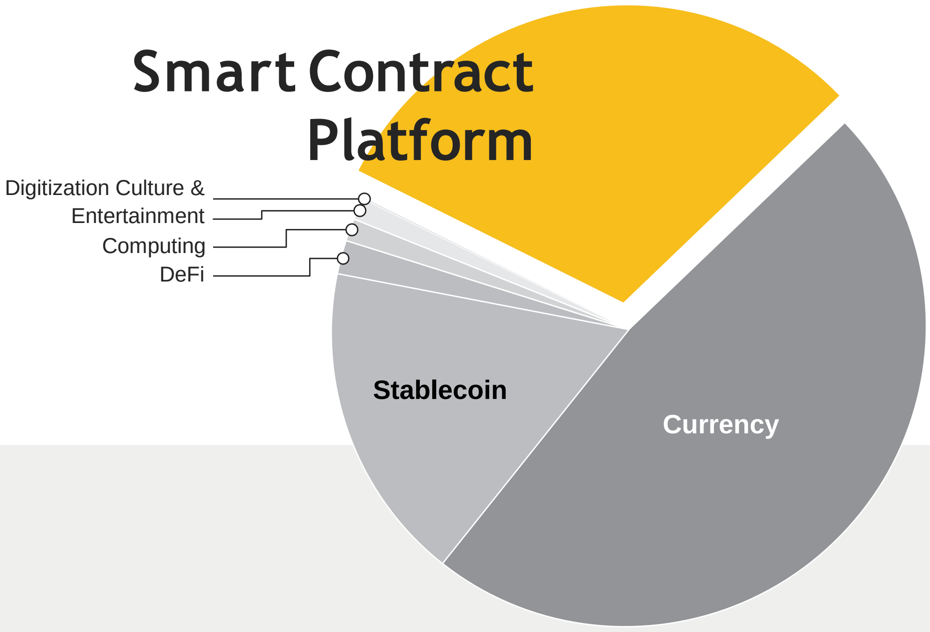
April 2024

DECENTRALIZING CROWDFUNDING: A SMART CONTRACT SOLUTION FOR KICKSTARTER

Asset Tokenisation and Distributed Ledger Technologies



Lecturers: Paul Laird

Group members: Thi Thuy Trang Cao – 20008109
Gauri Shingane – 20018204

Introduction

Crowdfunding platforms, such as Kickstarter, have become popular for individuals to realize their creative projects. However, traditional platforms suffer from issues such as centralization, lack of transparency, and inefficiency. As a result, blockchain technology and smart contracts offer a feasible solution.

This report discusses the creation of a smart contract to power the Kickstarter-like crowdfunding platform. We intend to make the crowdfunding process more transparent, secure, and efficient by utilizing blockchain's decentralization and the automation abilities of smart contracts. Our research demonstrates the tremendous prospects for implementing blockchain applications in crowdfunding, leading the way to a more democratic and inclusive fundraising landscape.

Background and Rationale

What is a smart contract?

A smart contract is a digital agreement encoded within a computer program and deployed on a network. It operates on the "if/then" logic principle, where predefined conditions trigger the automated execution of contract terms. Smart contracts automate processes by monitoring events and executing actions based on predetermined rules. See figure below for a detail illustration of smart contracts.

Step 1: Contract implementation

Public ledger
Stakeholders
Contractor

Transcribe a construction agreement into a smart contract between interested parties

- Computer program incorporates terms and conditions of construction agreement into a computer code running on a network.
- Smart contract is agreed and validated via consensus vote in a blockchain network and stored in each node

The Smart contracts enable:

- Consensus based validation
- Standardization and storage at all nodes

Step 2: Contract Execution

Contract self-executes executes based on the "if/then" principle, where situations are monitored, terms are executed and transactions are allocated by the computer program, based on the situation

- Sends the transaction via blockchain network for validation against set rules
- Validation can be manual or automated based on how it is setup

