Function Specification

Harry Dunne
C00220865@itcarlow.ie
Supervisor
Patrick Tobin

Contents

Application

Service

Digital Inheritance refers to the transfer of assets which exist on electronic systems. These assets, making up the owner's *digital estate*, include passwords, usernames, online accounts, contracts, receipts, financial transactions, and medical information. The transfer of the digital estate occurs when there is a prolonged, or permanent absence of the original data owner.

This application seeks to address the issue of data inheritance to specific beneficiaries, at a time when the original owner is no longer present.

Target Market

This application can be used to store any information that can be written through text digitally. The market for this technology would be any individual or organisation who has information they wish to bestow to others in case of an unforeseen event happening. However, it is most likely that those in cryptocurrency would find the most use out of such an application, as it allows for the secure transfer of highly sensitive cryptocurrency data.

Success Metrics

The project will be considered a success if information can reliably be secured and recovered with this form of decentralised encryption method. The web application that performs such a task, should be something secure, highly portable, and open source. For this reason the JAMstack technology stack was chosen and the website should adhere to it to be considered a success.

Inspiration

The methods used in this digital inheritance system were not inspired from any other product or application. However, the design of this digital inheritance system went through many iterations where concepts used in other inheritance solutions such as – dead man's switches, blockchain smart contracts, and encrypted database storage; were considered. These methods have flaws and were iterated out of the design of the project until the final design of the project was reached.

Functionality

The service is run through a web application, by doing this, anyone with an internet connection can use the service. In previous iterations of the project design, it was contemplated if an open source desktop application was superior to a web application. This

was because an open source desktop application could guarantee users that no unseen backend code was stealing their input. However, by using a JAMstack website, there is no server-side code. Allowing for the web application to be considered open source and thus publicly auditable.

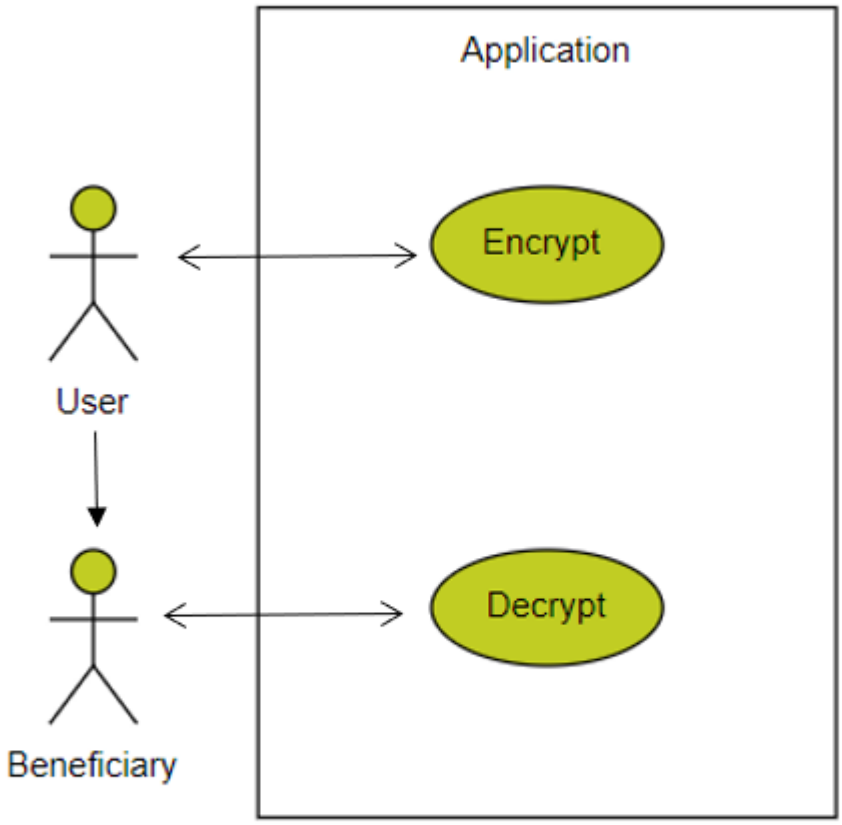
The pages that user input and decrypt their data are specially made so that no inbound or outbound connections are made from the page. The encryption is done locally on the user's end, simulating a desktop application. By doing this, the encryption and decryption web pages can be made into offline portable applications, that can be stored and transported on hardware devices such as USBs, simply by downloading the page itself.

Core Functionality

The user should be able to access the encryption page and the decryption page, depending on which task they wish to complete. On the encryption page, the user will be walked through the encryption process. This process will involve then inputting the information they wish to use in the inheritance system, how many beneficiaries they have, and a final password. After these steps are completed, the user will be given the pieces they need to decrypt the information and told who they should give each decryption piece to.

On the decryption page, a similar process is done, except in reverse. The beneficiaries will first be asked how many keys they have, then the secret password. After this the information will be decrypted, and if the decryption pieces were correct, this should be the correct output, however, if the decryption pieces are incorrect, only ciphertext is returned.

