Frontier Multisig

# Functional Specification

**Dara Lynch - 19324446**
**Alan McGrath - 19392951**

**Date of completion: 18/11/2022**

# Contents

# 1. Introduction

## 1.1 Overview

The aim of this project is to build a web application that allows users to create and use multisignature cryptocurrency wallets. A regular cryptocurrency wallet only requires one signature for a transaction to be processed. However, a multisignature wallet requires multiple signatures in order for transactions to take place. This ensures a higher level of security when making transactions, as multiple users would have to approve or deny the transaction in order for it to be completed.

If an account within the wallet should be breached, the funds within the wallet will remain safe unless all wallet holder accounts are also breached. The wallets will be on the ethereum blockchain which means that all transactions are stored in an immutable database, where users can view any transactions that have been previously made.

## 1.2 Business Context

A multisignature cryptocurrency wallet would prove useful within a business so that multiple wallet holders could have access to the funds within a wallet. A transaction could not be made without the approval of the other wallet users. This allows extra security within the company as each transaction would have to be discussed or approved by others.

A primary wallet holder would be held responsible for adding other users to the multisignature wallet. This encourages better collaboration within a business as it gives the company the opportunity to discuss whether or not a transaction would be necessary. When a business uses our wallet they can allow more freedom for spending business funds while keeping these funds highly secure.

### 1.3 Glossary

***Blockchain*** - A blockchain is a public digital ledger where data is stored.

***Ethereum*** - Ethereum is an open source blockchain which allows transactions through the use of smart contracts.

**Cryptocurrency** - A token issued on the blockchain which can generally be traded or exchanged

**Cryptocurrency wallet** - A place where cryptocurrency tokens can be stored.

***Gas Fees*** - The fee paid to the network to complete a transaction.

***Metamask Wallet*** - This is a commonly used Ethereum soft wallet.

***FIAT currency*** - Government issued currency that is not backed by a commodity such as gold (e.g. US Dollar, Euro, GBP etc.)

# 2. General Description

## 2.1 Product / System Functions

To use our web app the user must have previously created a Metamask wallet. This will be essential for logging into the application as this is how users will create or be added to wallets. The user's wallet address will have to be shared with the primary wallet holder so that they can allow the user access to the funds inside the wallet, although the option to make a transaction would still depend on the approval of the other wallet holders.

Before a transaction can be made, funds must be added to the wallet. To do this a wallet owner must use a cryptocurrency exchange to purchase the funds. Once there are funds available in the wallet the user may start a transaction that will need to be approved or denied by the users with access to this wallet. This is an additional layer of security for our web application so that if one account is breached transactions may not be processed if all users within a wallet have to approve it.

Another main characteristic of our project is that all transactions will be stored on the ethereum blockchain. This means that each transaction made will be stored in an immutable database which can be accessed by anyone, this gives the ability to see where and how many funds were sent at each transaction.

## 2.2 User Characteristics and Objectives

The audience targeted for this application are businesses that purchase goods and services using cryptocurrency. The option to create a group wallet allows more collaboration within a company for colleagues to discuss which transactions are necessary or not, while still maintaining exceptional security standards.

### 2.2.1 - Metamask

Users will be required to have a Metamask wallet which is extremely easy to set up. This allows users with little or no knowledge of blockchain technology to be able to easily use our platform.

### 2.2.2 - Cryptocurrency Volatility

If users are to hold cryptocurrencies they should make themselves aware of the volatility of these assets and how the prices can rapidly vary.

### 2.2.3 - Shared Account/Company Card

It would also be preferred if a user had previous experience using FIAT currency shared/company accounts or cards previously. This would mean they would already be familiar with the process of transactions being approved.

## 2.3 Operational Scenarios

### 2.3.1 - Create Wallet:

1. Log into the web app by connecting your Metamask wallet.
2. Click “Create new Wallet”.
3. Your metamask wallet will pop up automatically and ask you to confirm this. Click confirm.
4. Your wallet is created and ready to be used.