Step 3: Contract Amendment

Transcribe amendments and change orders into a modified smart contract between parties

- The contract amendment that includes new or modified contractual terms and logic is communicated to all parties
- Modified smart contract is agreed and validated via consensus vote in a blockchain network and stored in each node

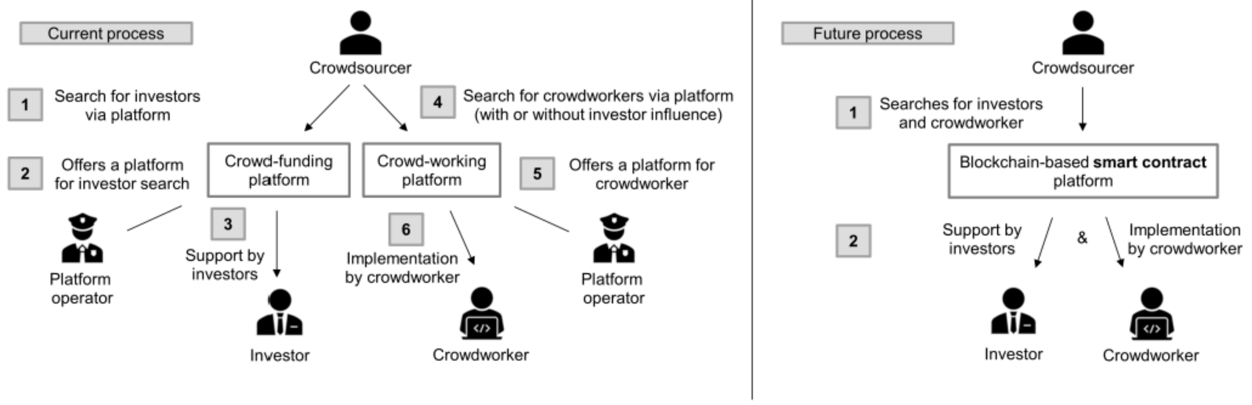
Benefits of smart contracts

1. **Reduced or complete avoidance** of intermediary and overall project costs
2. **Maximized transparency** of cost, time and scope of project
3. **Limited complexity**, enabling informed decision-making
4. **Efficient resolution** of contractual grey areas

Benefits of using a blockchain-based crowdfunding platform

Backers contribute funds to the campaign via cryptocurrency transactions recorded on the blockchain. The smart contract automatically executes actions such as updating funds raised and releasing funds based on predefined milestones. This process ensures transparency, accountability, and efficient distribution of funds to project creators upon campaign completion.

Exhibit 1: Blockchain-based crowdfunding platform process



Current process
Crowdsourcer
1 Search for investors via platform
4 Search for crowdworkers via platform (with or without investor influence)
2 Offers a platform for investor search
Crowd-funding platform
Crowd-working platform
5 Offers a platform for crowdworker
3 Support by investors
6 Implementation by crowdworker
Platform operator
Investor
Crowdworker
Platform operator
Future process
Crowdsourcer
1 Searches for investors and crowdworker
Blockchain-based smart contract platform
2 Support by investors & Implementation by crowdworker
Investor
Crowdworker

Source: M. Billert, 2019

Use case for smart contracts

Smart contracts could combine the search for investors and crowd workers and minimize the interactions between stakeholders, reducing transaction costs. Automating these processes through smart contracts eliminates intermediaries, streamlining the crowdfunding process and ensuring greater efficiency. This approach simplifies the engagement between creators and backers and enhances transparency and trust within the crowdfunding ecosystem.

Purchase order/Scope of work: This document outlines the agreed-upon terms and conditions between the buyer and seller for goods or services. It specifies the scope of work, including deliverables, timelines, and pricing, ensuring clarity and alignment between parties.

Certified Progress: This contains verified milestones or stages of completion within a project or contract. Certification of progress confirms that predefined criteria or benchmarks have been met, triggering further actions or payments as per the agreement.