Context Diagram

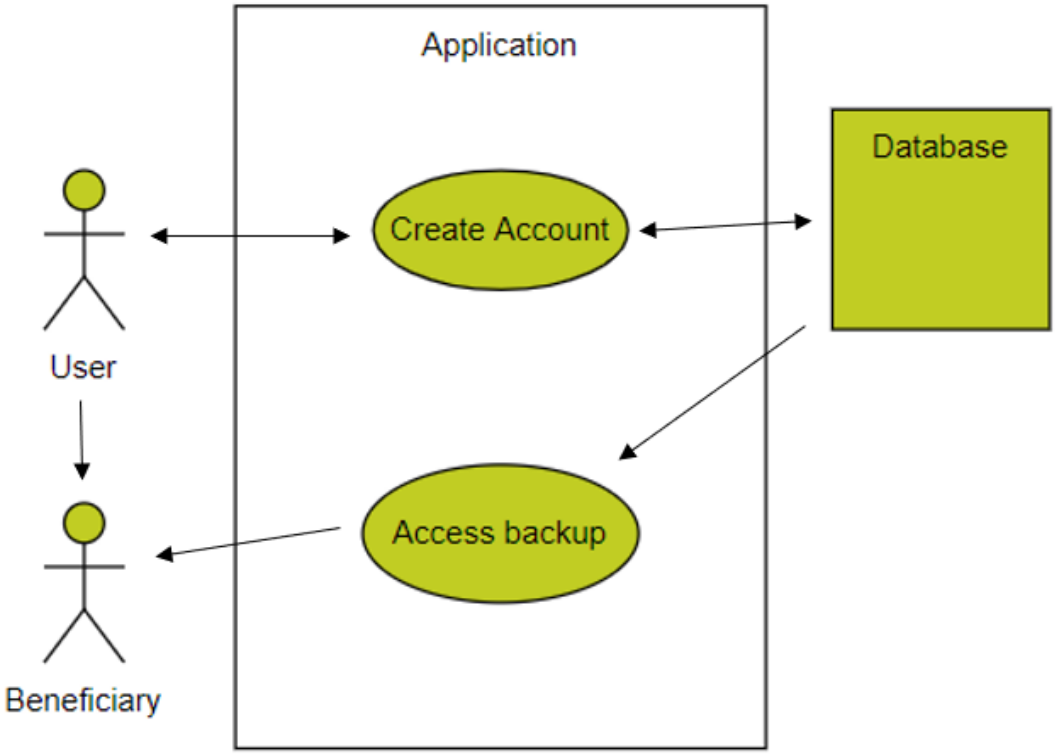


Secondary Functionality

Users will be given the option to create an account on the website. With this account they will be able to store some of the beneficiary's decryption keys. It is safe to do this, as not enough information will be stored in the database so that the user's information can be decrypted.

The storage of decryption pieces adds data redundancy and allows for a backup of that information. This feature can be used if a beneficiary loses their decryption piece and asks the original owner for it again. Beneficiaries who want to retrieve their lost decryption pieces after the owner is gone, may also be given a way to access this information with authentication that only they would know.

Context Diagram



Use Cases

Encrypt

Primary Actor:

User

Preconditions:

The user has accessed the website and is on the encrypt page either by accessing it online or downloading it and accessing it locally.

Success Guarantee:

The user has the information they inputted encrypted and have been returned the decryption pieces they and their beneficiaries will need to decrypt the data.

Main Success Scenario:

1. The user goes to the encrypt page, online or locally.
2. They fill in the user input section and click next.
3. They choose the number of beneficiaries and click next.
4. They choose their secret password and click next.
5. They are presented with all the information they will need to decrypt the data.
6. They give this information to the appropriate beneficiaries and store the rest safely themselves.

Alternative Flows:

2. The user has not entered information
    a. Stop the “next” action and display box telling the user to fill out the field
4. The user does not enter a password
    a. Stop the “next” action and display box telling the user to fill out the field

Decrypt

Primary Actor:

Beneficiary

Preconditions:

The user has accessed the website and is on the decrypt page either by accessing it online or downloading it and accessing it locally.

Success Guarantee:

The beneficiary has the information they inputted decrypted and have the decrypted information returned to them.

Main Success Scenario:

1. The beneficiaries go to the decrypt page, online or locally.
2. They select how many decrypt keys are being used based on counting the decryption keys all of them have. After doing this, they click next.
3. They input all the decryption keys into the boxes and click next.
4. They input the secret password they have received from the original user's will and click next.
5. They are presented with the decrypted information.

Alternative Flows:

3. The beneficiaries have not entered all the keys
    a. Stop the "next" action and display box telling the user to fill out the field
4. The beneficiaries have not entered the password
    a. Stop the "next" action and display box telling the user to fill out the field
5. The decryption didn't work due to faulty information
    a. Present the information, which will be garbled ciphertext, and suggest that they either do not have all the information or they entered some information in wrong.

Login

Primary Actor:

User

Preconditions:

The user has accessed the website and is on the Sign in page.

Success Guarantee:

The user inputs the correct credentials.

Main Success Scenario:

1. The user goes to the sign in page
2. They input their credentials
3. They click the login button
4. They are given an authenticated session and returned to their user page

Alternative Flows:

1. The user enters the wrong username
    a. Display error for wrong credentials
2. The user enters wrong password
    a. Display error for wrong password

Logout

Primary Actor:

User

Preconditions:

The user has already logged in.

Success Guarantee:

The user is logged in and they click the logout button/

Main Success Scenario:

1. The user is logged in
2. They click the logout button
3. Their user session is unauthenticated
4. They are returned to the home page

Alternative Flows: