University of Nottingham
UK | CHINA | MALAYSIA

G52GRP Interim Group Project

Interim Report

Project Title:

Blockchain-based Dead Man's Switch

Group ID: O

Student Name	Student ID	School of CS Username
Carmel Natasha Barnabas	20509430	hcycb2
Anshana Manoharan	20506329	hcyam4
Lai Ken Siang	20409289	hfykl13
Adyan Dean bin Wafdi	20413774	hfyaw2

Supervisor:

Dr Ioini El Nabil

Date:

14 December 2023

Table of Contents

1 Introduction

1.1 Introduction

Within the blockchain realm, many users are owners of a *blockchain wallet* which can hold a person's funds as well as sensitive content (such as personal documents). For every wallet formed, there is *unique* public key which is inscribed onto each blockchain for every transaction made as well as a private key that is only known to the owner of the wallet. This is where the problem arises; while it is a secure measure, the private key being only known to the owner makes it very difficult to obtain their assets when they pass away. Prior to this, the main method of passing blockchain assets down to another person is by providing the private key within their will however there is a risk of unauthorized access (i.e. through theft) which leads to the loss of funds or illegal transactions.

Thus, to counter this issue, we have proposed a solution that mimics a real-life dead man's switch in a blockchain storage system. To preface, a dead man's switch is found on exercise machines such as treadmills or Stairmasters, but is commonly found on assistive technology such as wheelchairs (Zhang & Hansen, 2022) or on a train's vigilance device (Macii, Avancini, Benciolini, Dalpez, & Corrà, 2014). Its purpose is to halt all processes if the human operator is no longer able to operate the machine. In this report, we will discuss existing solutions, explain the technologies that we will be using as well as an overview of our solution.

1.2 Aims of the Project

Through this project, we would like to propose a new storage space that implements a *dead man's switch*. Following are the aims of this project:

1. To develop a secure and efficient blockchain-based application for transferring ownership of personal data upon the benefactor's passing.
2. To provide a user-friendly interface for benefactors to elect and prioritise beneficiaries.
3. To implement a dead man's switch mechanism to ensure timely access to personal data in case of the benefactor's death.
4. To minimise server-side scripting and focus on blockchain implementation for enhanced security and decentralisation.

1.3 Objectives of the Project

Following are the objectives of this project:

1. Implement the connection of the user's wallet through MetaMask with the application.
2. Design and implement a smart contract that enables the benefactor to register beneficiaries and their priority levels.
3. Store personal data securely on a decentralised storage system.
4. Provide standard uploading, downloading, and viewing of personal data/files for the benefactor.
5. Implement a dead man's switch mechanism through a notification system that triggers if the benefactor remains inactive for a specified period.
6. Provide a user-friendly interface for the benefactor to switch off the countdown if they are still alive.
7. Deploy and test the application on a blockchain network.

Note that for every account created on this platform, one must *instantaneously* assign a beneficiary in the occasion that the benefactor passes away.

2 Background

2.1 Literature review

This digital solution, akin to a dynamic digital living will, constitutes a novel approach to safeguarding personal data and digital assets in the event of the owner's incapacity or death. Such systems, often termed "dead man's switches," activate pre-defined actions upon the owner's absence, ensuring the secure and responsible management of their digital legacy. Smart contracts, self-executing agreements stored on blockchain networks, offer a compelling alternative to traditional legacy systems involving lawyers and wills. Their inherent automation eliminates the potential for delays and exorbitant costs associated with traditional legal processes. Furthermore, the decentralized nature of blockchain networks presents significant advantages in terms of censorship resistance Merre, 2020). Unlike centralized servers, blockchain networks operate without a single point of control, minimizing the risk of data manipulation or access restrictions. This inherent resilience makes decentralized dead man's switches particularly suitable for protecting sensitive data and ensuring the execution of the owner's wishes, even in challenging or unpredictable circumstances.

2.2 Existing Solutions

One existing solution is the Sarcophagus. It is an unaudited decentralized dead man's switch built on blockchain technology. Sarcophagus uses Arweave, a file storage system on the blockchain. Arweave stores data permanently on this blockchain. On the other hand, IPFS, which provides the same services as Arweave, allows data to be deleted from the blockchain if required. Because of this, our project will use IPFS instead of Arweave.

Furthermore, Bequest Finance is an existing solution that serves the purpose of storing financial assets. However, our application is intended for general use and enables users to store personal data. This includes (but is not restricted to) cryptocurrency, NFTs or sensitive documents.

Similar posthumous communication services include *Se eu morrer primeiro* (translates to If I die first in Portuguese). The main difference this solution and our project solution is the use of decentralised blockchain technology. Se eu morrer primeiro ensures to store encrypted user data at two different offline locations. Our project however will be storing the encrypted data into a decentralised blockchain peer-peer network. Additionally, *Se eu morrer primeiro* also requires the user to allocate 2 trustees who will notify the service provider if anything happens to the user. Instead, our project will be minimising unnecessary user interaction (PEREIRA, PRATES, MACIEL, & PEREIRA, 2017). Hence, our project will be implementing a countdown timer that will need to be switched off by the user when it was set on. This proves that the user is alive.

2.3 Technical research

2.3.1 Platforms

The application will be deployed as a Progressive Web App (PWA), enabling users to access its functionalities across diverse platforms and devices (PCs, phones, tablets) through their web browsers. This decision prioritizes accessibility, as users simply need an existing MetaMask account and an internet connection to log in and engage with the platform. Additionally, the choice of Next.js as the development framework further underscores our focus on efficiency and rapid deployment, allowing us to concentrate on core functionalities like uploading and notification systems without compromising on user experience.

Key benefits of this approach include:

- Open Access: Users can access the application from any device with a web browser, regardless of their location.
- Seamless Experience: The PWA design mimics native app functionalities, providing a familiar and intuitive user experience.

- Simplified Authentication: Existing MetaMask accounts offer a convenient and secure login process.
- Streamlined Deployment: Next.js facilitates efficient development and deployment through Vercel, enabling us to deliver the application to users quickly.

2.3.2 APIs used

After thorough research, we have chosen to use a framework called thirdweb. It allows us to easily integrate wallets, deploy smart contracts and pin/unpin files from the IPFS storage. This all-in-one solution facilitates a multitude of functionalities, including seamless wallet connection, effortless deployment, and secure file upload to our chosen storage system, IPFS. We opted for thirdweb due to its accessibility and versatility, offering a centralized hub for these essential tasks, streamlining development, and expediting our app's progress.

Here is how thirdweb contributes:

1**. Wallet integration**: incorporates smart wallets like MetaMask, Coinbase, etc.
2. **Deployment**: easily deploy a smart contract to any EVM compatible network without configuring RPC URLs, exposing your private keys, writing scripts, and other additional setup such as verifying your contract.
3. **IPFS storage**: easy pinning and unpinning of files.

Overall, thirdweb's comprehensive functionality and ease of use have significantly contributed to our development progress, allowing us to focus on core app development.

2.3.3 Storage

Our team has strategically chosen IPFS for content storage, capitalizing on its innovative features and alignment with our app's core principles. IPFS, a decentralized peer-to-peer network, eliminates the need for centralized servers. IPFS offers several potential benefits for dead man's switch dApp, particularly in terms of decentralization, data security, and immutability. Following are some features of IPFS that are advantageous to our project:

1**. Enhanced Security and Transparency**: Data uploaded to IPFS is addressed by cryptographic hashes, ensuring verifiability and immutability. This aligns perfectly with our app's focus on secure transfers between owners and beneficiaries, as any manipulation becomes immediately detectable.

2. **Decentralization and Resilience**: IPFS distributes data across a vast network of nodes, eliminating single points of failure. This enhances resilience and censorship resistance, crucial for our app's reliable operation and accessibility.

3. **Cost-Effectiveness and Efficiency**: Compared to traditional storage solutions, IPFS offers cost-effective data storage and efficient retrieval. Data is automatically replicated across nodes, minimizing latency, and ensuring smooth access for users.

4. **Seamless Integration:** IPFS seamlessly integrates with our chosen framework, thirdweb, enabling frictionless communication between users and the file system. This simplifies development and streamlines data management within our app.

2.3.4 Authentication Methods

One of the most important factors in the entire app is the legitimacy of accounts. It is vital that we will be dealing with verified accounts as triggering the dead man's switch should not be an action that is taken lightly. To provide authentication, we will be working with *MetaMask* (MetaMask, n.d.) which is a browser-based application that allows users to store and manage account keys, make Ethereum-based

transactions and most importantly, securely connect to decentralized apps. We opted for MetaMask as it is an incredibly popular wallet choice which means it can support a large number of users.

2.3.5 Programming languages and libraries

The website itself is built primarily on Next.js, utilizing Tailwind and Typescript. Wallet connections and storage are done through thirdweb, a development toolkit customized for building web3 apps. The backbone of our app is built on an Ethereum Virtual Machine (EVM) which vital in communicating with the blockchain and the deployment of smart contracts. All smart contracts are built using Solidity, as it is made for generating Ethereum-based smart contracts. Using Solidity is more appropriate for our group because web3 development is out of the scope of our knowledge, hence, it is considered a good programming language to start with.

2.3.6 Collaborative software

In order to work on the project together, we are utilizing GitHub to upload our code. Each member has access to the repository – however, since we are all still experimenting with the blockchain technologies, our commits thus far have been minimal. In terms of task divisions, all tasks are verbally stated within meetings or through WhatsApp and is documented in the meeting minutes.