Master data: Contains critical information that serves as a foundation for business operations, such as

- Customer details
- Product specifications
- Vendor information

Matching with the invoice validates that the contractual details are represented correctly, and that cost limits and operational ratios are within set parameters.

Benefits

The use of smart contracts could potentially provide the following benefits:

- Transparency: Blockchain technology ensures transparent and immutable record-keeping of all transactions. Every contribution, transfer, and distribution of funds is recorded on the blockchain, providing stakeholders with a transparent view of the crowdfunding process.
- Security: The decentralized nature of blockchain networks makes them highly resistant to tampering and hacking. Smart contracts underpin blockchain-based crowdfunding platforms, enforce pre-defined rules and automate transaction execution, reducing the risk of fraud and manipulation.
- Lower Costs: By eliminating intermediaries such as banks or payment processors, blockchain-based crowdfunding platforms reduce transaction fees and administrative costs associated with traditional crowdfunding models.
- Global Accessibility: Blockchain technology enables crowdfunding campaigns to reach a global audience without the limitations of traditional financial systems.

Requirements Analysis

Functional Requirements

- **Create Campaign:** Smart contract should have functionality to initiate a new crowdfunding campaign with specific funding goals, a list of whitelisted vendors' wallet addresses, and token specifications for the campaign contributors.
- **Accept funds:** Smart contract should facilitate the contributors to fund the campaign and seamlessly keep track of the contributors.
- **Reward tokens:** All the contributors should get a rewards token depending on the amount they have contributed to the campaign. This token can be redeemed for certain benefits by the contributor at the end of the campaign.
- **Spending funds:** Smart contract should facilitate the campaign owner to spend the funds collected by the campaign by creating a fund transfer request, which are then sent to all the contributors of the campaign.
- **Consensus for fund transfer requests:** The fund transfer request should go through an approval process with the contributors and if more than 51% of the contributors approve the fund transfer request, the funds are transferred to the mentioned whitelisted vendor wallet.
- **Token reward:** At the end of campaign the campaign owner can decide what reward or benefits to be awarded to the contributors based on the type of token they possess.
- **Security:** For the security of our smart contract, we have implemented checks in our code using require statements, restricted certain parts of smart contract as creator-only methods and maintained mappings for keeping track of all important elements of the campaign.
- **End campaign:** Once the campaign reached expected milestone, the campaign owner can end the campaign and all the remaining funds will be refunded to the campaign contributors' wallets.

Technical Requirements

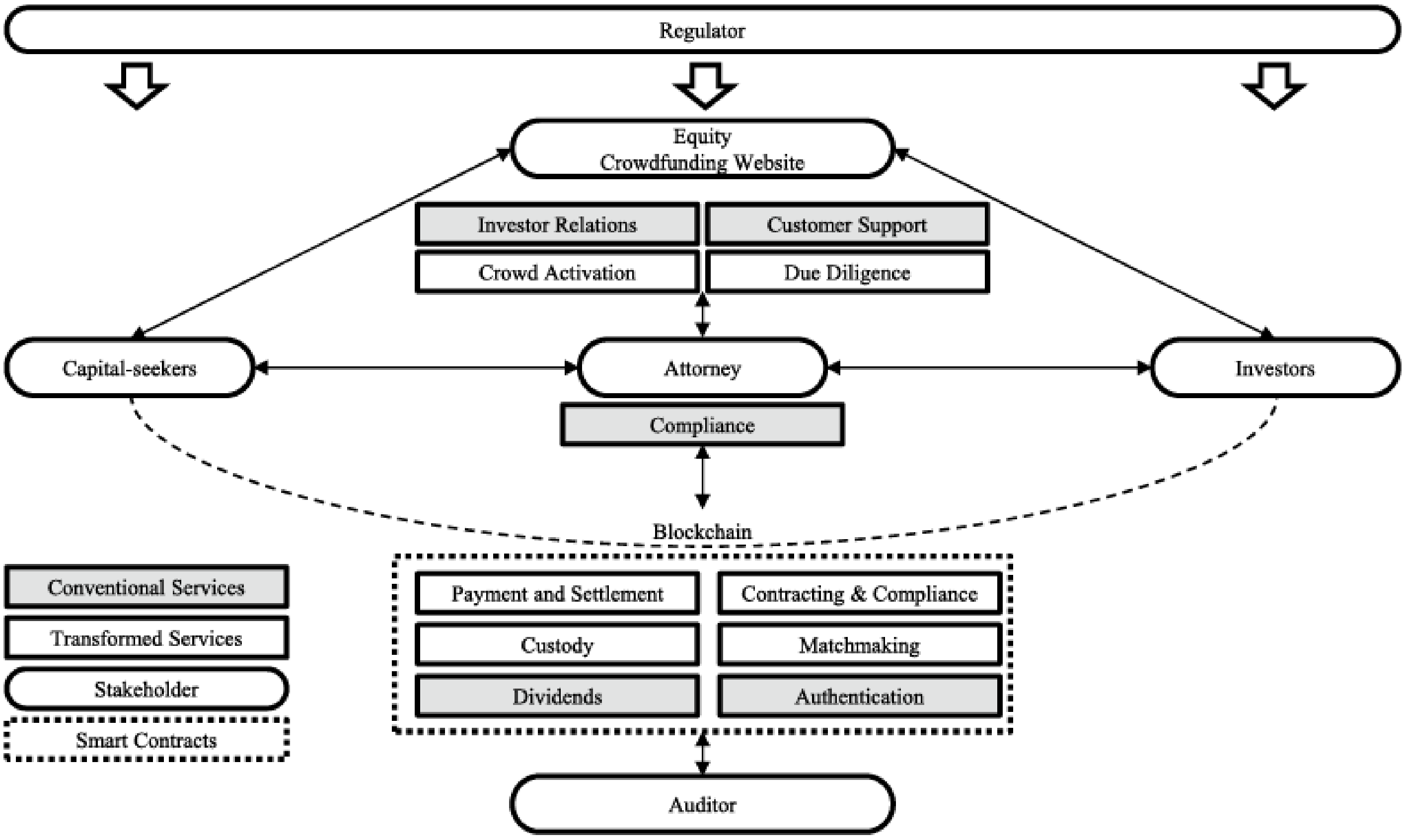
Table 1: Technical requirements

Requirement	Explanation
Blockchain platform	For this smart contract we have chosen the Ethereum blockchain as our Blockchain platform.
Programming language	The programming language for a smart contract on Ethereum is Solidity, so we plan to use Solidity for coding the smart contract.
Ganache and Mocha	For testing and developing the smart contract, we have set up local blockchain using Ganache and used testing library functions provided by Mocha.
Web3.js	We have used web3.js library to facilitate the interaction between Ethereum and the browser.

Stakeholders

- **Campaign Creators**: Individuals or entities who initiate and manage the crowdfunding campaign to fund their projects or ventures. They are responsible for creating compelling campaign content, setting fundraising goals, and delivering promised rewards or outcomes.
- **Contributors/Investors**: Individuals or entities who contribute funds to support projects on the crowdfunding platform.
- **Platform Operators:** The administrators or operators of the crowdfunding platform who provide the infrastructure and services for hosting campaigns, managing transactions, and facilitating communication between creators and backers.

Exhibit 2: Stakeholders involved in the crowdfunding process



Source: Haas et al., 2015

- **Regulators and Legal Authorities:** Government agencies or regulatory bodies oversee crowdfunding activities and ensure compliance with relevant laws and regulations.
- **Service Providers:** Third-party providers offering ancillary services such as payment processing, marketing, legal advice, and fulfilment services.
- **Community and Supporters:** Individuals who are not direct backers but play a crucial role in spreading awareness and generating interest in crowdfunding campaigns.
- **Financial Institutions:** Banks or financial institutions that facilitate transactions between backers and project creators, particularly for platforms that accept traditional payment methods alongside cryptocurrencies.

Design process

Defining the components

- **Campaign:** Campaign component is the main class of the smart contract having multiple elements and methods to help the smart contract work as expected. This component has elements which help keep record of all objects such as a variable manager to keep track of the campaign owner, *minContribution* variable, contributors mapping to keep track of all wallet IDs of contributors, *approverCount* variable to keep count of total contributors, requests mapping for keeping record of all fund transfer requests, count of total fund transfer requests *totalRequests*.
- **Contributors:** Contributor component is the wallets IDs(people) who will contribute and back the campaigns. Multiple methods help contributors interact with the smart contract.
- **Vendors:** Vendors are the wallet IDs that will receive funds from the smart contract in exchange of service or products that will help campaign owner to fulfil the needs of the campaign.
- **Requests:** Request components help the campaign owners to get a consensus from the contributors on whether to send the funds to indicated vendor(wallet ID) to get services in exchange. The contributors can verify the wallet ID of the vendors and approve/reject the spending transaction.

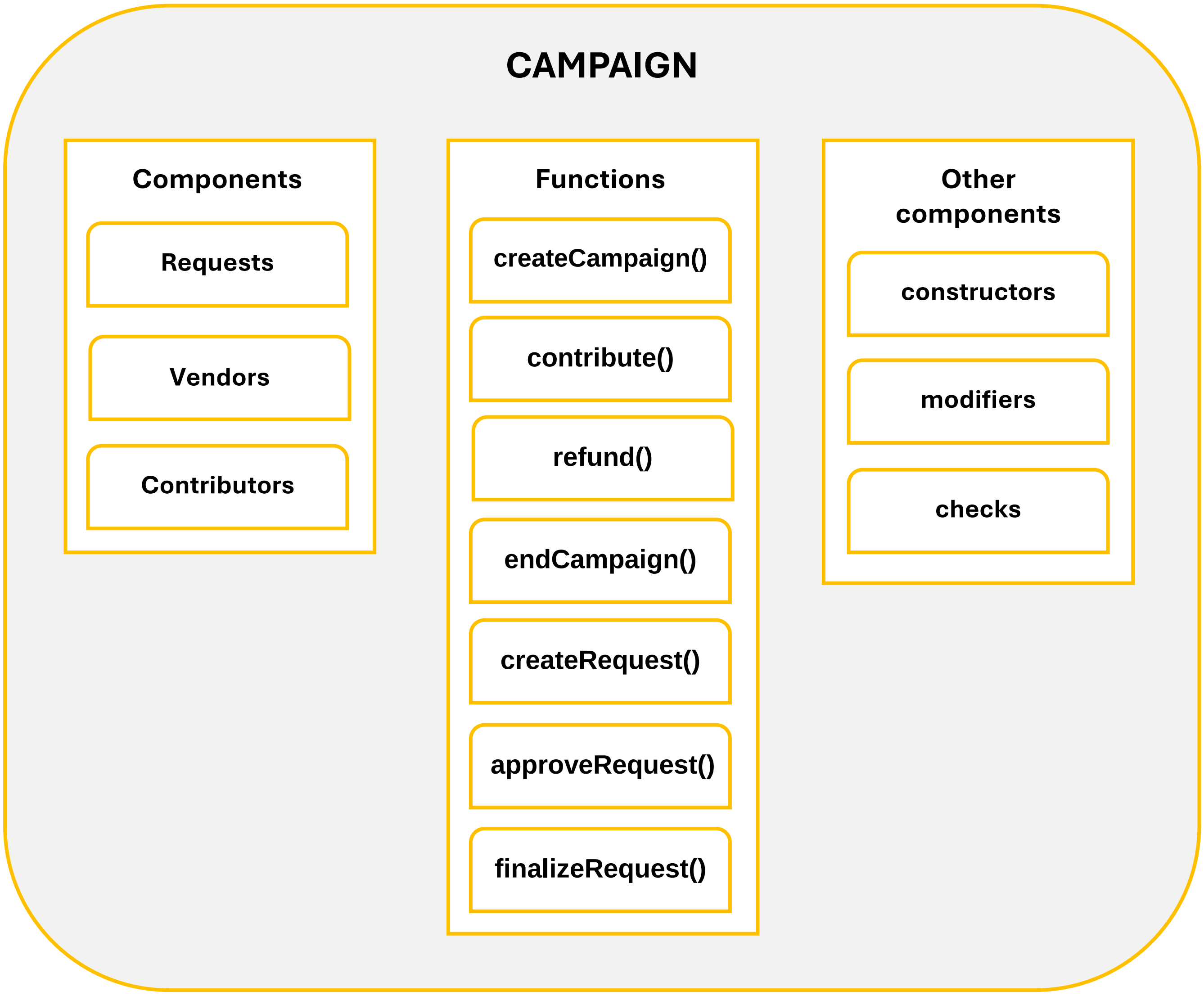
Developing smart contract functions

- createCampaign(*minContribution*): This function allows the campaign owner to initiate a new campaign by providing the minimum contributions that can be sent by the contributors.
- contribute(*amount*): This function helps the contributors to donate funds to the campaign and also get back a token which proves their contribution to this particular campaign.
- createRequest(*description*, *amount*, *recipient*): This method helps the campaign owner to create a fund transfer request by providing appropriate explanation of where and why funds are being transferred to, the amount of funds being transferred, the recipient’s wallet address.
- approveRequest(*campaignId*): Once the contributor approves any fund transfer request, this method registers this vote and the outcome is used by *finalizeRequest* method.
- finalizeRequest(*requestId*): This method uses the data registered by *approveRequest* methods and decides whether the consensus has reached and finally whether to complete the transfer of funds or to discard the transfer request initiated by the campaign owner.
