

Astana IT University

**KALIYEV MADIYAR
KOZHAKHMETOV NARBEEK**

Development of a mobile blockchain application

6B06101 — Computer Science

Diploma project

Supervisor:
Kuatbayeva A.A.
PH.D. in Computer Science
assistant-prof., Department of
Computing and Data Science

Kazakhstan Republic
Astana, 2024

CONTENTS

Definitions	4
Designations and abbreviations	6
Introduction	7
1 Theoretical Part	10
1.1 Introduction to Crowdfunding	10
1.1.1 Definition and Concept of Crowdfunding	10
1.1.2 Types of Crowdfunding	10
1.1.3 Issues with Existing Crowdfunding Platform	12
1.1.4 Benefits of Decentralized Crowdfunding Platform	13
1.2 Overview of Blockchain Technology	14
1.2.1 Understanding Blockchain Basics	14
1.2.2 Impact on Financial Services	17
1.2.3 Blockchain Core Concepts	18
1.2.4 Blockchain Security	25
1.3 Strategies for Effective Crowdfunding: Risk Management, Success Maximization, and Case Studies	27
1.3.1 Addressing Risks	27
1.3.2 Success Factors	28
1.3.3 Crowdfunding Platforms Case-Study	29
2 Practical Part	31
2.1 Introduction of the Practical Part	31
2.2 Blockchain Structure and Operation Overview	32
2.3 Architecture of application	33
2.3.1 System Architecture	33
2.4 Diagrams and Graphs	34
2.4.1 Entity Relationship Diagram	34
2.4.2 Component Diagram	36
2.4.3 User Data Collection Method	39
2.4.4 Use-Case Diagram	40
2.4.5 Class Diagram	43
2.5 Development Environment Setup	45
2.5.1 Description of the Tools and Technologies Used	46
2.5.2 Steps for Setting Up the Development Environment	46
2.5.3 Frontend Development with Expo and React Native	48
2.5.4 Smart Contract Development with Solidity	50
2.5.5 Integration with Wallets Using WalletConnect and Wagmi	53
2.5.6 Security Measures	54
2.5.7 Results and Achievements	57
3 Conclusion	59

Bibliography	61
A Smart Contract	64
1.0.1 Smart Contract representation	64
1.0.2 Blockchain Transaction Object Structure	66
1.0.3 Smart Contract for Storing and Retrieving a Value	66
B Development code samples	67
2.0.1 Menu screen	67
2.0.2 React Native redirect	67
2.0.3 Project details	67
2.0.4 Modal.tsx	68
2.0.5 List of projects	69
2.0.6 Main menu screen	70
2.0.7 Abis of smart contract	72
C Mobile Application User Interface	78
3.0.1 Screenshots	78

DEFINITIONS

The following terminologies are used in this work:

Blockchain	A digital protocol that functions as a decentralized and immutable ledger. It maintains a chronological record of transactions, ensuring security, transparency, and verifiability across multiple nodes in the network.
Smart Contract	A computer protocol designed to digitally facilitate, validate, or enforce the negotiation or execution of a contract. It allows for trustworthy transactions without the need for intermediaries or third parties, with these transactions being traceable and irreversible.
CF	Crowdfunding emerged as a method for individuals to contribute limited amounts of money to various campaigns via online platforms.
Ethereum	Ethereum is a decentralized blockchain platform that executes smart contracts. This platform allows decentralized applications (dApps) to operate without downtime, censorship, fraud, or third-party interference, ensuring they function exactly as programmed.
MM	MetaMask serves as a cryptocurrency wallet and gateway to decentralized apps, allowing users to connect to blockchain networks, implement contracts, and manage cryptocurrencies not just through browsers but also mobile devices.
Solidity	Solidity is a programming language with static typing and object-oriented features, specifically crafted for creating smart contracts and applicable across different blockchain platforms such as Ethereum.
Gas	Gas is the unit used to determine the total amount of ether expended for executing an operation on the Ethereum network, with the sender setting the gas price.
RN	React Native is an open-source mobile application development framework that allows developers to build native mobile apps using JavaScript and React.js.
Expo	Expo is a platform for creating mobile applications using React Native that provides a simplified development

experience, including a controlled workflow, built-in APIs, and over-the-air update capabilities.

WalletConnect A protocol that enables the connection of various cryptocurrency wallets, such as MetaMask, Trust Wallet, Safe, and others.

Wagmi A decentralized exchange prioritizing liquidity, offering user-friendly navigation and advanced strategies for stable pricing and growth.

Mobile
Application A software program designed to run on smartphones, tablets, or other mobile devices, providing specific functionalities or services to users.

DESIGNATIONS AND ABBREVIATIONS

Following designations and abbreviations are used in this work:

CFP	Crowdfunding Platform
MM	Metamask
RN	React Native

INTRODUCTION

Work relevance. The platform where entrepreneurs and others who are in financial need can have access to funding by receiving contributions from a comparatively large number of individuals is called crowdfunding. Whether aspiring business owners are raising funds for their startups from investors or a person who needs expensive surgery to cure an illness is seeking help from caring individuals, there is always a need for such a system.

The enormous potential of the crowdfunding market means that it is poised for a promising future, which in turn necessitates the development of a digital and easy-to-use platform that will distribute funds securely without any issues in the process. Traditional crowdfunding applications are available; however, there are several issues with the current platforms.

The main problems existing in the current crowdfunding industry worldwide are constantly changing agreements on project delivery, such as extending deadlines, altering the initial project objectives, and the possibility of funds being distributed unevenly, meaning that only a tiny fraction will achieve the original goals. Additionally, creators of campaigns may falsify project descriptions to easily obtain funds for their own gain. Moreover, to have faith in crowdfunding services delivering cash correctly, there is a necessity to rely on intermediaries that deduct a particular fraction from the total generated money. In fact, the problem with available campaigns on platforms not being rigorously regulated and the risk of fraudulent operations significantly undermines user trust and the integrity of the platform, necessitating a solution to eliminate such issues and ensure credibility for the crowdfunding platform.

The addition of blockchain technology can eliminate these problems. For example, the data that is stored on the blockchain cannot be changed. All financial transactions are open and can be tracked. In addition, smart contracts contain certain rules to control financial transactions, and there are no third parties, eliminating fees on every transaction. Implementing a blockchain-based crowdfunding idea in a mobile application will improve accessibility for users since opening an application is faster, easier, and more convenient than manually entering the address in a browser. Mobile apps also offer features like biometric authentication and saved logins for secure and seamless access. The optimized user interface on mobile ensures intuitive navigation and contribution processes.

Crowdfunding may help start-up businesses, victims of natural catastrophes, and individuals seeking funds for medical procedures. Consequently, this helps to advance economic success and community development both domestically and internationally. Creating such a platform in the form of a mobile application powered by blockchain would prevent issues and make the procedure more convenient for various users.

Goal of the work. The fundamental aim of the work is to show how blockchain technologies, particularly smart contracts, can be used to combat issues with the existing crowdfunding applications. The project's overall focus is to make the process in financial transactions more transparent and prevent any malicious acts that could misuse the generated cash from either investors or caring people. By developing smart contracts on Solidity, the project aims to create a secure and automated system for executing crowdfunding transactions. RN is intended to develop a mobile application with a user-friendly interface for accessing and participating in crowdfunding campaigns. Digital wallet such as MM is utilized for securely managing digital identities and facilitating secure transactions within the application. Additionally, blockchain technology leverages the immutability and integrity of transaction records, enhancing trust and transparency in the crowdfunding process.

Research object. The diploma project's research objective is to look into a number of important issues. To begin, it will identify and compare the characteristics of existing crowdfunding platforms to those of the proposed platform, weighing the benefits and drawbacks of each. This evaluation will also contain a thorough description of crowdfunding and its different forms, which are the project's primary emphasis. Second, the project wants to investigate key principles in blockchain technology, including as consensus mechanisms, smart contracts, and decentralisation, in order to get a thorough grasp of its possible applications in crowdfunding. Third, the study will involve the hands-on development of an RN-based mobile application, incorporating essential elements such as smart contracts, Solidity, and MM. Added to that, the study will evaluate the possible impact of adopting blockchain technology into the crowdfunding system, including the advantages, obstacles, and consequences for campaign writers and stakeholders. This research will be complemented by case studies from current blockchain-based crowdfunding platforms. Overall, the study seeks to get a better understanding of blockchain-powered crowdfunding and its consequences for both campaign organisers and donors.

Novelty. This project proposes a new method for mobile app financing that utilizes the capabilities of the Ethereum blockchain. Integrating Ethereum's decentralized platform with smart contracts transforms traditional crowdfunding practices. The utilization of smart contracts automates agreement implementation, thereby enhancing transparency and reducing the reliance on intermediaries. Additionally, Ethereum's DApps provide a secure and adaptable framework for conducting crowdfunding operations. By merging blockchain technology with the inherent advantages of crowdfunding, the project aims to revolutionize fundraising processes, fostering a decentralized and transparent fundraising ecosystem.

Research methodology.

The methodology of the diploma project includes gathering information from various literature sources for the theoretical part and describing the development process to fulfill the main research topic. The literature consists of research articles, books and academic publications that focus on important areas such as crowdfunding, blockchain and especially smart contracts, which will help to obtain key information for the research. Consequently, the practical implementation of the system enhances traditional crowdfunding models that generally lack transparency, fraud prevention mechanisms, and safety, which could compromise the flow of contributed funds. These vulnerabilities might lead fund seekers to dismay and decrease trust levels, where trust is a critical element. A blockchain infrastructure including smart contracts is in place and cryptocurrency wallets are integrated using WalletConnect. Solidity is used to create smart contracts and financial transactions are stored in immutable blocks. This entire system is implemented in an RN-based mobile app. In general, the main goal of the project is to improve current crowdfunding systems using blockchain, with a methodology specifically tailored to analyze the literature review in depth and develop a blockchain-enabled mobile application.

Practical relevance of the work. The importance of the project is not only about fundraising for campaigns. Rather, it is about building a community where people are united while at the same time encouraging others that are compassionate to help those who are faced with specific problems like disaster management as well as community development projects. This influence is more than just financial support because it encourages unity, compassion and together they can work to feel responsible and empowered in a group or shared empowerment among members of society. In addition, the mobile app is perfect for many startups attempting to move their businesses forward. This makes their campaign more trustworthy, transparent, and available thanks to blockchain technology. It is these technologies that ensures secure and immutable transactions which is also important when it comes to confidence by individuals who hold stakes in these enterprises. Besides the mobile app's interface being user-friendly, creates room for wide public engagement with crowdfunding efforts, hence making it more inclusive and important.

Objectives:

- 1 Explore blockchain applications in crowdfunding, establish a blockchain infrastructure, implement smart contracts, and integrate digital wallets such as MM for transactions.
- 2 Create an easy-to-use interface and develop a mobile application on RN with all necessary features to enhance user experience, ensuring smooth accessibility and functionality.

1 Theoretical Part

1.1 Introduction to Crowdfunding

1.1.1 Definition and Concept of Crowdfunding

Crowdfunding is commonly used to back various initiatives, spanning from businesses to humanitarian efforts. For starters, it encourages the expansion of businesses and creativity in fields such as medical research, social entrepreneurial projects, and tourism. On the other side, it is critical in providing funds for humanitarian causes such as disaster relief, access to healthcare in impoverished areas, educational programs in developing nations, and environmental conservation. It means gathering contributions from the general public via online platforms or social networks and utilising their global reach without being limited by geography, all through rapid expansion. For crowdfunding to be successful, it requires active involvement from contributors, crowdfunding platforms, and campaign creators. This approach offers advantages including enhanced global alignment between contributors and fundraisers, early access to comprehensive project details, heightened investment interest and crucial assistance to those in need. Despite the benefits of traditional crowdfunding, it often encounters fraud, evident in false or cancelled projects as well as unfulfilled pledges. Trust in crowdfunding platforms is hindered by information asymmetry, characterized by the lack of clarity and the demand for specific details. Due to these issues, it is predicted that blockchain technology might help improve efficacy by enhancing operational clarity among crowdfunding platform users. [1]

1.1.2 Types of Crowdfunding

There are three categories of crowdfunding that are focused in project, specifically donation-based, reward-based, and equity-based crowdfunding platforms that will be discussed.

- People who give and can be seen giving in donation-based fundraising for humanitarian or artistic projects such as philanthropists try to help less privileged persons in different ways through supporting various charitable organisations so that they meet their needs. Just like a reward-based crowdfunding campaign, winning in such a type of crowdfunding usually lies on what people seeking funds want and how they set their campaigns. These projects only depend upon donations people can give them out of their own will for public good but have no monetary benefits as they are primarily a sense of belonging to a given society. For instance, such operations may involve charity work done by NGOs which often need financial support from individuals who sympathize with their goals. For instance, "goFundme," as per its website, has raised \$15 billion since 2010, with one donation made every second (see <https://www.gofundme.com/c/gofundme-giving-report-2021>). [2] Fundraising through the platform can be

used for various reasons ranging from helping an individual who has a challenging circumstance to specific causes with friends or family members taking the lead. They usually charge a total of 2.9 percent processing fee plus 30 cents for every donation. Hence, if one received \$500 at Gofundme from five donors that would result into \$484.00 being deposited into their account whereas a transaction fee takes \$16.00. Donation-based funding for instance is buying equipment for public schools which lack funds; Supporting cancer diagnoses/treatment that can save life or Humanitarian activities like provision of water and food to refugees. These cases show why such endeavors are important. Using donation-based fundraising platforms, such as charities and non-governmental organizations can connect campaigns to make it easier over traditional methods like adverts and mail sent to donors who had given before. When all is said and done, crowd funding donations are nothing short of a potent tool for pooling funds toward resolving human issues as well as nurturing the society's artistry by means of reliance on good will from people trying out their best to be of help to others. Through it all, donation-based crowdfunding is all about bringing together individuals' efforts in regard to supporting significant projects and boosting positivity within the society. [1]

- In reward-based crowdfunding platforms, funders operate as makers in essence while smaller contributors are in most cases not really looking at it from a money-making end. This kind of situation allows for project originators to get most of their backing by basically selling a future product already without the loss chances as seen from their perspective, though not for sure in any scenario given that we mainly deal with goods and services here. At the same time, such design may bring unpredictable outcomes from the supporter's perspective on whether or not it meets their needs as they see it. Nevertheless, funding can predict potential future demand and identify future funding rounds through established methods such as venture capital or bank loans. Funders can also promote the product by acting as ambassadors through platforms like Facebook, and in return receive more credits. People who contribute to the reward-based crowdfunding platforms get rewards that are material and non-monetary in return for their support, which has been a major motivation for them. This instantiation points to taste heterogeneity, indicating its more consequential nature in reward-based crowdfunding relative to equity-based platforms. For a project to succeed, it has to meet the planned goals and assure that money is disbursed contingent on the accomplishment of a project. The computer game named "Star Citizen" raised 104 million dollars via reward crowdfunding in 2013, being the most successful project at all times on the basis of total cash collected. Most of these belonged to common enthusiasts. In exchange for the promised outcomes, they received virtual spacecraft to use in the game, early access to unfinished versions, t-shirts, and so on. Kickstarter stands out as a crucial reward-based platform, having supported

257,086 projects with nearly \$7.95 billion in funding and over 94 million pledges since its launch (see: <https://www.kickstarter.com/>). [2] These projects span various categories, from art to technology, and offer a range of rewards from small contributions to substantial amounts. Prizes mostly involve the end product that is personal items, public acknowledgements just to show how the platform can be used to fund various creative projects according to varied needs. In general, reward-centric crowdfunding environments have shifted the power dynamic from mere recipients to sources of funds with many more advantages than only money gained in return. The efficacy of this model in various forms of art is underscored by their wide use as well as by the triumphs of certain games like Star Citizen and platforms such as Kickstarter; these are powered by different rewards and the involvement of a wider section of people that have diverse inclinations towards forms of amusement. [1]

- In a firm that is equity-based, donors are treated as investors or as lenders. To run a successful campaign, they would have to evaluate the risks of making an investment. The doubt that surrounds the funders is if the product of the project will be enough for the demands of the large prospective customer base. Fundraisers on these websites use the liken stock options as a way of backing up their campaigns. They often identify a particular goal related to money, and if this goal is not achieved, none of the money already set aside for such task remains; so it is a way of securing the interests of donors while setting reasonable targets for the financial plan needed by the project. Instead of starting with venture capital or bank financing in the form of loans and overdrafts as was previously deemed necessary, equity crowdfunding gives companies the ability to request online donations from a wide range of people interested in helping them out financially. It normally is just one stage in the fundraising processes that involves getting money from supporters at various intervals. Crowdcube, based in the United Kingdom, boasts to have sponsored over 200 initiatives, including 28 of the 30 most popular European equity crowdfunding campaigns, raising an average of £450,000 per campaign in the first half of 2023. (see on <https://www.crowdcube.com/explore/raising>). [2] As you use these platforms to invest, a fee of 2.49% will be charged on any amount deposited and this fee applies all through the investment process. In the happening of making a payment, this specific proportion amounting to any figure from £2.49 up to £250 is withdrawn just as is the case with any other crowdfunding sites where transactions are involved. This type of crowdfunding allows investors to acquire shares and become owners of the company, with returns based on earnings. [1]

1.1.3 Issues with Existing Crowdfunding Platform

Blockchain based solutions outperform traditional crowdfunding systems because they cannot hide or modify transaction-related information, which might

result in low cash generation and unreliable campaigns. Furthermore, this enables for a better understanding of the authenticity of the offered campaigns by modifying some of their content. Trust is destroyed by the ability to deceive; spreading false information through misleading campaigns that may result in fundraising for non-existent projects or misuse of funds, making it difficult to obtain support for genuine causes. Another major problem lies in dependence on external agents who collect money for each activity to meet such expenditure possibilities like charges on connection, platform maintenance and payment system operations. These middlemen are not credible in front of potential shareholders hence making everything more complicated and lowering the speed while at the same time denying people power over their own savings and the related undertakings reducing thereby their involvement in making choices. A further factor to consider is that financing is limited in these centralised crowdfunding platforms owing to the fact that they always have certain goals for how much money must be gathered before any campaigns can begin, as well as time constraints within which such targets must be achieved. If the company needs more money, this restriction will create problems in obtaining it and the consequences of not having additional funds if targets are not met or funding expires. On the other hand, people who hold crowdfunding sites have strict guidelines about the way money is generated which can deprive creators of the project and those who support them an opportunity to decide for themselves. Frustration arises due to a lack of control, which in turn results in loss of trust among all parties involved. Backers should weigh the risks of traditional crowdfunding, which include potential financial loss, lesser returns, project failure, delays, insufficient updates, money mismanagement, and a lack of clarity. This will help them avoid such eventualities as much as possible, and prevent falling victim to any financial problems that may arise as a result of these risks [3].

1.1.4 Benefits of Decentralized Crowdfunding Platform

Switching from a centralised paradigm to a blockchain-centered pattern in traditional fundraising eliminates a plethora of potential vulnerabilities. When it comes to dependability, it is worth noting that all campaign information, including financial transactional details, is safely kept and retrievable by anybody who uses blockchain technology. Unlike other databases, blockchain ensures that stored data cannot be changed without altering subsequent data in the interconnected chain, which offers a more secure and reliable way for crowdfunding systems.

Furthermore, blockchain technology improves processes by requiring consensus before any transaction can proceed. The distributed storing of transaction information among several computers in the network, together with this consensus method, prevents the ledger from being controlled by a single party because the authenticity of each block must be validated by others. In simple terms,

blockchain’s immutability capabilities allow for tracing, verifying, and validating any transaction mistakes, information modifications, or donation records, ensuring public control over the platform. Blockchain technology provides responses to prevention of fraud in crowdfunding platforms regarding trustworthiness. The adoption of blockchain technologies by crowdfunding platforms aims at boosting trust between the contributors and campaigns owners, providing mechanisms that will help in checking fraud and enhancing the security as well as credibility of the platform. This involves distinguishing between public and private blockchains. Open blockchains (such as Bitcoin or Ethereum) do not require an administrator to be invited and anyone can participate in the agreement protocol, while closed blockchains have their limitations. [4]

Further, the integration of blockchain technology results into lowering of operating costs by employing various cryptocurrencies and lesser connection shares plus payments from third parties including financial institutions. Automated agreement-formed systems avoid these agents by reducing the transaction charges as well as other costs involved. The savings can be enjoyed by developers who may have access to more money when they need it or the investors who find their contribution limited by fee charges. Moreover, the operation process is made simpler through automation and digitization of tasks such as fund collection, verification and disbursement by use of blockchain technology. By removing the need for manual intervention, the likelihood of human errors is reduced. Fundraisers and contributors would be more interested and trusting to online fundraising platforms if blockchain applications cybersecurity capabilities are anything to go by because it fosters proper accountability when handling contracts relating to cash through secure checks that have been put in place. Achieving sustainable financing along with improving the performance of social organizations becomes a reality. [5]

The integration of blockchain technology ensures unchangeable records, clear visibility, and decentralized management, effectively tackling the trust and security issues commonly found in traditional crowdfunding platforms. Our project contributes to the advancement of blockchain-based crowdfunding, offering a promising solution for the crowdfunding ecosystem. This perspective aligns with scholars who recognize blockchain technology’s potential to serve as an alternative structure for social crowdfunding and focusing on creative approaches to contribute social value [6].

1.2 Overview of Blockchain Technology

1.2.1 Understanding Blockchain Basics

Blockchain technology is a software protocol that revolutionizes digital record-keeping by creating a decentralized, unalterable, sequential ledger that ensures security, clarity, and authenticity. It functions as a decentralized

database that securely stores transactions in immutable blocks. This method maintains chronological records without a central authority. Blockchain's primary characteristics include decentralisation, data consistency through consensus methods, longevity, making information difficult to wipe, user anonymity, and transaction traceability. It comes in a variety of formats, including public, private, and hybrid blockchains, each with its own set of advantages. [7]

The main benefit is that encrypted transactions can be easily trusted, which helps to transfer money securely without the involvement of intermediaries. The stability of digital transactions and record keeping is due to the fact that they utilize distributed ledgers that prevent fraudulent online activities. [8]

As Blockchain has changed quite a lot during its existence, understanding the progress of this technology is essential.

- Bitcoin was launched as a programmable currency by Blockchain 1.0 to lead to the development of a decentralized digital transfer system where keys are required for transactions.

- Blockchain 2.0 was the beginning of a programmable society in which blockchain applications spread to other social sectors, including finance, peer-to-peer transactions, registration of trustworthy information, verification of property and copyright rights, and smart governance.

- Blockchain 3.0 expanded blockchain's utility to decentralized applications, enhancing operational efficiency and societal trust through decentralization and trusted data sharing.

- Blockchain 4.0: Another improvement from the previous generation is seen in the case of applying the consensus protocol which governs inside the network and makes possible the use of DApps in real-time business situations during Industry Revolution (IR) 4.0 times.

- Blockchain 5.0 addressed traditional blockchain limitations by improving virtual connections, resulting in enhanced processing speed and security. [9]

Traditional centralized systems work by having a central authority in charge of data control but blockchain is decentralized in nature then censorship, single point failures, and manipulation in the entire network are avoided. Consensus methods are the rules that all blockchain users agree upon, like Proof of Work (PoW), Proof of Stake (PoS), or Delegated Proof of Stake (DPoS). The methods promote the legitimacy of transactions between various users, in turn ensuring transactions are always conducted in a safe and secure manner via blockchain technology that brings consensus and trust among distributed networks. Since blockchain transactions are recorded on a public ledger, it is difficult for anybody to edit or erase them. This attribute promotes trustworthiness in information preservation while also increasing the reliability of facts, attesting to their truth; as a result, most people believe them. Again, these cryptographic approaches

safeguard all transactions so that they cannot be changed. Transactions on blockchain are encrypted using cryptography to prevent unauthorised access. Each transaction is linked to the previous one using a chain-like structure, creating a secure and tamper-proof record. Blockchain's advanced security features, such as cryptographic encryption and interlinked transactions, guarantee the authenticity and permanence of recorded data. [10] These measures improve the integrity and reliability of blockchain-based systems. Finally, self-executing agreements, also known as smart contracts, automate actions when certain circumstances are satisfied, eliminating the need for third parties. Applications include crowdfunding, supply chain management, and decentralized finance, among others. Crowdfunding involves automated fund collecting and release depending on defined steps. They make transactions more transparent and traceable. Smart contracts enable automated financial transactions that eliminates the need for traditional intermediaries. Smart contracts improve procedures, lower costs, and boost trust across industries.

The potential application of Blockchain Technology in the finance sector could be observed. Transaction fees, which are usually imposed by traditional institutions, could be reduced or even eliminated. Consumers must depend on banks or outside organizations to conduct transactions for each money transfer. The use of blockchain technology eliminates dependence on intermediaries such as banks, eliminating commissions and other potential costs associated with these services. Furthermore, by allowing for new levels of interactivity and flexibility in services, assets, and holdings, the digitization of financial instruments, such as digital assets and smart contracts further extends benefits of blockchain technology. Digitization allows asset authentication and a complete transaction history to be verified in a single common source of truth, ensuring data integrity. Blockchain is a technology that makes a readily available, secure, and impenetrable record of online transactions. It shares the record of transactions spread across a continuous network of users and has no central authority, meaning that it's akin to the working process of the internet. The structure consists of several data blocks, each of which contains a collection of transactions. Those blocks are connected and protected by sophisticated cryptography. [11] To add, it increases security while automating compliance with smart contracts. The financial services sector utilizes blockchain technology more often where this invention has transformed the global financial system and enhanced its efficiency and security. There are several ways how blockchain is improving the global financial sector. The main advantage of blockchain is the principle of building a worldwide network that is both cost-effective and transparent that is known as cross border settlements. Costs are decreased while the users receive overall value addition.

1.2.2 Impact on Financial Services

The financial services sector that includes banking, fraud prevention, credit scoring, transaction monitoring and privacy maintenance is rapidly adopting blockchain to enhance services, combat fraud, and reduce client costs. Blockchain improves international transactions' efficiency and affordability. Its widespread adoption across businesses of all sizes is reshaping the financial landscape. It is very important to implement blockchain on a large scale as it encourages the standardization of handling transactions in financial services, for instance, if banks adopt blockchain it allows for peer-to-peer transactions as well as smart contracts that can help in sending money quickly, making payment settlements easy and at the same time enable cross-border transactions move at a faster speed. Blockchain is important for everyone to join directly in industries that demand rapid, easy-to-verify, and difficult-to-alter actions and information since it has the world's greatest network security capabilities, which prevent alterations and reduce fraud instances. [12] Blockchain technology can revolutionize the financial sector's efficiency by streamlining document management, although its decentralized nature poses challenges. It also reduces fraud by speeding up secure transactions; financial institutions are vulnerable to fraud and will as such benefit from the use of this technology. There is a great importance in making contract management easier through self-executing smart contracts leading to more effective results. Significant benefits it gives contain reduction of fraud, rapid and secure exchanges, and improved risk management. Blockchain is having a significant impact on the calculation of credit scores in banking and finance, driving innovation, encouraging collaborative models, facilitating efficient processes, reducing costs and securing business networks. Its unchangeable nature speeds up auditing, ensuring compliance and effective activity monitoring. When one manages credit scores on the blockchain, one ensures data protection, which guarantees a partnership between people who need each other's data through smart contracts. Furthermore, these blockchains not only help financial institutions, specifically in managing trusts, by maintaining clients' data, but also enhance investment process, safeguard privacy, ensure confidentiality and efficiently monitor transactions. The blockchain is a decentralized ledger for keeping track of transactions that is synonymous with smart contracts, rapid and low-priced transaction processes, and upgraded safety methods that benefit banks. An extensive application of this idea is being undertaken. [13] Some banks have started incorporating it into their system in order to achieve money transfer within 3 minutes because it is faster than the traditional banking system, at the same time ensuring protection. Blockchain's decentralized structure enhances transaction secrecy and record safety, which helps to bring down disputes around missing or wrong transactions. Same as that, countless transactions can be

recorded in a faster, more secure way within the clearing and settlement systems in investment banks, through a blockchain augmentation of existing systems. It maintains privacy and confidentiality while promoting openness, trust, and performance, especially in private and hybrid systems that can handle a large number of transactions. Blockchain significantly reduces processing time and costs by automating approval process flows and clearing computations via smart contracts, making them an effective tool for storing vast amounts of financial information throughout the world. As a result, financial service providers and their consumers may reduce costs while simultaneously improving credibility, payment speed, clarity, and security. Before such technology, payments between banks could take up to a week to transfer. However, due to digital currencies and distributed ledger technologies, payments, as a result, are quicker, less expensive, and more convenient. Leaving a digital mark on the blockchain will also help entities whose origins determines their worth. A platform for truth and trust is an immutable distributed ledger of digital assets. The consequences are enormous for practically every sphere of society, not just the financial services sector. [14]

1.2.3 Blockchain Core Concepts

- Smart Contracts: Smart contracts are essential in blockchain technology, namely Bitcoin, as it simplifies transactions and accords securely for people and firms. This serve as a mediator cuts off the middle-man between the two parties such as individuals' organisation. Unlike any other contract type, every smart contract is self-executable after being triggered by specific pre-written conditions leading to their enforcement without involving any middlemen including; a central government or external regulatory agencies. Code's decentralized and shared application during transactions guarantees their permanence and traceability; moreover, it makes it possible to use blockchain technology. Smart contracts promise to have a major impact on several industries, from finance to engineering, by serving as computer protocols in the virtual machine (VM) of blockchain. But their implementation is not easy and depends on factors such as the type of VM used e.g. Ethereum or Corda and the specific contract area. [15] However, the right virtual machine chosen greatly affects how fast transactions take place and the general efficiency. To begin a blockchain smart agreement, a transaction is required. The best way to do this is to implement a constructor function. It is this constructor function that will eventually be deployed as the complete source code onto the network for this specific smart contract. Consequently, developers of such contracts enjoy various return types among them being the address of the contract. Users can interact with the smart contract by executing accessible functions, facilitated by returned parameters like the contract address. Smart contracts are meant to incorporate various types of relevant information that aids their correct operation across several applications. Status variables are

stable information stored in a smart contract and are commonly used to monitor and manage ongoing actions and situations. Functions are executable blocks of code that execute certain actions or operations within a contractual agreement, allowing it to do calculations internally while also interacting with other entities outside. Function modifiers improve functions by having pre-execution checks or conditions, which make the code clearer and can easily be reused. Through events, smart contracts can interact with external tools, logging tools, triggering certain actions when certain circumstances or events occur including notifications. A structure is a customisable data type that combines many variables to produce a single structure, allowing for easy arrangement and management of compound information in a software. Enumerations are special data types that represent a limited number of status options and may be used to manage predetermined states or categories inside the contract. They are used to handle well-defined statuses or categories within a contract and these can be used by programmers when creating smart contracts depending on what they want. Each kind of enumeration serves its own purpose so that blockchain applications can become more resourceful. [16] Blockchain technological enhancements do not only substitute the conventional contracts with smart contract but they also enable automatic agreement execution within decentralized contexts. This automation together with the built-in blockchain technology safety as well as transparent characteristics have dramatically transformed many areas including keeping records; customer screening; managing data securely; protecting individual information; trading processes among others thereby establishing a new model for transactional procedures across varied sectors. [17]

The smart contract (see Appendix A), known as SimpleContract houses an integer value referred to as the value and also the owner's address. Upon deployment of the contract, value and the address for the owner are set to zero and the address of the person deploying it respectively, msg.sender. setValue and getValue are two functions in the contract that will enable the owner insert new values or display the existing value respectively. setValue function is a restricted access area meant for the owner more so for the purposes of security.

- A decentralized environment necessitates the consensus mechanism for the authentication of network transactions, safety upkeep and trust that is needed among partners who are unfamiliar to each other. Consensus algorithms, crucial components of any blockchain, enable nodes that are connected to a network of computers terrain on identical grounds regarding what is happening currently in the ledger that is dispersed thereby enabling untrustworthy nodes to trust each other. Trust and reliability are guaranteed in various network types using agreement mechanisms. These consensus mechanisms require that the nodes in the network agree and cooperate, which guarantees equal rights between them

all. Subsequently, every new block that comes is regarded as being worldwide accepted by every node. [18]

- 1 Bitcoin network uses Proof of Work (PoW) as its most commonly employed consensus algorithm. It selects miners based on solving complicated mathematical problems that have high computational requirements determining who will generate blocks. Whoever solves one such problem becomes eligible for mining the next block and earning an award from it. Then, validators are responsible with determining if the block contains any potentially damaging material. [19]
- 2 Proof of Stake (PoS) is a less energy intensive method in which participants in blockchain technology invest in coins rather than using expensive hardware, saving energy by reducing power consumption. An example of this change is Ethereum's move to PoS in the Serenity update. Unlike PoW, which places a high value on investing in hardware, PoS requires players to lock up a portion of their coin reserve so that it can be used to invest in the network. In this process, they earn money by betting on them once they are confirmed. The payout they receive depends on the degree of risk they take, which entitles them to a reward. Consequently, this program encourages an agreement between people and a chapter with an economic interest in the emergence of a hero is chosen to create a block. Last but not least, the PoS principle requires that a person has at least 32 Ethers, which they use for betting to easily make profits as well as recoup their own money from what others give them for transactions made.
- 3 Proof of Burn (PoB) is an alternative strategy in which validators burn money irretrievably in order to earn mining privileges, emphasising long-term commitment in return for short-term loss. This strategy, despite demanding a significant stake, has been criticised for wasting resources and giving an edge to those ready to burn more.
- 4 Validators in the Proof of Capacity system gamble with electronic disk drive capacity as opposed to hardware or tokens. The more storage the more likely they are to receive mining awards
- 5 Proof of Elapsed Time (PoET) is one of the fairest consensus algorithms, especially popular in permissioned blockchain networks. To ensure fairness for all validators, each validator has an equal probability of creating a block. To accomplish this, validators wait to be selected at random times and then publish their wait duration data on the blockchain. These blocks are sent out across the network for validation. The node that confirmed its block before others will be included in the blockchain because it waited the least amount of time. The method includes precautions to prevent one node from winning all the time.
- 6 Proof of Authority (PoA) stands out as a reliable, efficient, and secure

consensus algorithm, granting block generation rights to pre-approved nodes based on their authority, not stake or hardware. [18]

The successful PoA implementation in Microsoft Azure emphasizes the significance of choosing the right consensus algorithm depending on the needs of the business network. The types of consensus mechanisms are very important for transaction validation in blockchain networks.

It is common that in blockchain networks two miners mine simultaneously resulting in simultaneous creation of blocks. Each node broadcasts the blockchain network in half by sending the block to neighboring nodes. [18] These segments exhibit differing blockchains, characterized by variations in the last appended block. As a result, each node in the blockchain network possesses a distinct version of the blockchain, contributing to the decentralized nature of the system, as shown in the following figure:

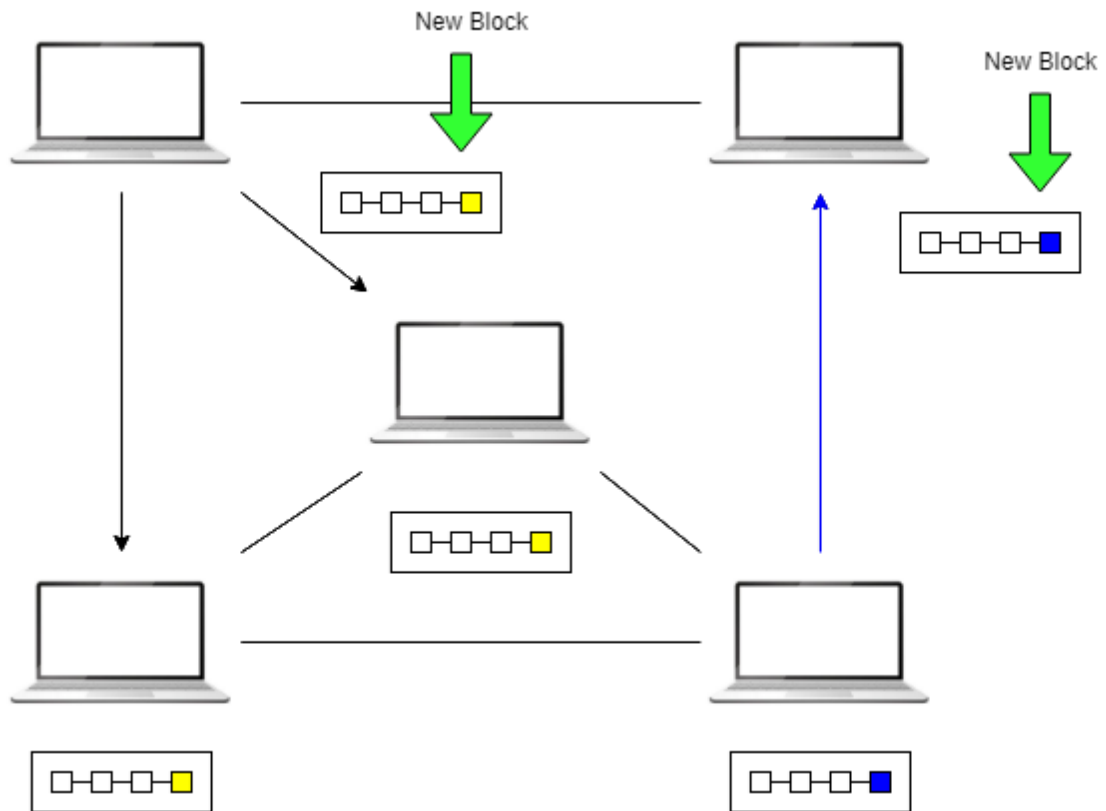


Figure 1.1 – Blockchain Segmentation

When many blocks are mined at the same time, the longest chain rule determines which block is accepted. Among the two resulting segments, the segment that successfully mines the next block establishes the longest chain, while the other segment must discard its block and integrate the new blocks into its blockchain. The block deleted from the shorter section is referred to as a 'Orphan

Block', and all transactions are disregarded, with no reward for the miner. The result emphasises the crucial role blockchain mining plays in maintaining system security and trust. [18]

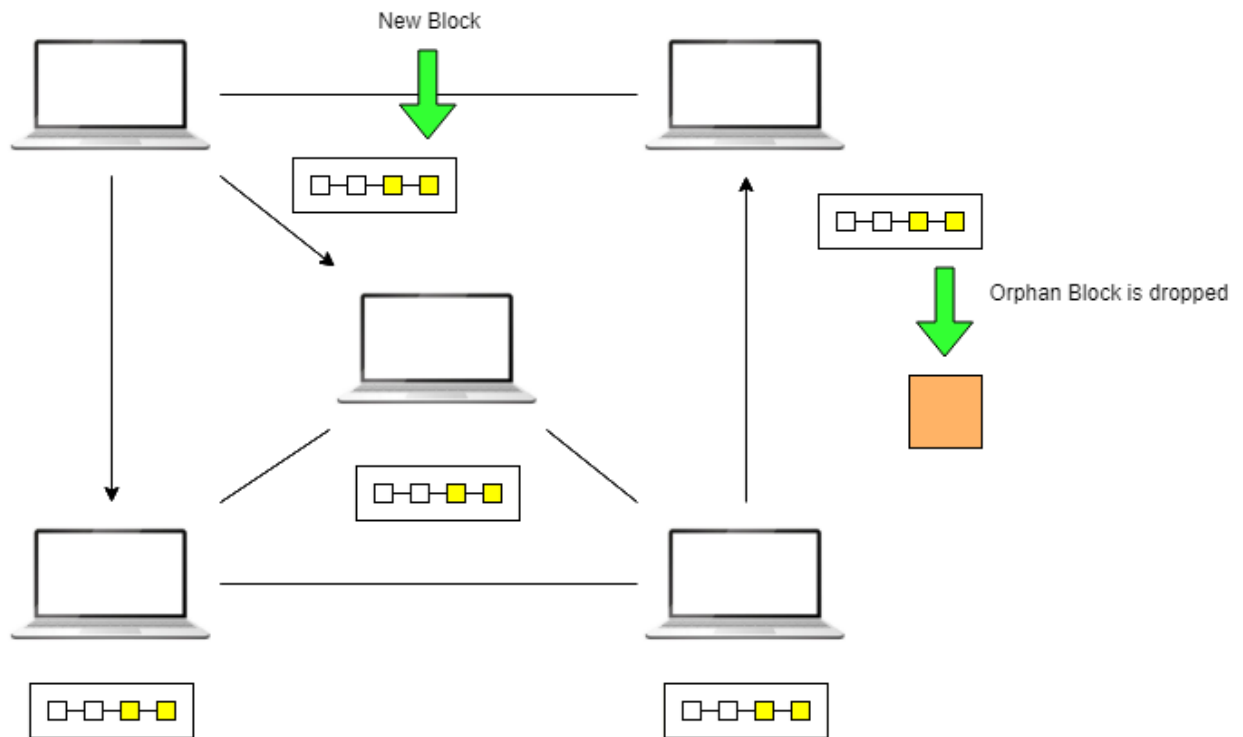


Figure 1.2 – Orphan Block

- Several critical features increase the security and dependability of blockchain transactions. For example, blockchain permanently archives transaction records, ensuring that all transactions cannot be reversed. These transactions are largely safe due to the use of advanced cryptographic primitives that ensure their secrecy and integrity. Additionally, transactions are systematically arranged by time, where each one comes after the other one, which is good because one can easily follow the audit trail without confusion. Moreover, transactions are verifiable such that anybody can know where they came from and whether they are genuine. A change is made on the blockchain when it is recorded and cannot be reversed or altered in any way. All data in the ledger is safely kept and accurately maintained via cryptography, with access limited to authorised keys and cryptographic signatures. Each transaction is authenticated using the sender's private key, which validates its legitimacy and creates confidence in the process. [18]

The provided code's structure in Appendix A depicts a usual Ethereum transaction; it has a nonce which allows monitoring transactions in order, gas price to show transaction fee rates, gas limit that establishes a cap on the gas

usage in a transaction, receiver's account (to), amount of Ether to transfer (value), miscellaneous data attached to transactions and cryptographic functions (v, r, s) employed for the purpose of checking the signature. Each element is necessary to carry out and validate transactions on the Ethereum blockchain. [18]

- Ethereum operates as a network that's structured on blockchain technology. It uses smart contracts to allow applications to function without delays, fraudulent activities, or the need for intermediaries. The programmes and transactions are distributed and processed via a peer-to-peer (P2P) network, which also allows for the issue of tokens to execute smart contracts. In Ethereum network, various nodes handle different tasks. Full nodes do two things – contain all the blockchain information locally and check blocks. Besides, light nodes have only block headers, and they were created to handle transactions mostly. Archive nodes keep historical blockchain data. Ethereum accounts store balances of Ethereum (ETH) and are capable of conducting transactions within the network. There are two types of account: externally owned accounts (EOAs) generated by wallets, based on private keys for access and which communicate with smart contracts for activities such as checking balances, transferring ether, or contracting. Contract accounts (CA) are generated when a contract is deployed and are controlled by the contract code. In Ethereum fees are charged for transactions which is known as gas, a computational unit. Gas prices are determined in gwei, it indicates how much each unit of gas costs for the sender or receiver. If miners charge higher gas prices they will be likely to process transactions more quickly. The gas caps deal with the potential abuse threats that come with the cost of resourceful transactions. Ethereum offers various ether units (wei, szabo, finney, and ether) to specify transaction amounts. For example, 1 gwei is equivalent to 10 power of minus 9 Ether. Solidity programming language used for smart contracts on Ethereum allows developers to control gas consumption, so they can save on transaction charges. Total ether needed for execution is directly proportional to gas used multiplied by gas used which both require money. It is important that we correctly manage gas when operating decentralized apps over this network if they are to be efficient and safe. [18]

Amount of Ether needed to process a transaction:

$$UtilizedEthers = spentGas * priceOfGas$$

- Blockchain technology has numerous critical features. Foremost among these is that it has been set up as an open system which means that anyone can access blockchain data. The outcome of such openness is inclusivity throughout the network. Secondly, identity verification and validation of each operation require the consensus mechanism that often requires that specific nodes vote. An instance is the acceleration of consensus building by this method when different nodes are operating without conflicting interest but agree to the same

transaction. A network can only guarantee the legitimacy of a transaction once an agreement has been reached. Furthermore, the actions of all the nodes in the blockchain network are calculated using algorithms. To add on, the algorithms also assign accounting rights to these nodes thereby encouraging fair competition and preventing the network from being taken over by a single entity. Also, under supervision, authenticity and integrity of each and every record are preserved using blockchain technology by recording them precisely. This has led to secure storing of each transaction and makes it impossible for alteration later on further increasing reliance and trust. Blockchain security is all about data encryption and cryptography, which are its most basic components. These two make it impossible for malicious tampering or faking because they keep our information safe while at the same time allowing them retain their original form whether it is being encrypted or decrypted. By employing encryption standards that require each node to possess its own unique key, we can find and diffuse potential attackers in blockchain systems. Key-based encryption only allows secure transmission of data packets after confirming that the sender's key is valid. Such processes like checksum division further enhance security of information within the blockchain system. [9]

- Bitcoin is a kind of digital currency that serves to make transactions between people directly, without any intermediaries. These transfers are simple but secure, and usually take only a few minutes. Number of operations that Bitcoin can handle is pretty big, although it depends on their size since there is only 1MB block size. The sender starts the Bitcoin transaction process when he sends the receiver bitcoins. At this point in time, the receiver's public address is utilized for authorization of the transaction and ownership is verified with the sender's private key. The Bitcoin network then receives the transaction. After this, miners gather the transactions and start forming a new block in them. Transactions remain unverified yet until it is confirmed and added to a block by a miner. Miners receive newly generated bitcoins as well as transaction fees for doing this job. Upon successful inclusion of a block into the blockchain by a miner, it is distributed equally to the whole network so as to be confirmed. A block is considered valid and added to the blockchain once a majority of nodes (over 50%) confirm its validity. This process ensures the integrity and security of the Bitcoin network, completing the Bitcoin transfer successfully. [18]

- MM is a cryptocurrency wallet that functions as a gateway to decentralized apps (DApps), allowing users to interface directly with blockchain networks via web browsers. It further enables users to connect to a variety of blockchain networks, including test networks for deploying smart contracts. MM lets you manage and spend cryptocurrency anytime, anywhere. It provides a range of features, such as opening new accounts, checking transactions, get ethers into the

MM wallet, and sending ether from one account to another right through the application interface. [18]

1.2.4 Blockchain Security

Using the blockchain technology in crowdfunding raises certain security worries that need to be solved for a platform to function well. Those challenges are privacy breaches, logic errors, transactions ordering difficulties, time stamp problems, call stack size constraints, structural weaknesses, scaling difficulties, computing power control, transaction confirmation delays, anonymous questions, emergency stops, integration costs and internet of things. Despite the transparency of transactions, blockchain compromises the privacy of its users. The range of well-known vulnerabilities involve selfish mining as well as the threat of Sybil attacks. Dishonest miners tend to withdraw new blocks from honest ones by means of withholding them within selfish mining. Whenever a person creates many accounts on this network without any proper explanation, they may do Sybil attacks thereby affecting its overall efficiency. In addition, independent blockchain-run smart contracts may result in extra unsafe factors like blockchain flaws, faulty smart contract code etc. as well as the virtual machine which carries out these actions. Building smart contracts within blockchains is essential being that once these are put into place, they can never be undone easily. However, mistakes made when writing the program for such software-based agreements could have very negative consequences since they can never be altered post-dissemination. This emphasizes the importance of verification before deployment. Most smart contracts do not make publicly available their source codes which hampers audits and over 77% of them do not disclose these codes. Government auditors lack control over whether contracts remain safe or dependable due to lack of clear understanding. If all transactions are not ordered properly in blockchain systems, then their future states will not be determined. Failure in arranging or still waiting transactions can lead to concurrency problem, which can be utilized by criminal nodes. Particularly, when a single block has a lot of dependent transactions which execute the same smart contract multiple occasions, this matter becomes more pressing. Using Ethereum's smart contracts, miners can create newer blocks having random timestamps inside a given period. This flexibility can be exploited by attackers who modify timestamp within the permitted window, thereby affecting smart contracts outcome as well as opening it to attacks. The number of calls to an individual smart contract, which is a result of call stack size, is always limited to 1024 calls. The blockchain grows large and more calls are made, thus surpassing this limit, thus causing exceptions. It is extremely important, as it has a direct impact on the reliability of smart contract execution, to adhere to this constraint. There is a set of basic security issues that are associated with Blockchain-based smart contracts. For instance, an individual transactions may

lose ether once it is directed to an irregular or homeless address on Ethereum. It becomes necessary, therefore, for the system to ensure that destination address is correct and corresponds to a smart contract in order to prevent permanent loss of funds. Moreover, it is essential to note that powerful computer systems can retrace and change block hashes contained in blocks immediately following any given block, leading to vulnerability attacks. The fast growth in blockchain data and the standard block size of 1MB in addition to the high level of security that it provides means that it grows faster than many other technologies. As a result of rapidly growing amount of information, different kinds of systems may find it difficult to become larger. One possibility is to reduce the amount of data stored in the local cache by controlling the headers. It gets rid of scalability problems of sorts. On one hand, blockchains guarantee the integrity of data but remain vulnerable to attack if any user has over fifty percent computing power of the network and can thus manipulate or disrupt transactions which pose risks towards honesty of blockchains. Blockchain transactions are generally confirmed within 10 minutes. However, this is way too slow for businesses. Simplified Payment Verification (SPV) and other approaches can cut the validation time by checking only the block headers. Still, there is a notable shortcoming of the entire blockchain concept. There is no secrecy in public blockchain participation. Despite the fact that blockchain uses hash keys for identification purposes, it is possible for public platforms to follow up on user information by tracking their IP address, or through third-party applications and the like, thereby putting their personal privacy at risk. As such traditional smart contracts do not allow for closure of transactions already consummated something that could slow down the whole system especially when there is an error. This is because the basic nature of blockchain does not allow alteration of information upon recording these transactions. Integrating current systems with blockchain technology might come with its own costs. Software and technology might have to be upgraded while employees are also retrained all in the bid to help firms achieve the required standards of performance. Most companies find these outlays on mergers too high hence they fail to utilize them. Dependencies between IoT devices and nodes that interact in decentralised systems create new security challenges. These difficulties encompass proximity based physical assaults on devices, network attacks exploiting vulnerabilities in IoT networks, software attacks focusing on the soft spots of IoT related software, and data attacks concentrating on unauthorized access and data breaches. You need to have a robust security mechanism in place for the security of data and IoT devices stored and regular updates must be made. To recap, for instance, blockchain technology has numerous ways it can enhance crowdfunding activities but there are many security risks associated with it as well. For these problems to be solved in a reliable and secure manner, we must ensure that the servers

used are well protected from hackers by means of strong encryption methods; always keeping watch on them so that their security does not go down; and also making sure that security procedures are regularly updated in such computing environment. [20]

1.3 Strategies for Effective Crowdfunding: Risk Management, Success Maximization, and Case Studies

1.3.1 Addressing Risks

There may be concerns in Crowdfunding Blockchain-Based Mobile Applications that must be properly considered. The following are the potential risks listed.

- 1 Overfunded campaigns tend to benefit the platform since they exceed the funding threshold that they were set to achieve; nonetheless, this situation can be overwhelming and risky for the organizers mainly because they may fail to fulfill all promises as expected. When equity crowdfunding campaigns reach an overfunding point, it is required that more shares be produced thereby leading to ownership dilution; however, for this problem's solution purposes, platforms could let fundraisers keep on gathering additional monies past their targets or return them to donors.
- 2 If a campaign does not meet its financial goals it can damage the reputation of both the platform and its fundraisers. This makes it unattractive to both investors and venture capitalists and it does not leave behind a good image considering the marketing strategies or concept attached to the campaign. The risk containment strategies include "all-or-nothing" funding model that only allows refunding if objectives are not met or "keep-it-all" where fundraisers retain all money received.
- 3 Intellectual property laws can be broken in crowdfunding campaigns, or they can be the target of intellectual property piracy from other parties. This occurs when trademarked ideas are unwittingly adopted by the campaign, or when campaign details are stolen for the purpose of isolating production. Fundraisers may counteract this by educating themselves on the dangers of copyrighted material and barring non-members from using their campaigns.
- 4 Platforms that fail to produce items as promised or ensure that their money is spent appropriately undermine an investor's faith and tarnish the project's image. To elaborate, examples might be shown in which money raising has not been supported by the delivery of items or services as promised. To combat this sin, service providers must thoroughly investigate the validity of fundraisers so that people suspected of fraudulent activities are identified.
- 5 One source of regulatory uncertainty is the introduction of legislation requiring crowdfunding platforms to be compliant. Uncertain laws may limit such platforms' activities and increase accountability. As a result, in order to reduce

these risks, regulations governing such firms must be watched and their policies updated in every given context of concurrent or withdrawal in accordance with applicable requirements.

- 6 Uncertain user interfaces can lead to user discontent, less engagement, and higher abandonment rates during the crowdfunding process, particularly on small-screen devices, and this has a substantial impact on the usability of crowdfunding mobile apps. To mitigate such risks, one must concentrate on simplicity, device consistency and accessibility features in order to ensure a smooth and interesting user experience. It is also crucial to optimize UI components for speed as well as incorporate strong security measures in order to increase the app's usefulness among individuals and build users' trust.

1.3.2 Success Factors

Within an efficient market, the success of a fundraiser is greatly dependent on the quality of their campaign. As a result, with money only reaching a few initiatives and different campaign elements absent, financing for successful programmes should be prioritised. Funding should instead prioritise initiatives that are of high quality when evaluated abstractly, despite having limited money. Nonetheless, quality is often unobservable, requiring funders to rely on quality signals. Besides, sponsor characteristics may not always correlate to the quality of the project, but they may impact the formulation of financial decisions. According to this analysis, social networks are vital. This section focuses on social networks and how sponsors, fundraisers, and Crowdfunding Platforms (CFPs) use them to increase crowdfunding efficacy.

- 1 Funders, lacking information about project quality, may base their decisions on past funding behavior. Another source of quality signals could be 'social buzz' or 'eWOM (electronic word-of-mouth)' manifested through support a campaign garners on social networks like Facebook shares or Twitter tweets. This social buzz could complement campaign descriptions on a CFP and significantly impact campaign success if funders give more weight to recommendations from friends in their social network than general information. The combined effects of popularity information (past funders' decisions) and social buzz on the likelihood of success of crowdfunded campaigns were examined. Data from over 6,000 projects on Indiegogo revealed that social buzz, especially Facebook shares, positively influences project backing. Although the reverse influence, where previous funders create additional eWOM to attract further funders, seems logical, it wasn't supported by the data.
- 2 In addition to obtaining information from their own social network, funders may glean insights from the social network of the fundraisers they intend to support. A fundraiser's number of Facebook connections, indicating the size

of their social network, serves as a reliable predictor of successful fundraising. Furthermore, a fundraiser’s social ties significantly impact the evolution of a fundraising campaign. Local funders (those co-located with the artist they fund) exhibit less responsiveness to the cumulative funding level than distant funders, possibly due to their offline access to the artist. This demonstrates how important geography is in financing decisions, which are most likely influenced by specific knowledge or cultural biases. A selected group of Kickstarter fundraisers that supported alternative projects saw higher success rates, attracting more sponsors and monetary donations. Such a scenario might entail both direct and indirect incentives in crowdfunding settlements.

- 3 Even in platforms where fundraisers typically appear only once, CFPs can utilize social network information to update funders’ beliefs. For example, a CFP with access to LinkedIn data could report success rates of contacts who previously launched projects and are connected to the current fundraiser. Utilizing this information, the CFP can provide insights to funders, influencing their anticipation of project success. Recommender systems within CFPs may leverage funders’ social network characteristics for personalized project suggestions, especially when funders are uncertain about horizontal match values. This may intensify information cascades, potentially leading to both positive and negative outcomes. [2]

1.3.3 Crowdfunding Platforms Case-Study

There are platforms currently utilizing Blockchain Technology in the crowdfunding system, similar to our proposed project. In this case study, the focus lies on existing systems and research linked to the functionality, effectiveness, and potential impact of integrating blockchain technology into crowdfunding platforms.

- 1 KickICO is a platform for collecting donations that supports AIO-based fundraising, with auction sales available both on the KickICO platform and campaign tokens that are automatically admitted for listing at KickEX exchange. The joint participation of the two platforms and exchanges increases significantly the demand for traded tokens leading to more involvement from various community members. Upon successful campaign completion and necessary checks, the company’s tokens become available for trading on the KickEX exchange with automatic listing. Auction based Initial Offering (AIO), utilized on the platform, is a form of crypto fundraising based on fair pricing, developed by the Kick Ecosystem team. Token prices are determined by users through auctions rather than being fixed as in traditional fundraising methods, forming the basis of demand and supply. It has helped reduce corporations’ direct impact on token valuations, as well as pre-sale discounts. Whenever this is the case, it reflects users’ evaluation of the products presented and the

market demand for them, which reduces the chances of losing all their assets once they are listed on an exchange or launched on some public platform; as a result, they have become more expensive than before, even for smaller companies. In the token sale process, the issuer determines the number of tokens and the duration of the sale in advance, and sets a daily quota of token availability. The participants offer a price and the number of tokens, with tokens being given out to the ones who pay more for them. Daily disabling of unused tokens helps to maintain the natural order in the system. KickEX Exchange set its values with market needs in mind making it different from other token sale methods with preset values; this has promoted price stability for secondary markets like we've never seen before. To sum it up, KickICO uses tokens in fundraising where they can represent various assets, entitlements or utilities according to a project's profile. Crowdfunding campaigns on blockchain profit from more safety and variability. This secondary sell as a cryptocurrency at an exchange allows for tokenholders' profits and instant liquidity.

- 2 PledgeCamp is a crowdfunding platform that leverages blockchain technology to provide additional security and responsibility. It operates as a decentralised platform without a central authority as a result resisting platform censorship. Any person is able to show a project on the platform while those who support it employ smart contracts to enable creators to get paid. Any time any project is funded, smart contracts are used to create a device called "Backer Insurance" which is useful in maintaining responsibility within PledgeCamp. Under this system, a portion of the backer's funds is securely held in trust until specified project milestones are met, granting the creator access to these funds. In contrast to profit-driven listing fees, PledgeCamp takes a "smart crowd" strategy, focusing on community building to promote platform growth and utility. The idea behind this method stems from PledgeCamp's token system, which is effectively a token generator that avoids reinventing the wheel. The system contains two currencies: PLG, an ERC-20 utility token that helps ecosystem projects, and Camp Share (CS), which is mirrored from individual accounts whose owners staked PLG tokens for CS tokens at a 1:1 ratio therefore entering the system as 'Moderators'. Moderators maintain platform integrity, earning a portion of listing fees proportional to their staked tokens. In campaign listings, PledgeCamp initially sets aside 50 billion PLG tokens (5% of the initial token supply) to reward Moderators during the platform's early stages.

2 Practical Part

2.1 Introduction of the Practical Part

Within this regard, this thesis project deals with the development of the mentioned mobile application in relation to crowdfunding using blockchain technology for transparent, reliable, and secure transactions that are also convenient for users. Focused on the elaboration of a workable but secure solution to all of such problems, the project will gain the capability of solving the most acute questions of modern-era crowdfunding platforms: fraud, a change in the condition of projects, and course, lack of confidence.

Also, the following technologies will be applied for developing the general interface functionality: React Native with Expo to more easily create the cross-platform mobile app, Solidity to more safely develop intelligent contracts, WalletConnect to allow connection of users' wallets securely to the cryptocurrency, and Wagmi for friendlier ways of interacting with Ethereum's blockchain. This is because these kinds of technologies have many opportunities and are compatible with each other, which will permit an individual to make strong and dependable applications.

The project will involve work with the following:

- User interface development: Develop a user-friendly interface from React Native and Expo.
- Innovative contract development: Develop intelligent contracts in Solidity and test them to manage crowdfunding campaigns.
- Wallet integration: Integration with cryptocurrency wallets for staking and yield farming.
- Security solutions: Implement security solutions and protect user data with seamless transactions.
- Testing and debugging: Functional tests for applications and smart contracts and detecting and removing problems.
- Such verification not only proves the potential of blockchain technology but assures that crowdfunding has been improved. This very thesis, in its practical implementation, will be of significance for the creation of the platform that will help startups and ordinary people in need of financial support to accumulate the needed amount of resources with minimal risks and a maximum level of transparency.

The following sections will provide a description of every stage in the development process, from the configuration of the development environment up to the testing and the results demonstration of the application.

2.2 Blockchain Structure and Operation Overview

A block in a blockchain has two primary components – the block header as well as the block body. Each block has its own hash that is specifically defined using the SHA256 cryptographic hash method applied to its header. The block header usually contains PrevHashBlock which is the hash of the previous block in the chain, timestamp, Tx root and Merkle Root along with Nonce as random number. The block body contains information about the transactions that is kept inside. A Merkle Tree structure in the block body securely keeps many transactions from the prior block. In Merkle's tree, leaf nodes store hashes of transactional data and internal nodes store hashes received from child nodes. As transactions occur, nodes within a consensus-based mechanism compete for the right to validate transactions. During a specific time period, all transactions are associated with the winning node. Subsequently, all nodes across the network verify the block. Upon successful authentication by the majority of nodes, the block is added to the blockchain. [21]

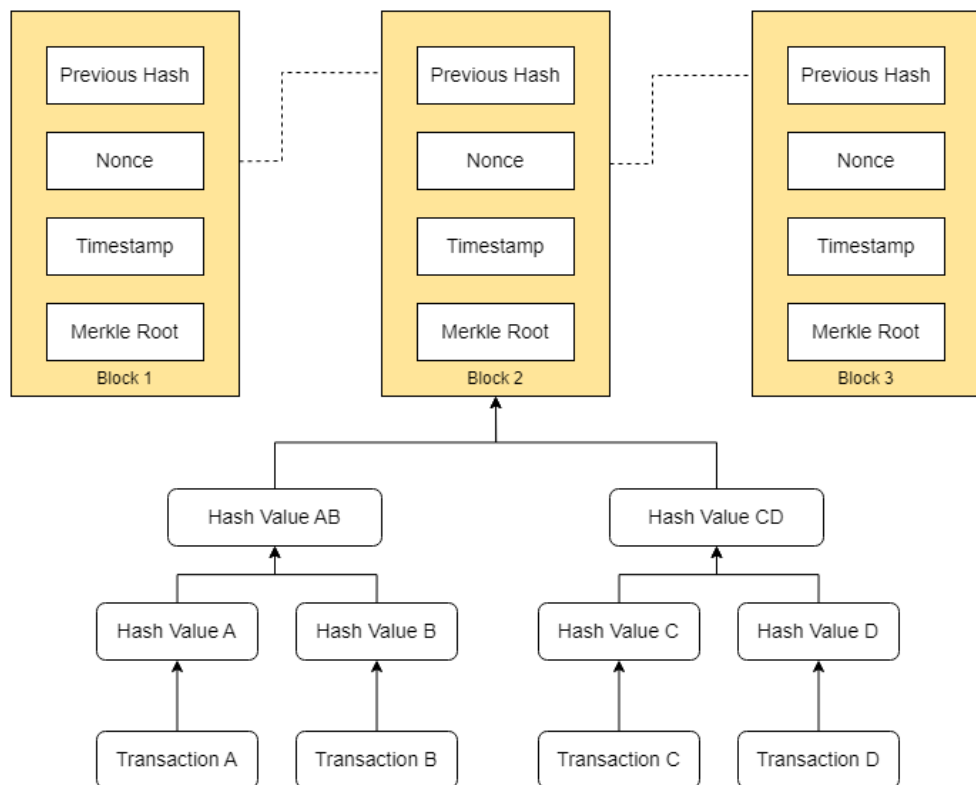


Figure 2.1 – Block Structure in Blockchain

2.3 Architecture of application

2.3.1 System Architecture

In that light, the mobile crowdfunding application is developed with an adequate, secure, scalable, and user-friendly architecture approach and is integrated with state-of-the-art technologies. That structure comprises the most critical three layers: frontend, backend, and blockchain integration. Such a structure would allow continuous interaction across the user interface, intelligent contracts, and blockchain components.

- 1 Frontend (React Native with Expo) Frontend: This is related to the user interface. It's made from React Native; therefore, it's cross-platform. The rendering will take place on iOS as well as Android.

- UI Components: This is how the screens and components to render the UI relate, including lists of projects, forms for creating projects, and user profiles.

- Navigation: It makes it very easy to move from one screen to another with the help of React Navigation.

- State Management: It should have one way of application state management using either Redux or the Context API. In this way, data flow and state are managed consistently throughout components.

Detailed Description: React Native is a powerful concept for building native apps with React. It has a great ecosystem and its friendly community. Expo is a free and open-source platform that pairs with React Native to build native app projects.

- 2 Backend (Solidity Smart Contracts): The backend is implemented using smart contracts written in Solidity and deployed on the Ethereum blockchain. These smart contracts handle the core logic of crowdfunding, including campaign creation, funding, and fund distribution.

- Campaign Contract: Manages the creation and lifecycle of crowdfunding campaigns. Each campaign has attributes such as the owner, title, description, target, deadline, amount collected, image, donators, and donations.

- Functions:

- createCampaign: Allows users to create new crowdfunding campaigns.

- donateToCampaign: Enables users to donate to a specific campaign.

- getDonators: Returns the list of donators and their donations for a specific campaign.

- getCampaigns: Returns all the campaigns created.

Detailed Description:

- Solidity: A programming language designed for developing smart contracts on the Ethereum blockchain, offering robust security features and a growing ecosystem of tools and libraries.

- Smart Contract Functions:
 - createCampaign: Allows users to create new crowdfunding campaigns by specifying details like funding goals and deadlines.
 - donateToCampaign: Enables users to contribute funds to a campaign.
 - finalizeCampaign: Distributes funds to the campaign creator if the funding goal is met or refunds contributors if the goal is not met.
- 3 Blockchain Integration (WalletConnect and Wagmi): Blockchain integration is facilitated by WalletConnect and Wagmi, ensuring secure connections and interactions with users' cryptocurrency wallets and the Ethereum blockchain.
 - WalletConnect: It allows the user to connect to his cryptocurrency wallet securely. In other words, this mobile app will direct the user to sign off on something, but his private key will never leave his device.
 - Wagmi: This allows one to interact with the Ethereum blockchain, from reading and sending smart contract data up to transactions.

Detailed Description:

- WalletConnect: Open protocol connecting dapps to mobile using a QR code scanning or deep-links.
- Wagmi: This library is abstracted over most interactions by smart contracts for Ethereum. It contains routines connecting with wallets, reading from and writing to the blockchain, and managing states of transactions.

2.4 Diagrams and Graphs

2.4.1 Entity Relationship Diagram

The visual representation of different entities and their relationship with each other in a system is called Entity-Relationship Diagram. Each entity has its own respective attribute that is going to complement it. The Implementation of a crowdfunding mobile application based on blockchain consists from objects and their relationships.

The first entity is the user who interacts with the system. The attributes included in it are walletAddress, name and avatar with variable type string. Here walletAddress serves as the primary key, as a consequence it identifies each user.

The Campaign entity represents a user-created project that will generate funds until the goal is achieved. This entity includes data types such as id, owner, title, description, target, deadline, amountCollected and image with their data types beginning from integer to float and ending with string . The id is the primary key identifying each campaign, while the owner attribute acts as a foreign key referencing the walletAddress in the User entity, indicating which user created the campaign.

The next entity is Transaction, which contains information about the funds contributed to the application. This is important object due to the fact that each transaction data is stored in a block of chains making the financial process

transparent and immutable. Its variables are as follows: `_id`, `from`, `to` amount, timestamp and `campaignId`. The `_id` is the primary key that identifies each transaction. Both `from` and `to` are foreign keys that reference the `walletAddress` in the User entity, indicating the sender and receiver of the transaction, respectively. The `campaignId` is a foreign key referencing the `id` in the Campaign entity that links each transaction to a particular campaign.

In general, a user can create multiple campaigns, which is represented by the `owner` attribute in the Campaign entity, which points to the `walletAddress` in the User entity. The user can then make multiple transactions as a sender and a receiver, represented by the `"from"` and `"to"` attributes in the transaction entity that point to the `walletAddress` in the User entity. Moreover, what are represented by the `campaign id` attribute in the transaction entity are numerous transactions linked with a single campaign.

Entities and Attributes:

1 User

- `walletAddress` (Primary Key): A unique identifier for the user's cryptocurrency wallet.
- `name`: The user's name.
- `avatar`: The user's profile image.

2 Campaign

- `id` (Primary Key): A unique identifier for the campaign.
- `owner` (Foreign Key): References the `walletAddress` from the User entity, indicating the campaign creator.
- `title`: The name of the campaign.
- `description`: Details about the campaign.
- `target`: The funding goal for the campaign.
- `deadline`: The end date for the campaign.
- `amountCollected`: The total amount of funds raised.
- `image`: An image representing the campaign.

3 Transaction

- `_id` (Primary Key): A unique identifier for the transaction.
- `from` (Foreign Key): References the `walletAddress` from the User entity, indicating the sender.
- `to` (Foreign Key): References the `walletAddress` from the User entity, indicating the recipient.
- `amount`: The amount of funds transferred in the transaction.
- `timestamp`: The time when the transaction occurred.
- `campaignId` (Foreign Key): References the `id` from the Campaign entity, indicating the associated campaign.

Relationships:

- 1 **User to Campaign:**
 - One-to-Many relationship, where a user can create multiple campaigns, but each campaign is owned by a single user.
- 2 **User to Transaction:**
 - One-to-Many relationship, where a user can be involved in multiple transactions, either as the sender or the recipient.
- 3 **Campaign to Transaction:**
 - One-to-Many relationship, where a single campaign can have multiple transactions associated with it.

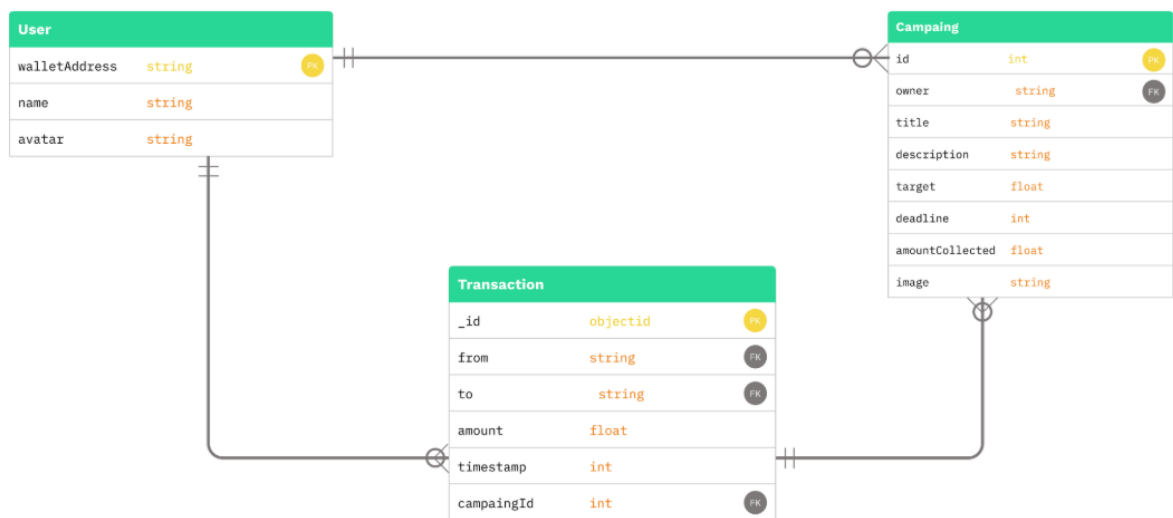


Figure 2.2 – Entity Relationship Diagram (ERD)

2.4.2 Component Diagram

A component diagram is a type of UML diagram that shows the organization and dependencies among different components of a system. For a mobile crowdfunding application, the component diagram will show the interaction between the interface, backend, and blockchain integration components. It will help one understand how different parts of the system collaborate among each other to deliver some common functionality.

Components in the diagram:

- 1 **Interface (React Native with Expo):**
 - User Interface Components: These are the visual parts of the application, namely project list screens, project creation forms, and user profile screens.
 - Navigation: Deals with the transition between various screens with the help of React Navigation.
 - State Management: This allows the information flow and state

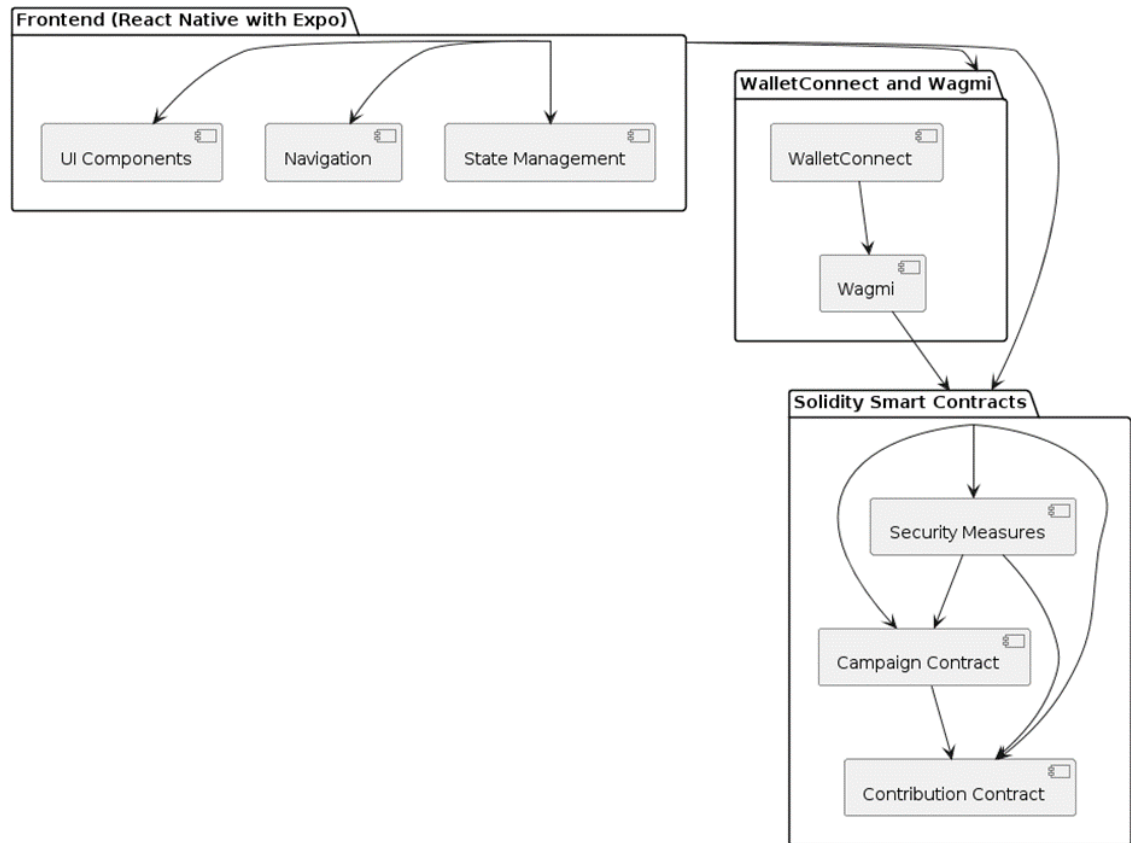


Figure 2.3 – Component Diagram

management among different components in a smooth and consistent way with the use of Redux or Context API.

2 Server side (Solidity Smart Contracts):

- Campaign Contract: Creation and life cycle of crowdfunding campaigns.
- Contribution contract: Manages user contributions and makes sure that money is collected and allocated properly, depending on the success of the campaign.
- Security features: Implements re-entry protection, access control, and more security features to ensure the contracts are safe from common vulnerabilities.

3 Blockchain Integration (WalletConnect and Wagmi):

- WalletConnect: this way, user cryptocurrency wallets will securely be connected with the mobile application.
- Wagmi: Thus, interacting with the Ethereum blockchain will be much easier when reading smart contract data, sending transactions, and more.

Interaction between the components:

1 User interaction:

- The interaction would be through the user interface components in React Native.

- Users will be able to create new campaigns, see their campaigns, and contribute to campaigns right from inside their mobile application.

2 Interaction with the smart contracts:

- When a user creates a campaign, the application will call the createCampaign function on the Ethereum blockchain through Wagmi.
- The user sends the amount to the donateToCampaign function, which processes and ensures that each transaction is securely recorded.

3 Blockchain and wallet integration:

- WalletConnect secures user-linked wallets with the app binding and allows users to authorize without showing private keys.
- Wagmi will manage the interactions with the blockchain to ensure the integrity of the data and the secure execution of the functions of the smart contract.

Description of the diagram component:

1 Interface (React Native with Expo):

- User Interface Components: The interaction with the user is done with several components of the interface, such as forms, lists, and so on.
- Navigation: It controls the transition between the screens of the application.
- State Management: It controls the state of the application, maintaining the consistency of data.

2 Backend (Solidity Smart Contracts):

- Campaign Contract: Campaign information: owner, name, description, goal, deadline, image, donors, donations.
- Contribution Contract: Full information from users who registered and made a transfer.

3 Integration with the blockchain (WalletConnect and Wagmi):

- WalletConnect: It safely links users' cryptocurrency wallets to applications.
- Wagmi: It interacts with the Ethereum blockchain. It increases usability: Contracts and transactions are more convenient to find.

Example of the interaction process:

1 Create campaign:

- The user fills out a form in the interface, thereby creating a new campaign.
- On the backend, createCampaign triggers a function in the smart contract through Wagmi.
- The campaign is stored on the blockchain.

2 Donation:

- The user chooses one of the campaigns to which he/she wants to donate.

- The user authorizes the transaction through his connected wallet using WalletConnect.
- It is a function of the smart contract donateToCampaign, which is called through Wagmi.
- A contribution event is emitted, and a safe transfer of the funds is executed.

The above component diagram explains the logical flow with the mobile crowdfunding application about the interaction between the interface, the back-end, and blockchain. This framework will provide users with a reliable, scalable, and secure platform for them to create and finance their projects in a transparent and efficient manner.

2.4.3 User Data Collection Method

Our application uses WalletConnect to enroll users, allowing them to authenticate and access the platform through their blockchain wallets. This decentralized authentication process ensures that users retain full control over their identity and personal information, without having to disclose additional data such as names or email addresses.

In developing our blockchain-integrated crowdfunding mobile app, a thorough data collection methodology was necessary to ensure that user information, project details, and transaction data were effectively managed while prioritizing user privacy and data security. Our approach to data collection includes the following aspects:

- 1 **Wallet Authentication:** Users authenticate their blockchain wallets using the WalletConnect protocol to initiate the registration process. This authentication mechanism verifies a user's identity directly through their wallet address, eliminating the need for traditional registration forms and disclosure of personal information.
- 2 **Transaction Details:** Users could be asked to sign transactions as part of the authentication procedure in order to verify their identity and provide access to the application. These transactions are safely executed on the blockchain and transaction details are stored for simple verification.
- 3 **User Preferences:** While the main goal of WalletConnect authentication is to confirm the user's identity, our software is still able to record user settings that are associated with its features and settings. These preferences can be encrypted and kept in the blockchain for later use, or they can be saved locally on the user's device.
- 4 **Consent and Privacy Measures:** Users are given with concise consent forms outlining the data access permissions that WalletConnect needs before the authentication process starts. Users maintain complete control over the permissions they grant, guaranteeing adherence to privacy regulations and

upholding user autonomy.

- 5 Security Considerations: To protect user authentication and transaction signing procedures, WalletConnect makes use of robust encryption and cryptographic technologies. This guarantees that private keys and wallet addresses—among other critical user data—remain shielded from unwanted access and eavesdropping.

Our application streamlines the registration process while protecting users' privacy and security by utilizing WalletConnect for user registration and authentication. By adhering to the tenets of blockchain technology, this decentralized method gives people control over their digital identities and private information.

2.4.4 Use-Case Diagram

A use case diagram is a type of UML diagram that captures the functional requirements of a system by illustrating the interactions between users (actors) and various use cases (system functionalities). For the crowdfunding platform, the use case diagram helps to visualize the primary activities that users and administrators can perform and how these activities interact with the system.

Key Components of the Use Case Diagram

Actors:

- 1 User:
 - Represents the end-users who interact with the platform to create projects, view projects, manage their profiles, and contribute to projects.
- 2 Admin:
 - Represents the administrators who manage the projects, approve or reject new projects, and oversee the platform operations.
- 3 Blockchain System:
 - Represents the underlying blockchain infrastructure that verifies transactions and ensures the security and integrity of the data.

Use Cases:

- 1 Create Project:
 - Allows users to create new crowdfunding campaigns by providing necessary details such as project title, description, funding goal, and deadline.
- 2 View Project:
 - Enables users to browse and view the details of existing crowdfunding projects.
- 3 Manage User Profile:
 - Allows users to update their personal information, preferences, and account settings.
- 4 Contribute Project:

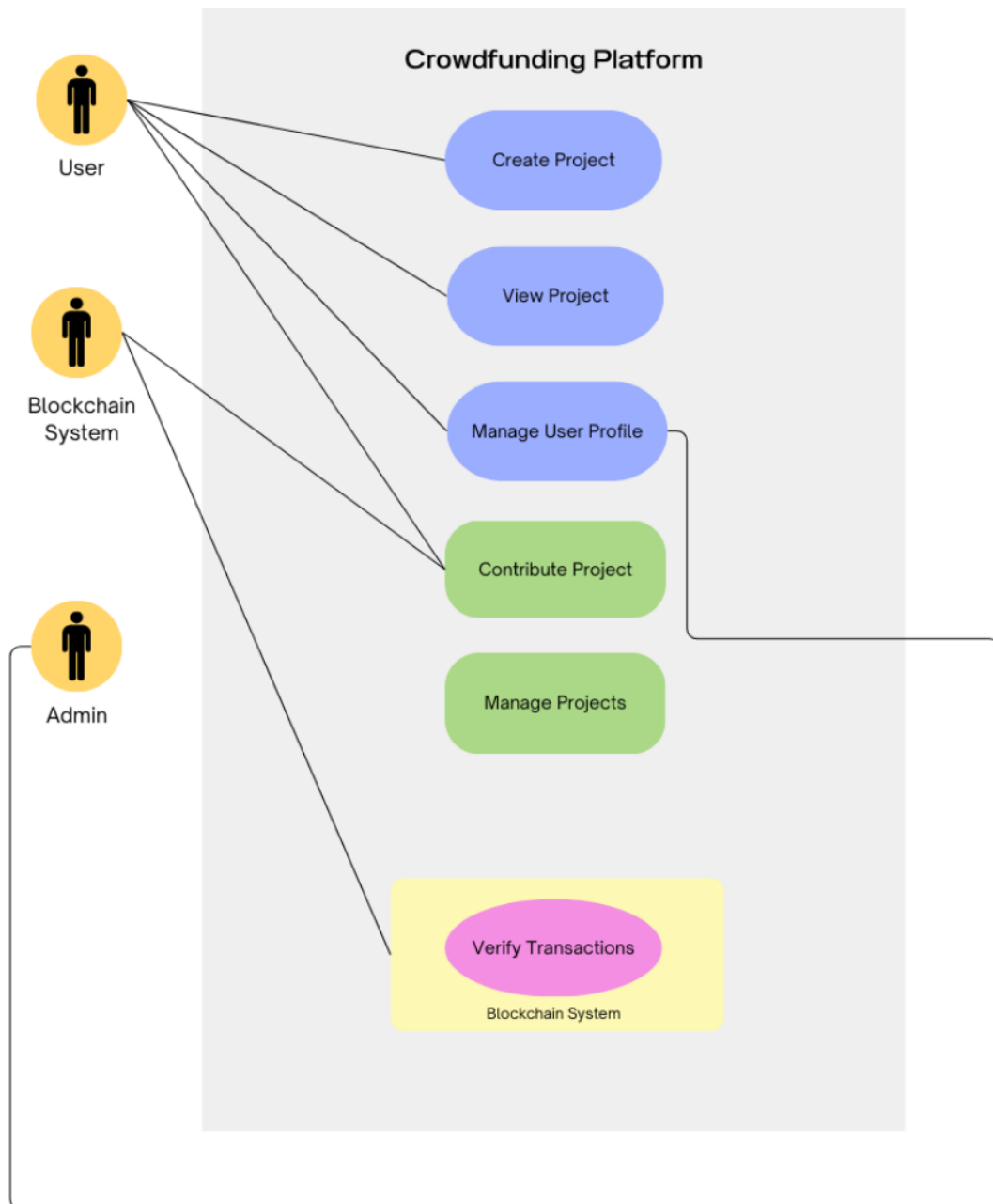


Figure 2.4 – UML Use-Case Diagram

- Facilitates users in contributing funds to a crowdfunding project. This involves transferring funds to the project's smart contract on the blockchain.
- 5 Manage Projects:
 - Allows users to update their personal information, preferences, and account settings.
- 6 Verify Transactions:
 - Handled by the blockchain system, this use case involves verifying and recording transactions to ensure their integrity and immutability.

Detailed Description

- **Create Project:**
 - Actor: User
 - Description: The user fills out a form specifying the project details, which is then sent to the backend. The backend interacts with the blockchain to create a new smart contract for the project. The project is subsequently listed on the platform for other users to view and contribute to.
- **View Project:**
 - Actor: User
 - Description: Users can browse through a list of projects and select one to view its details. The details are retrieved from the backend, which fetches data from the blockchain.
- **Manage User Profile:**
 - Actor: User
 - Description: Users can update their personal information and account settings. These changes are processed by the backend and updated in the platform's database.
- **Contribute Project:**
 - Actor: User
 - Description: Users select a project to contribute to and specify the amount. The transaction is initiated through the user's cryptocurrency wallet. The blockchain verifies and processes the transaction, ensuring that the funds are securely transferred to the project's smart contract.
- **Manage Projects:**
 - Actor: Admin
 - Description: Administrators have access to a dashboard where they can view and manage all projects. They can approve new projects, monitor ongoing campaigns, and handle any issues that arise. Admin actions are recorded on the blockchain for transparency.
- **Verify Transactions:**
 - Actor: Blockchain System
 - Description: Every transaction is verified by the blockchain system to ensure its integrity. The immutable nature of blockchain ensures that all contributions are recorded transparently and cannot be altered.

Explanation of the Diagram

- **User:**
 - Interacts with the following use cases: Create Project, View Project, Manage User Profile, and Contribute Project.
 - These interactions are depicted by lines connecting the User actor to the respective use cases.
- **Admin:**

- Interacts with the Manage Projects use case.
- This interaction is depicted by a line connecting the Admin actor to the Manage Projects use case.

- **Blockchain System:**

- Interacts with the Contribute Project and Verify Transactions use cases.
- These interactions are depicted by lines connecting the Blockchain System actor to the respective use cases.

The use case diagram effectively illustrates the various functionalities provided by the crowdfunding platform and the interactions between the system and its users. This high-level overview helps in understanding the scope of the system and the different roles played by each actor.

2.4.5 Class Diagram

User Class:

- **Attributes:**

- id: int - A unique identifier for the user.
- name: string - The name of the user.
- email: string - The email address of the user.
- password: string - The password for user authentication..

- **Methods:**

- createProject(): Method to create a new crowdfunding project.
- viewProject(): Method to view existing projects.
- manageProfile(): Method to manage user profile information.
- contributeToProject(): Method to contribute funds to a project.

Admin Class:

- **Methods:**

- approveProject(): Method to approve new projects.
- manageProjects(): Method to manage all projects on the platform.

Project Class:

- **Attributes:**

- id: int - A unique identifier for the project.
- title: string - The title of the project.
- description: string - A brief description of the project
- targetAmount: float - The funding goal for the project.
- collectedAmount: float - The amount of funds collected so far.
- deadline: date - The deadline for the project to reach its funding goal.

- **Methods:**

- getDetails(): Method to get the details of the project.
- addContribution(): Method to add a contribution to the project.

Relationships

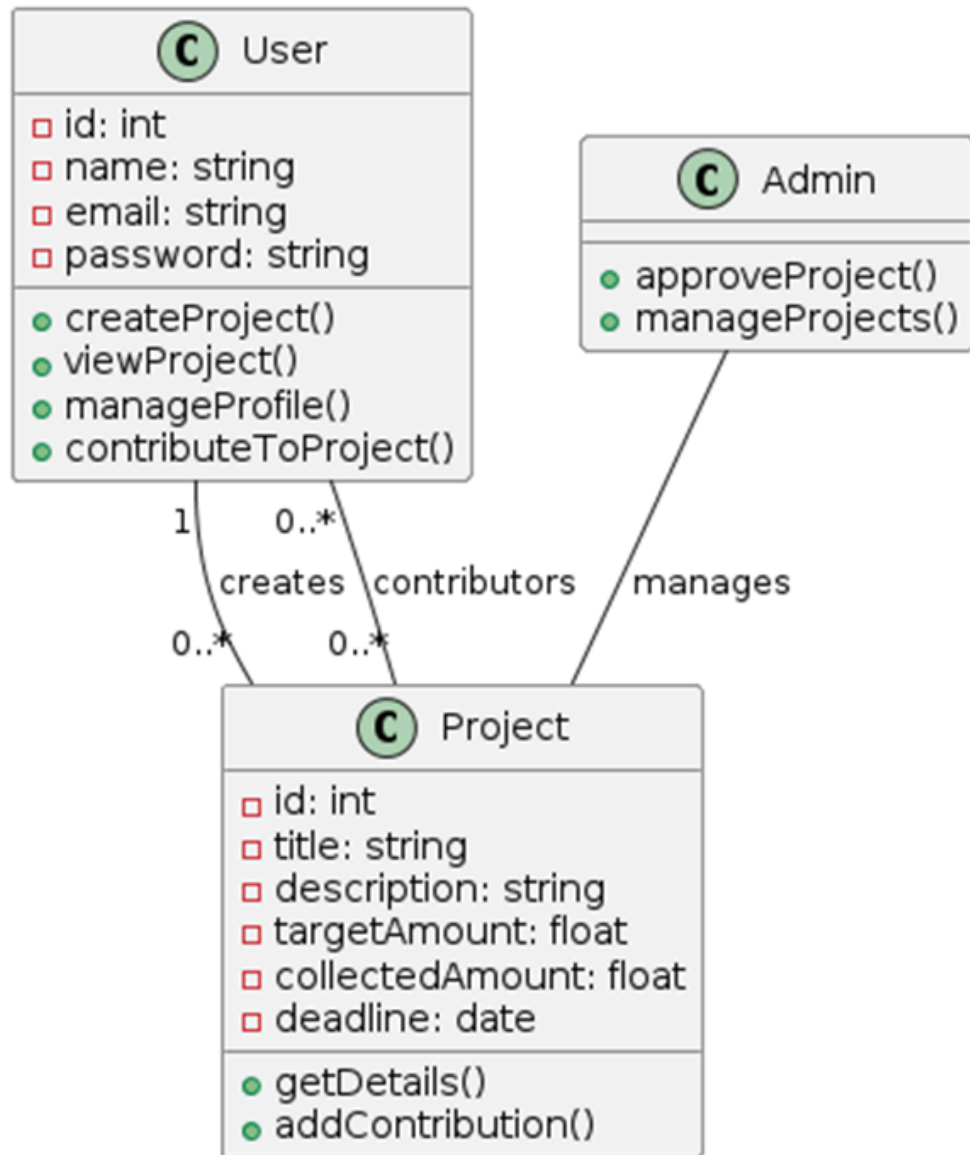


Figure 2.5 – UML Class Diagram

- **User - Project:**
- A user can create multiple projects (1 to 0..* relationship).
- A project can have multiple contributors, which are users (0..* to 0..* relationship).

- **Admin - Project:**

- An admin manages multiple projects (manages relationship).

Detailed Description, User Class:

Attributes:

- `id: int` - A unique identifier for the user.
- `name: string` - The name of the user.
- `email: string` - The email address of the user.
- `password: string` - The password for user authentication.

Methods:

- createProject(): This allows users to initiate new projects on the platform.
- viewProject(): Users can browse through various projects listed on the platform.
- manageProfile(): This allows users to update their personal details and account settings.
- contributeToProject(): Users can support projects by donating money.

Admin Class:**Methods:**

- approveProject(): Admins have the authority to review and approve projects before they go live.
- manageProjects(): Admins oversee the entire lifecycle of projects, ensuring compliance and resolving any issues.

Project Class:**Attributes:**

- id: int - A unique identifier for the project.
- title: string - The name by which the project is identified.
- description: string - This provides details about the project's goals and purpose.
- targetAmount: float - The amount of money the project aims to raise.
- collectedAmount: float - This tracks the progress towards the funding goal.
- deadline: date - Projects have a time limit to achieve their target.

Methods:

- getDetails(): Users can view comprehensive information about the project.
- addContribution(): This updates the collected amount and records the contribution

In summary, this class diagram provides a clear and detailed view of the main entities involved in the crowdfunding platform and their interactions. By understanding the attributes and methods of each class, as well as the relationships between them, developers and stakeholders can gain a better understanding of the system's design and functionality. This diagram serves as a blueprint for implementing the platform's features and ensuring a robust and efficient system.

2.5 Development Environment Setup

To develop the crowdfunding mobile application, we need to set up a robust and efficient development environment. This section outlines the tools and technologies used, along with detailed steps for setting up the development environment. The setup process involves installing and configuring Node.js, Expo CLI, and other necessary tools to ensure a smooth development workflow.

2.5.1 Description of the Tools and Technologies Used

- Node.js: Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server side and is essential for managing the development environment. Node.js is also used to run various build tools and scripts. [22]

- npm (Node Package Manager): npm is the default package manager for Node.js. It is used to install, manage, and update packages (libraries and tools) needed for the project. npm helps in managing project dependencies efficiently. [22]

- Expo: Expo is an open-source platform for making universal native apps that run on Android, iOS, and the web. It includes a universal runtime and libraries that let you build native apps by writing React and JavaScript. [23]

- React Native: React Native is a framework for building native apps using React, a JavaScript library for building user interfaces. It allows developers to create mobile applications that run on both iOS and Android from a single codebase. [24]

- Solidity: Solidity is a statically-typed programming language designed for developing smart contracts on the Ethereum blockchain. It is used to implement the backend logic for decentralized applications. [25]

- WalletConnect: WalletConnect is an open protocol for connecting decentralized applications to mobile wallets using QR code scanning or deep linking. It ensures secure connections and transactions between the app and users' cryptocurrency wallets. [26]

- Wagmi: Wagmi is a library for Ethereum that simplifies interactions with smart contracts. It provides tools for connecting to wallets, reading data from smart contracts, and managing transactions. [27]

2.5.2 Steps for Setting Up the Development Environment

Step 1: Install Node.js and npm

Step 2: Create a new expo project

Run the create-expo-app command:

Open a terminal or command prompt.

Run the following command to create a new Expo project named CrowdfundingApp

This command initializes a new Expo project and prompts you to choose a template.

Choose the Navigation (TypeScript) Template:

After running the command, you will see a prompt to choose a template. Use the arrow keys to navigate to the "Navigation (TypeScript)" option and press Enter to select it.

Navigate to the Project Directory:

```
create-expo-app x + v
PS C:\Users\Narbe\OneDrive\Рабочий стол\App> npx create-expo-app CrowdfundingApp --template
? Choose a template: » - Use arrow-keys. Return to submit.
> Default - includes tools recommended for most app developers
  Blank
? Choose a template: » - Use arrow-keys. Return to submit.
  Default
  Blank
  Blank (TypeScript)
> Navigation (TypeScript)
  File-based routing with TypeScript enabled
  Blank (Bare)
```

Figure 2.6 – Create a new expo project

```
added 1538 packages, and audited 1539 packages in 2m

146 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

✔ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd CrowdfundingApp
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you need to do iOS development without a Mac
- npm run web
```

Figure 2.7 – Installed vulnerabilities

Once the project is created, navigate to the project directory

```
PS C:\Users\Narbe\OneDrive\Рабочий стол\App> cd CrowdfundingApp
PS C:\Users\Narbe\OneDrive\Рабочий стол\App\CrowdfundingApp>
```

Figure 2.8 – Navigate to the Project Directory

Start the Development Server:

Start the Expo development server to begin working on your project. This will open a new tab in your browser with the Expo DevTools, where you can see the status of your project and scan the QR code to run the app on a physical device using the Expo Go app.

After running `npx expo start`, the Expo DevTools will launch in your default web browser. You can scan the displayed QR code with the Expo Go app on your mobile device to preview the application in real-time.

Step 3: Install additional dependencies:

In the quest to be in a position to make secure wallet connections and be in a position to interact with the Ethereum blockchain from the crowdfunding mobile application, it is necessary to install some specific blockchain integration libraries. The steps that follow are intended to serve as a guide in the process of installing these dependencies through the use of wagmi, @web3modal/wagmi-react-native, and viem. Installation of Blockchain Integration Libraries Blockchain Libraries for Integration: We will also implement wagmi, @web3modal/wagmi-react-native,

```
PS C:\Users\Narbe\OneDrive\Рабочий стол\App\CrowdfundingApp> npx expo start
Starting project at C:\Users\Narbe\OneDrive\Рабочий стол\App\CrowdfundingApp
(node:10520) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
Starting Metro Bundler



> Metro waiting on exp://10.200.2.12:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Web is waiting on http://localhost:8081

> Using Expo Go
> Press s | switch to development build

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> Press o | open project code in your editor

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```

Figure 2.9 – Development Server

and viem for safe connections to wallets and communication with the Ethereum blockchain.

Step 4:Configuring WalletConnect and Wagmi:

- 1 Get project ID at <https://cloud.walletconnect.com>
- 2 Create config
- 3 Create modal

2.5.3 Frontend Development with Expo and React Native

In the case of crowdfunding mobile application frontend development, the aim is to develop the screens of the application and navigation among the screens of the application is to be set. React Native with the help of Expo, is providing a very solid groundwork to develop mobile applications for both platforms through one source code.

Key areas in Frontend Development

Developing the Project:

Start by creating a new Expo project with the Navigation (TypeScript) template. The template includes all the dependencies we need and prepares the structure for building a React Native application with support for TypeScript and navigation included.

Designing the Main Screens:

Project List Screen: On this screen, all the listed crowdfunding projects are displayed. The screen is made in such a way that it gives a snapshot of each project, its name, and a brief description along with it. One can go through the projects, and one of those can be clicked to view that particular project in detail.

Create Project Screen: This screen is used to create a new crowdfunding project. It has several input fields for the project title, description, fund target and other necessary information. The users can fill out the form and submit it to create a new project.

User Profile Management Screen: It is the screen that provides the user with an opportunity to manage his or her profile information. The user is able to edit his or her name and email address. A feature is given to edit the details that are in the user profile and create the changes.

Providing Navigation: Make navigation between the various screens using React Navigation. Navigation is a key aspect of any mobile application that enables the users to move conveniently between the various sections of the application. The Navigation (TypeScript) template has included support for React Navigation to help you set up and manage the navigation within your application in a more convenient way.

Designing the Screens: Always have uniform styling of the screens to have a similar look and feel all through the application. Have defined styles for the various UI components such as buttons, text inputs, and containers by making use of the built-in styling of React Native.

Managing State:

Use state management for managing the dynamic data in the application. Any of the state management libraries, like Redux or Context API can be used to manage the state of the application in efficient ways. That will make the data shared and updated within a scope of different parts of the application.

Establish a Link to Backend Services:

Integrate the frontend with the backend services to fetch and display data from the server. This includes making API calls to fetch project information, submit new projects, and update user profiles. This complete integration makes the application dynamic in nature and responsive to user interactions.

Testing and Debugging: Test the application thoroughly in devices and various screen sizes to ensure the users have a seamless experience. Make use of the development tools and simulators integrated with Expo to test the application in various use cases. Debug the issues that come up during testing of the application to ensure that the application responds as desired. Optimization of Performance Optimize the application performance by making it so that it does not re-render unnecessarily, compress the assets, and optimize network requests. Through performance optimization, it is possible to perfectly and smoothly run

the application on all the devices. Summary Crowdfunding mobile application frontend development includes project setup, design of the major screen, addition of navigation, and integration with backend services. While doing these processes and keeping in mind the user experience, state management, and performance tuning, it becomes an option to develop a great and user-satisfying application that fulfills the requirements of its users.

2.5.4 Smart Contract Development with Solidity

The backend for the Crowdfunding mobile app is coded using smart contracts that are coded using Solidity and deployed on the Ethereum blockchain. The smart contract was developed through the use of the Remix IDE and deployed on the Sepolia testnet to ensure that the deployment cost is within a test environment where the cost is controllable.

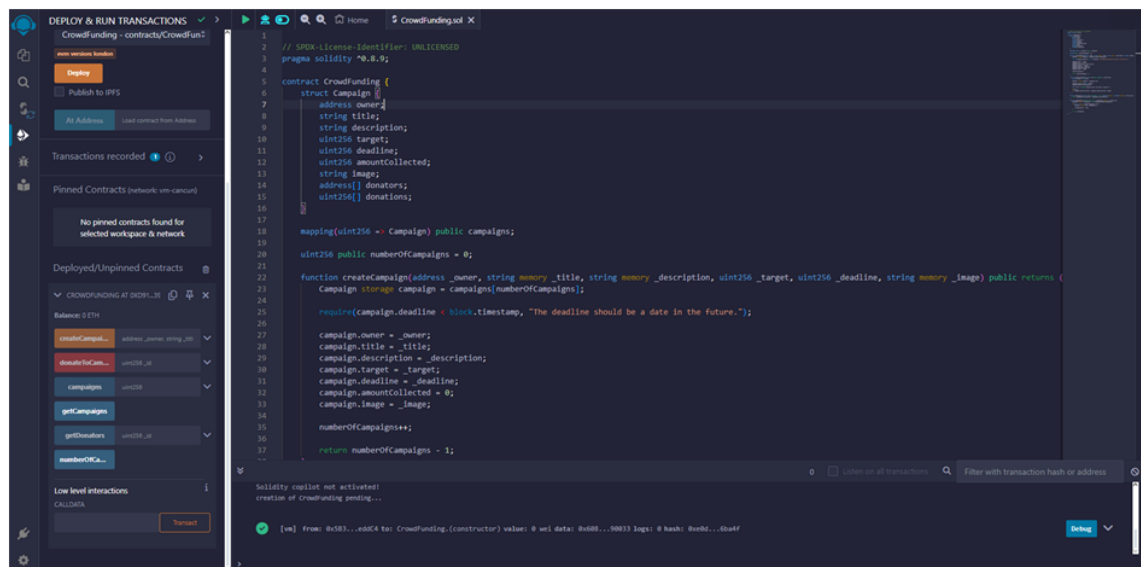


Figure 2.10 – Smart Contract on Remix IDE

Overview of the Smart Contract

The smart contract of the crowdfunding application is the coordinator of the campaign creation, the maintenance of the contribution recording, and the processing of the fund distribution. The contract has several most important functions and data structures:

- 1 Campaign Struct: It defines the schema of a crowdfunding campaign, which includes various properties such as owner, title, description, target amount, deadline, amount collected, and lists of donators and donations.
- 2 Functions:
 - createCampaign: Can create a new crowdfunding campaign passing the details like the owner, title, description, target amount, deadline, and an image.
 - donateToCampaign: Allows the user to send Ether to a particular

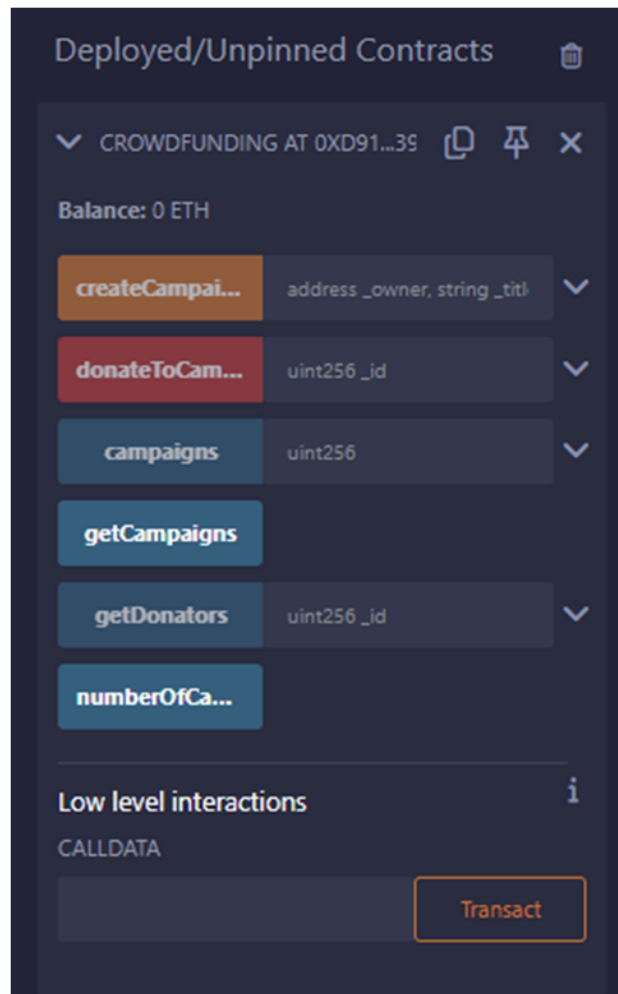


Figure 2.11 – Deployed Smart Contract

campaign and thereby donate money. The function updates the collection of the campaign and maintains the address of the donator along with the amount of donation offered.

- `getDonators`: It returns the list of donators and their associated donations for a specified campaign, thereby keeping transparency intact in the donation process.

- `getCampaigns`: Fetches all the campaigns that are created on the platform through which the user can go through and select the campaigns that they want to fund.

Key Features and Logic

- 1 Campaign Creation: The `createCampaign` function is used to instantiate a new campaign with provided information. Also, it ensures that the campaign deadline is a future date so that the expired campaign is not created. This function registers all the details about the campaign, including the owner, title, description, target amount, deadline, and image.
- 2 Handling Contributions: The `donateToCampaign` function allows the users to

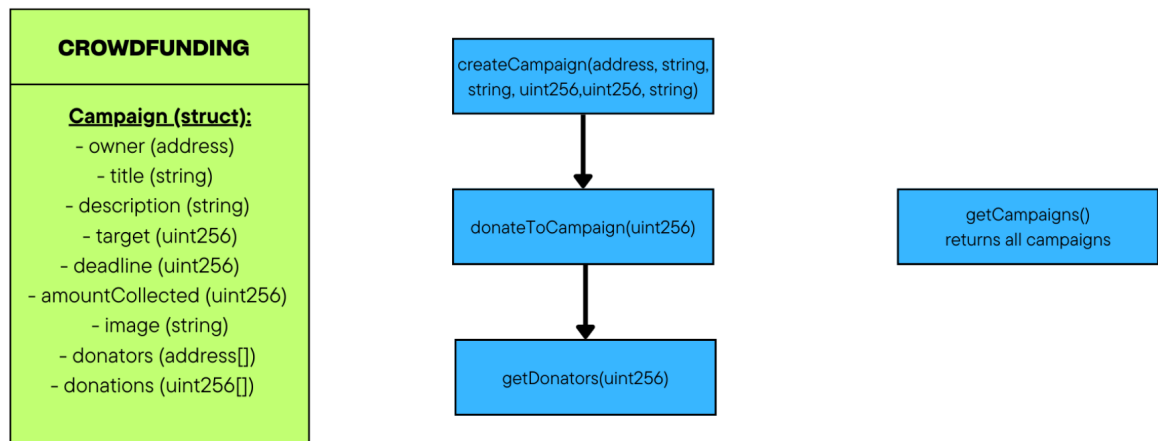


Figure 2.12 – Smart Contract Structure

donate to a campaign in Ether. This function updates the total fund collected by the campaign and stores all the donations by storing the address of the donator and the amount donated. It also helps in enabling the safe transfer of funds to the owner of the campaign.

- 3 Data Retrieval: The getDonators function enables getting access to the list of the addresses and the amount of contribution of a particular campaign, which gives openness and accountability in the process of fundraising. getCampaigns is an endpoint that allows the retrieval of all the campaigns that are there on the platform, and the users can see and go through the various options of fundraising available to them.
- 4 Security Measures: The smart contract also does some security check in an effort to prevent some common security vulnerabilities. For example, it checks that the campaign end date is a date in the future when a campaign is being created. This can help prevent corruption of the crowdfunding process and make valid campaign setup possible.

Sepolia Testnet deployment

The smart contract is deployed on the Sepolia testnet, which is a public network for testing on Ethereum so that developers can deploy the smart contract in order to test it anywhere without incurring any cost of deploying it on the Ethereum main network. Sepolia testnet is a network environment that is very close to the mainnet, whereby it ensures that the way things are handled in regards to the contract is well tested before receiving any kind of deployment on the mainnet.

Advantages of Using Sepolia Testnet:

- Low-Cost Testing: There is a whole lot of testing and iteration you can

afford, as the testnet doesn't cost real Ether for transactions.

- Mainnet-Like Environment: The Sepolia testnet is an accurate simulation of the mainnet environment, and the contract deployed on the main network will behave as desired. Risk Mitigation: Likely contract issues can be identified and mitigated while testing on the testnet before deployment on the mainnet, and consequently, the risks of bugs and vulnerabilities can be reduced.

Key Components and Functions:

- 1 Campaign Management: It will enable users to create and update crowdfunding campaigns. This campaign is identified by ID and contains all the information about the project, such as the address of the owner, title of the project, description, funding goal, deadline, and an optional image URL.
- 2 Donation Processing: Users can participate in the campaigns by sending the Ether to the `donateToCampaign` function. The contract logs each donation, updates the total amount collected, and ensures the full amount that has been sent to the owner of the campaign is sent securely.
- 3 Accountability and Transparency: The contract has the capability to get in-depth information regarding campaigns and contributions. The `getDonators` functionality retrieves details of all the donors and the amount donated for any specific campaign, and the `getCampaigns` functionality retrieves the list of all the campaigns currently running on the site. This kind of openness helps in creating trust among the users, and hence, accountability also comes in the scenario.
- 4 Security and Validation :The use of a smart contract also has inbuilt security measures to validate inputs that shall see to it that the integrity of the fundraising process is maintained. For instance, the `createCampaign` function also checks that the deadline for the campaign is in the future, so it is not possible to create invalid campaigns. The contract also makes use of secure means of transferring the funds so that reentrancy attacks and other vulnerabilities can be eliminated.

The crowdfunding application Smart Contract has the most basic functionalities provided for the administration of campaigns, contributions, and visibility features. Through the use of Solidity and deployment on the Sepolia testnet, the application takes the maximum advantage of the Ethereum blockchain security and immutability properties to come up with a secure and strong crowdfunding platform. This setup allows for adequate testing and validation before any potential mainnet deployment, ensuring that the contract will securely function as intended.

2.5.5 Integration with Wallets Using WalletConnect and Wagmi

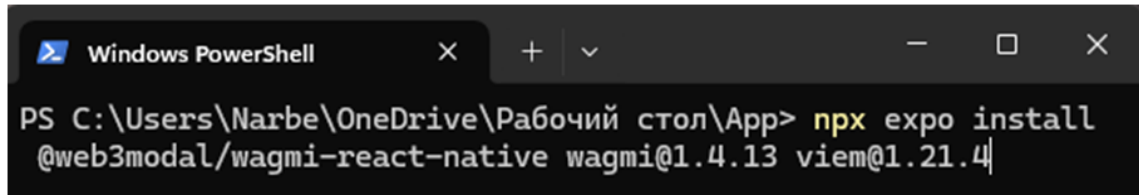
Integrating wallets into the crowdfunding mobile application involves setting up WalletConnect for secure wallet connections and using Wagmi for blockchain

interactions. This integration allows users to connect their cryptocurrency wallets to the application, enabling them to perform transactions securely.

Setting Up WalletConnect

WalletConnect is an open protocol that enables mobile applications to connect to cryptocurrency wallets using QR code scanning or deep linking. This ensures secure and seamless interactions between the app and users' wallets.

- 1 Install WalletConnect libraries: Use the following command to install the required WalletConnect and Wagmi libraries:



```
Windows PowerShell
PS C:\Users\Narbe\OneDrive\Рабочий стол\App> npm expo install @web3modal/wagmi-react-native wagmi@1.4.13 viem@1.21.4
```

Figure 2.13 – Installing WalletConnect and Wagmi Libraries

- 2 Configure WalletConnect: Create a configuration for WalletConnect in your project to set up the connection with users' wallets.

Using Wagmi for Blockchain Interactions

Wagmi is a library that simplifies interactions with the Ethereum blockchain. It provides tools to connect to wallets, read data from smart contracts, and manage transactions efficiently.

- Initialize Wagmi Client: Set up the Wagmi client to handle blockchain interactions. This involves configuring providers and connectors for the Ethereum network.

Example Code for Connecting Wallets and Performing Transactions

Below are examples of how to connect wallets using WalletConnect and perform transactions using Wagmi:

Integrating WalletConnect and Wagmi into the crowdfunding mobile application enables secure wallet connections and efficient blockchain interactions. By following the steps outlined above and using the provided example code, you can set up WalletConnect for wallet connections and use Wagmi to handle transactions and interactions with the Ethereum blockchain. This integration enhances the functionality of your application, providing users with a secure and seamless experience.

2.5.6 Security Measures

Ensuring the security of the smart contracts and user data is paramount in the development of a crowdfunding mobile application. This section outlines the

```

1  import { useWalletConnect } from '@web3modal/wagmi-react-native';
2  import { configureChains, createClient, WagmiConfig } from 'wagmi';
3  import { sepolia, mainnet } from 'wagmi/chains';
4  import { publicProvider } from 'wagmi/providers/public';
5
6  const { chains, provider } = configureChains(
7    [sepolia, mainnet],
8    [publicProvider()]
9  );
10
11  const client = createClient({
12    autoConnect: true,
13    provider,
14  });
15
16
17  export const WalletProvider = ({ children }) => (
18    <WagmiConfig client={client}>{children}</WagmiConfig>
19  );

```

Figure 2.14 – Connecting Wallets

measures implemented to secure smart contracts, practices for protecting user data and transactions, and examples of security tests and their results.

Measures Implemented to Ensure the Security of Smart Contracts

- 1 Input Validation: Validate all inputs to smart contract functions to ensure they meet expected formats and ranges. For example, ensuring that deadlines are set in the future and that the target amount is a positive number.
- 2 Access Control: Implement access control mechanisms to restrict access to sensitive functions. For instance, only the owner of a campaign should be able to modify its details or withdraw funds.
- 3 Reentrancy Guards: Use reentrancy guards to prevent reentrancy attacks, where an external contract makes recursive calls back into the contract before the initial execution is complete. This is achieved using the `nonReentrant` modifier from the OpenZeppelin library.
- 4 SafeMath for Arithmetic Operations: Utilize SafeMath library for all arithmetic operations to prevent overflows and underflows.
- 5 Event Logging: Emit events for significant actions such as campaign creation, donations, and fund withdrawals. This provides an audit trail and helps in tracking and debugging.
- 6 Fail-Safe Mechanisms: Implement fail-safe mechanisms such as circuit breakers that can pause contract operations in case of detected anomalies or security breaches.

Security Practices for Protecting User Data and Transactions

- 1 Encryption: Use encryption to protect sensitive user data both in transit and at

rest. Data exchanged between the frontend and backend should be encrypted using SSL/TLS.

- 2 Secure Wallet Connections: Utilize WalletConnect for secure connections between the application and users' wallets. This ensures that private keys are never exposed to the application.
- 3 Secure Communication: Ensure all communications between the application, blockchain, and backend services are secured using industry-standard encryption protocols.
- 4 Authentication and Authorization: Implement robust authentication and authorization mechanisms to ensure that only authorized users can perform certain actions. For example, only the campaign owner should be able to withdraw funds from a campaign.
- 5 Regular Security Audits: Conduct regular security audits of the smart contracts and the application codebase. Use both automated tools and manual reviews to identify and mitigate vulnerabilities.
- 6 User Education: Educate users on best practices for securing their wallets and protecting their private keys. Provide guidance on recognizing phishing attempts and other common threats.

Examples of Security Tests and Their Results

- 1 Unit Tests: Implement comprehensive unit tests for all smart contract functions to ensure they behave as expected under various conditions. For example, tests can verify that only valid inputs are accepted and that unauthorized access attempts are rejected.
- 2 Integration Tests: Perform integration tests to validate the interaction between smart contracts and the frontend application. These tests ensure that data flows correctly and that security measures are enforced across the entire system.
- 3 Static Analysis: Use static analysis tools such as MythX or Slither to analyze the smart contract code for vulnerabilities. These tools can detect common issues like reentrancy, integer overflows, and access control flaws.
- 4 Manual Code Reviews: Conduct manual code reviews by experienced developers to identify potential security issues that automated tools might miss. Reviewers check for logic errors, insecure coding practices, and adherence to best practices.

Example Security Test Results:

- 1 Test: Reentrancy Attack Prevention:

Description: Ensure that the contract is protected against reentrancy attacks.

Result: Passed. The nonReentrant modifier was successfully applied to all functions that perform external calls.

2 Test: Access Control Enforcement

Description: Verify that only authorized users can call restricted functions.
Result: Passed. Unauthorized access attempts were correctly rejected, and only the campaign owner could withdraw funds.

3 Test: Input Validation:

Description: Ensure that all inputs to smart contract functions are validated.

Result: Passed. Invalid inputs were rejected, and appropriate error messages were returned.

4 Test: SafeMath Operations

Description: Verify that arithmetic operations use SafeMath to prevent overflows and underflows.

Result: Passed. All arithmetic operations were safe, and no overflows or underflows occurred.

Implementing robust security measures is crucial for the safe and reliable operation of the crowdfunding mobile application. By validating inputs, enforcing access controls, using reentrancy guards, and encrypting sensitive data, the application ensures the security of smart contracts and user transactions. Regular security audits and comprehensive testing further enhance the security posture, protecting the application and its users from potential threats.

2.5.7 Results and Achievements

The development of the crowdfunding mobile application has culminated in the successful implementation of several key features and functionalities. This section highlights the completed components of the application, provides a demonstration of its working state, and includes any available user feedback and metrics.

Description of the Completed Features and Functionalities

WalletConnect-Based Registration and Authorization:

- Users can register and log in to the application using WalletConnect, which eliminates the need for traditional email and password authentication.
- Secure wallet connections ensure that user data and funds are protected.

Project Creation:

- Users can create new crowdfunding projects by providing necessary details such as title, description, target amount, deadline, and an optional image.
- Projects are stored on the Ethereum blockchain, ensuring transparency and immutability.

Project Listing and Browsing:

- A comprehensive list of all active crowdfunding projects is displayed to users.
- Users can browse through projects, view detailed information, and decide which projects to support.

Fund Contribution:

- Users can contribute funds to their chosen projects using their connected cryptocurrency wallets.

- The application supports secure wallet connections through WalletConnect and manages transactions using Wagmi.

User Profile Management:

- Users can view and update their profile information, including their wallet address and display name.

- Profile management is integrated with secure authentication mechanisms.

Transaction History and Transparency:

- Detailed records of all contributions are maintained and can be viewed by users.

- The application provides transparency by showing the list of donators and their contributions for each project.

Security and Data Integrity:

- Smart contracts are secured with input validation, access control, reentrancy guards, and other best practices.

- User data is protected using encryption and secure communication protocols.

3 Conclusion

In conclusion, developing the blockchain-powered mobile crowdfunding app is an arduous yet extremely rewarding project due to the new potential it will unlock through decentralized platforms, bringing security, transparency, and efficiency to fundraising. In this project, these goals were addressed both theoretically and practically through studies, resulting in a well-performing mobile application that effectively resolves several issues associated with traditional crowdfunding platforms.

It unveils the theory of the significant conceptions of blockchain technology and smart contracts in general and its application in crowdfunding. At the same time, it ensures robust solutions by using its immutable and verifiable ledger, reassuring the clarity of paramount importance of transparency, safety, and decentralization in modern financial systems. This relates to the problems of fraud, lack of transparency, and inefficiencies faced on traditional crowdfunding platforms. Such groundwork laid away on how the practical development of the application would look and could guide design and implementation into a secured and reliable app.

The pragmatic project implementation involves developing the crowdfunding application using React Native and Expo for the front end by providing an intuitively flawless user interface. The user could make project creation, browser projects, edit profiles, and the funding contributor. It ensures that every wallet connection facilitated by WalletConnect and Wagmi retains a user's registration and logged-in state; one can perform any action using only their crypto wallet to execute a transaction, no longer needing traditional email and password auth. It provides functionalities including campaign creation, fund contribution, and data retrieval by a back-end interacting with intelligent contracts developed in Solidity and deployed in the Ethereum blockchain to ensure transparency and safety.

All possible measures regarding user data and transaction safety have been taken, including input validation, access control, reentrancy guards, and encryption. Measures in the direction of input validation, access control, reentrancy guards, and encryption have been thoroughly implemented. Now, the proper working and safety of the front end and smart contracts will be examined through extensive testing by different methods, some of which are: unit tests, integration tests, static analyzers, and manual code reviews. User reviews are now starting to be posted, offering evidence about user-friendliness and safety features. Well-accepted metrics already indicate that the user base is steadily growing, with the majority successfully using the platform for crowdfunding.

The appropriateness of the project is, in fact, evident in the attainment of its goals: securing, making transparent, and, most of all, making a user-friendly platform for crowdfunding. Blockchain technology ensures verifiability

and tamper-proofing of every deal, guaranteeing immutability of the process against fraud and a lack of transparency. The solution urged on the feedback of a user-friendly interface and promised secure integration of a wallet, performed to be visible and attractive in practice.

Now is the time when the application will be deployed to the masses. This could be achieved with a few steps. The first would be aggressive marketing and user education to ensure that potential users know the benefits and utilization of blockchain technology and WalletConnect. Then, it is through the subsequent partnerships with the right stakeholders to create the necessary resilience and agility. Thirdly, support would be needed to give technical aid in case of issues so that the user experience is smooth.

Given below are the areas in which the proposed solution can be enhanced in developing features like project updates, social sharing, and advanced search features. There will be an enormous increase in user interaction and user satisfaction. This process will make the application scale better with many more users and far beyond the current number of projects as the application scales up. It is also a step toward enhancing user experience: full compatibility, and the expected performance and working of an application on any device or operation system. Regular scheduled security audits and updating protect against new threats and vulnerabilities. Furthermore, educating the user on the technology behind the blockchain and the safety within the wallet and best practices for the user can be used to inform them about the proper action to be taken in the platform.

In other words, this diploma project here presents the vibrancy of blockchain technology is revolutionizing the olden structure of crowdfunding, making it more credible and transparent among project creators and backers. Decentralized applications appear to present their real potential and demonstrate their independent capability upon proven principles and experience acquired from this project related to the actual critical state of modern financial systems. Therefore, the project may act as a preliminary base for further research and development of scientists in this fast-changing area of blockchain technologies enabled by the design of mechanisms for its application in many places to provide better security, transparency, and efficiency. With more improvements and strategic implementations, this fundraising application is set to be mind-blowing in the industry.

BIBLIOGRAPHY

- 1 Yadav. Venturing Crowdfunding using Smart Contracts in Blockchain / Yadav, Nikhil, Sarasvathi V. // *IEEE Xplore*. — 2020. — Pp. 1–6. <https://ieeexplore.ieee.org/document/9214295>.
- 2 Omrani. The Economics of Crowdfunding Platforms / Omrani, Nessrine // *SSRN Electronic Journal*. — 2015. — Pp. 1–52. https://www.researchgate.net/publication/315435388_The_Economics_of_Crowdfunding_Platforms.
- 3 Trusted Crowdfunding Platform using Smart Contracts / Tejashwini P S, B Gowda, Anshul Shukla et al. // *IEEE Xplore*. — 2023. — Pp. 1–7. <https://ieeexplore.ieee.org/document/10234998>.
- 4 Dr. Poornima G. Naika, Dr. Kavita S. Ozab. Leveraging the Power of Blockchain Technology for Building a Resilient Crowdfunding Solution / Dr. Kavita S. Ozab Dr. Poornima G. Naika // *ScienceDirect*. — 2023. — Pp. 11–20. <https://www.sciencedirect.com/science/article/pii/S1877050923020616>.
- 5 The role of blockchain technology-based social crowdfunding in advancing social value creation / Loan T.Q. Nguyen, Thinh G. Hoang, Linh H. Do et al. // *Technological Forecasting Social Change*. — 2021. — Vol. 170. — Pp. 1–12.
- 6 Zheng. Value Drivers of Blockchain Technology: A Case Study of Blockchain-Enabled Online Community / Zheng, Yujie // *Telematics and Informatics*. — 2021. — Vol. 58. — Pp. 1–17.
- 7 Lee, Winson. Decentralized Application for Charity Organization Crowdfunding using Smart Contract and Blockchain / Winson Lee, Hong Yee, Nordiana Rahim // *Applied Information Technology And Computer Science*. — 2021. — Vol. 2, no. 2. — Pp. 1–13. <https://publisher.uthm.edu.my/periodicals/index.php/aitcs/article/view/220>.
- 8 Nik Azlina Nik Ahmad, Syed Abdul Halim Syed Abdul Rahman. Applying Ethereum Smart Contracts to Blockchain-Based Crowdfunding System to Increase Trust and Information Symmetry / Syed Abdul Halim Syed Abdul Rahman Nik Azlina Nik Ahmad // *ICCTA '21: Proceedings of the 2021 7th International Conference on Computer Technology Applications*. — 2021. — Pp. 53–59. <https://dl.acm.org/doi/abs/10.1145/3477911.3477920>.
- 9 Wenhua Zhang, Faizan Qamar Taj-Aldeen Naser Abdali Rosilah Hassan Syed Talib Abbas Jafri. Blockchain Technology: Security Issues, Healthcare Applications, Challenges and Future Trends / Faizan Qamar Taj-Aldeen Naser Abdali Rosilah Hassan Syed Talib Abbas Jafri Wenhua, Zhang, Quang Ngoc Nguyen // *Electronics*. — 2023. — Vol. 12, no. 3. — Pp. 1–28. <https://doi.org/10.3390/electronics12030546>.
- 10 Marten Risius1 Christoph F, Breidbach1-Mathieu Chanson2 Ruben von

Krannichfeldt³ Felix Wortmann³. On the performance of blockchain-based token offerings / Breidbach¹-Mathieu Chanson² Ruben von Krannichfeldt³ Felix Wortmann³ Marten Risius¹, Christoph F // *SpringerLink*. — 2023. — Pp. 1–19. <https://link.springer.com/article/10.1007/s12525-023-00652-5>.

11 Firmansyah Ashari¹ Tetuko Catonsukmoro², Wilyu Mahendra Bad³ Sfenranto⁴ Gunawan Wang⁵. Smart Contract and Blockchain for Crowdfunding Platform / Wilyu Mahendra Bad³ Sfenranto⁴ Gunawan Wang⁵ Firmansyah Ashari¹, Tetuko Catonsukmoro² // *International Journal of Advanced Trends in Computer Science and Engineering*. — 2020. — Vol. 9, no. 3. — Pp. 1–7. <https://www.warse.org/IJATCSE/static/pdf/file/ijatcse83932020.pdf>.

12 Hassija, Vikas. BitFund: A Blockchain-based Crowd Funding Platform for Future Smart and Connected Nation / Vikas Hassija, Vinay Chamola, Sherali Zeadally // *Sustainable Cities and Society*. — 2020. — Vol. 60, no. 102145. — Pp. 1–18. https://www.researchgate.net/publication/341450390_BitFund_A_Blockchain-based_Crowd_Funding_Platform_for_Future_Smart_and_Connected_Nation.

13 An Approach for Crowdfunding Using Blockchain / K. M. Anandkumar, Shravan Vignesh, M. E, Syed I // *E3S Web of Conferences*. — 2024. — Vol. 491, no. 02021. — Pp. 1–14. https://www.researchgate.net/publication/378366684_An_approach_for_crowd_funding_using_blockchain.

14 Mohd Javaid Abid Haleem, Ravi Pratap Singh Rajiv Suman. A review of Blockchain Technology applications for financial services / Ravi Pratap Singh Rajiv Suman Mohd Javaid, Abid Haleem, Shahbaz Khan // *ScienceDirect*. — 2022. — Vol. 2, no. 3. — Pp. 1–18. <https://www.sciencedirect.com/science/article/pii/S2772485922000606>.

15 Mithsara, Malithi. Blockchain-based Distributed Secure Crowdfunding and Decision-Making Platform for Large-scale Business Projects in Public and Private Sectors / Malithi Mithsara, T. M. K. K. Jinasena // *European Modern Studies Journal*. — 2020. — Vol. 4, no. 5. — Pp. 1–11. https://www.researchgate.net/publication/344425077_Blockchain-based_Distributed_Secure_Crowdfunding_and_DecisionMaking_Platform_for_Largescale_Business_Projects_in_Public_and_Private_Sectors.

16 Rathore, A. Decentralized Crowdfunding Platform Using Blockchain / A. Rathore, S. W. A. Rizvi // *Journal of InformaticsElectrical and Electronics Engineering*. — 2023. — Vol. 04, no. 079. — Pp. 1–12. <https://jieee.a2zjournals.com/index.php/ieee/article/view/79>.

17 Smart Contracts Security Application and Challenges: A Review / Alaba Fadele, Hakeem Sulaimon, Ifeyinwa Marisa Madu, Owamoyo Najeem // *Cloud Computing and Data Science*. — 2023. — Pp. 15–41. <https://ojs>.

wiserpub.com/index.php/CCDS/article/view/3271.

18 *Poornima, Dr.* Must Explore Tools and Techniques for Novice Blockchain Developers (Delving into Blockchain Technology) / Dr. Poornima, G. Naik. — Shashwat Publication, 2023.

19 *Tiganoaia, B.* Building a Blockchain-Based Decentralized Crowdfunding Platform for Social and Educational Causes in the Context of Sustainable Development / B. Tiganoaia, G. Alexandru // *MDPI*. — 2023. — Vol. 15, no. 23. — Pp. 1–19. <https://www.mdpi.com/2071-1050/15/23/16205>.

20 *Amin, Md. Ratul.* CROWDFUNDING SMART CONTRACT: SECURITY AND CHALLENGES / Md. Ratul Amin, Megat Zuhairi // *Indian Journal of Computer Science and Engineering*. — 2021. — Vol. 12. — Pp. 202–209. https://www.researchgate.net/publication/349473746_CROWDFUNDING_SMART_CONTRACT_SECURITY_AND_CHALLENGES.

21 *Bamber, Sukhvinder Singh.* CrowdFund: CrowdFunding Decentralized Implementation on Ethereum Blockchain / Sukhvinder Singh Bamber // *International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING*. — 2023. — Vol. 11, no. 3. — Pp. 1–6. <https://ijisae.org/index.php/IJISAE/article/view/2587>.

22 *Node.js.* Node.js Documentation. <https://nodejs.org/en>.

23 *Expo.* Expo Documentation. <https://docs.expo.dev/>.

24 *Source, Meta Open.* React Native: Learn once, write anywhere. <https://reactnative.dev/>.

25 *Solidity.* Solidity Documentation. <https://docs.soliditylang.org/en/v0.8.25/>.

26 *WalletConnect.* Getting Started. <https://docs.walletconnect.com/getting-started>.

27 *WAGMI.* Our Mission. <https://docs.wagmi.com/wagmi/about-wagmi/our-mission>.

Appendix A Smart Contract

1.0.1 Smart Contract representation

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

contract CrowdFunding {
    struct Campaign {
        address owner;
        string title;
        string description;
        uint256 target;
        uint256 deadline;
        uint256 amountCollected;
        string image;
        address[] donators;
        uint256[] donations;
    }
    mapping(uint256 => Campaign) public campaigns;
    uint256 public numberOfCampaigns = 0;

    function createCampaign(address _owner, string memory _title,
        string memory _description, uint256 _target, uint256 _deadline,
        string memory _image) public returns (uint256) {
        Campaign storage campaign = campaigns[numberOfCampaigns];

        require(campaign.deadline < block.timestamp, "The deadline
            should be a date in the future.");

        campaign.owner = _owner;
        campaign.title = _title;
        campaign.description = _description;
        campaign.target = _target;
        campaign.deadline = _deadline;
        campaign.amountCollected = 0;
        campaign.image = _image;
        numberOfCampaigns++;
        return numberOfCampaigns - 1;
    }
}
```



```

function donateToCampaign(uint256 _id) public payable {
    uint256 amount = msg.value;
    Campaign storage campaign = campaigns[_id];

    campaign.donators.push(msg.sender);
    campaign.donations.push(amount);

    (bool sent,) = payable(campaign.owner).call{value:
    amount}("");

    if(sent) {

        campaign.amountCollected =
        campaign.amountCollected + amount;
    }
}

function getDonators(uint256 _id) view

public returns (address[] memory,

uint256[] memory) {
    return (campaigns[_id].donators, campaigns[_id].donations);
}

function getCampaigns() public view returns (Campaign[] memory)
{
    Campaign[] memory allCampaigns =

    new Campaign[](numberOfCampaigns);

    for(uint i = 0; i < numberOfCampaigns; i++) {
        Campaign storage item = campaigns[i];

        allCampaigns[i] = item;
    }

    return allCampaigns;
}
}

```

1.0.2 Blockchain Transaction Object Structure

```
const transaction = {
  nonce: <nonce_value>,
  gasPrice: <gas_price>
  gasLimit: <gas_limit>
  to: <public_key_receiver>,
  value: <amount>,
  data: <data>,
  v:<recovery_id>,
  r:<ECDSA Signature Output>,
  s:<ECDSA Signature Output>
}
```

1.0.3 Smart Contract for Storing and Retrieving a Value

```
contract SimpleContract {
  uint256 public value;
  address public owner;

  constructor() {
    value = 0;
    owner = msg.sender;
  }

  function setValue(uint256 _newValue) public {
    require(msg.sender == owner,
      "Only the owner can set the value");
    value = _newValue;
  }

  function getValue() public view returns (uint256) {
    return value;
  }
}
```

Appendix B Development code samples

2.0.1 Menu screen

```
import { FlatList, View } from 'react-native';
import projects from '@assets/data/projects';
import ProjectListItem from '@components/ProjectListItem';

export default function MenuScreen() {
  return (
    <View>
      {/* <ProjectListItem project={projects[4]} />
      <ProjectListItem project={projects[1]} /> */}
      <FlatList
        data={projects}
        renderItem={({item}) => <ProjectListItem project={item}/>}
        numColumns={2}
        contentContainerStyle={{gap: 10, padding: 10}}
        columnWrapperStyle={{gap: 10}}
      />

    </View>
  );
}
```

2.0.2 React Native redirect

```
import { Redirect } from "expo-router";

export default function TabIndex(){
  return <Redirect href={'/startup/'}/>
}
```

2.0.3 Project details

```
import { View, Text } from 'react-native'
import React from 'react'
import { useLocalSearchParams } from 'expo-router'

const ProjectDetails = () => {
  const {id} = useLocalSearchParams();
```

```

    return (
      <View>
        <Text style={{fontSize: 20}}> ProjectDetails for id:
          {id} </Text>
        </View>
      )
    }

export default ProjectDetails

```

2.0.4 Modal.tsx

```

import { StatusBar } from 'expo-status-bar';
import { Platform, StyleSheet } from 'react-native';

import EditScreenInfo from '@components/EditScreenInfo';
import { Text, View } from '@components/Themed';

export default function ModalScreen() {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Modal</Text>
      <View style={styles.separator}
        lightColor="#eee"

        darkColor="rgba(255,255,255,0.1)" />
      <EditScreenInfo path="app/modal.tsx" />

      {/* Use a light status bar on iOS to account for
        the black space above the modal */}
      <StatusBar style={Platform.OS ===
        'ios' ? 'light' : 'auto'} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',

```

```

    },
    title: {
      fontSize: 20,
      fontWeight: 'bold',
    },
    separator: {
      marginVertical: 30,
      height: 1,
      width: '80%',
    },
  },
});

```

2.0.5 List of projects

```

import { StyleSheet, Text, View, Image, Pressable }
from 'react-native';
import Colors from '../constants/Colors';
import { Project } from '../types'
import { Link } from 'expo-router'
export const defaultProjectImage =
'https://mobilaser.kz/wp-content/uploads/2022/02/
the-untitled-ventures-2.jpg';

type ProjectListItemProps = {
  project: Project;
}

const ProjectListItem = ({ project }: ProjectListItemProps) => {
  return(
    <Link href={`../startup/${project.id}`} asChild>
      <Pressable style={styles.container}>
        <Image
          source = {{uri: project.image || defaultProjectImage}}
          style = {styles.image}
          resizeMode='cover'
        />
        <Text style = {styles.title}>{project.name}</Text>
        <Text style = {styles.goal}>${project.goal}</Text>
      </Pressable>
    </Link>
  )
}

```

```

        </Link>
    );
};

export default ProjectListItem;

const styles = StyleSheet.create({
  container: {
    backgroundColor: 'white',
    padding: 10,
    borderRadius: 20,
    flex: 1,
    maxWidth: '50%',
  },
  title: {
    fontSize: 18,
    fontWeight: '600',
    marginVertical: 10,
  },
  goal: {
    color: Colors.light.tint,
    fontWeight: '700',
  },
  image: {
    width: '100%',
    aspectRatio: 1,
  },
});

```

2.0.6 Main menu screen

```

import '@walletconnect/react-native-compat'
import { WagmiConfig } from 'wagmi'
import { sepolia } from 'viem/chains'
import { createWeb3Modal, defaultWagmiConfig, Web3Modal, W3mButton }
from '@web3modal/wagmi-react-native'

import { StyleSheet } from 'react-native';

```

```

import EditScreenInfo from '@components/EditScreenInfo';
import { Text, View } from '@components/Themed';

// 1. Get projectId at https://cloud.walletconnect.com
const projectId = '87f1d17369dc69480330b59edb852665'

// 2. Create config
const metadata = {
  name: 'Web3Modal RN diploma app',
  description: 'Web3Modal RN crowdfunding diploma app',
  url: 'https://web3modal.com',
  icons: ['https://avatars.githubusercontent.com/u/37784886'],
  redirect: {
    native: 'YOUR_APP_SCHEME://',
    universal: 'YOUR_APP_UNIVERSAL_LINK.com'
  }
}

const chains = [sepolia]

const wagmiConfig = defaultWagmiConfig({ chains,
projectId, metadata })

// 3. Create modal
createWeb3Modal({
  projectId,
  chains,
  wagmiConfig,
  enableAnalytics: true
  // Optional - defaults to your Cloud configuration
})

export default function TabOneScreen() {
  return (
    <WagmiConfig config={wagmiConfig}>

      <View style={styles.container}>
        <Text style={styles.title}>Tab One</Text>
        <View style={styles.separator} lightColor="#eee"

```

```

        darkColor=
        "rgba(255,255,255,0.1)" />
        <EditScreenInfo path=
        "app/(tabs)/index.tsx" />
        <W3mButton />
    </View>

    <Web3Modal />
  </WagmiConfig>
);
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: '80%',
  },
});

```

2.0.7 Abis of smart contract

```

[
  {
    "inputs": [
      {
        "internalType": "address",
        "name": "_owner",
        "type": "address"
      },
      {

```



```

        "internalType": "string",
        "name": "_title",
        "type": "string"
    },
    {
        "internalType": "string",
        "name": "_description",
        "type": "string"
    },
    {
        "internalType": "uint256",
        "name": "_target",
        "type": "uint256"
    },
    {
        "internalType": "uint256",
        "name": "_deadline",
        "type": "uint256"
    },
    {
        "internalType": "string",
        "name": "_image",
        "type": "string"
    }
],
"name": "createCampaign",
"outputs": [
    {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
    }
],
"stateMutability": "nonpayable",
"type": "function"
},
{
    "inputs": [
        {
            "internalType": "uint256",

```

```

        "name": "_id",
        "type": "uint256"
    }
],
"name": "donateToCampaign",
"outputs": [],
"stateMutability": "payable",
"type": "function"
},
{
    "inputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "name": "campaigns",
    "outputs": [
        {
            "internalType": "address",
            "name": "owner",
            "type": "address"
        },
        {
            "internalType": "string",
            "name": "title",
            "type": "string"
        },
        {
            "internalType": "string",
            "name": "description",
            "type": "string"
        },
        {
            "internalType": "uint256",
            "name": "target",
            "type": "uint256"
        }
    ]
}

```

```

        "internalType": "uint256",
        "name": "deadline",
        "type": "uint256"
    },
    {
        "internalType": "uint256",
        "name": "amountCollected",
        "type": "uint256"
    },
    {
        "internalType": "string",
        "name": "image",
        "type": "string"
    }
],
"stateMutability": "view",
"type": "function"
},
{
    "inputs": [],
    "name": "getCampaigns",
    "outputs": [
        {
            "components": [
                {
                    "internalType": "address",
                    "name": "owner",
                    "type": "address"
                },
                {
                    "internalType": "string",
                    "name": "title",
                    "type": "string"
                },
                {
                    "internalType": "string",
                    "name": "description",
                    "type": "string"
                }
            ],
            "type": "tuple"
        }
    ],
    "name": "getCampaigns",
    "outputs": [
        {
            "type": "tuple"
        }
    ],
    "type": "function"
}

```

```

        "internalType": "uint256",
        "name": "target",
        "type": "uint256"
    },
    {
        "internalType": "uint256",
        "name": "deadline",
        "type": "uint256"
    },
    {
        "internalType": "uint256",
        "name": "amountCollected",
        "type": "uint256"
    },
    {
        "internalType": "string",
        "name": "image",
        "type": "string"
    },
    {
        "internalType": "address[]",
        "name": "donators",
        "type": "address[]"
    },
    {
        "internalType": "uint256[]",
        "name": "donations",
        "type": "uint256[]"
    }
],
"internalType": "struct CrowdFunding.Campaign[]",
"name": "",
"type": "tuple[]"
}
],
"stateMutability": "view",
"type": "function"
},
{
    "inputs": [

```

```

        {
            "internalType": "uint256",
            "name": "_id",
            "type": "uint256"
        }
    ],
    "name": "getDonators",
    "outputs": [
        {
            "internalType": "address[] ",
            "name": "",
            "type": "address[] "
        },
        {
            "internalType": "uint256[] ",
            "name": "",
            "type": "uint256[] "
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "numberOfCampaigns",
    "outputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
}
]

```

Appendix C Mobile Application User Interface

3.0.1 Screenshots

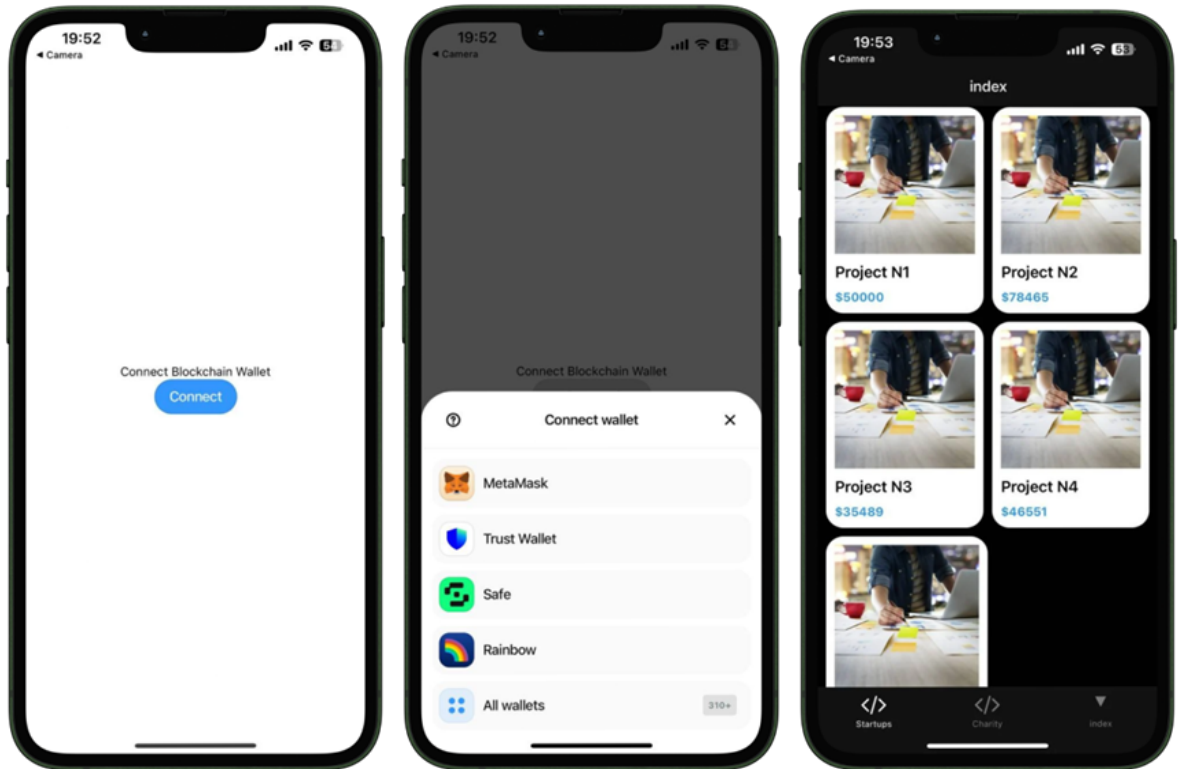


Figure C.1 – Authorization screen

Figure C.2 – Authorizing with WalletConnect

Figure C.3 – Main screen with campaigns