### 2.3.2 - Add other owners to wallet:

1. Open the home screen of your wallet.
2. Click “Add users”.
3. Paste the wallet address of the user you would like to add.
4. Your metamask wallet will pop up automatically and ask you to confirm this. Click confirm.
5. The user has now been added to the wallet.

### 2.3.4 - Remove owners from wallet:

1. Open the home screen of your wallet.
2. Click “Remove users”.
3. Find the wallet address matching the user that you would like to remove.
4. Click “Remove this user” and you will be prompted with an “Are you sure…” popup. Click “Confirm”.
5. The user has now been removed from the wallet.

### 2.3.5 - Create a transaction for approval:

1. Open the home screen of your wallet.
2. Click start transaction.
3. Enter the address that the funds will be sent to.
4. (Optional) Tag the transaction or leave a note so others know where the funds are being sent.
5. Wait for the other wallet owners to approve or deny the transaction.

### 2.3.6 - Approve/deny transaction:

1. Open wallet to view pending transactions.
2. Click “Approve” or “Deny”.
3. Transaction will be completed or cancelled depending on approval process.

#### 2.3.7 - Add funds to wallet:

1. Purchase funds from a cryptocurrency exchange of your choice (Binance, Coinbase etc.)
2. Open your multisig wallet and copy the wallet address.
3. Go back to the exchange where your funds are and follow their instructions to make a withdrawal.
4. When prompted for an address to send the funds, paste the address that you previously copied.
5. Double check that the wallet address you are sending the funds to matches the address of your multisig wallet and complete the transaction.
6. Wait a short period of time (varies based on network traffic and the exchange you use) and the funds will appear in your multisig wallet**.**

### 2.4 Constraints

#### 2.4.1 - Immutability and Loss

Due to the nature of blockchain technology, if our wallets were to not work properly, the funds in that wallet could be lost forever. The blockchain is immutable and nobody has the power to recover lost funds or reverse transactions. While this may sound like a flaw in blockchains, it is important that this is the case in order to maintain decentralisation

#### 2.4.2 - Network Fees

On Ethereum there are gas fees which must be paid for every interaction with the blockchain such as adding funds to your wallet and taking the funds out of your wallet. Users should keep enough gas fees in their wallet to be able to cover the gas fees for withdrawing.

#### 2.4.3 - Test Network Downtime

Towards the end of our testing we will deploy our smart contracts to an Ethereum testnet. While the Ethereum network itself does not have downtime, the testnet periodically does. In this case we may need to fall back to deploying contracts locally if this occurs.

# 3. Functional Requirements

## 3.1. **Functional requirement 1 -** Create Wallet

**Description -** A multisignature wallet must be created. The user who creates this wallet will be the 'Primary Owner' with the privileges to add more users to the wallet.

**Criticality -** Essential for funds to be stored and transactions to be made.

**Technical issues -** Must be designed differently from regular wallets in order to allow multiple owners.

**Dependencies with other requirements -** The foundation of all other requirements.

## 3.2. **Functional requirement 2 -** Add other owners to wallet

**Description -** Secondary users to be added by the primary user so that transactions can be made, approved or denied. This action can only be done by the Primary Owner (the user who first created the wallet). The secondary users must send their personal wallet address to the Primary Owner in order to be added to the wallet.

**Criticality -** Not essential but improves security having multiple owners. If not implemented properly, the system may require approval from an invalid user

**Technical issues -**

**Dependencies with other requirements -** Depends on functional requirement 1 being completed.

### 3.3. **Functional requirement 3 -** Remove owners from wallet

**Description -** A primary owner can remove secondary owners from accessing the funds in the wallet. This will mean that the user no longer has access to the funds stored on the wallet and will not be able to vote on future transactions.

**Criticality -** Essential for security.

**Technical issues -**

**Dependencies with other requirements -** Depends on functional requirement 2 being completed.

### 3.4. **Functional requirement 4 -** Add funds to wallet

**Description -** Ability to transfer funds from a cryptocurrency exchange and store them in the wallet.

**Criticality -** Funds are necessary for transactions to be made. Improper implementation may mean loss of funds.

**Technical issues -** Funds may be delayed appearing in the wallet depending on the exchange the user may use and network congestion.

**Dependencies with other requirements -** Depends on functional requirement 1 being completed.

### 3.5. **Functional requirement 5 -** Create a transaction

**Description -** A user will create a transaction and may leave a note or tag so others can see the purpose of the transaction. This transaction is then placed on hold until voting is completed by the wallet users to approve or deny the transaction.

**Criticality -** Essential so that funds can be spent.

**Technical issues -** Systems should be designed to be configurable with how many approvals are required for a transaction to be completed.

**Dependencies with other requirements -** Depends on functional requirement 1 and 4 being completed.

### 3.6. **Functional requirement 6 -** Approve/Deny transaction

**Description -** After a transaction is created, it will be pending until it passes the approval process. The wallet users will be able to vote to approve or deny the transaction. Only once the required number of approvals are received will the transaction be completed.

**Criticality -** Essential for security.

**Technical issues -** Transactions should be designed to automatically complete as soon as necessary approvals are met.

**Dependencies with other requirements -** Depends on functional requirement 5 being completed.

## 4. System Architecture

Ethereum dApp System Overview

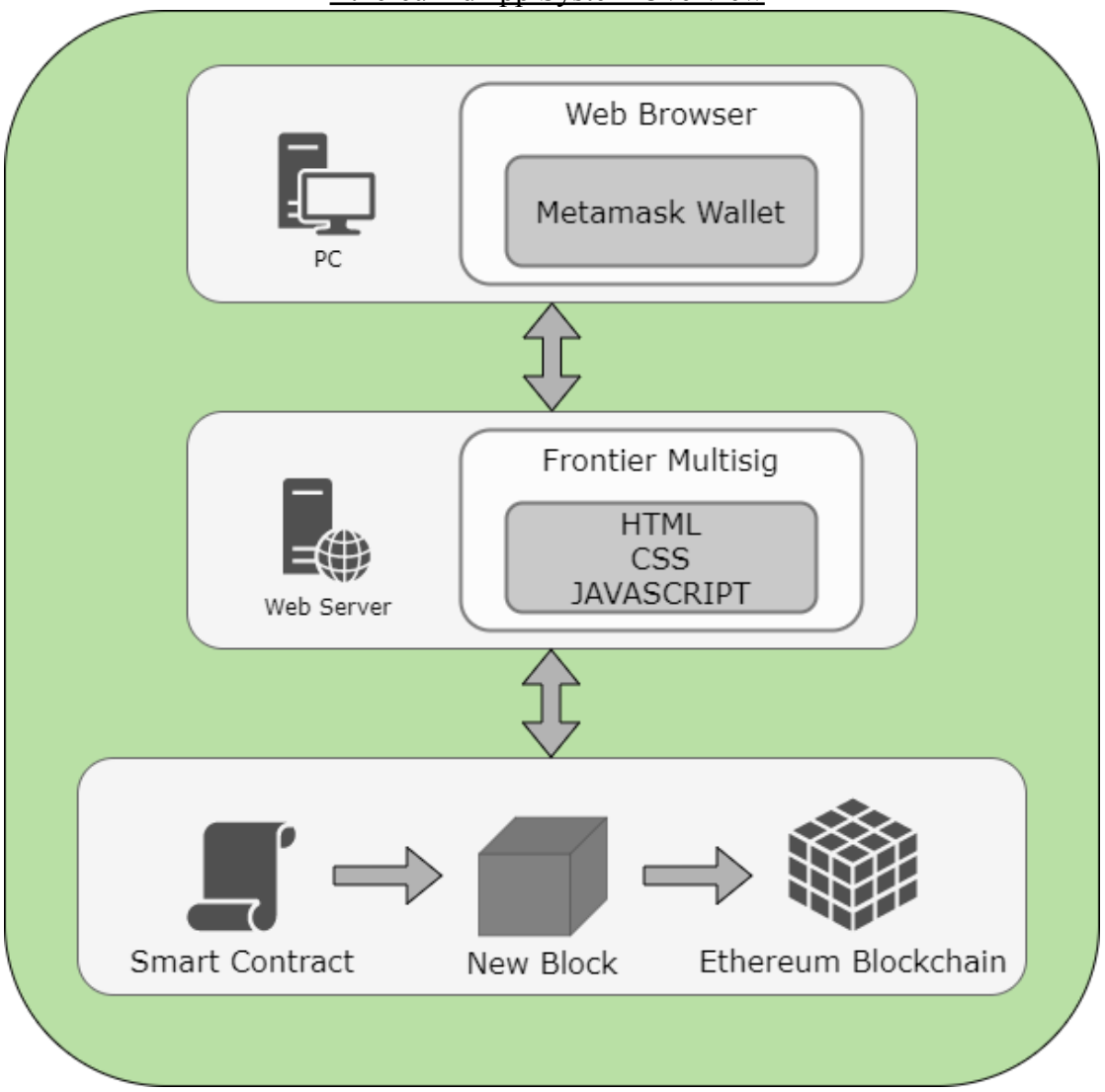


*Fig 4.1*

In the Ethereum dApp System Overview (*Fig 4.1*) we can see a high level architecture of how the user, web server and blockchain interact. The user, on their desktop PC or laptop will access the web app through their web browser, which will need a Metamask wallet extension. The browser and extension will then interact with our web application, which will be made up of HTML, CSS and Javascript. When a user makes an input which requires communication with the blockchain, our javascript function will call one of our smart contracts. The smart contracts are capable of writing new information on the blockchain. The smart contract will put the information onto a 'new block' which is then verified and added to the Ethereum blockchain.

