

Development of a Decentralised Blockchain System for Community Funding



Project Team

Sl. No.	Reg. No.	Student Name
1	18ETCS002104	Sahil Salim
2	18ETCS002121	Subhendu Maji
3	18ETCS002131	Tanishq R Porwar

Supervisors:1. Prof. Hari Krishna S M

January – 2022

**B. Tech. in Computer Science and Engineering
FACULTY OF ENGINEERING AND TECHNOLOGY
M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES
Bengaluru -560 054**

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled “Development of a Decentralised Blockchain System for Community Funding” is a bonafide work carried out in the Department of Computer Science and Engineering by Mr. Sahil Salim bearing Reg. No. 18ETCS002104 in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

August – 2018

**Name of first supervisor
Designation**

**Name of second supervisor
Designation**

**Dr Pushphavathi T P
Professor and Head – Dept. of CSE**

**Dr Govind R. Kadambi
Professor and Dean-FET**

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled “Development of a Decentralised Blockchain System for Community Funding” is a bonafide work carried out in the Department of Computer Science and Engineering by Mr. Subhendu Maji bearing Reg. No. 18ETCS002121 in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

August – 2018

**Name of first supervisor
Designation**

**Name of second supervisor
Designation**

**Dr Pushphavathi T P
Professor and Head – Dept. of CSE**

**Dr Govind R. Kadambi
Professor and Dean-FET**

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled “Development of a Decentralised Blockchain System for Community Funding” is a bonafide work carried out in the Department of Computer Science and Engineering by Mr. Tanishq Porwar bearing Reg. No. 18ETCS002131 in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

August – 2018

Name of first supervisor
Designation

Name of second supervisor
Designation

Dr Pushphavathi T P
Professor and Head – Dept. of CSE

Dr Govind R. Kadambi
Professor and Dean-FET



Declaration

Development of a Decentralised Blockchain System for Community Funding

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of university regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
1	18ETCS002104	Sahil Salim	
2	18ETCS002121	Subhendu Maji	
3	18ETCS002131	Tanishq R Porwar	

Date :



Acknowledgements

First and foremost, we offer our sincere gratitude to our project supervisor Mr. Hari Krishna S M, Professor, Department of Computer Science Engineering, MSRUAS for his valuable suggestions, guidance, and encouragement which he has provided throughout the duration of this project. The experience, which we have gained by working under him, is an invaluable possession.

We express our sincere thanks to Dr Govind R Kadambi, Dean, Faculty of Engineering and Technology, MSRUAS for providing necessary support for this project. We are also very thankful to Dr Pushphavathi T P, Head of Department, Computer Science Engineering, MSRUAS for providing all the help and facilities to carry out the research work.

We are grateful to our seniors and batchmates, who made suggestions, pointed out errors, responded to our questions, and helped in many ways directly and indirectly. Finally, we deeply acknowledge our own teammates for their sheer hard work and drive that drove this project to what it is.

Summary

In today's age of the internet, start-ups are opting to acquire funds through crowdfunding rather than conventional methods such as venture capitalists and bank loans. This shift is mainly due to the notion that conventional methods such as bank loans usually prove to be a lengthy and time-consuming procedure. Crowdfunding not only provides easy access to funds but also helps to market the product and establish a proof of concept instantly. However, the current architecture that most crowdfunding platforms use is heavily centralised, where the platform is only acting as a mediator between the two parties. This gives rise to a plethora of problems such as lack of transparency, security issues and also forces the contributors into trusting the organisation to have their funds transferred to the desired purpose.

A decentralised system helps solve all these issues by eliminating chances of a single point of failure and ensuring complete transparency of all transactions.

We aim to achieve such a decentralised system by making use of smart contracts on the Ethereum Blockchain along with an easy to use and friendly user interface. The aforementioned system has been successfully deployed at

<https://cryptofund-blockchain.vercel.app/>.



Table of Contents

Declaration	1
Acknowledgements	2
Summary	3
Table of Contents.....	4
List of Tables.....	6
Abbreviation and Acronyms	8
1. Introduction.....	1
1.1 Introduction	1
1.2 Conclusion.....	3
2. Background Theory	4
2.1 Background Theory	4
2.1.1 Blockchain	4
2.1.2 Ethereum	5
2.1.3 Gas	5
2.1.4 Smart contract	5
2.1.5 dApp.....	6
2.1.6 MetaMask.....	6
2.1.7 NextJS.....	6
2.1.8 Chakra UI.....	6
2.1.9 Solidity	6
2.1.10 Web3.....	6
2.2 Conclusion	7
3. Aim and Objectives	8
3.1 Title	8
3.2 Aim	8



3.3 Objectives.....	8
3.4 Methods and Methodology/Approach to attain each objective.....	9
3.5 Conclusion	10
4. Problem Solving	11
4.1 Identifying stakeholders	11
4.2 Functional Requirements.....	12
4.3 Non-Functional Requirements.....	12
4.4 Design.....	17
4.4.1 Class diagram	18
4.4.2 Block diagram for the decentralised architecture.....	19
4.4.3 Block diagram for interaction with the smart contract.....	20
4.5 Simulation and implementation	20
4.6 Testing.....	24
4.7 Conclusion	25
5. Results	26
6. Project Costing	29
6.1 Project Cost Estimation.....	29
6.2 Summary	29
7. Conclusions and Suggestions for Future Work.....	30
7.1 Suggestions for future work	30
References.....	32
Appendix	34
Appendix-A: Source code of the Smart Contract.....	34

List of Tables

Table 1: Methods and Methodology	9
Table 2: Specification table for FR 1	13
Table 3: Specification table for FR 2	13
Table 4: Specification table for FR 3	14
Table 5: Specification table for FR 4	14
Table 6: Specification table for FR 5	15
Table 7 Specification table for FR 6	15
Table 8 Specification table for FR 7	16
Table 9 Specification table for FR 8	16
Table 10 Specification table for FR 9	17
Table 11: Test Cases.....	24
Table 12 Project Cost Estimation	29

List of Figures

Figure 1 Class Diagram of smart contract.....	18
Figure 2 Block Diagram of Decentralised Architecture.....	19
Figure 3 Block diagram of interaction of smart contract.....	20
Figure 4 Create a new campaign	21
Figure 5 Campaign details.....	22
Figure 6 MetaMask interaction	22
Figure 7 Create a withdrawal request	23
Figure 8 Withdrawal requests of a campaign	23
Figure 9 Homepage of web-app	26
Figure 10 Example of Metamask interaction.....	27
Figure 11 How cryptofund works- about page	27
Figure 12 Campaign details.....	28
Figure 13 Withdrawal request for a campaign	28



Abbreviation and Acronyms

ETH Ether

DAO Decentralized Autonomous Organization

dApp Decentralized application

1. Introduction

In this chapter, the project is introduced with its theme, followed by its purpose. Then a literature survey is documented stating the most popular existing applications under the domain of the current project and their key features. In the end, the current project and its output will be compared with the existing application, and the key features of the current project will be evaluated.

1.1 Introduction

In simple terms, crowdfunding can be defined as raising funds for a project or a campaign by a group of people instead of using established entities such as banks or venture capitalists. Freedman and Nutting [1] defined crowdfunding as a method of collecting many small contributions, by means of an online funding platform to finance or capitalize a popular enterprise. The crowdfunding action mainly involves three parties, which are the contributors, crowdfunding platform and project managers. The main benefit of crowdfunding is that it can raise the amount of money needed in a short amount of time owing to the fact that the majority of people today are a part of some social network and gain a large exposure to crowdfunding campaigns seamlessly [2]. Nowadays, many start-ups have chosen crowdfunding to raise money for their projects as it is harder to gain loans from banks or other investors [3]. Even if the loan does get sanctioned, it often takes an indefinite amount of time to actually acquire the funds.

Besides being able to acquire funds faster through crowdfunding, the contributors to the project can also provide value-added involvement and feedback to the project while also creating public awareness of the campaign [4]. Moreover, investors also have access to

more information in the initial phase of the project which may boost their eagerness to further invest in such crowdfunding projects. However, despite having several advantages, centralised crowdfunding platforms still have many flaws that bring their entire architecture into question. One of the main issues linked with traditional crowdfunding platforms is that all operations rely on trusting the platform. This trust ranges from trusting that the platform transfers the donated funds to the designated campaign owner and trusting the campaign owner to actually use these funds for the stated purpose. Thus, in a centralised system, there is a lack of transparency due to which potential contributors might hesitate to actually invest in a campaign they come across. Establishing a decentralised platform helps mitigate these issues by providing a truly serverless, trust-less and transparent platform [5].

Our project aims to implement such a decentralised system by making use of a smart contract deployed on the Ethereum Blockchain. By implementing smart contracts in the crowdfunding system, we can create a contract that will hold a contributor's money until more than half of the contributors approve the withdrawal of the funds by the campaign owner. Depending on the outcome, the funds will either be given to the project owner or safely returned to the contributors. In such a system, every single transaction made can be traced via Etherscan based on the unique smart contract address. Smart contracts being immutable and self-executing ensure that the contract cannot be breached once it has been initiated [6].

The remainder of the paper is organised as follows. The background review explains the core concepts involved in forming the decentralised platform. The aims and objectives highlight what our project does by explaining the methodology. Results present the deployed application. The final section summarises the proposed system.

1.2 Conclusion

In this chapter, the existing crowdfunding platforms were compared with the proposed system. Vital differences were brought out between the centralised and decentralised architectures. A clear understanding of the drawbacks of the conventional platforms were understood, and better alternatives were proposed. This will help in getting a clear idea of the innovation associated with this proposed project.

2. Background Theory

In this chapter, all theories related to the proposed project including technical aspects and resources are explained. This section starts with a discussion of the key technologies to be used to build the proposed platform including all the frameworks used. This chapter ends with a summary of all the stated concepts.

2.1 Background Theory

This section gives complete knowledge and understanding of different technologies and frameworks that were required in this project for its completion. The background Theory is listed below:

2.1.1 Blockchain

A blockchain is a distributed database that is shared among the nodes of a computer network. As a database, a blockchain stores information electronically in digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralised record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party [7].

One key difference between a typical database and a blockchain is the way the data is structured. A blockchain collects information together in groups, known as "**blocks**" that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the "blockchain." All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

2.1.2 Ethereum

Ethereum is a blockchain platform with its own cryptocurrency, called Ether (ETH) or Ethereum, and its own programming language, called Solidity.

As a blockchain network, Ethereum is a decentralised public ledger for verifying and recording transactions. The network's users can create, publish, monetise, and use applications on the platform and use its Ether cryptocurrency as payment. Insiders call the decentralised applications on the network "dApps."

Ethereum was created to enable developers to build and publish smart contracts and distributed applications (dApps) that can be used without the risks of downtime, fraud, or interference from a third party [8].

2.1.3 Gas

Gas refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network.

Since each Ethereum transaction requires computational resources to execute, each transaction requires a fee. Gas refers to the fee required to conduct a transaction on Ethereum successfully [9].

2.1.4 Smart contract

Smart contracts are programs that reside within decentralised blockchains and are executed pursuant to triggered instructions. A smart contract acts in a similar way to a traditional agreement but negates the necessity for the involvement of a third party. Smart contracts are capable of initiating their commands automatically, thus eliminating the involvement of a regulatory body [10]. As a consequence of blockchain's immutable feature, smart contracts are developed in a manner that is distinct from traditional software. Once deployed to the blockchain, a smart contract cannot be modified or updated for security patches, thus encouraging developers to implement strong security strategies before deployment in order to avoid potential exploitation at a later time.

2.1.5 dApp

A decentralised application (dApp) is an application built on a decentralised network that combines a smart contract and a frontend user interface [11]. A dApp has its backend code running on a decentralised peer-to-peer network. Contrast this with an app where the backend code is running on centralised servers. A dApp can have frontend code and user interfaces written in any language (just like an app) to make calls to its backend.

2.1.6 MetaMask

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications [12].

2.1.7 NextJS

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

2.1.8 Chakra UI

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

2.1.9 Solidity

It is the programming language for implementing Ethereum based Smart Contracts.

2.1.10 Web3

web3.js is a collection of libraries that allow you to interact with a local or remote Ethereum node using HTTP, IPC or WebSocket.

2.2 Conclusion

In this chapter, all the required technologies and frameworks have been explained in depth along with their use in achieving our final product. Thus, an insight into all the key concepts has been attained. All the appropriate tools and frameworks can now be selected accordingly to facilitate the functioning of the platform.

3. Aim and Objectives

This chapter focuses on defining the title and aim of the project concisely. Later this chapter also includes the required objectives that need to be fulfilled in order to complete this project. This is followed by methods and methodologies that tabulates the procedure that will be followed in order to complete the objectives. This section then ends with a summary.

3.1 Title

- ❖ Development of a decentralised blockchain system for community funding

3.2 Aim

- ❖ To create a truly serverless, trust less, and transparent system to facilitate community funding.

3.3 Objectives

- ❖ To conduct a literature survey on the existing community funding platforms.
- ❖ To acquire functional and non-functional requirements based on the literature survey.
- ❖ To gain insight on Ethereum framework, solidity programming and smart contracts.
- ❖ To develop a smart contract based on the identified functional requirements.
- ❖ To develop and deploy a web3 application to facilitate user interaction.
- ❖ To deploy a smart contract on Ethereum Blockchain.
- ❖ To document the report consolidating all relevant results.

3.4 Methods and Methodology/Approach to attain each objective

Table 1: Methods and Methodology

Objective No.	Statement of the Objective	Method/ Methodology	Resources Utilised
1	To conduct a literature survey on the existing community funding platforms	To review the literature on existing community funding platforms, their working, functionality and software used from peer-reviewed journals, reference books and other authentic sources.	Reputed journals and books on decentralised systems.
2	To acquire functional and non-functional requirements based on the literature survey	On conducting literature survey, the functional and non-functional requirements are obtained. Based on the identified requirements, a high-level design will be created in UML	Reputed journals and books on decentralised systems.
3	To gain insight on Ethereum framework, solidity programming and smart contracts	Understand Ethereum blockchain ecosystem and learn Solidity programming language to create smart contracts. Get familiar with web3.js framework to establish a communication between smart contract and the React frontend.	Ethereum official documentation.
4	To develop smart contract based on the identified functional requirements	Identify the different terms of contracts for the FR's. To create the algorithm/pseudo code.	Solidity programming language. Ethereum official documentation.

		To implement smart contract using Solidity programming language.	
5	To develop and deploy a web3 application for facilitate user interaction	Design a User-friendly frontend. Develop a ReactJS application. Make use of web3.js library to communicate with smart contract	Next.JS, ChakraUI, node, web3.js
6	To deploy smart contract on Ethereum Blockchain	Deploy the smart contract to Ethereum Rinkeby Test Network for the frontend to interact.	Infura and Rinkeby testnet

3.5 Conclusion

This particular chapter focused on defining the title and aim of the project concisely. Later this chapter included required objectives that were needed to be fulfilled to complete this project. The section is then followed by methods and methodologies, which are successfully tabulated in order to know the steps used in completing the objectives, including the resources used.

4. Problem Solving

In this section, the actual dissection of the project is done, and each module is built piece by piece in order to complete the project. In the design section, the application has been built in accordance with functional requirements based on which diagrams like Class Diagram and High-Level block diagrams are drawn. In the implementation and simulation section, the important features of the platform are described with snippets. In the testing section, all functional requirements are tested, and the result is analysed, resulting in the status of the test condition.

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

4.1 Identifying stakeholders

The stakeholders can be divided into two parts:

- ❖ Campaign Creators: These are the users who have created a Campaign.
- ❖ Contributors & Approvers: Contributors are the users who contribute and fund the campaigns. Approvers are Contributors who have contributed more than the Minimum Contribution, and they can approve the withdrawal requests.

Any web-based application is a centralized application which means that anything we do on the platform is managed by a server which is owned by a single company.

We propose a Decentralized Application powered by Ethereum Blockchain, where all the information about campaigns, contributions, withdrawal requests and funds are kept on a Blockchain Network, visible to all and decentralized. This means the funds and

transactions are visible to and stored at every node on the blockchain, and prevents the data from being stored in a centralized server, single location.

Hence not letting the money get into the hands of anyone and eliminating every possibility of it getting misused — an elegant and logical solution to the problem in hand.

4.2 Functional Requirements

FR1: The system should allow the user to connect to the cryptocurrency wallet.

FR2: The system should allow the user to create a campaign.

FR3: The system should allow the user to browse all existing campaigns.

FR4: The system should allow the user to view campaign details.

FR5: The system should allow the user to contribute to a campaign.

FR6: The system should allow only the campaign owner to place a withdrawal request to acquire the collected funds.

FR7: The system should allow verified contributors to approve a withdrawal request.

FR8: The system should transfer the funds to the campaign owner's wallet on successful withdrawal request.

FR9: The system should allow contributors to claim a refund after the deadline has lapsed.

4.3 Non-Functional Requirements

NFR1: The system should be secure.

NFR2: The system should be easily maintainable.

NFR3: The system should be reusable.

NFR4: The system should be scalable.

NFR5: The system should be portable.

Table 2: Specification table for FR 1

Item	Detail
Requirement tag	FR 1
Requirement Statement	The system should allow the user to connect to the cryptocurrency wallet
Dependent on Requirements	none
Stakeholder Owning the Requirement	Campaign owner and contributors
Example of user/system interaction for this requirement	The users of the platform connect to their wallet through Metamask

Table 3: Specification table for FR 2

Item	Detail
Requirement tag	FR 2
Requirement Statement	The system should allow the user to create a campaign
Dependent on Requirements	FR 1
Stakeholder Owning the Requirement	Campaign owner
Example of user/system interaction for this requirement	Campaign owners can post a campaign on the platform stating their purpose along with the goal amount to be reached.

Table 4: Specification table for FR 3

Item	Detail
Requirement tag	FR 3
Requirement Statement	The system should allow the user to browse all existing campaigns.
Dependent on Requirements	FR 2
Stakeholder Owning the Requirement	Campaign contributors
Example of user/system interaction for this requirement	Users on the platform can browse through a list of all the available campaigns.

Table 5: Specification table for FR 4

Item	Detail
Requirement tag	FR 4
Requirement Statement	The system should allow the user to view campaign details.
Dependent on Requirements	FR 2, FR 3
Stakeholder Owning the Requirement	Campaign contributors
Example of user/system interaction for this requirement	Users on the platform can view all the details of a selected campaign that has been mentioned by the campaign owner at the time of creation.

Table 6: Specification table for FR 5

Item	Detail
Requirement tag	FR 5
Requirement Statement	The system should allow the user to contribute to a campaign.
Dependent on Requirements	FR 1, FR 3, FR 4
Stakeholder Owning the Requirement	Campaign contributors
Example of user/system interaction for this requirement	Users of the platform can contribute any amount they wish to the campaign of their choice.

Table 7 Specification table for FR 6

Item	Detail
Requirement tag	FR 6
Requirement Statement	The system should allow only the campaign owner to place a withdrawal request to acquire the collected funds.
Dependent on Requirements	FR 1
Stakeholder Owning the Requirement	Campaign owner

Example of user/system interaction for this requirement	Campaign owners may place a withdrawal request in order to acquire the donated funds in their wallet.
---	---

Table 8 Specification table for FR 7

Item	Detail
Requirement tag	FR 7
Requirement Statement	The system should allow verified contributors to approve a withdrawal request.
Dependent on Requirements	FR 1, FR 6
Stakeholder Owning the Requirement	Campaign contributors
Example of user/system interaction for this requirement	Contributors of a campaign may choose to approve or reject a withdrawal request placed by the campaign owner.

Table 9 Specification table for FR 8

Item	Detail
Requirement tag	FR 8
Requirement Statement	The system should transfer the funds to the campaign owner's wallet on successful withdrawal request.
Dependent on	FR1, FR7

Requirements	
Stake Holder Owning the Requirement	Campaign owner
Example of user/system interaction for this requirement	The requested amount is transferred to the campaign owner's wallet on successful withdrawal.

Table 10 Specification table for FR 9

Item	Detail
Requirement tag	FR 9
Requirement Statement	The system should allow contributors to claim a refund after the deadline has lapsed.
Dependent on Requirements	FR1, FR7
Stake Holder Owning the Requirement	Campaign owner
Example of user/system interaction for this requirement	The contributed amount is sent back to the respective contributor based on a ratio which is determined by their overall contribution to the campaign.

4.4 Design

Design is necessary when it comes to development since it acts as a blueprint for the entire process from requirement making to finished (final product). Hence, in this section designs specific to this project like class-diagram, block-diagram, etc. are attached.

4.4.1 Class diagram

The Classes defined are:

- Campaign
- CampaignFactory
- Requests
- connectWallet

The Relationships defined are:

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

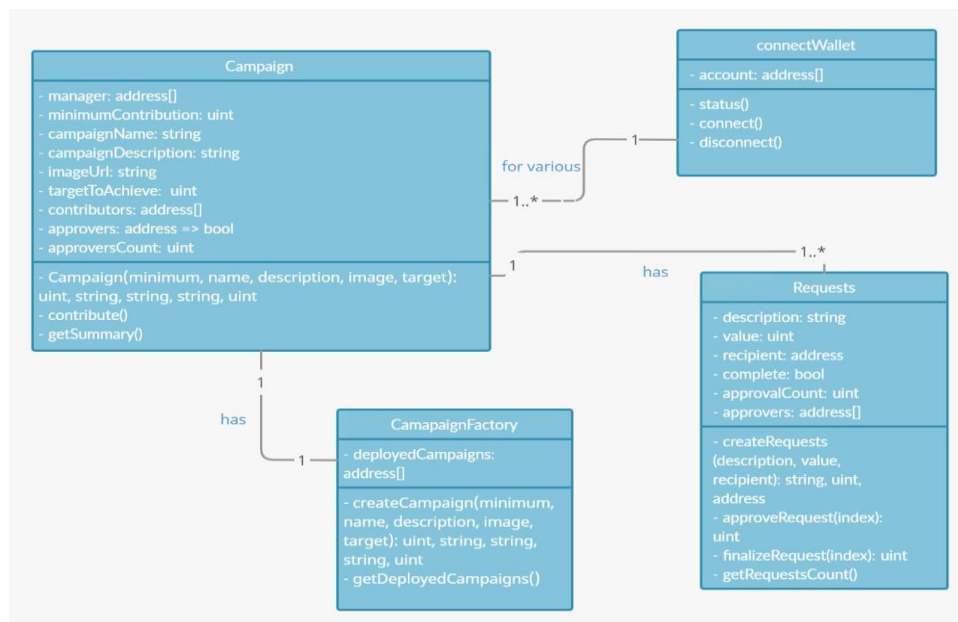


Figure 1 Class Diagram of smart contract

4.4.2 Block diagram for the decentralised architecture

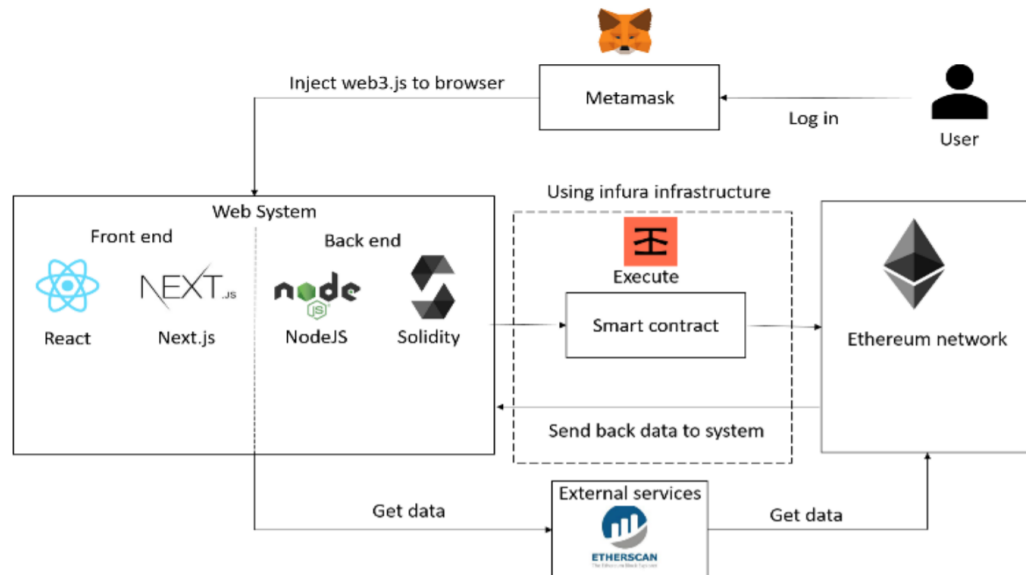


Figure 2 Block Diagram of Decentralised Architecture

Initially, the user connects to the platform through the front end which is built on React. To login, the user connects their wallet through Metamask. Each account has a unique public address which will be used by the dApp to identify them and facilitate the transfer of funds accordingly.

To start or contribute to a campaign the frontend interacts with the smart contract by injecting the web3.js framework which is deployed on the Infura infrastructure. On meeting the terms of the smart contract, it self-executes and updates all the nodes accordingly on the Ethereum Rinkeby test network. All the transactions can be viewed on Etherscan enforcing complete transparency.

4.4.3 Block diagram for interaction with the smart contract

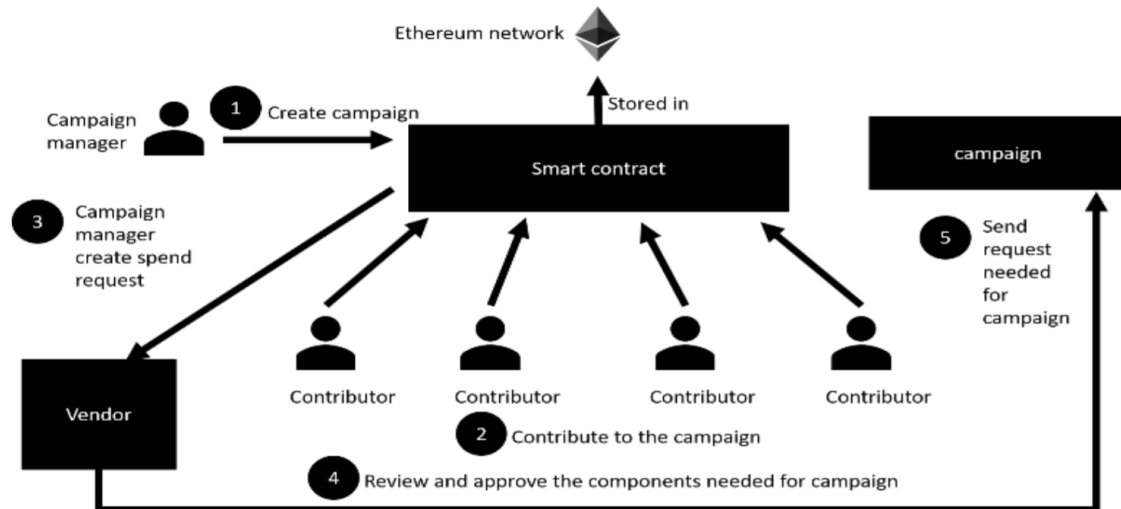


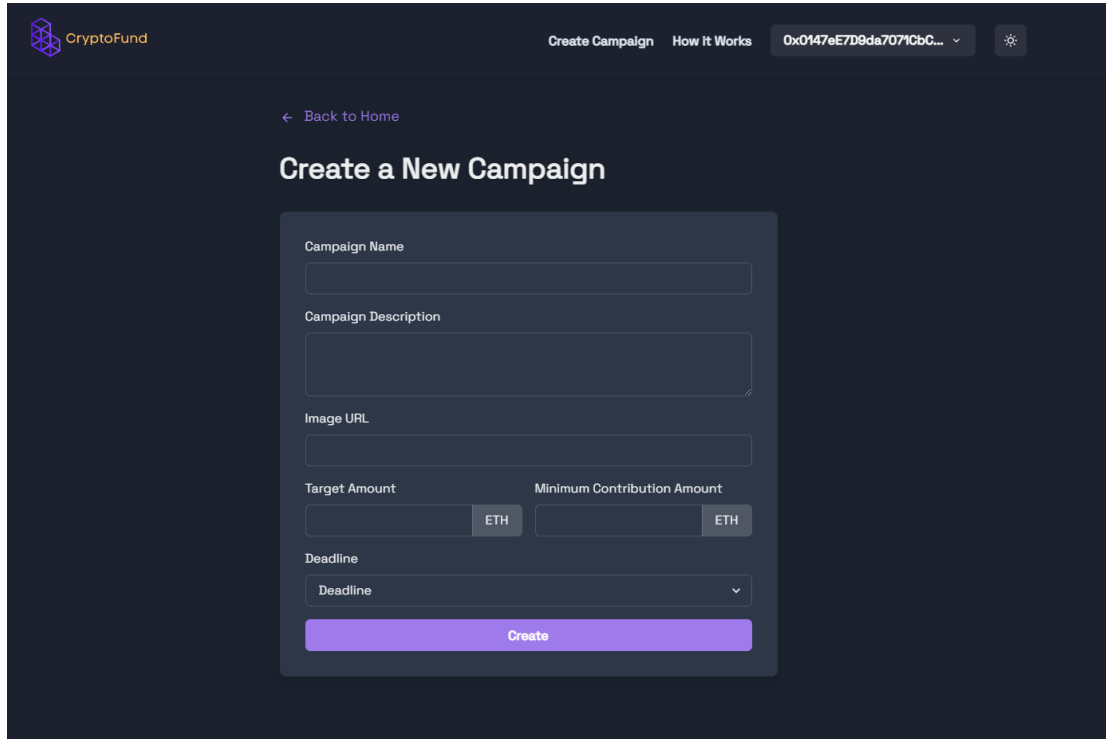
Figure 3 Block diagram of interaction of smart contract

- 1) The campaign owner first creates a campaign by interacting with the deployed smart contract. All created campaigns are stored on the smart contract.
- 2) Interested contributors may choose to donate funds to a particular campaign of their choice. At this point, all the donated funds reside solely on the smart contract.
- 3) Once a desired amount of funds has been collected, the campaign owner must place a withdrawal request stating the amount to be withdrawn.
- 4) Contributors of the campaign have the ability to view all requests made by the campaign owner and can choose to either accept or reject the withdrawal request. This establishes some sense of ownership between the contributors and the project.
- 5) Once more than 50% of the contributors have approved the withdrawal request, the requested funds

4.5 Simulation and implementation

- ❖ **Creating a Campaign:** Just like Crowdfunding in the real world as well as on other crowdfunding platforms, anyone can create a campaign in a few minutes. The

campaign information will be managed by the Ethereum-based smart contract and thus cannot be tampered with.



The screenshot shows the 'Create a New Campaign' interface on the CryptoFund website. The form is centered on a dark background. It contains the following fields and elements:

- Campaign Name:** A text input field.
- Campaign Description:** A larger text area for description.
- Image URL:** A text input field for the campaign image.
- Target Amount:** A text input field with an 'ETH' button next to it.
- Minimum Contribution Amount:** A text input field with an 'ETH' button next to it.
- Deadline:** A dropdown menu with 'Deadline' selected.
- Create:** A prominent purple button at the bottom of the form.

Figure 4 Create a new campaign

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

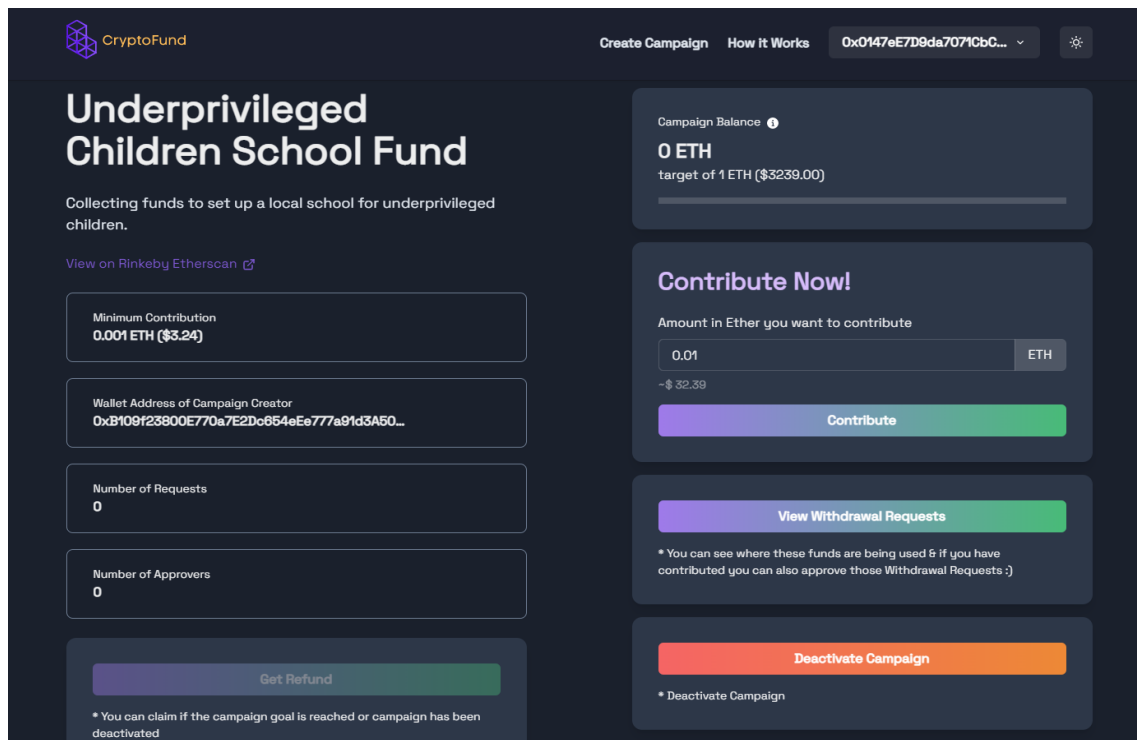


Figure 5 Campaign details

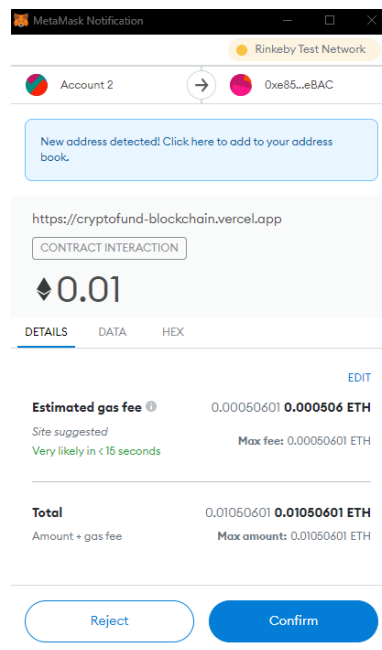
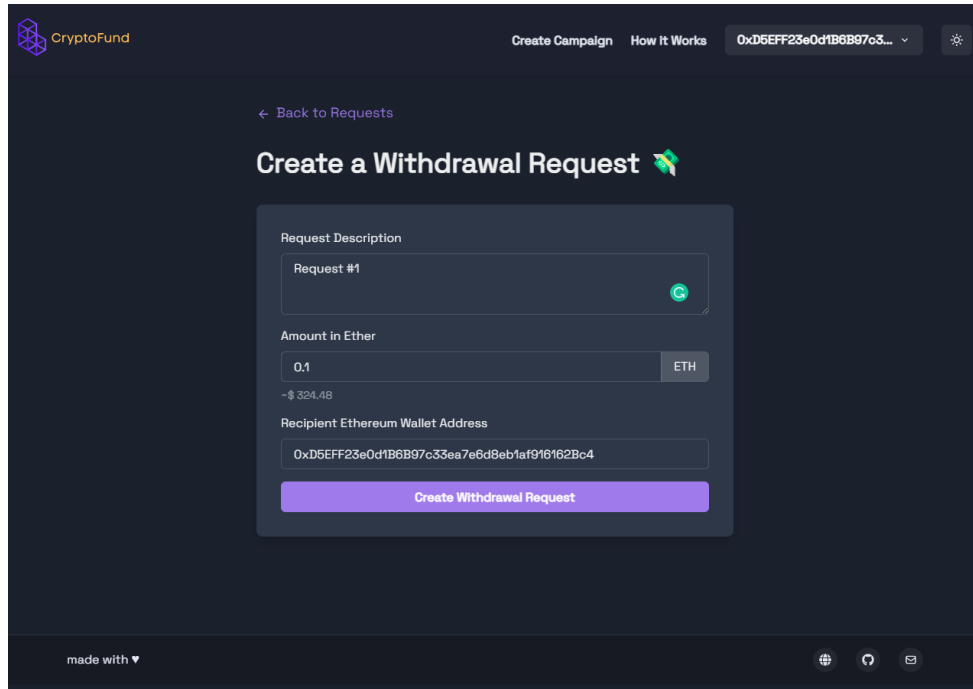


Figure 6 MetaMask interaction

- ❖ **Withdrawal of Funds:** The Creator of a Campaign can propose how to use the funds in the form of a Withdrawal Request. Anybody who contributes more than a particular amount is called an approver, and will be able to approve or deny the request. Funds can't be withdrawn without the approval of 50% approvers.

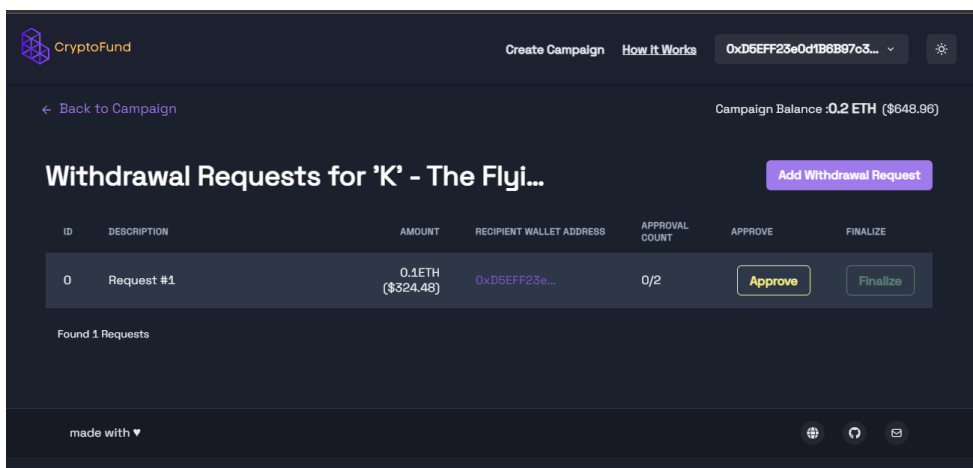


The screenshot shows the 'Create a Withdrawal Request' form in the CryptoFund interface. The form is titled 'Create a Withdrawal Request' with a green flag icon. It contains the following fields:

- Request Description:** A text input field with the value 'Request #1'.
- Amount in Ether:** A text input field with the value '0.1' and a dropdown menu set to 'ETH'. Below the input, it shows '~\$ 324.48'.
- Recipient Ethereum Wallet Address:** A text input field with the value '0xD5EFF23e0d1B6B97c33ea7e6d8eb1af916162Bc4'.

At the bottom of the form is a purple button labeled 'Create Withdrawal Request'. The interface also includes a 'Back to Requests' link, a 'Create Campaign' button, a 'How It Works' link, and a dropdown menu showing the campaign ID '0xD5EFF23e0d1B6B97c3...'.

Figure 7 Create a withdrawal request



The screenshot shows the 'Withdrawal Requests for 'K' - The Flyi...' page in the CryptoFund interface. The page displays a table of withdrawal requests and includes the following elements:

- Back to Campaign:** A link to return to the campaign page.
- Campaign Balance:** 0.2 ETH (\$648.96).
- Add Withdrawal Request:** A purple button to create a new request.
- Table of Requests:**

ID	DESCRIPTION	AMOUNT	RECIPIENT WALLET ADDRESS	APPROVAL COUNT	APPROVE	FINALIZE
0	Request #1	0.1ETH (\$324.48)	0xD5EFF23e...	0/2	<button>Approve</button>	<button>Finalize</button>
- Found 1 Requests:** A summary of the number of requests found.

The interface also includes a 'made with' logo and social media icons at the bottom.

Figure 8 Withdrawal requests of a campaign

4.6 Testing

Table 11: Test Cases

SI No.	FR No.	Expected	Obtained	Result
1	FR 1	The system should allow the user to connect to the cryptocurrency wallet	The user was successfully able to connect to the cryptocurrency wallet	PASS
2	FR 2	The system should allow the user to create a campaign	The user was successfully able to create a campaign	PASS
3	FR 3	The system should allow the user to browse all existing campaigns	The user was successfully able to browse all existing campaigns	PASS
4	FR 4	The system should allow the user to view campaign details	The user was successfully able to view campaign details	PASS
5	FR 5	The system should allow the user to contribute to a campaign	The user was successfully able to contribute to a campaign	PASS

6	FR 6	The system should allow only the campaign owner to place a withdrawal request to acquire the collected funds.	The campaign owner was successfully able to place a withdrawal request to acquire the collected funds	PASS
7	FR 7	The system should allow verified contributors to approve a withdrawal request	The user was successfully able to approve withdrawal requests	PASS
8	FR 8	The system should transfer the funds to the campaign owner's wallet on successful withdrawal request	The smart contract was successful in transferring the appropriate funds to the corresponding campaign owner	PASS

4.7 Conclusion

This chapter focused on identifying the stakeholders and defining all the required functional and non-functional requirements. Useful UML diagrams were designed based on these requirements, which helped to further formulate the system. Finally, the system was tested against all the requirements and tabulated accordingly.

5. Results

The chapter presents the results of the proposed system. This contains the total outcome of our project. The objectives defined in the previous chapter are fulfilled by defining the objectives and then showing the output respectively.

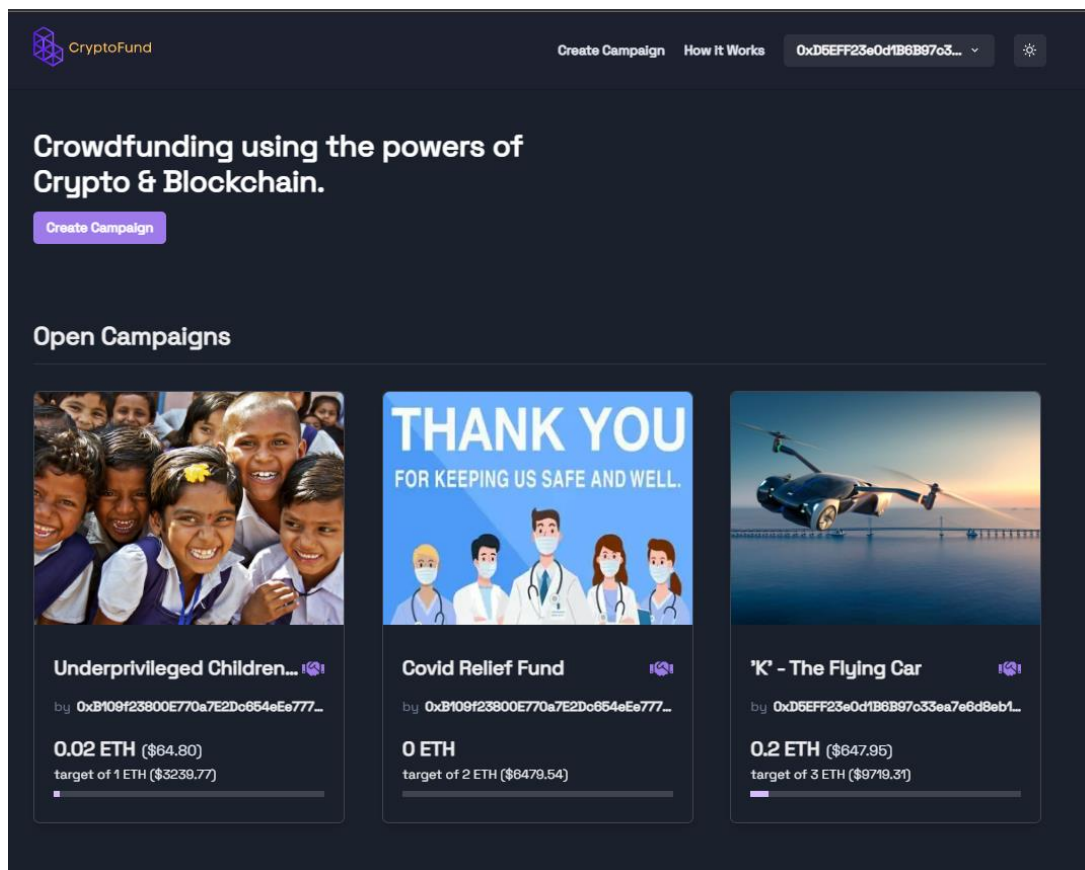


Figure 9 Homepage of web-app

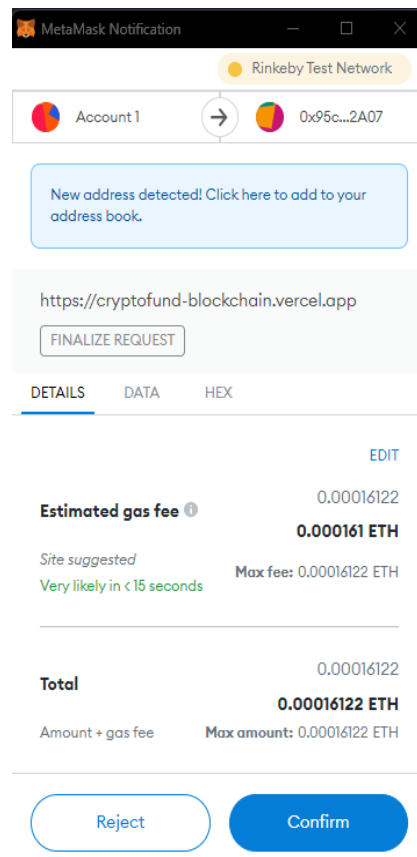


Figure 10 Example of Metamask interaction

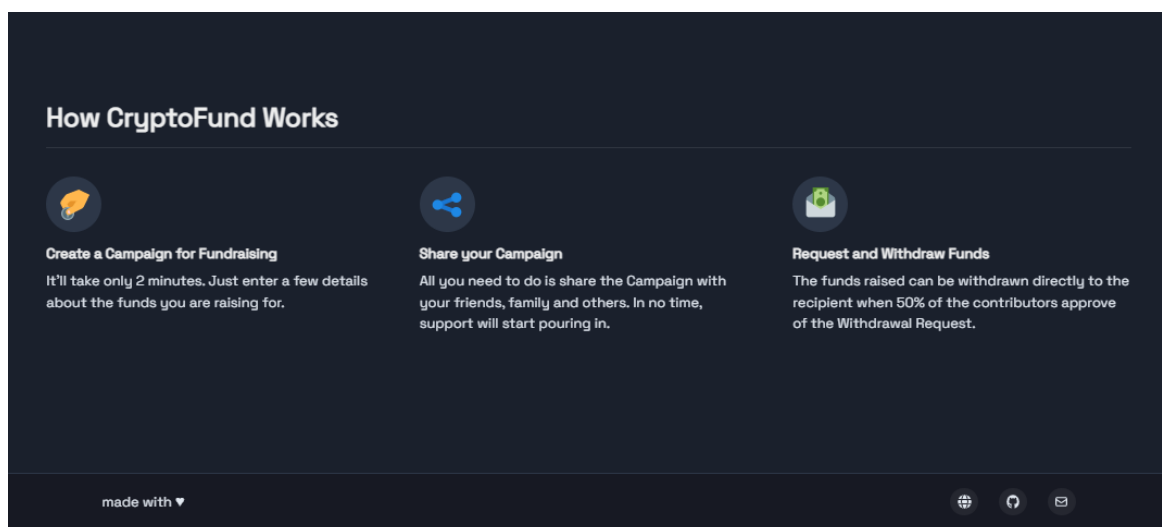


Figure 11 How cryptofund works- about page

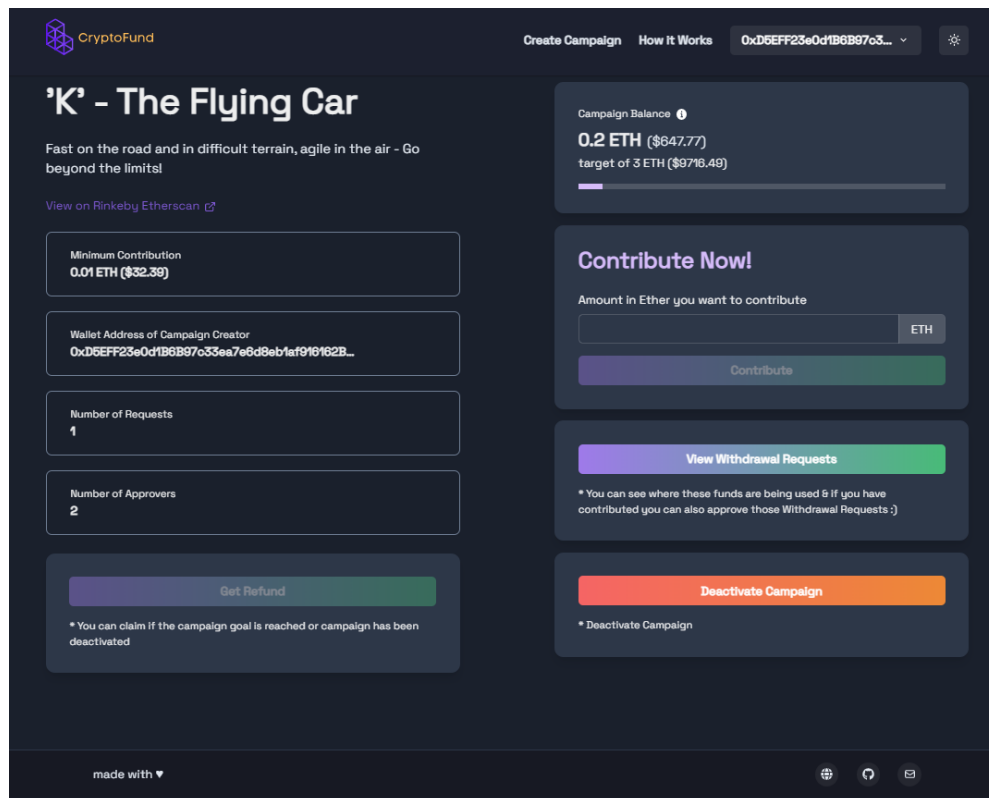


Figure 12 Campaign details

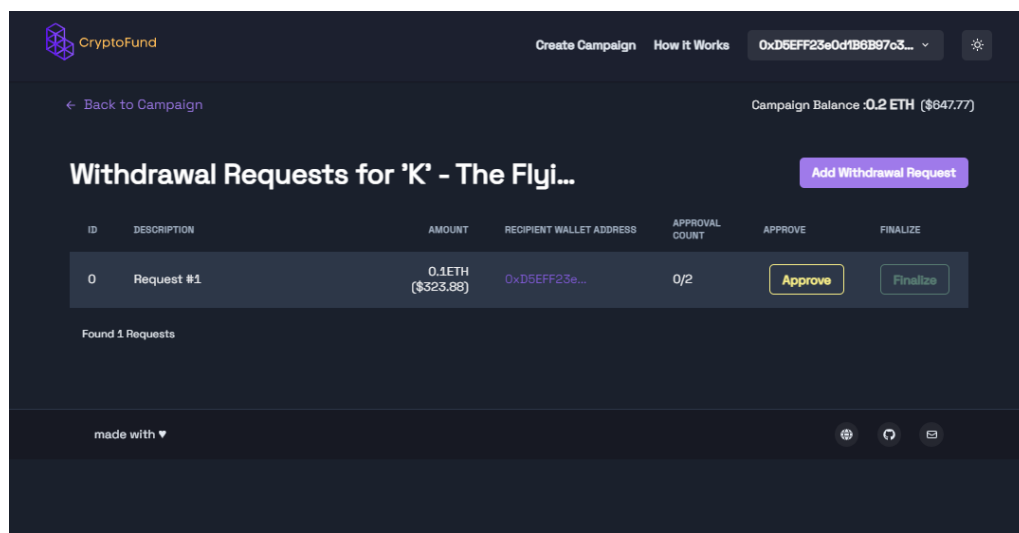


Figure 13 Withdrawal request for a campaign

6. Project Costing

This chapter deals with the costing of this project which gives an overall estimation of expenses that was required to complete this project. This Covers expenses of testing devices, Platform and Hardware cost, Human Resource Cost and a grand total of the entire cost.

6.1 Project Cost Estimation

The cost of project is summarised in a tabular form displayed below:

Table 12 Project Cost Estimation

Serial Number	Resources and Work	Cost (Rs)
1	Laptop	3 * 50,000 = ₹ 1,50,000
2	Human Resources	3 * 20,000 = ₹ 60,000
	TOTAL	₹ 2,10,000

6.2 Summary

Since this project didn't have any physical model, hence no expenses were made for the same. However, effort on making the software via parallel learning of new technology raised the Human Resource cost which can be seen in Table 12.

7. Conclusions and Suggestions for Future Work

In this project, we first explored the reasons for using a community funding platform rather than conventional methods such as banks and venture capitalists. The merits and demerits of current crowdfunding platforms was then drawn out wherein it was established that a decentralised platform would be best suited to mitigate these issues. Based on the proposed architecture, appropriate frameworks and technologies were chosen and explained in depth. The desired platform was then built keeping in mind the formulated functional and non-functional requirements. Extensive testing was then performed on this system based on all the listed objectives.

Although the proposed system overcomes most of the problems associated with the centralised approach, it lacks in certain aspects such as high gas fees and long computation times. However, the Ethereum Blockchain is ever-expanding and soon aims to significantly reduce the gas prices with the release of Ethereum 2.0 where a transition is made from using proof-of-work to proof-of-stake as its consensus algorithm.

The proposed system could also be built on the Solana Blockchain which would result in significantly reduced gas prices and exceedingly fast computation. However, building decentralised applications on the Solana Blockchain is an increasingly difficult task as compared to the Ethereum Blockchain.

7.1 Suggestions for future work

The proposed system could be used to set up a Decentralised Autonomous Organisation (DAO) which is an organization represented by rules encoded as a computer program that is transparent, controlled by the organization members and not influenced by a central

government. Here, each contributor could have a proportional vote on the decisions made in the organisation based on the amount contributed by them.

Other crowdfunding mechanisms could also be integrated into the platform such as Initial Coin Offerings (ICO). An initial coin offering (ICO) is the cryptocurrency industry's equivalent to an initial public offering (IPO). A company looking to raise money to create a new coin, app, or service may launch an ICO as a way to raise funds.

Interested investors can buy into the offering and receive a new cryptocurrency token issued by the company. This token may have some utility in using the product or service the company is offering, or it may just represent a stake in the company or project.

References

- [1] Freedman et al. "The Foundations of Online Crowdfunding", Equity Crowdfunding for Investors (eds D. M. Freedman and M. R. Nutting), 2015, doi:10.1002/9781118864876.ch1.
- [2] T. Dannberg, "Advantages and Disadvantages with Crowdfunding : - and Who are the Users?", Dissertation, 2017.
- [3] Schwienbacher et al, "Crowdfunding of Small Entrepreneurial Ventures", Handbook Of Entrepreneurial Finance, Oxford University Press.
<http://dx.doi.org/10.2139/ssrn.1699183>.
- [4] Macht et al, " The Benefits of Online Crowdfunding for Fund-Seeking Business Ventures", Strategic Change, 2014, 23 (1-2). pp. 1-14. ISSN 10861718.
- [5] Schlueter et al, "Underlying Benefits and Drawbacks of Crowdfunding from the Perspective of Entrepreneurs in Germany", 5th IBA Bachelor Thesis Conference, University of Twente. Available at:
http://essay.utwente.nl/67409/1/Schlueter_BA_MB.pdf [Accessed 15 Aug 2018].
- [6] Z. Zheng et al, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends", IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, 2017, pp. 557-564.doi: 10.1109/BigDataCongress.2017.85.
- [7] Miraz et al., "Applications of Blockchain Technology beyond Cryptocurrency", Annals of Emerging Technologies in Computing (AETiC), 2018. 2. 1-6.
- [8] Chen et al., "Exploring Blockchain Technology and its Potential Applications for Education. Smart Learning Environments", 5. 10.1186/s40561-017-0050-x.
- [9] A. Angrish et al., "A Case Study for Blockchain in Manufacturing: "FabRec": A Prototype for Peer-to-Peer Network of Manufacturing Nodes", Procedia Manufacturing, vol. 26, pp. 1180-1192, 2018.

- [10] Clack, C.D., Bakshi, V.A. and Braine, L., 2016. Smart contract templates: foundations, design landscape and research directions. arXiv preprint arXiv:1608.00771.
- [11] Taş, R. and Tanrıöver, Ö.Ö., 2019, October. Building a decentralized application on the Ethereum blockchain. In 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) (pp. 1-4). IEEE.
- [12] Lee, W.M., 2019. Using the metamask chrome extension. In Beginning Ethereum Smart Contracts Programming (pp. 93-126). Apress, Berkeley, CA.

Appendix

Appendix-A: Source code of the Smart Contract

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.5.0 <0.9.0;

contract CampaignFactory {
    Campaign[] public deployedCampaigns;

    function createCampaign(
        uint256 minimum,
        string calldata name,
        string calldata description,
        string calldata image,
        uint256 target,
        string calldata deadline
    ) public {
        Campaign newCampaign = new Campaign(
            minimum,
            msg.sender,
            name,
            description,
            image,
            target,
            deadline
        );
        deployedCampaigns.push(newCampaign);
    }

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

contract Campaign {
    struct Request {
        string description;
        uint256 value;
        address payable recipient;
        bool complete;
        uint256 approvalCount;
    }
}
```

```

        mapping(address => bool) approvals;
    }

    address public manager;
    uint256 public minimumContribution;
    string public CampaignName;
    string public CampaignDescription;
    string public campaignDeadline;
    string public imageUrl;
    uint256 public targetToAchieve;
    uint256 private isCampaignActive = 1;

    address[] public contributors;
    mapping(address => uint256) public contributorsMap;
    mapping(address => bool) public approvers;
    uint256 public approversCount;

    modifier onlyManager() {
        require(
            msg.sender == manager,
            "Only the manager can call this function"
        );
        _;
    }

    constructor(
        uint256 minimum,
        address creator,
        string memory name,
        string memory description,
        string memory image,
        uint256 target,
        string memory deadline
    ) {
        manager = creator;
        minimumContribution = minimum;
        CampaignName = name;
        CampaignDescription = description;
        imageUrl = image;
        targetToAchieve = target;
        campaignDeadline = deadline;
    }

    uint256 numRequests;
    mapping(uint256 => Request) public requests;

    function contribute() public payable {

```

```
// check if the sender is already a contributor
for (uint256 i = 0; i < contributors.length; i++) {
    if (msg.sender == contributors[i]) {
        contributorsMap[msg.sender] =
            contributorsMap[msg.sender] +
            msg.value;

        return;
    }
}

contributors.push(msg.sender);
contributorsMap[msg.sender] = msg.value;

if (msg.value >= minimumContribution) {
    if (approvers[msg.sender] != true) {
        approvers[msg.sender] = true;
        approversCount++;
    }
}
}

function createRequest(
    string memory description,
    uint256 value,
    address payable recipient
) public onlyManager {
    Request storage r = requests[numRequests++];
    r.description = description;
    r.value = value;
    r.recipient = recipient;
    r.complete = false;
    r.approvalCount = 0;
}

function approveRequest(uint256 index) public {
    Request storage request = requests[index];
    require(
        approvers[msg.sender],
        "You are not an approver! Only contributors can approve a
specific payment request"
    );
    require(
        !request.approvals[msg.sender],
        "You have already approved this request"
    );
}
```



```

    );

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

function finalizeRequest(uint256 index) public onlyManager {
    Request storage request = requests[index];
    require(
        request.approvalCount >= (approversCount / 2),
        "Not enough approvals"
    );
    require(!(request.complete), "Request already finalized");

    (bool success, ) = request.recipient.call{value:
request.value}("");
    if (success) {
        request.complete = true;
    }
}

function getSummary()
public
view
returns (
    uint256,
    uint256,
    uint256,
    uint256,
    address,
    string memory,
    string memory,
    string memory,
    uint256,
    string memory,
    uint256
)
{
    return (
        minimumContribution,
        address(this).balance,
        numRequests,
        approversCount,
        manager,
        CampaignName,
        CampaignDescription,
        imageUrl,
        targetToAchieve,

```

```

        campaignDeadline,
        isCampaignActive
    );
}

function getRequestsCount() public view returns (uint256) {
    return numRequests;
}

function claimFunds() public payable onlyManager returns (bool) {
    require(
        address(this).balance >= targetToAchieve,
        "target not achieved"
    ); // funding goal met

    // transfer all funds to the campaign manager
    (bool success, ) = payable(manager).call{value:
address(this).balance}(
    ""
    );
    if (success) {
        isCampaignActive = 0;
    }
    return success;
}

function getCampaignIsActive() public view returns (uint256) {
    return isCampaignActive;
}

function setCampaignIsActive() public returns (uint256) {
    isCampaignActive = 0;
    return isCampaignActive;
}

function getMyAmount(address sender) public view returns (uint256)
{
    return contributorsMap[sender];
}

function getRefundAmount() public payable returns (bool) {
    require(contributorsMap[msg.sender] > 0);
    require(isCampaignActive == 0);

    uint256 senderContribution = contributorsMap[msg.sender];
    uint256 totalContributions = 0;

```

```
        for (uint256 i = 0; i < contributors.length; i++) {
            totalContributions += contributorsMap[contributors[i]];
        }
        uint256 senderRatio = (senderContribution * 100) /
totalContributions;
        uint256 refundAmount = (address(this).balance * senderRatio) /
100;

        (bool success, ) = msg.sender.call{value: refundAmount}("");
        require(success, "Transfer failed.");

        if (success) {
            contributorsMap[msg.sender] = 0;
        }
        if (address(this).balance == 0) {
            isCampaignActive = 0;
        }

        return success;
    }
}
```