3 Requirements Specification

3.1 Functional Requirements

1. **Benefactor subscription**: the application should allow benefactors to register their wallet.
2. **Beneficiary management**: Benefactors should be able to elect beneficiaries, remove and prioritize beneficiaries.
3. **Data storage**: Encrypted data files should be stored on a decentralised storage system.
4. **Encrypted files**: data files must be encrypted by the benefactor.
5. **Decryption key handover**: Benefactor must securely hand over the decryption key to the beneficiaries.
6. **Dead man's switch**: Beneficiaries should be able to trigger the dead man's switch if they suspect the benefactor is deceased.
7. **Access control**: Beneficiaries should only gain access to data after the confirmation of the benefactor's passing and according to their priority level.
8. **Authentication**: all triggers and interactions with the application should be authenticated with user's wallet.

3.2 Non-Functional Requirements

1. **Security**: the application should be secure and protect user data from unauthorised access, modification, or deleted.
2. **Privacy**: user data should be kept private and confidential so ensure that it is uploaded into the decentralised file storage securely.
3. **Availability**: the application should be highly available and accessible to users at all times.
4. **Performance**: the application should be performant and handle transactions efficiently.
5. **Scalability**: the application should be scalable to accommodate a growing number of users and data.
6. **Usability**: the application should be user-friendly for both benefactors and beneficiaries.
7. **Maintainability**: the application code be well-structured and easy to maintain.

4 Design

4.1 Application architecture

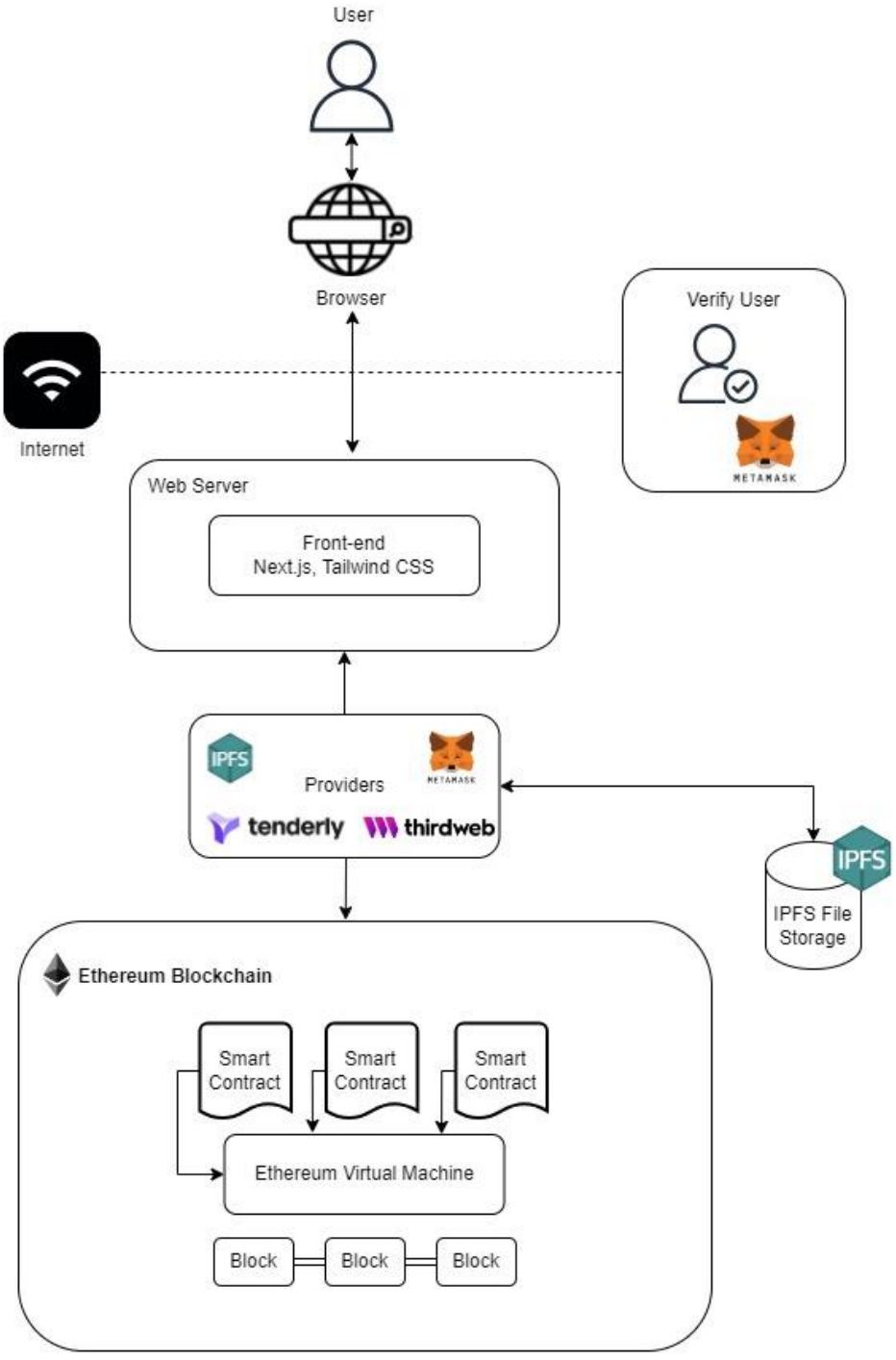


Figure 4.1.a: This diagram demonstrates the main architecture of the dead man switch dApp

4.2 Use case diagram

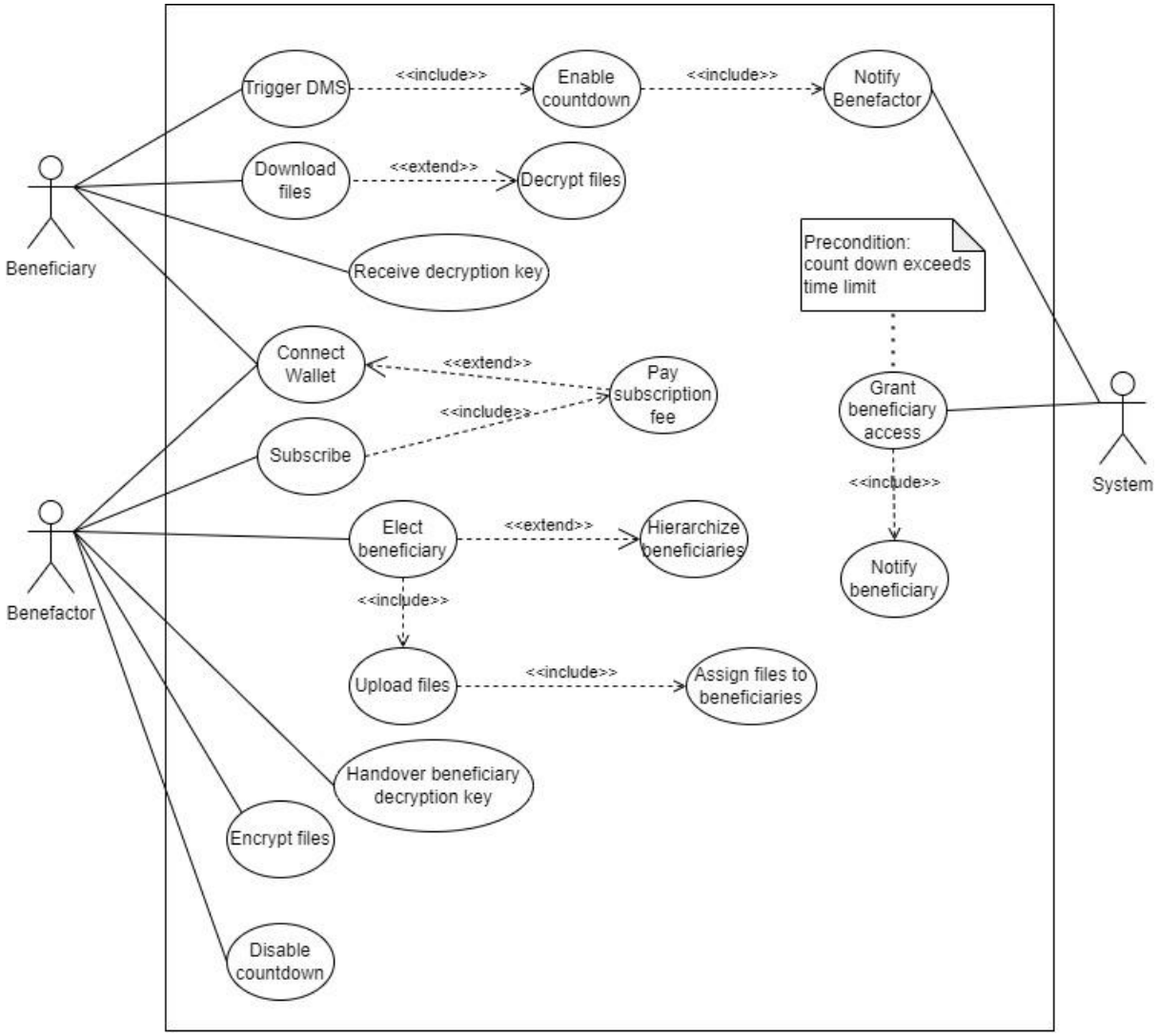


Figure 4.2.a: Use case diagram for the digital dead man's switch application: this shows all the actors and use cases involved.

4.3 Activity diagram

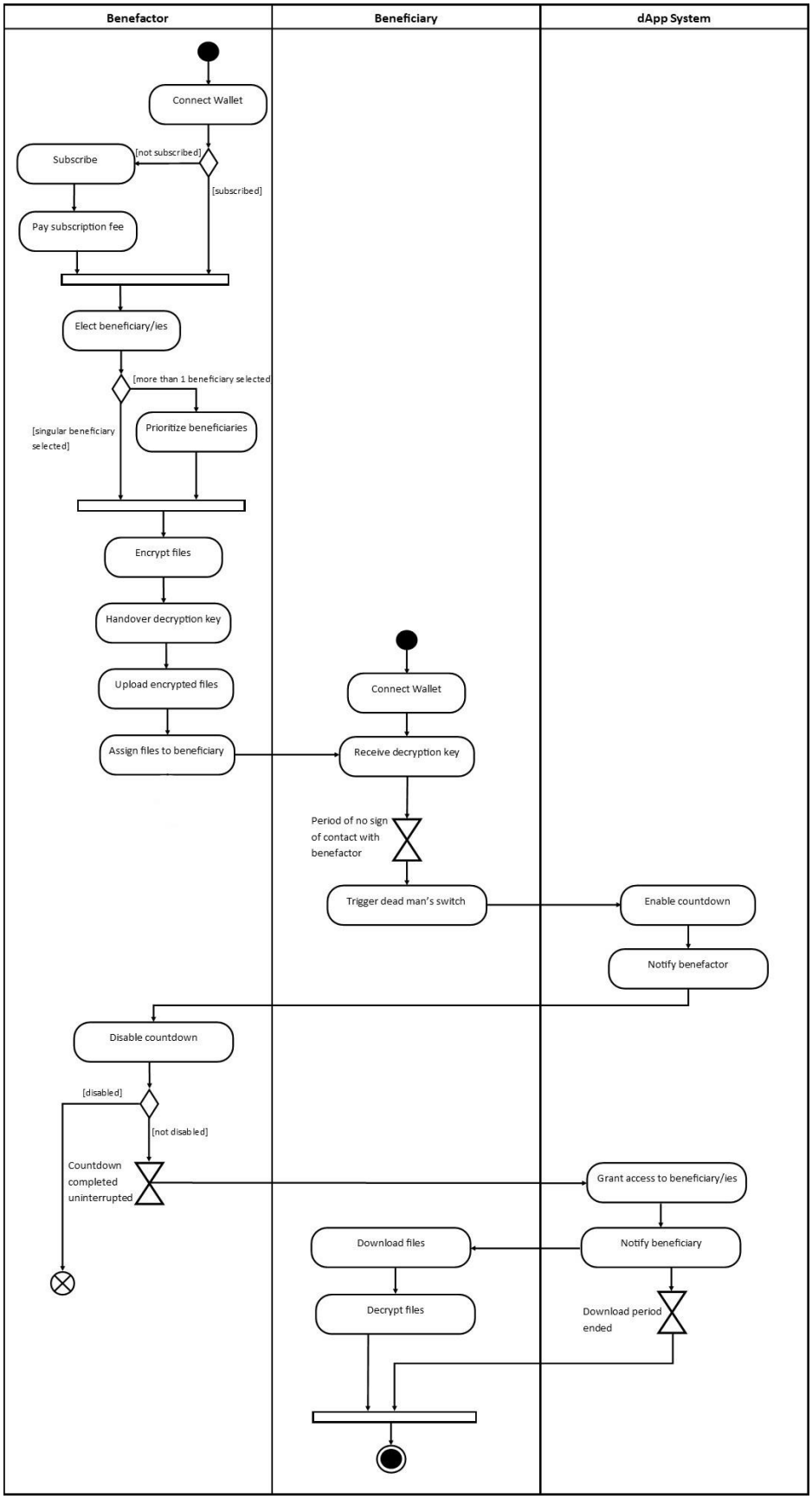


Figure 4.3.a: This activity diagram outlines the flow of actions for the mechanism of the dead man's switch dApp

4.4 Low Fidelity Prototype

4.4.1 New User

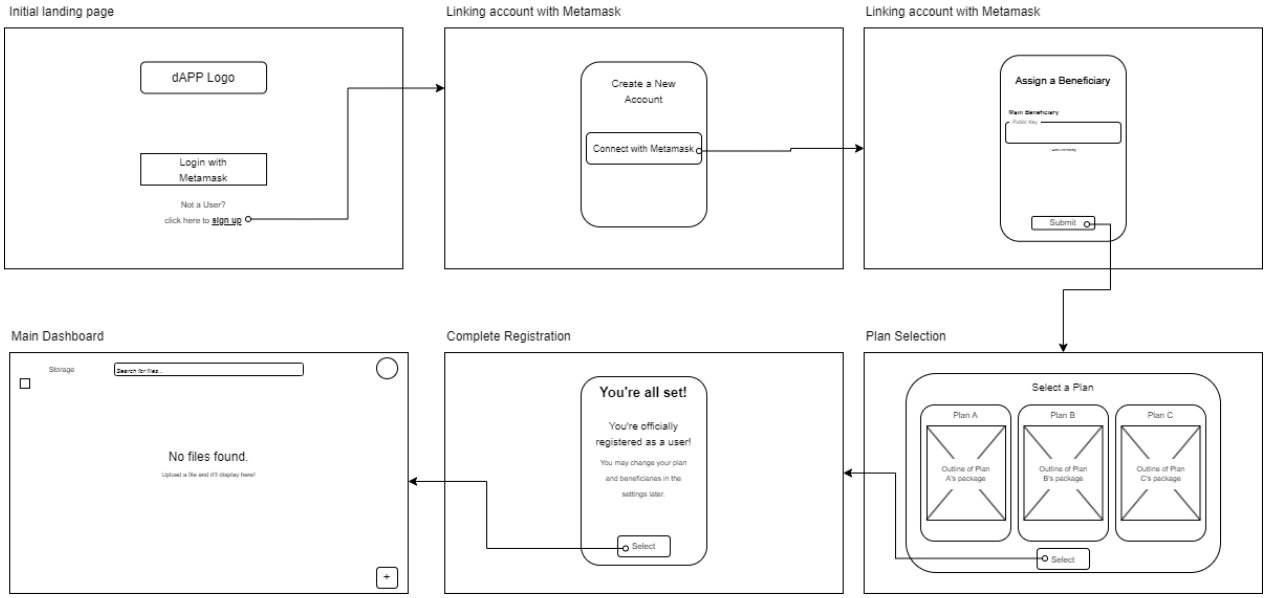


Figure 4.4.1.a: Low fidelity prototype demonstrating the actions of a new user

In Figure 4.4.1a, we have outlined a very simple flow of events that demonstrates the standard actions of a new user. First, when arriving at the landing page, the user will be prompted to sign-up if they have not registered previously. To begin their registration, they first need to connect using MetaMask; to check if the user has already registered or not, the function will compare the chosen wallet address among all the wallets that have been stored on IPFS. Then, the user will be prompted to assign a *main beneficiary* – they also have the option add multiple beneficiaries where they can assign their priority levels, which affects the content that they receive. Next, they are prompted to select a plan and pay using crypto to officially set up their account. Finally, the user will see a completion page notifying them that the registration is complete and will be redirected to the dashboard.

4.4.2 Uploading Files

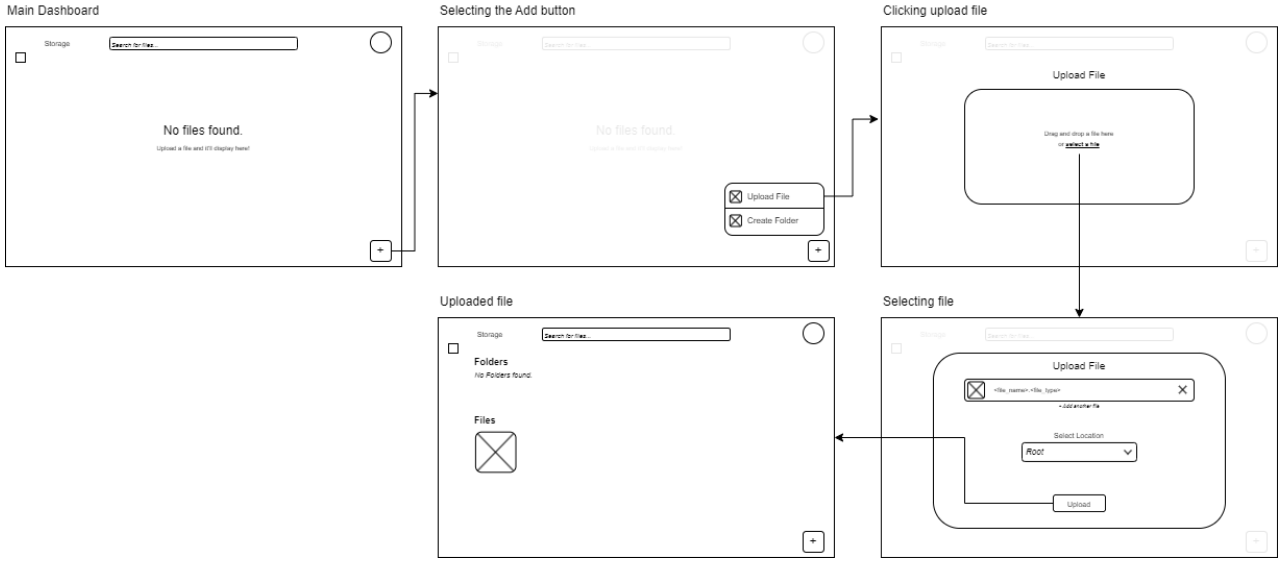
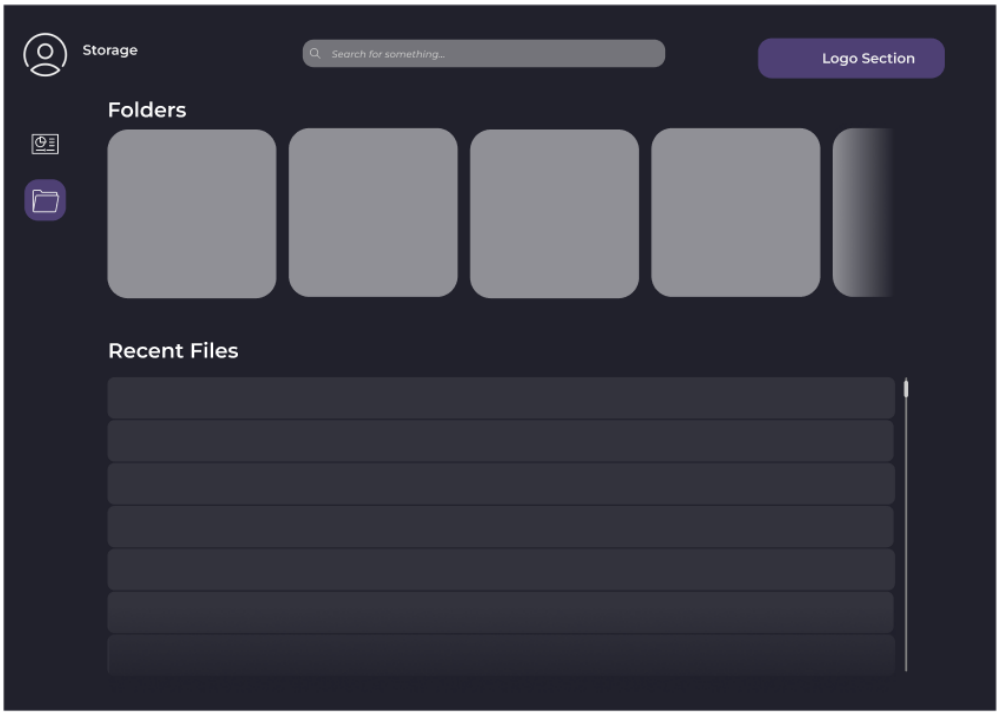


Figure 4.4.2.a: Low fidelity prototype demonstrating the actions of on how to upload files as a first-time user.

Figure 4.4.2.a shows the standard flow of events to upload files. First, the user will have to click the plus icon on the bottom right to open a small menu where they can either 'Upload File' or 'Create Folder'. Upon clicking 'Upload File', they will be prompted a popup where they can drag in files or select from their file explorer before choosing a storage location within their drive. It is worth noting that the main storage location (i.e. outside of creating folders) will always be called 'Root' and that the user must first initialize a folder if they plan to move it. Finally, the file will be uploaded and the user is free to view their file in their storage area.

4.5 High Fidelity Prototype



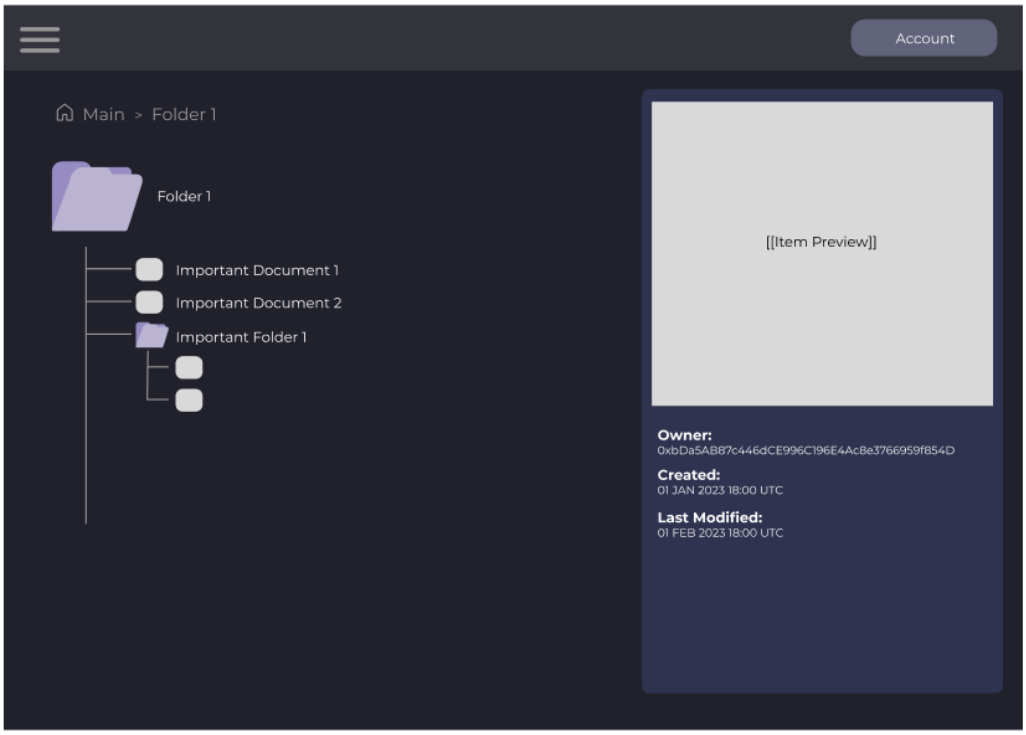


Figure 4.4.a: Drafts of the Storage and Folder designs for the app

5 Implementation

5.1 Implemented Features

5.1.1 Smart contract

In our DApp, the smart contract plays a crucial role in providing the logic and functions that handle the behaviours and functionalities of the application. Since our DApp is in the early stage of development we only focused on the core functionalities. As we continue to build the project, we will be adding more features and implementation to provide better user experience on our application. Our test smart contracts are written in Solidity and deployed on the Goerli test network using Remix IDE. When using Remix, we deploy and run the smart contract in the injected provider environment, which is connected to MetaMask. This allows us to interact with the smart contract using our MetaMask account and simulate transactions in a local development environment. Once the smart contract is deployed, Remix provides "Deployed Contracts" panel where you can see the deployed contract along with the functions. You can interact with these functions by sending transactions and calling read-only functions directly from Remix. The following are the core functionalities of our two smart contracts *OwnerContract* and *BeneficiaryContract*. Note, in the following descriptions, owner and benefactor are used interchangeably.

1. **OwnerContract**

This Owner smart contract is designed to manage ownership and data access control in our application. It ensures that only the designated owner can perform certain actions such as setting and removing a beneficiary, storing, and retrieving encrypted data, and managing countdown related functionalities. Here is a list of its functionalities:

a. **Beneficiary management**: When the user first creates an account, they are prompted to select a beneficiary by inserting their public key (account address), this is done by the `setBeneficiary` function (see Figure 5.1.1.1.a).

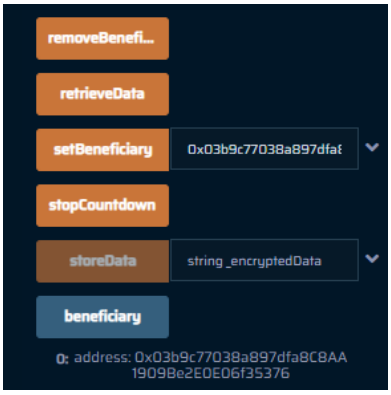


Figure 5.1.1.1.a: Set Beneficiary and Beneficiary Address

b. Alongside this, the user is also able to remove a beneficiary's address, this is done by the `removeBeneficiary` function (see Figure 5.1.1.1.b).

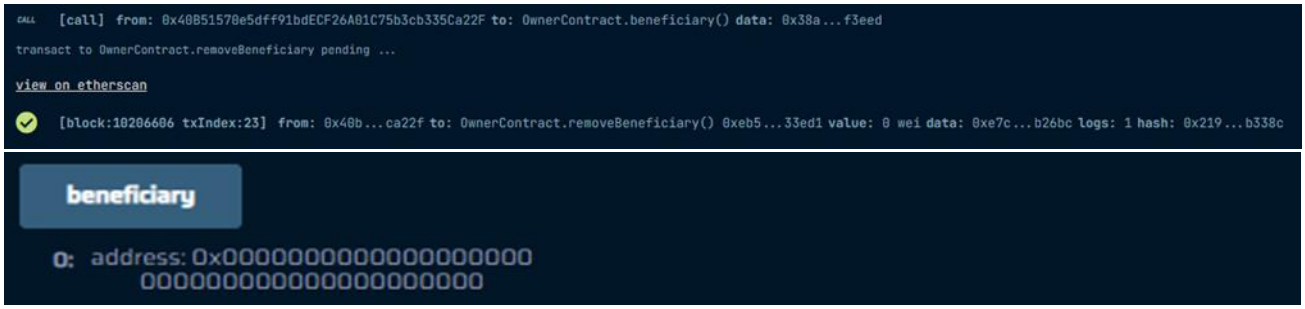


Figure 5.1.1.1.b: Results of the transaction of Remove Beneficiary

c. **Data storage and retrieval**: Alongside this, the owner can store and retrieve the data. `storeData` will be used to the file that has been uploaded while `getEncryptedData` will return what was stored by the user (see Figure 5.1.1.1.c). For demonstration purposes, we will be excluding the implementation of a hashing function as it would be optimal if the user uploaded a file that was already encrypted.

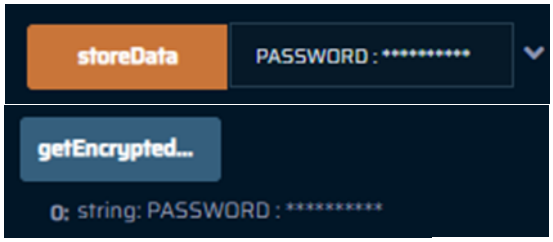


Figure 5.1.1.1.c Functions for storing and retrieving data

d. **Countdown functionality**: A major functionality of our app includes a countdown that begins when a beneficiary triggers the switch. We have included an external view function that allows the owner to check the status of the countdown. The function `countdownDuration` will return the amount of time allocated to the owner for disabling the switch, `countdownStart` enables the countdown and decrements the pre-set countdown value until it equals zero. When it reaches zero with no action from the benefactor, the phase where the account will start terminating begins. Finally, `getCountdown` will take two parameters, `isActive`, a Boolean that signifies if there is an existing countdown, and `timeRemaining`, a value signifying how much time is left in the countdown(see Figure 5.1.1.1.d).

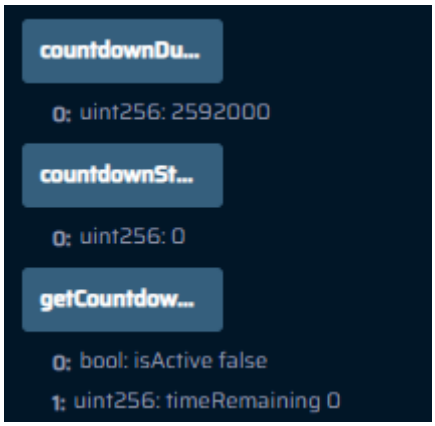


Figure 5.1.1.1.d: Countdown Status

e. **Heartbeat mechanism**: This heartbeat action is to signal that the owner is still active and is enabled when the countdown is enabled (see Figure 5.1.1.1.e). This will update the time the benefactor last accessed the application.

Figure 5.1.1.1.e: Transaction of Owner Trigger Heartbeat

2. BeneficiaryContract

This smart contract is designed to facilitate interactions between a beneficiary and the *OwnerContract* in our application. It allows the beneficiary to trigger countdown, check the status of the countdown, and claim data when the countdown is disabled with no heartbeat sent by the beneficiary. By combining these features, this smart contract enhances the functionality of the application by providing a secure and structured way for beneficiaries to interact with the owner.

a. **Beneficiary (owner) address**: The function `owner` returns the address of the owner.

Figure 5.1.1.2.a: Owner Address

b. **Heartbeat mechanism**: `Owner.lastHeartbeat` is called in the beneficiary contract to check if the benefactor is alive. This is to notify the beneficiary that they need to trigger the dead man's switch. We included a `lastHeartbeat` function in the beneficiary's view from the `OwnerContract` where the function is updated with the latest heartbeat's block timestamp. The purpose of this is to prevent beneficiaries to constantly spamming the switch (see Figure 5.1.1.2.b).

Figure 5.1.1.2.b: Transaction of Latest Heartbeat Triggered

c. **Countdown functionality**: Same functionalities as stated in in `OwnerContract` (see Figure 5.1.1.d). The beneficiary will need access to `countDownStart` (to trigger the dead man's switch) and `getCountDown` (if the benefactor is alive, count down will be reset and `getCountDown` returns 0) .

5.1.2 Notifications

To keep the owner informed about beneficiary actions, we have been experimenting with Tenderly's alert system. The Tenderly alert system allows us design custom prompts and notifications beyond just crypto transactions. Tenderly's "contextual triggers" mean users get alerted for specific events, like a beneficiary triggering the dead man's switch. Figure 5.1.2.a shows an example of how the notification looks like when sent through email, and while Figure 5.1.2.b shows how the notification will show on Discord. We decided to alert and display notification on email, however it may be a feature where the user can select other platforms such as Discord to receiving notifications because it increases the chance of the benefactor seeing the notification and acting upon it.

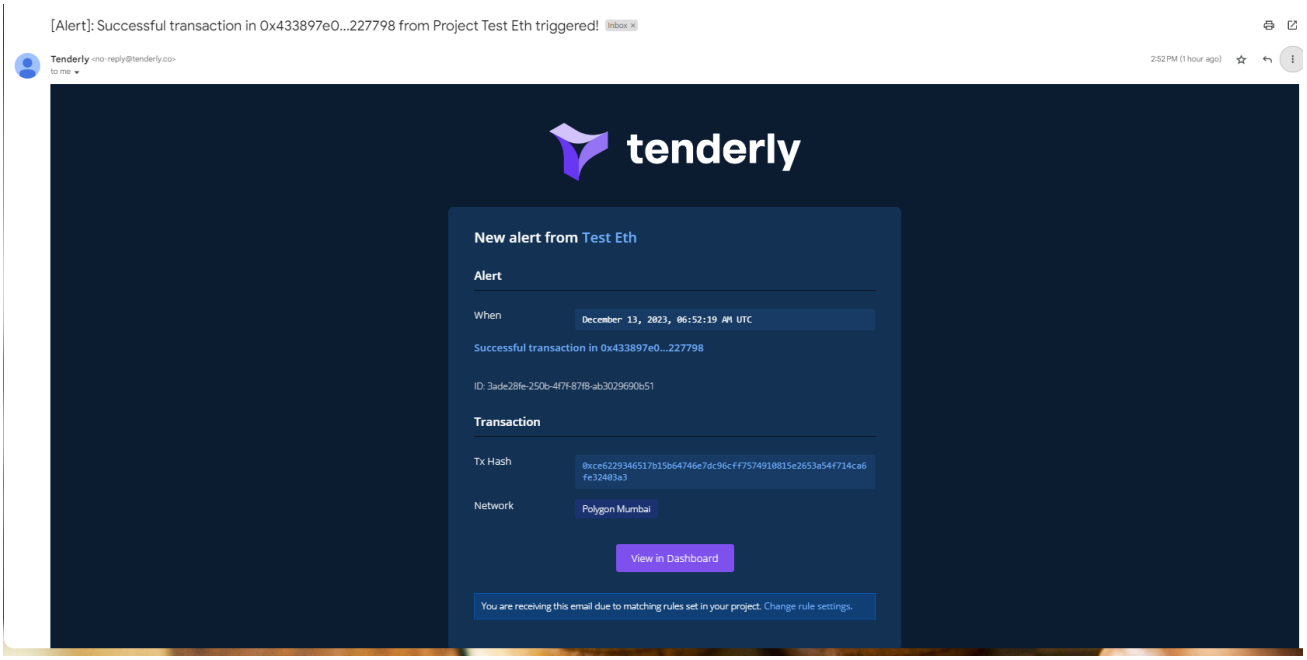


Figure 5.1.2.a Example of tenderly sending an alert to the user's emails

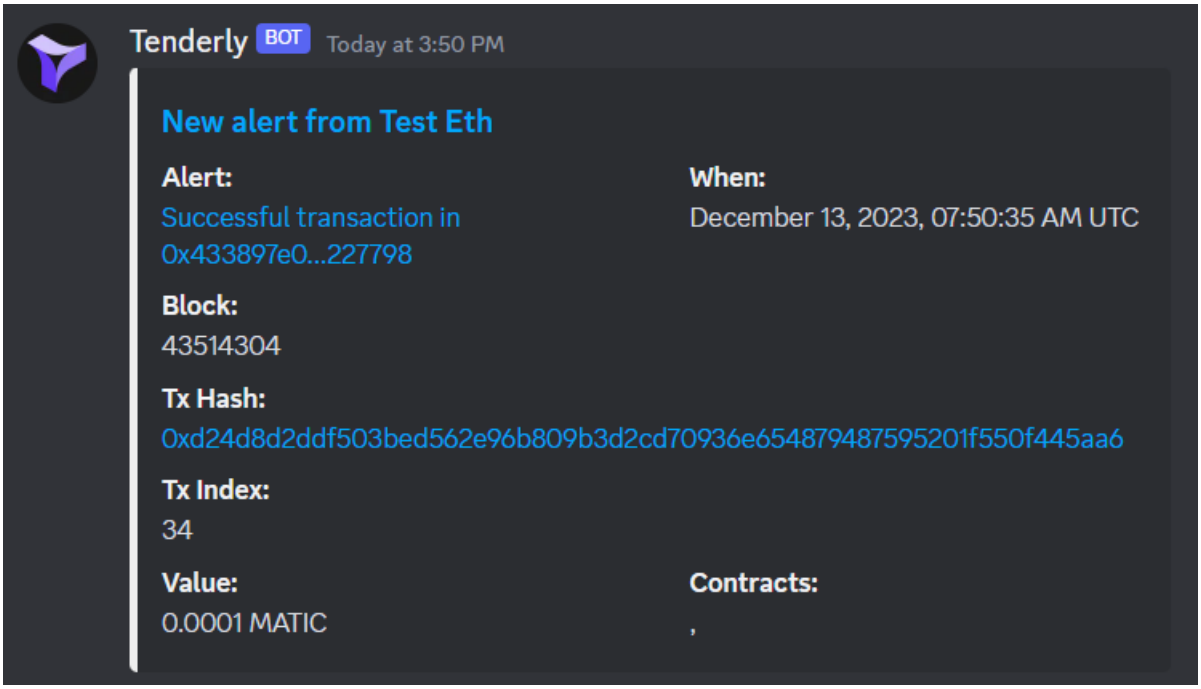


Figure 5.1.2.b: Example of tenderly sending the same notification to Discord

5.1.3 File Storage

The integration of thirdweb has significantly enhanced our platform's capabilities, particularly with file management. Our system now seamlessly allows for file uploads and storage with the added advantage of utilizing IPFS pinning. While the Figure 5.3.a demonstrates the repeated uploading of an image, our system's does actually allow users to upload various file formats beyond images, such as .pdf, .mp3 etc but they also enjoy the convenience of having these files securely stored. Additionally, our platform offers a user-friendly interface that enables easy navigation and display of the uploaded files, ensuring accessibility and convenience for all users.

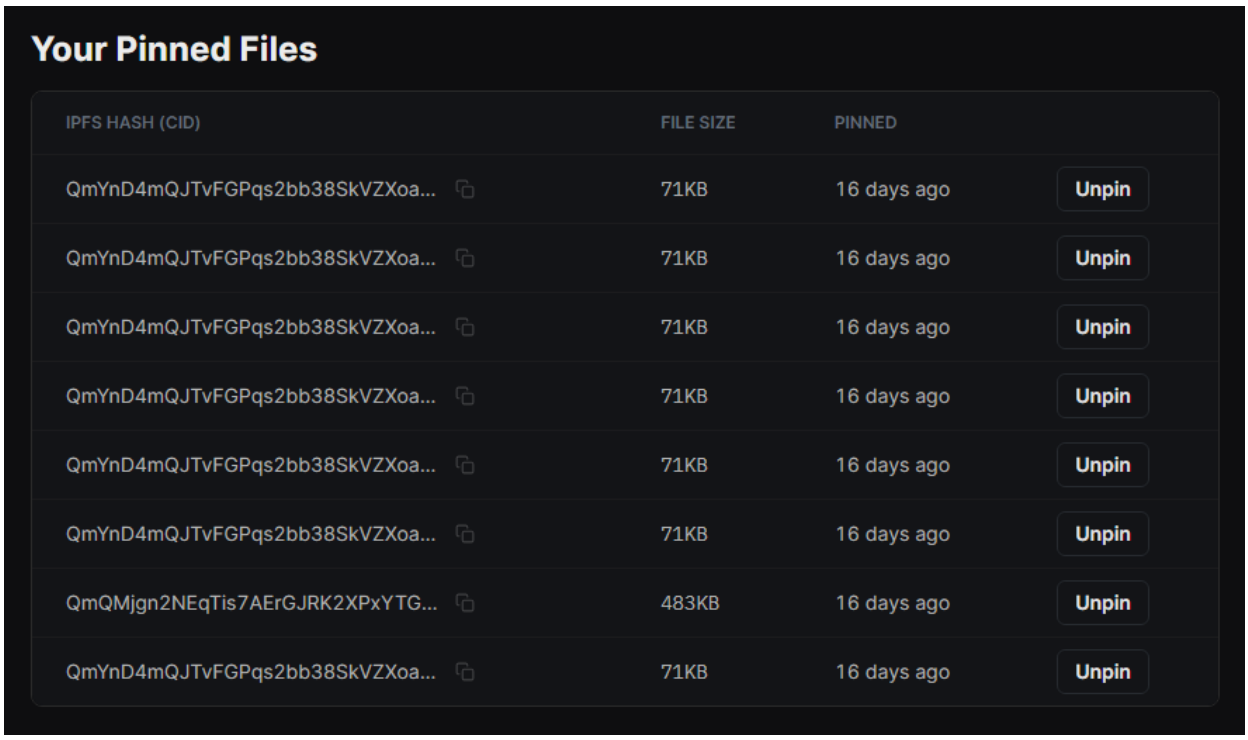


Figure 5.3.a: Example of files being uploaded being pinned to IPFS

5.2 Yet to be implemented

5.2.1 Countdown

The activation of the "dead man's switch" countdown will be contingent solely upon the designated beneficiary's initiation of the trigger mechanism. While the actual implementation of this countdown functionality has yet to commence, our team has conducted extensive research into the potential of utilizing the "Ethereum-Alarm-Clock" **framework** as a viable solution. This framework, if implemented, would enable the display of a dynamic countdown timer within the beneficiary's account interface, providing real-time visual indication of the remaining time before the trigger's effects take place. Furthermore, the framework possesses the capability to prevent the beneficiary from initiating subsequent triggers within a predetermined timeframe, thereby ensuring the efficacy and integrity of the dead man's switch protocol.

The activation of the "dead man's switch" countdown will be contingent solely upon the designated beneficiary's initiation of the trigger mechanism. While the actual implementation of this countdown functionality has yet to commence, our team has conducted extensive research into the potential of utilizing the "Ethereum-Alarm-Clock" **framework** as a viable solution. This framework, if implemented, would enable the display of a dynamic countdown timer within the beneficiary's account interface, providing real-time visual indication of the remaining time before the trigger's effects take place. Furthermore, the framework possesses the capability to prevent the beneficiary from initiating subsequent triggers within a predetermined timeframe, thereby ensuring the efficacy and integrity of the dead man's switch protocol.

5.2.2 Subscription to service

After a beneficiary selects their beneficiary (see Figure 4.4.2.a), they will be prompted to choose a subscription plan and upon subscribing and paying for the service will they be able to upload files to the application. This can be done using the `payable` attribute in Solidity. This payable attribute allows transactions to be made to the smart contract where crypto currency can be sent to the smart contract. Additional functions need to be created to facilitate the need to check when a payment is due after a set time period, ensure the correct amount is sent in correspondence to the chosen payment plan and renew the subscription.

6 Discussion

6.1 Issues faced and solutions/decisions made.

Issue 1: Establishing a secure connection between the wallet and MetaMask:
While attempting to integrate the wallet functionality into the application, I faced hurdles in establishing a stable connection. Although the app technically connects with MetaMask, a crucial issue arose wherein the wallet's address was visibly exposed within the stack.

Solution 1: Additional coding was required to ensure the secure display of the address within the application interface. Resolving this matter involved implementing specific code modifications within the application to safeguard and appropriately present the wallet's address while maintaining a secure connection with MetaMask.

Solution 2: After extensive attempts to resolve the issue, the team engaged in discussions to evaluate alternative solutions. Considering the time sensitivity of the project, the decision has been made to shift towards web development using Tenderly for the time being, solely for the purpose of sending a notifications after the Dead Man’s Switch is pushed. We initially intended to utilize QuickNode for the project's infrastructure, however, encountered consistent issues and inconsistencies, prompting the shift towards

Tenderly for a more reliable infrastructure setup. It further enables us to establish a connection to the smart contract.

Issue 2: MetaMask Mobile Integration:

On desktop browser, MetaMask operates as a browser extension, however MetaMask doesn't work the same way on mobile browser. I attempted to use deep link to open MetaMask app on mobile device, however this approach presents challenges, especially in managing the user's return to react application after authenticating in MetaMask app. For example, after login into MetaMask and return to the app manually, the expected state update (transitioning from *ConnectWallet* view to *OwnerView*) is not occurring. This suggests that the wallet connection status is not being detected or handled correctly upon returning to the app.

Solution: Some potential solutions and decisions made are hardcoded button for manual transition. For example, implement a visible button upon user login in MetaMask to manually transition to the *OwnerView*. We also considered focusing solely on desktop platforms where MetaMask functions as a browser extension. For now, I am experimenting with alternative methods like using *WalletConnect* or enhancing state management for mobile MetaMask integration.

7 Mini-Scrum Implementation

7.1 Gantt chart

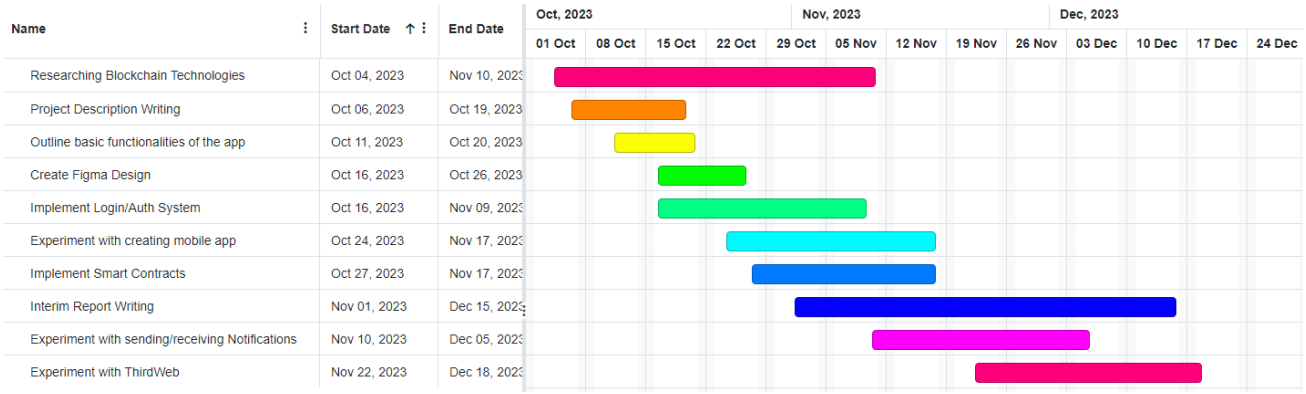


Figure 7.1.a: This Gantt chart demonstrates our expected progress throughout this semester

Figure 7.1.a outlines our expected progress throughout the semester. Although we completed all of the assigned tasks, we did not strictly adhere to the schedule because of a number of issues, including the need to meet deadlines for other modules, the challenge of implementing the features, and some inconsistencies with the logic and flow of the application. We had to rush through the last few weeks because we had to spend longer than anticipated on each section. Before the semester ended, we had hoped to have a small, functioning prototype; but, because of the previously mentioned problems, we were only able to create fragments of the final app that function independently.

7.2 Burndown chart

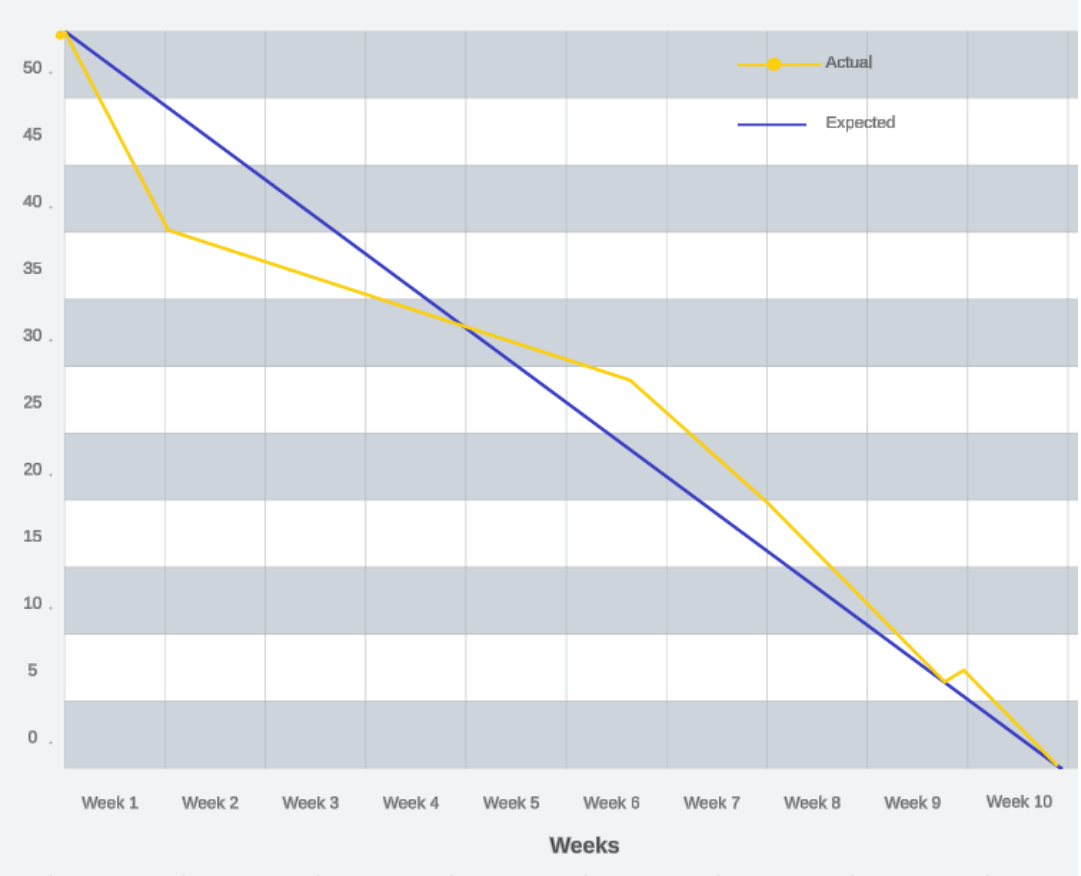


Figure 7.2.a: This burndown chart demonstrates our actual progress for this semester

7.3 Time plan

Note that we received our project description on 3rd October, hence we had 10 weeks in this Autumn semester to work on the project.

No.	Task	Duration	Start Date	End Date
1	Researching blockchain technologies	21 hours	4th October 2023	1st November 2023
2	Finding appropriate APIs	10 hours	4th October	10th December 2023
3	Project description writing	4 hours	6th October 2023	17th October 2023
4	Outline basic functionalities of the app	5 hours	7th October 2023	1st November 2023
5	Design use case, activity, and architecture diagrams	4 hours	10th October 2023	28th October 2023
5	Create Figma design	4 hours	15th October 2023	30th October 2023
6	Implement Login/Authentication system	5 hours	5th November 2023	14th November 2023
7	Experiment with mobile application	8 hours	5th November 2023	25th November 2023
8	Implement test smart contracts	6 hours	20th November 2023	30th November 2023
9	Interim report writing	11 hours	24th October 2023	14th December 2023
10	Experiment sending notifications	7 hours	30th November 2023	12th December 2023
11	Experiment with thirdweb	8 hours	22nd November 2023	11th December 2023
Total Hours		93 hours		

Table 7.3.a: This table demonstrates the tasks allocated and the approximate time taken to complete them

7.4 Mini-scrum implementation

Following the initial project briefing and supervisor meeting, we conducted a detailed role allocation process based on individual skillsets and project requirements. As our four-member team faced a significant workload and new technical territory, both the Scrum Master and Product Owner actively contributed to development tasks alongside their leadership roles. To optimize workflow and maximize efficiency, each team member took on additional responsibilities throughout the project lifecycle.

7.4.1 Role assignment

Carmel Natasha Barnabas was assigned as the Product Owner. Her assigned tasks were to monitor the overall progress of the sprint, maintain the sprint backlog, delegate tasks to the Team, organise formal and informal meetings, chair informal meetings as well as take the role of design specialist and developer.

Lai Ken Siang and Adyan Dean bin Wafdi were assigned as the senior developers, testers and experimental specialists. Their tasks were to develop and test the application, as well as experiment with different APIs and frameworks to find the most suitable and efficient tools to utilise in development.

Anshana Manoharan was assigned as the Scrum Master. Her assigned tasks were to take down meeting minutes, monitor how the Team members are working and whether they require extra help in their task, remove any potential impediments as well as take the role of system architect and developer.

7.4.2 Sprints

Requirements Gathering and Planning (Oct 2023 - Mid Oct 2023):

- Collaborative requirement definition through multiple meetings with the supervisor and team.
- Finalization of the product vision and scope.

Sprint 1 (Oct 4th 2023 - Mid Nov 2023):

- Initial sprint commenced with product backlog creation and role allocation within the team.
- Research and experimentation phase initiated, focusing on identifying suitable APIs and frameworks.
- Intensive research efforts culminated in the selection of appropriate development tools for the project.

Sprint 2 (Mid Nov 2023 - Mid Dec 2023):

- Implementation of test smart contracts to validate core functionalities.
- Continued experimentation with chosen frameworks and APIs for optimal integration.
- Successful development of standalone features like notifications, file storage, and test smart contracts.

7.4.3 Meetings

Project progress was effectively communicated through a combination of formal and informal meetings. Regular updates on sprint progress were provided to the supervisor via three dedicated meetings. Additionally, four informal meetings facilitated team member discussions on individual task progress and fostered collaboration. While the number of informal meetings may appear limited, frequent, and documented communication between team members occurred through WhatsApp. The project's complex nature, requiring access to updated learning resources, posed a challenge for scheduling frequent in-person meetings. To mitigate this, we adopted a meticulous approach to meeting documentation. Detailed records of tasks completed before and after each

meeting were captured in the meeting minutes, ensuring transparency and continuity of progress even in the absence of physical gatherings.

8 Conclusion

In summary, our app development aims to address the persistent challenge of permanent loss of access to content for deceased blockchain users. The solution involves creating a decentralized app with a beneficiary system. Account owners must designate someone to activate the dead man's switch and claim ownership of their content after their passing. The report comprehensively outlines the requirements and technical resources utilized in the project, including implementation details. Despite encountering challenges, particularly due to our team's limited experience with blockchain technologies and its steep learning curve, we persevered and implemented the requirements to the best of our ability. Moving forward, armed with the knowledge gained in working with APIs, writing smart contracts, and mastering basic features, we plan to assemble our app during the break. The goal is to produce a final prototype in time for Presentation Day and Open Day.

Word Count (excluding Table of Contents, Appendix and References): 4856

9 Appendix

9.1 Meeting Minutes

<table>
<tr><td>Meeting type</td><td colspan="3">Formal</td></tr>
<tr><td>Meeting date</td><td>3rd October 2023</td><td>Meeting duration</td><td>1 hour</td></tr>
<tr><td>Minutes prepared by</td><td colspan="3">Anshana Manoharan</td></tr>
<tr><td colspan="4">Meeting objective</td></tr>
<tr><td colspan="4">Receive project briefing and pick project idea</td></tr>
<tr><td colspan="4">Attendees</td></tr>
<tr><td colspan="4">Dr Nabil, Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan</td></tr>
<tr><td colspan="4">Tasks done prior the meeting</td></tr>
<tr><td colspan="4">Nothing as this is the first meeting</td></tr>
<tr><td colspan="4">Agenda and notes, decisions, issues</td></tr>
<tr><td>Topic</td><td colspan="3">Discussion</td></tr>
<tr><td>Choosing a Project idea</td><td colspan="3">Dr Nabil briefed the group on different potential project ideas; application to track and add sCPD points, LinkedIn verifiability feature where professional and educational institutions verify a user’s achievements and a feature which automatically generates a resume, digital dead man’s switch implemented on the blockchain. After thorough discussion, the group chose to work on the dead man’s switch using blockchain technology.</td></tr>
<tr><td>Additional Notes</td><td colspan="3">1. Data can be stored on an existing web3 file storage system like IPFS.
2. Benefactor uploads data to the app.
3. The app encrypts and stores the data into IPFS.
4. The ideology where the beneficiary has the key to the benefactor’s “house” but does not know its “location”.
5. Benefactor physically hands over the decryption key to the beneficiary.
6. If beneficiary suspects the benefactor is dead, the beneficiary must trigger the countdown.
7. If the benefactor is alive, the benefactor must switch off the countdown, else assumed dead.
8. Beneficiary is only directed to location of data after the benefactor is confirmed to be dead.</td></tr>
<tr><td colspan="4">Post-meeting objective</td></tr>
<tr><td colspan="4">Research on blockchain technology, web3 development. Figure out how to implement the smart contracts and familiarize with programming languages to be used. Understand the main infrastructure of the app to be built.</td></tr>
</table>

Table 9.1.a This meeting minutes includes details about project ideation and decision-making: Dead man's switch using blockchain. Post-meeting objectives include research on blockchain tech, web3 development, and understanding app infrastructure.

Meeting type	Informal		
Meeting date	4th October 2023	**Meeting duration**	1 hour
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Brainstorm project idea and get concrete understanding of the objectives and aims of the project			
Attendees			
Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
General research on web3 development. Checked existing ideas of the same idea.			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Features	Discussed all the nice-to-have and must-have features of the app.		
Existing ideas	Looked into existing ideas like Sarcophagus, BeQuest and discussed how differently the project idea could be implemented.		
Additional Notes	1. Users must subscribe to the service provided. 2. Either one or many beneficiaries can be assigned by the benefactor. 3. Beneficiary hierarchy can be implemented where different beneficiaries have different priorities, and based off of their priority, specific data files are assigned to them. 4. For authentication, biometrics can be used (eg: fingerprint/ facial recognition). 5. Users can only use the app on one device at a time. 6. "Back-up" beneficiaries can be assigned, in the case where the direct beneficiary is "unavailable". 7. Notifications can be sent to benefactor to check if the benefactor is alive. 8. Set countdown time for 2 weeks. 9. Set grace period for data retrieval time for the beneficiary after benefactor's death as 2 months? 10. Will the beneficiary gain access to the benefactors account completely? Or will both the accounts be merged? 11. Will a benefactor be able to "blacklist" a certain user who could end up as the secondary beneficiary?		
Post-meeting objective			
Document our understanding of the brainstorm session and start gathering resources on how to implement the application.			

Table 9.1.b This meeting minutes includes feature brainstorming and exploration of existing ideas for the app. Post-meeting goals include documenting insights and gathering resources for application implementation.

Meeting type	Formal		
Meeting date	11th October 2023	**Meeting duration**	1 hour
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Clear up doubts from previous informal meeting			
Attendees			
Dr Nabil, Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
Searched up useful resources to be used for the implementation of the application			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Authentication	Instead of using biometrics, Dr Nabil suggested to use MetaMask wallets		
Grace period	Grace period can be either 1 or 2 months.		
Account merging	Accounts will not be merged, instead benefactor account should be deleted after the grace period is completed.		
Blacklisting	No blacklisting, as priority is only given to the current benefactor's wishes.		
Additional Notes	1. Either use 6 months or 12 months subscription 2. Storej, Filecoin, IPFS are some examples of file storage systems that can be used. 3. IPFS allows the location access to beneficiary after benefactor dies. This is to be done by the smart contract. 4. Backend needs to be used to send out notifications, listen to the smart contract and send announcements. 5. better to use NodeJS and JavaScript for back-end. 6. Beneficiary will decide when to trigger countdown, there will be no reminders for this. 7. After benefactor has passed, there will only be a transfer of ownership, no need to transfer any data files.		
Post-meeting objective			
Start drafting the project description and begin programming.			

Table 9.1.c This meeting minutes includes discussion on authentication, grace period, and account management. Post-meeting goals include drafting the project description and commencing programming.

Meeting type	Informal		
Meeting date	1st November 2023	**Meeting duration**	1 hour
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Update progress and allocate tasks to group members.			
Attendees			
Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
Submitted the project description. Members began testing out smart contracts on Solidity, and tested out mobile-app development using Ionic and React Native. Carmel created the final Figma design to be used.			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Subscription	Discussed whether to accept subscription payment with real-world currency or crypto currency. Final decision to be made after trying out different APIs available.		
Mobile app development	Due to issues arisen by using React Native and Ionic, Adyan was suggested to use Flutter and Dart.		
Figma UI	Accepted UI designs.		
Additional Notes	1. Check if Forging block will accept test Ethers on test networks. 2. Try to send Ethers directly to the smart contract. 3. Mobile application features include MetaMask authentication, viewing account overview, handling notifications.		
Post-meeting objective			
Each group member will work on allocated tasks, experimenting with react.js to open the application as a browser, complete final UI designs on Figma, implement subscription payment portal, work on mobile application development and connect with MetaMask.			

Table 9.1.d This meeting minutes includes discussion on payment methods, technology shift to Flutter, and UI approval. Post-meeting goals involve individual tasks, including experimenting mobile application, payment portals and MetaMask integration.

Meeting type	Formal		
Meeting date	22nd November 2023	**Meeting duration**	1 hour
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Discuss on current progress of the project			
Attendees			
Dr Nabil, Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
Experimenting with different methods to connect MetaMask with mobile app, web3.storage connection, and payment method for subscription.			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Handover of decryption key. Confirm updated use case diagram. Issue with redirecting back to mobile application call with MetaMask data	Required to be confirmed by members and Dr Nabil: both parties need to confirm the handover process, only then, the dead man's switch will be enabled. Concluded that since the beneficiary is a trustee, handover of the key will happen securely. This may be used as a nice-to-have feature. Remove beneficiary subscribe use case. Only benefactor needs to subscribe, because they are the one setting the dead man's switch. Dr Nabil suggested to connect Infura with an API Use the steps in Quicknode article "Connection with MetaMask wallet", and dev.to article "Connecting MetaMask with Flutter".		
Additional Notes	1. There is no requirement to make a mobile app but if the mobile app can provide notifications real time it can be advantageous. 2. Will focus on making the app just to turn off the countdown and for notifications. 3. Research about openzepplin - how to make erc20 token? Can use it as the currency for transactions (buying storage, etc)		
Post-meeting objective			
Update the use case diagram, create the activity diagram and flowchart. Work on how to make the MetaMask connect to the app when redirecting back to the call. Experiment with third web and deploy a test smart contract.			

Table 9.1.d This meeting minutes includes discussion on handover processes, use case adjustments, and technical implementation details. Post-meeting objectives include diagram updates, MetaMask connection experiments, and smart contract deployment.

Meeting type	Informal		
Meeting date	27th November 2023	**Meeting duration**	3 hours
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Discuss progress on tasks assigned at previous meeting and decide whether to continue with the mobile application based on results of the assigned tasks.			
Attendees			
Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
Adyan: Experimented with connections between Flutter and MetaMask for mobile application. Ken Siang: Experimented with connections between Native and MetaMask for the mobile application. Carmel: created basic front-end website (composing of dashboard and storage section) according to Figma design, experimented with thirdweb for uploading and pinning files into IPFS. Anshana: experimented with third web toolkit regarding PWAs, functions regarding subscription implemented but not executed yet, completed requirements specification and activity diagram.			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Final decision for mobile app PWA	Decided not to implement mobile application to send notifications because this can be done using other methods like on the website itself or using other decentralised methods of sending notifications. Research needs to be done in this aspect. Will start experimenting with PWA.		
Additional Notes	None		
Post-meeting objective			
Begin drafting the smart contract			

Table 9.1.e This meeting minutes includes discussion on the final decision for the mobile app and exploration of PWAs. Post-meeting objective involves initiating the drafting of the smart contract.

Meeting type	Informal		
Meeting date	13th December 2023	**Meeting duration**	3 hours
Minutes prepared by	Anshana Manoharan		
Meeting objective			
Finalise the interim report.			
Attendees			
Dr Nabil, Carmel Natasha Barnabas, Adyan Dean bin Wafdi, Lai Ken Siang, Anshana Manoharan			
Tasks done prior the meeting			
Adyan: Used Tenderly to experiment sending email and discord notifications in a decentralised way.			
Agenda and notes, decisions, issues			
Topic	**Discussion**		
Application architecture diagram	Discussed and created the final draft of the application architecture diagram		
Additional Notes	None		
Post-meeting objective			
Research and develop complete understanding of the building Web3 applications and get familiarised with deploying the solidity smart contracts			

Table 9.1.d This meeting minutes includes discussion on the application architecture diagram and prior experimentation with decentralized notifications. Post-meeting objectives include research on building Web3 applications and gaining familiarity with deploying Solidity smart contracts.

References

Macii, D., Avancini, M., Benciolini, L., Dalpez, S., & Corrà, M. (2014). Design of a redundant fpga-based safety system for railroad vehicles.

Merre, R. (2020). Safety beyond the grave with Chainlink and NGRAVE. *ResearchGate*, 8.

Merriam, P. (2015). *Welcome to Ethereum Alarm Clock's documentation!* Retrieved November 02, 2023 from https://ethereum-alarm-clock.readthedocs.io/en/latest/index.html

PEREIRA, F. H., PRATES, R. O., MACIEL, C., & PEREIRA, V. C. (2017). Combining Configurable Interaction Anticipation Challenges and Volitional Aspects in the Analysis of Digital Posthumous Communication Systems. *Journal on Interactive Systems.*, 12.

Zhang, G., & Hansen, J. (2022). Telepresence robots for people with special needs: a systematic review. . *International Journal of Human-Computer Interaction*, 38(17), 1651-1667.