- refund(*campaignId*): This method helps the contributors to get a refund of their contribution to the campaign. It also discards the token awarded to the contributor so that they cannot at any point in future claim any benefits from the campaign.
- endCampaign(*campaignId*): This method helps the campaign owner to end the campaign and transfers all the remaining funds to the contributors’ wallets.

Security considerations

- **Constructor initializations:** While creating a new campaign, the constructor stores the campaign creator’s wallet ID in a variable manager. This helps us check the wallet IDs of the user ensure that certain methods can only be accessed by the campaign owner.
- **Modifiers**: Modifiers are used to restrict certain methods of the smart contract from other users and ensure they are accessible to campaign owner only.
- **Checks**: Implementing checks in all methods using require statements helps filter unwanted and suspicious traffic from reaching the actual logic of the methods.

Final Design



Interaction Pattern

Creating a new campaign

The campaign owner can initiate a new campaign by providing data like,

1) description of the campaign to be created

2) target amount needed through the crowdfunding

3) minimum amount for contributing to the campaign

4) whitelisted wallet addresses of the vendors that will be providing their services and products to successfully reach the campaign goals

The smart contract does the following steps when a new campaign is created:

1) it assigns the creator's wallet address to the *owner* parameter of the smart contract – it helps when restricting certain functionalities of the smart contract to the owner.

2) it sets the *minimum_contribution* parameter to the minimum contribution amount provided by the campaign creator.

3) creates a mapping to keep track of all contributors

4) creates a count variable to keep count of all contributors

5) creates a mapping to keep track of all fund transfer requests to be created by the campaign owner

Once the campaign owner creates a new instance of the campaign smart contract, the people can start contributing to the campaign.

Contributing to the campaign

The public can freely contribute to the campaigns by adhering to the minimum contribution restriction set by the campaign owner.

The smart contract does the following processing steps when a new contributor contributes to the campaign:

1) check if the contributed amount is greater or equal to the minimum contribution amount set by the campaign owner

2) stores the contributor's wallet address in contributors mapping to keep track of all the contributors to the campaign

3) increases the count of contributors *approversCount*

4) sends a token to the contributor based on the amount of funds donated

Note that the contributed amount will be stored on the smart contract’s balance, not the campaign owner's wallet. This will ensure all spending activities are initiated through the smart contract and all contributors can verify the transactions.

Spending funds from the campaign

To spend the funds contributed by the public, the campaign owner needs to create a fund transfer request. The following details need to be provided by the campaign owner while creating a fund transfer request,

1) the reason for spending the funds, and a detailed description of why the fund transfer request was initiated.

2) amount of funds being spent by the campaign owner

3) the address of the recipient of funds

Once this request is created by the campaign owner, the smart contract executes the following steps,

1) creates a request object and assigns the description, amount, and recipient parameters

2) stores this object in the mapping requests to keep track of all fund transfer requests ever created by the campaign owner.

3) initiates a consensus by sending an approval request to all the contributors of the campaign, this step uses the contributors mapping updated in the contribution stage.

4) creates a count *approvalCount* to keep count of the contributors who have approved the fund transfer request, this variable will help in maintaining consensus

The notification sent to all the contributors helps create a consensus on whether the fund transfer request should be completed or should it be suspended.

The contributor can approve or reject the fund transfer request simply by interacting with the front end of the smart contract. When the campaign contributor approves the fund transfer request, the smart contract performs the following actions,

1) checks whether the address is in the contributors mapping, to make sure the address has indeed contributed to this campaign

2) check if the contributor has already approved/rejected the fund transfer request

3) Increase the *approvalCount* value by 1

Note that no action is performed by the smart contract if the contributor rejects the fund transfer request.

When the *approvalCount* value reaches more than 51% of the total contributors, the request can be manually completed by the campaign owner else the campaign owner can wait for all the contributors to approve/reject the fund transfer request.

If more than 51% of the contributors approve the fund transfer request then after finalizing the request the funds mentioned in the fund transfer request are automatically sent to the pre-decided vendor.

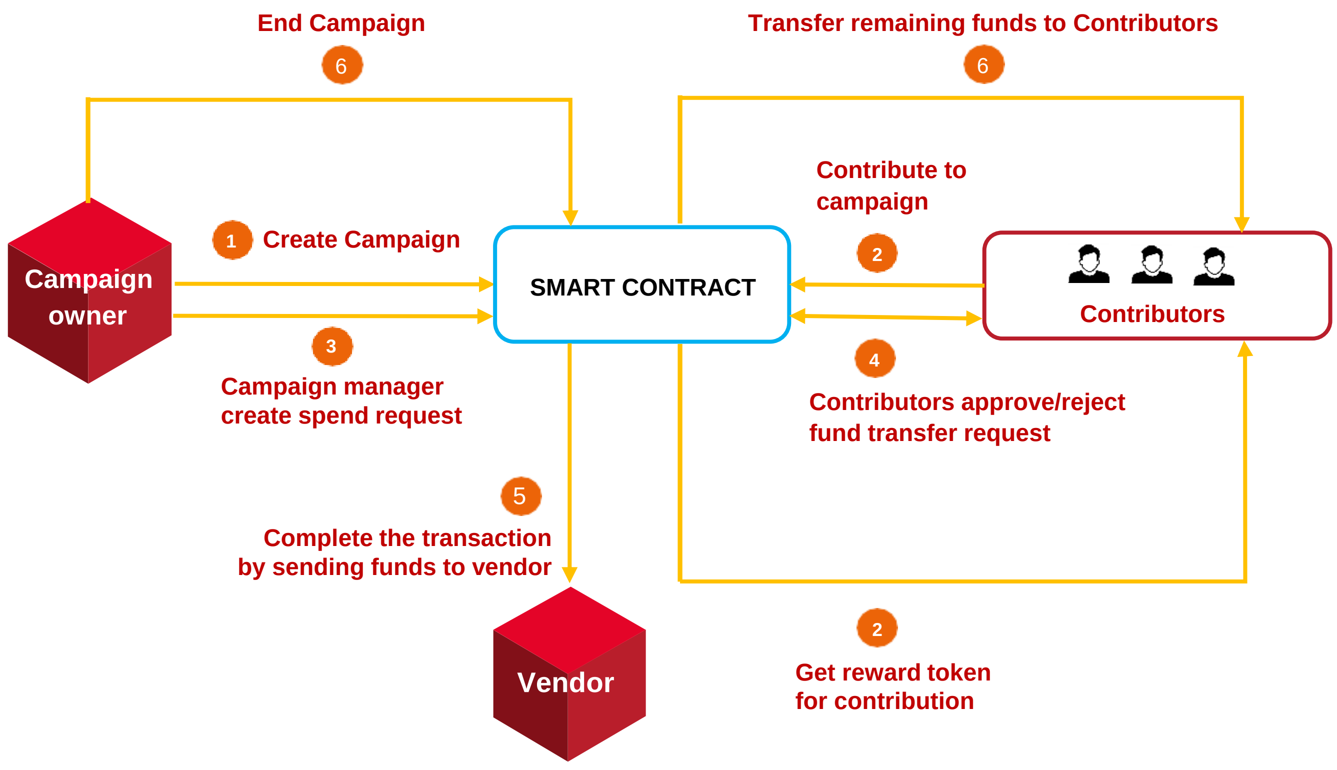
Getting refund

To get a refund of funds, the contributors can simply provide the *campaign_Id* and get the funds back. An interactive frontend is provided to the contributors to get back their funds easily.

Once the contributors get their refund the tokens rewarded while donating the funds will be rendered invalid and they cannot claim any benefits from the campaign anymore.

Ending a campaign

The campaign owner can terminate a campaign when it reaches certain milestone by simply using the terminate option on the frontend. This will delete the instance of smart contract deployed on the blockchain and transfer all the remaining funds to the contributors' wallets.


End Campaign
6
Transfer remaining funds to Contributors
6
Contribute to campaign
2
1
Create Campaign
Campaign owner
SMART CONTRACT
Contributors
3
Campaign manager create spend request
4
Contributors approve/reject fund transfer request
5
Complete the transaction by sending funds to vendor
Vendor
2
Get reward token for contribution

INDIVIDUAL CONTRIBUTION

In our group assignment, we collaborated closely on every aspect of the project, leveraging our combined strengths to achieve our goals effectively.

Thi Thuy Trang Cao

1. Introduction and Rationale:

- Provided an introduction to the project.
- Discussed the motivation behind developing a smart contract for a crowdfunding platform.
- Explored the limitations of traditional crowdfunding models and how smart contracts can address these issues.
- Outlined the benefits of using a blockchain-based crowdfunding platform.

2. Requirements Analysis:

- Defined the functional and non-functional requirements of the smart contract.
- Identified the stakeholders involved in the crowdfunding process, including project creators and backers.
- Discussed regulatory considerations, such as compliance with securities laws.

3. Design Process

- Describe the iterative design process followed to develop the smart contract.

Gauri Shingane

1. Final Design:

- Presented the finalized version of the smart contract.
- Discussed any modifications or improvements made during the design process.
- Highlighted key features and functionalities of the smart contract.

2. Model Deployment:

- Described a typical interaction flow between project creators and backers using the smart contract.
- Discussed how the smart contract facilitates the crowdfunding process, including project creation, funding, and project completion.

Collaborative Tasks:

1. Design Process:

- Discussed any design patterns or best practices used in the development.
- Actively participated in collaborative discussions and decision-making processes.

2. Review and Integration:

- Reviewing each other’s part to ensure quality, consistency, and adherence to best practices.
- Integrating individual components into a cohesive solution, ensuring seamless interaction between different parts of the project.
- Worked together to address any challenges or issues that arose during the project