfunding Using Blockchain (January 15, 2023). Available at SSRN: https://ssrn.com/abstract=4324640 or DOI: 10.2139/ssrn.4324640.

2. Zad, Saniya and Khan, Zishan and Warambhe, Tejas and Jadhav, Rushikesh and Alone, Vinod, Crowdfunding Using Blockchain Technology (December 31, 2022). Available at SSRN: https://ssrn.com/abstract=4330476 or http://dx.DOI.org/10.2139/ssrn.4330476.

3. Laxman Pandhare, Prof. Priyanka Shirbhate (2022). Review On Blockchain Technology. Ijraset Journal for Research in Applied Science and Engineering Technology. DOI Link: https://DOI.org/10.22214/ijraset.2022.39833.

4. Ritvik Gupta, Mayank Yadav, Usha Dhankar.
   Crowdfunding using Ethereum Blockchain. Ijraset Journal for Research in Applied Science and Engineering Technology (2022). DOI: 10.22214/ijraset.2022.43130.

5. Dhansri Sudhir Bawankule, Rushikesh Bharatrao Adhau, Shyamli Rajesh Deotale, Kiran Laxman Pandhare, Prof. Priyanka Shirbhate (2022). Review On Blockchain Technology. Ijraset Journal for Research in Applied Science and Engineering Technology. DOI Link: https://DOI.org/10.22214/ijraset.2022.39833.

6. Ashari, Firmansyah. Smart Contract and Blockchain for Crowdfunding Platform. International Journal of Advanced Trends in Computer Science and Engineering (2020). 9. 3036-3041. 10.30534/ijatcse/2020/83932020.

7. Hassija, Vikas & Chamola, Vinay & Zeadally, Sherali. (2020). BitFund: A Blockchain- based Crowdfunding

   Platform for Future Smart and Connected Nation. Sustainable Cities and Society. 60. 102145. 10.1016/j.scs.2020.102145.

8. Dhiman, Tushar and gulyani, Vidit and Bhushan, Bharat, Application, Classification and System Requirements of Blockchain Technology (May 14, 2020). Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Available atSSRN: https://ssrn.com/abstract=3600745 or http://dx.DOI.org/10.2139/ssrn.3600745.

9. Lilani, Siddharth & Modi, Jimit & Malani, Dhrumil & Soni, Fenil. (2019). Securing the Software Development Life Cycle (SDLC) with a Blockchain Oriented Development Approach.

10. Sarkar, Abhrajit, Crowd Funding in India: Issues & Challenges Available at Social Science Research Network February 27, 2016.SSRN: https://ssrn.com/abstract=2739008 or http://dx.DOI.org/10.2139/ssrn.2739008

CERTIFICATES OF PUBLICATION

IJSREM
e-Journal

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Asst Prof. Kavyashree J

in recognization to the publication of paper titled

Rise Together: A Crowdfunding Platform

published in IJSREM Journal on Volume 08 Issue 04 April, 2024

www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com

IJSREM
e-Journal

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Bhavana S K

in recognization to the publication of paper titled

Rise Together: A Crowdfunding Platform

published in IJSREM Journal on Volume 08 Issue 04 April, 2024

www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com

IJSREM
e-Journal

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Madhu S

in recognization to the publication of paper titled

Rise Together: A Crowdfunding Platform

published in IJSREM Journal on Volume 08 Issue 04 April, 2024

www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com

IJSREM
e-Journal

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

N Deepak

in recognization to the publication of paper titled

Rise Together: A Crowdfunding Platform

published in IJSREM Journal on Volume 08 Issue 04 April, 2024

www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com

IJSREM
e-Journal

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT
An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Santhosh P

in recognization to the publication of paper titled

Rise Together: A Crowdfunding Platform

published in IJSREM Journal on Volume 08 Issue 04 April, 2024

www.ijsrem.com
Editor-in-Chief
IJSREM Journal
e-mail: editor@ijsrem.com