## 5. High-Level Design

Wallet Owners Diagram

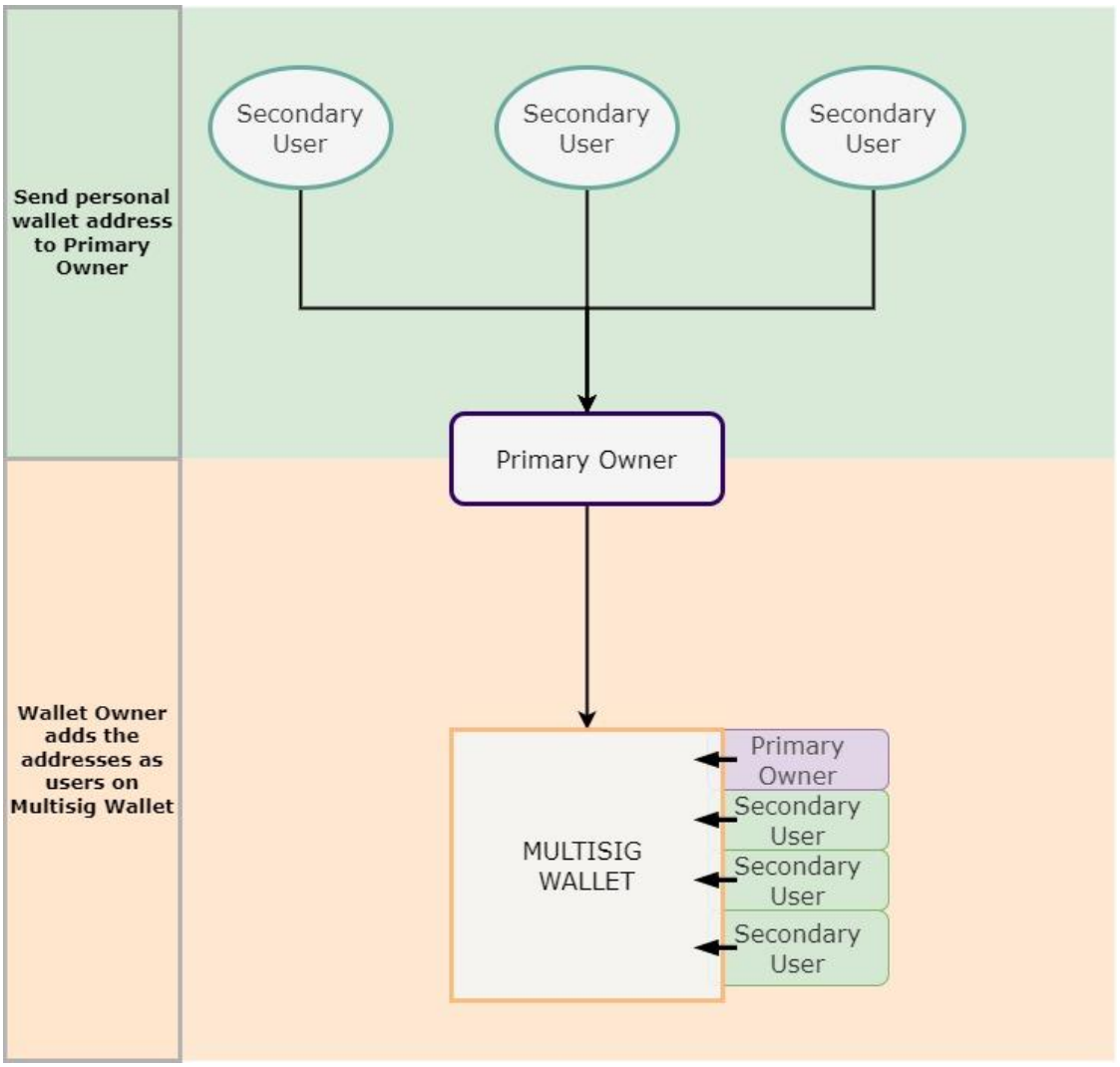


*Fig 5.1*

Using the Wallet Owners diagram (*Fig 5.1*) we can better understand how people can become owners of a Multisig wallet. The primary owner will create the wallet. Next the secondary users will send their personal wallet addresses to the primary owner. He/she will then enter these addresses in order to add the secondary users as wallet owners. Once this is complete, the secondary users can then have the privilege to begin transactions or be a part of the approval process for transactions started by other users.

Transaction Process Diagram

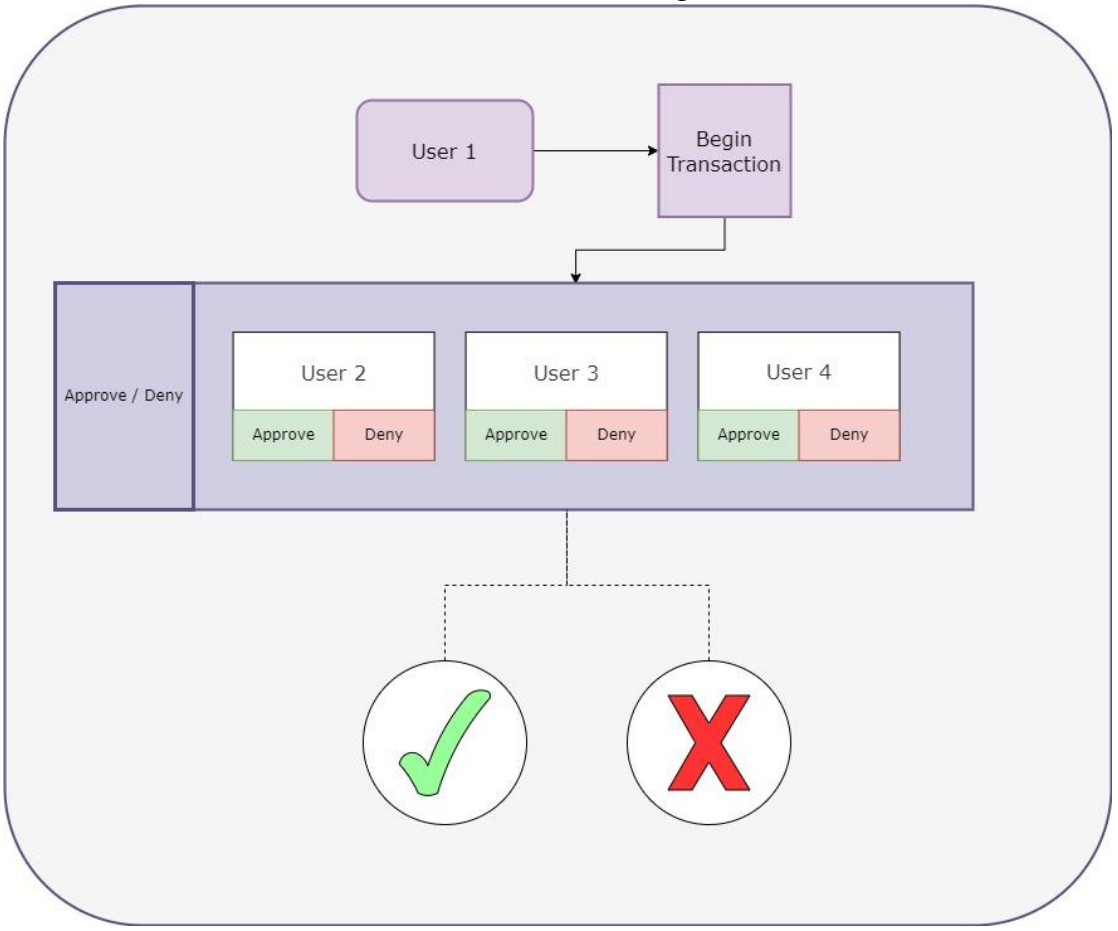


*Fig 5.2*

The Transaction Process diagram (*Fig 5.2*) shows the transaction process in the Multisig wallet. Any user can begin a transaction. They will enter the amount of funds, the address the transaction will be going to as well as a note so others will understand the purpose of the transaction. The transaction will then enter the approval stage. Here, the other wallet owners can choose to approve or deny the transaction at their discretion. If the required number of approvals are not received, the transaction will not be approved and the funds will not be sent.

## 6. Preliminary Schedule

| | Task Name | Duration | Start | ETA |
|---|---|---|---|---|
| 1 | Complete project execution | 230 days | 12/09/22 | 30/04/23 |
| 2 | Preperation | 82 days | 12/09/22 | 03/12/22 |
| 3 | Discuss Project Ideas | 7 days | 12/09/22 | 19/09/22 |
| 4 | Meet Project Supervisor | - | 29/09/22 | 29/09/22 |
| 5 | Complete Proposal Document | 7 days | 11/09/22 | 18/09/22 |
| 6 | Present Proprosal | - | 19/10/22 | 19/10/22 |
| 7 | Complete Functional Specification | 7 days | 4/11/22 | 11/11/22 |
| 8 | Solidity Learning | 21 days | 12/11/22 | 03/12/22 |
| 9 | Project Development | 66 days | 02/01/23 | 09/03/23 |
| 10 | Decide Testing Flow | 2 days | 02/01/23 | 03/01/23 |
| 11 | Decide Git Flow | 2 days | 02/01/23 | 03/01/23 |
| 12 | Smart Contracts | 46 days | 05/01/23 | 20/02/23 |
| 13 | Web Application | 46 days | 08/01/23 | 23/02/23 |
| 14 | Testing | 66 days | 05/01/23 | 12/03/23 |
| 15 | Bug Fixes | 14 days | 23/02/23 | 09/03/23 |
| 16 | Finalisation | 30 days | 01/04/23 | ~30/4/23 |
| 17 | UI/UX Tweaks | 14 days | 01/04/23 | 14/04/23 |
| 18 | Documentation/Technical Specification | 4 days | 14/04/23 | 28/4/23 |
| 19 | Final Delivery | 13 days | 4.05.12 | 28/4/23 |
| 20 | Prepare Presentation | - | 28/4/23 | TBD |
| 21 | Present Project | - | TBD | TBD |

## 7. References


1. MetaMask Wallet Extension
   Link

2. Ethereum Summary
   Link

3. Ethereum System Architecture
   Link

4. Blockchain Information & Tutorial
   Link