

Project Report

Harry Dunne
C00220865@itcarlow.ie
Supervisor
Patrick Tobin

Abstract

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

Complications arise in digital inheritance because of two reasons – The information being bestowed is confidential in nature and needs to be stored securely, the date at which the information should be transferred is unknown. In many cases, increasing security of stored data decreases future accessibility for beneficiaries, while increasing the future accessibility harms overall security of the data.

Currently, individuals looking for a digital inheritance solution, need to either intrust their data with 3rd party cloud services, or store their information physically so that beneficiaries can find it in the case of an unforeseen event. Both solutions create single points of compromise, that can be exploited by criminals, digitally and physically.

This project aims to address this problem by providing a decentralised solution to the storage and transfer of digital assets. The solution provided has been designed to give users full autonomy of their data and protect them against both digital and physical attacks. Using this technology, an individual can create a highly secure digital inheritance solution within minutes.

Contents

Abstract	1
Introduction	4
Project Description.....	5
Application Screenshots	6
Index Page.....	6
Encrypt Page	7
Decryption Page	8
Login Page	9
Development Challenges.....	10
Design of the inheritance system	10
Design of the application.....	10
Development of the whole web application	11
Conformance to Specification & Design.....	12
Elements that remained the same.....	12
Secret Sharing	12
User text input.....	12
Beneficiary selection	12
Elements that were changed	12
Application type	12
Blockchain	12
Elements that were deleted.....	13
Smart contracts.....	13
Dead man’s switch	13
Elements that were added.....	13
Helpful user features.....	13
Learning Outcomes	14

Tool options	14
JavaScript.....	15
jQuery	15
Bootstrap.....	15
Personal Learning	17
Project design & planning	17
Layout of the application.....	18
Testing	18
Frontend.....	18
Encryption/Decryption	19
Project Review	20
Successful components.....	20
Unsuccessful components.....	21
Future development changes based on uncompleted components.....	21
Future developments	22

Introduction

This report documents the development process of this project. The project involved the building of a web application that could perform the encryption and decryption of data in a way that would allow the user secure their most important data and split the decryption keys between themselves and beneficiaries. By doing this, the web application aims to solve the complex issue of digital inheritance. It does this in a novel approach, using a purely cryptography-based solution. Using the web application, a user can have their data encrypted, in such a way that they can make it transferrable to selected beneficiaries without losing any security in the meantime. The process had to meet the following specifications:

Criteria	Description
Free	If the service is free
Fast	If the service takes < 1 hour to setup
Decentralised	If the service is decentralised in nature
No single point of compromise	If there isn't a single point that can compromise schen
Open source	If the project is open source
Simple Setup	If the project can be considered easy to setup
Opt-in/opt-out	If the inheritance scheme can be ended at any time
Stores multiple types of data	If multiple types of data can be stored and transferred
Minimal beneficiary input	If the process doesn't involve > 1 beneficiary input
Doesn't need personal information	If the service doesn't require personal information

The first section of this document will describe the final result of the project. It will also include brief descriptions of the technologies used in the development of the application. The application screens will also be included in this section. After this the difficulties encountered during this project will be discussed. Then a comparison will be made between the original project plan and the end product. Finally, a holistic review of the completed project and the work timeline will be made. For more information about the sections of this document, refer to the bibliography.

Project Description

The primary objective of this project was a decentralised encryption stack that could encrypt important data, and have the data recovered at a later unknown date. The application that would perform this action was unknown at first, and only later in designing phase was a JAMstack web application settled on. By using a JAMstack web application, the website is opensource by nature and can be used offline. It also means that the website can be hosted on Netlify CDN for free, which saves costs and gives good worldwide performance. Such an application is highly portable and will only cost as much as the fee for the domain and database storage, which in most cases will be under \$20 a year. Cross browser compatibility was another important consideration for the project. By narrowing down the technology used to a small number of modern technologies and constantly testing compatibility throughout development, the website has good cross browser performance.

The languages used in this project were HTML, CSS, and JavaScript. Other technologies used were –

- jQuery: This technology was used to interact with items on the page in a cleaner fashion than JavaScript alone. JQuery has inbuilt features, which in a few lines, can do event handling and interactions, where in JavaScript, much more code is required.
- Bootstrap: Developed by Twitter, Bootstrap is a framework for front end web development. Within Bootstrap's styles such as colours, margins, grids and more, have been predeveloped. By using Bootstrap, a developer can integrate these preselected styles into their design, for faster development.
- Popper.js: Tooltip and popup framework.
- Font Awesome: Contains many vectorised icons for use in web development.
- Let's Encrypt: HTTPS encryption for the web application.

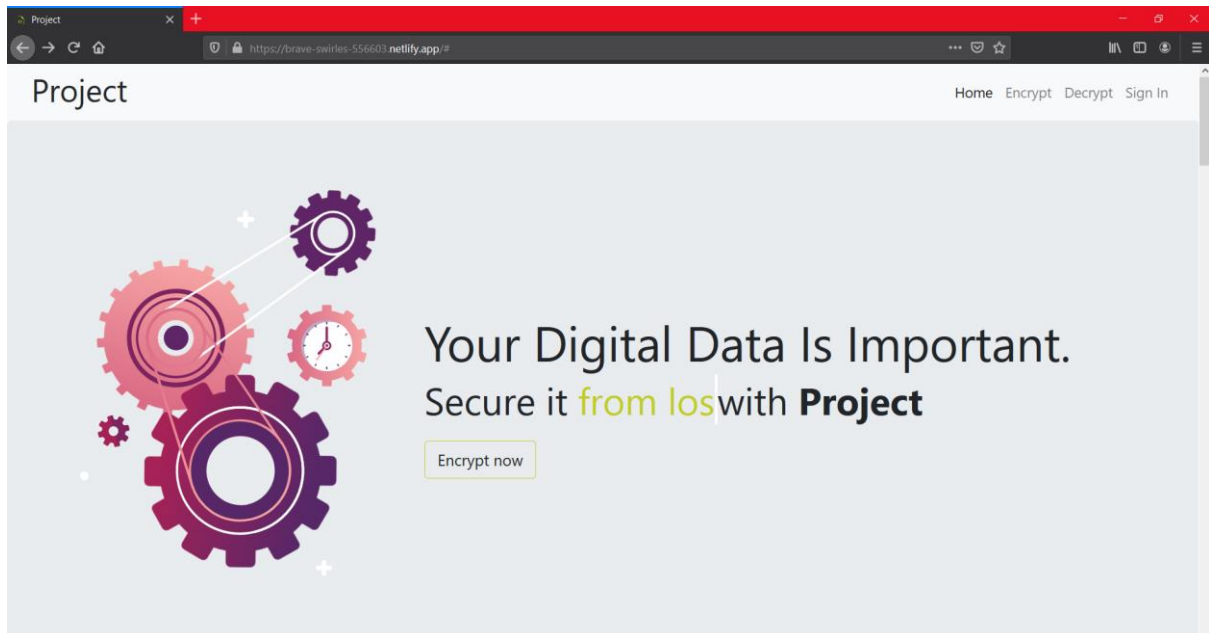
The website for the project is being hosted on Netlify CDN and can be found at:

<https://brave-swirles-556603.netlify.app>

Application Screenshots

Index Page

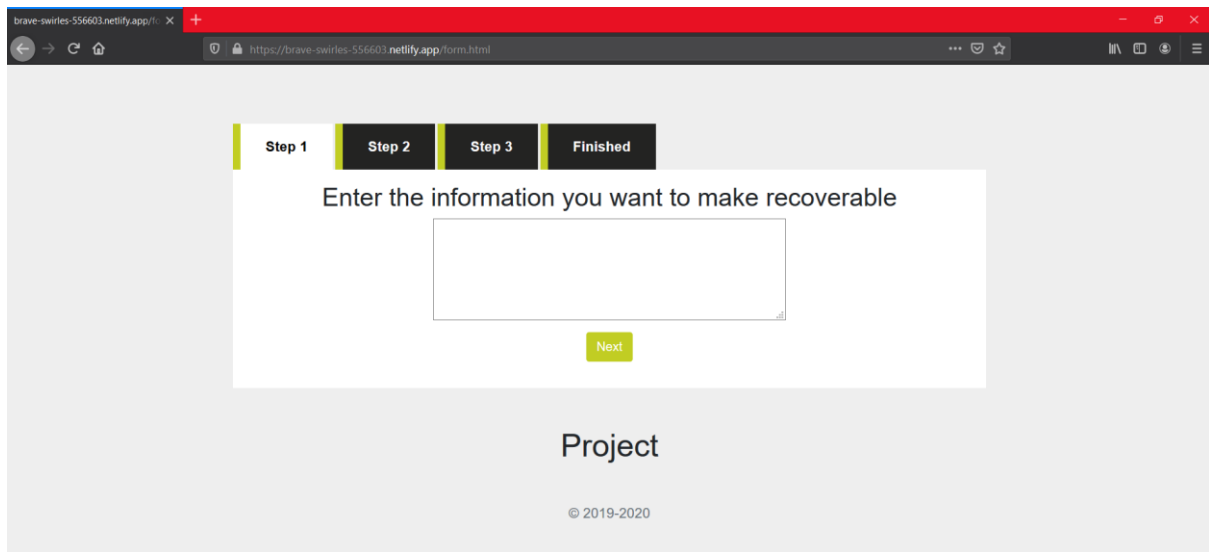
<https://brave-swirles-556603.netlify.app/>



The index page is the home page of a website. It is used as the landing page when a user goes to the domain. It is used to give the users an idea as to what the website is for. It also arranges the useful links and pages in a way that the user can navigate from the home page into other pages.

Encrypt Page

<https://brave-swirles-556603.netlify.app/form.html>

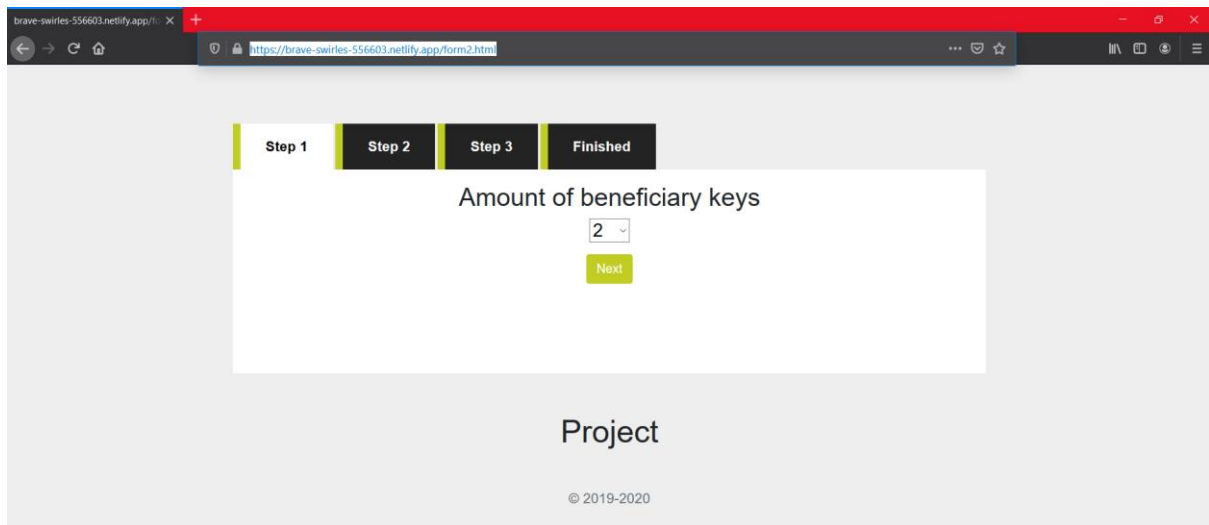


The screenshot shows a web browser window with a red title bar. The address bar displays the URL <https://brave-swirles-556603.netlify.app/form.html>. The main content area features a progress indicator at the top with four steps: 'Step 1', 'Step 2', 'Step 3', and 'Finished'. 'Step 2' is currently active. Below the progress indicator, a white box contains the text 'Enter the information you want to make recoverable' and a large empty text input field. A green 'Next' button is positioned below the input field. At the bottom of the page, the word 'Project' is centered, followed by a copyright notice '© 2019-2020'.

The encrypt page performs all the encryption parts of the project. This page has a simple to understand form, in which the user is guided through each part individually. Once the user has filled out a section of the form, they can click next and it goes to the next part without leaving the page. The user is also able to go backwards without leaving the page. Once the user completes the form, the encryption is done without leaving or refreshing the page.

Decryption Page

<https://brave-swirles-556603.netlify.app/form2.html>



Step 1 Step 2 Step 3 Finished

Amount of beneficiary keys

2

Next

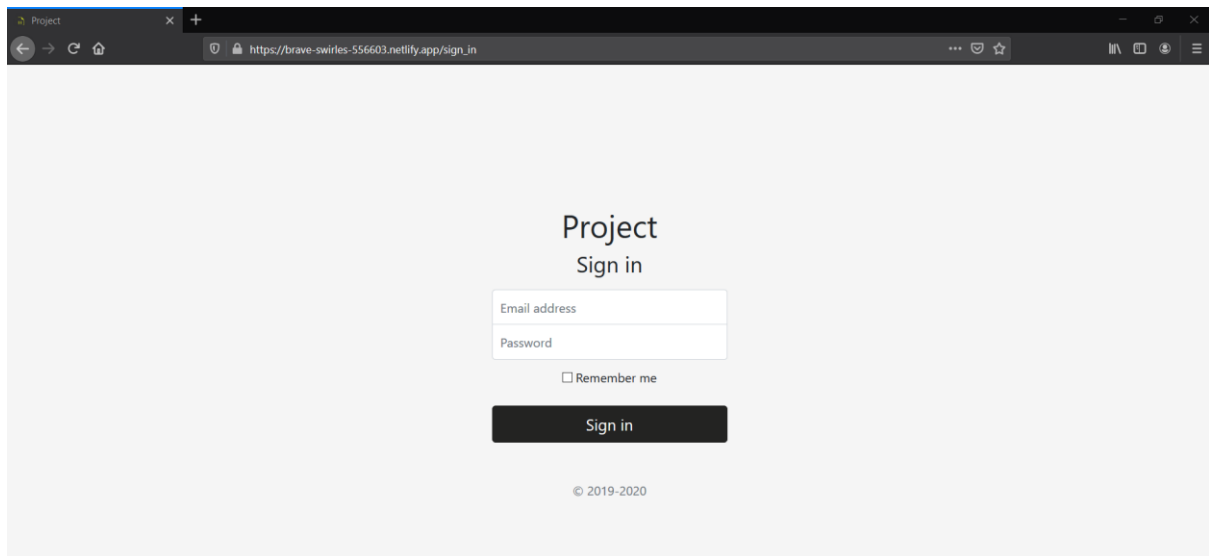
Project

© 2019-2020

The decryption page is split from the encryption page for easier user navigation. The user can decide which page to go to depending on whether they wish to encrypt or decrypt. To keep uniformity, the decryption page is the same design as the encryption page. Except the steps that are completed are changed to be for the purpose of decryption. As the user goes through the form, they are asked for certain decryption pieces. Once finished, the information is decrypted, and the user is returned the decrypted data. If the decryption is not successful as the decryption pieces were wrong, the decryption will not be successful, and the user will have to restart the process.

Login Page

https://brave-swirles-556603.netlify.app/sign_in



This page is designed to be the login page. As of now it is not functional as the backend is not there, however, the frontend is ready and prepared for the backend functionality whenever it is developed.

Development Challenges

Design of the inheritance system

The design of the inheritance system went through many iterations. Original design ideas included components such as:

- Dead man's switch
- Blockchain storage
- Smart contracts

In most cases, when designing digital inheritance systems people, including current alternatives on the market, use some form of dead man's switch. These dead man's switches can be as simple as a program that sends data once they are not interacted with. Or more complex when they are in some form of smart contract that interacts directly with a blockchain. When designing this project, a dead man's switch was the main component, although in later iteration of the design, they were considered far too intrusive and so it was taken out. Being forced to interact every month, week, or sometimes day, with a program that could prematurely release all your information, is unsettling for many users.

For this reason, a passive design had to be made, that was also decentralised and did not rely on database storage. The final method used by the project only requires encryption and ticks these boxes. Refining the design, delayed the start time of this project and was one of the first milestones that had to be broken before any of the development could start.

Design of the application

Like the design of the inheritance system, the application that would perform the encryption had to be decided upon. At first it was believed that it had to be an opensource desktop application, hosted on GitHub or similar, which could be downloaded and checked against a hash to prove authenticity. However, a desktop application brought along new difficulties such as operating system support and performance. Guessing from the way online website-based cryptocurrency wallets are popular, it was theorised that many of the public would prefer to use a website instead of an application they would have to download.

The main issue with using a website was that it could not guarantee the user it was as trustworthy as an open source desktop application. This was until the website was designed with Netlify + JAMstack, in which it would now not be possible to have backend code run, and all

code is instead run on the user's end. With this new design, the project can be within a web application, however, the encryption is still performed on the user's end instead of a server-side backend language. This was deemed to be the best of both worlds and used for the project. The decisions on what technology the application would use, did hold back development of the project.

Development of the whole web application

After it was decided that the project would be website based, there were 3 main sections of the website that needed to be completed.

- Frontend
- Encryption/Decryption
- User login

These 3 parts were all large sections of the project and each required a lot of time and different technical expertise. Having to pivot from designing the look of the website, to developing the encryption/decryption parts of the project, needed different development mindsets. The frontend and encryption development were more favoured than the database and user login development sections. Because of this the 3rd section tended to get less attention than the other two. Balancing the development of all 3 sections was challenging.

Conformance to Specification & Design

Overall, the core concept of the project - an application that can provide users with a digital inheritance system, remained the same and was achieved. Although, many technical elements within the project, did change during later design and development phases of the project. These changes involved adding new elements into the project, changing some elements, and deleting others.

Elements that remained the same

Secret Sharing

As proposed at the start, secret sharing encryption would be used to split keys so that the data can be recombined later when it is brought back together. This element was not changed, even though other extra security elements were added onto the encryption stack.

User text input

The design always had anticipated user input as being done through text. There were difficulties in designing this as the type of text and its data size had to be determined and issues that were caused by this were corrected for. Even with the minor development issues, this element stayed the same.

Beneficiary selection

The system was designed so that the user could select exactly how many beneficiaries they wanted. This was implemented and a maximum number of beneficiaries was set at 20, as it was believed no user would ever require more than that.

Elements that were changed

Application type

The application type was originally designed to be a desktop application, although later testing and development changed into becoming a JAMstack based web application. By doing this, the properties of an opensource desktop application could be achieved, while maintaining the usability, ease of access, performance, and design capability of a website.

Blockchain

Using a public blockchain to store separate decryption pieces was another consideration. But it was later scrapped as it would require users pay in the cryptocurrency that the blockchain was on. A private blockchain created just for the project was envisioned, in which user decryption keys would automatically be uploaded after they perform encryption. This idea

was scraped as it too would require a monetised token to protect against spamming of the blockchain. The blockchain idea was changed to a private blockchain, that the user would not interact with, although one which the database used as a backup. As it was now not necessary for the project to work, its development was moved to a later date.

Elements that were deleted

Smart contracts

Smart contracts are a way in which other blockchain based inheritance systems are performing this form of transaction. While they can give some advantages, such as being able to interact directly with user asset and transfer them when needed. They have many drawbacks, for one they really can only act of cryptocurrency and users who don't use cryptocurrency will not prefer this method. Smart contracts only advantage over traditional dead man's switches is that they can directly interact with assets, for many users this will be considered too invasive.

Dead man's switch

A dead man's switch was the most important part in the initial design of the inheritance system. However, it was later phased out of the design after it got considered too intrusive into the lives of users. Instead a plan which would be passive and wouldn't require user interaction after its setup was opted for.

Elements that were added

Helpful user features

There were many features designed to help users that could not have been predicted in planning. An example of this is the copy buttons and show key button on the finished section of the encryption stage. These buttons are there for the user's benefit, but their inception was not during planning, but during development, where it was discovered these would be helpful to the user.

Learning Outcomes

Tool options

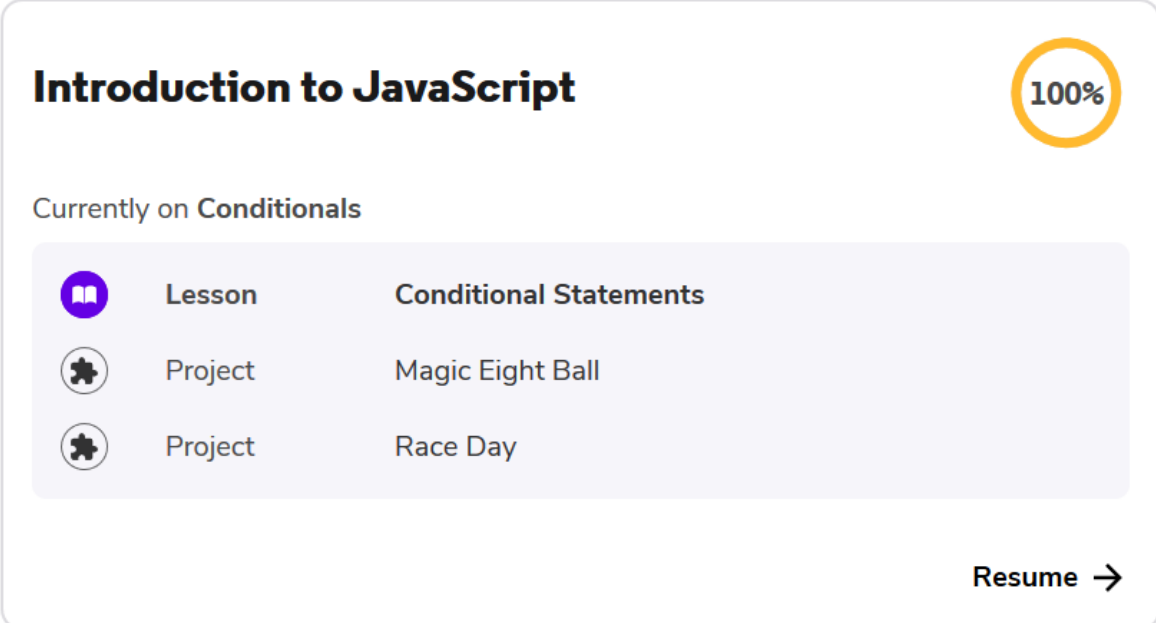
In order to pick out the best tools for the project, research went into the tools that were available for certain tasks. Some of these tools, such as IDEs and frameworks had to be tested in order to determine their usefulness. Some of the things that were heavily investigated during these phases include:

- **Integrated Development Environments:** Many different IDEs were tested to see how they could help in the development of the project. It was good to see the different IDEs for different languages, set them up and compare them to each other. Out of all the IDEs tested VSCode was the preferred one, even to other paid options.
- **Programming Languages:** Comparisons were made between different programming languages, the documentation for them available and the frameworks they had which could help in the project. This helped to get a solid grasp of the programming languages out there, what kind of resources are publicly available for them.
- **Web hosting:** After deciding to make a web application, research into hosting providers had to be done. Not only were the features of different web hosting looked at, but also the price of each. Netlify stood out compared to the others for its modern features and no fees.
- **Frameworks:** Frameworks that could help with this project were looked at. Especially ones that helped with frontend development of desktop and web applications. Seeing different frameworks and researching their performance gave good insight into what is available and which ones should be used for certain tasks.

JavaScript

While previous experience with JavaScript in 2nd year meant that the language was not entirely unfamiliar, it still was not enough to cover everything that was needed in a project like this. The site Codecademy had tutorials on JavaScript which helped to lay a groundwork on this language before attempting the project.

Learn



The screenshot shows the 'Introduction to JavaScript' course page on Codecademy. The title 'Introduction to JavaScript' is at the top left, and a '100%' completion badge is at the top right. Below the title, it says 'Currently on Conditionals'. A list of items is shown in a light blue box:

Icon	Type	Item Name
📖	Lesson	Conditional Statements
🧩	Project	Magic Eight Ball
🧩	Project	Race Day

At the bottom right of the box, there is a 'Resume →' button.

After completing this course, further help and learning of JavaScript was done mainly through Stack Overflow and a mixture of other online sources.

jQuery

Like JavaScript, jQuery was also completely new. Unlike JavaScript, it was only be used in some sections of the website and because of this no dedicated learning resource such as Codecademy was used. Instead when it looked like it could be used for a particular section, W3schools and Stack Overflow were used for help.

Bootstrap

Bootstrap is a framework that speeds up the process of frontend development. This framework has to be learnt from the ground up, in order to become familiar with it. This was done at the start of the project by watching several YouTube videos about Bootstrap and building websites with it. Once there was familiarity with Bootstrap, research into other

frameworks such as Bulma and Boilerplate were done to see if they had better features. It was decided that Bootstrap had what was needed for the project. The Bootstrap documents on the official website were also very helpful to understanding the more complex parts of the framework and were commonly referred to throughout the project.

Personal Learning

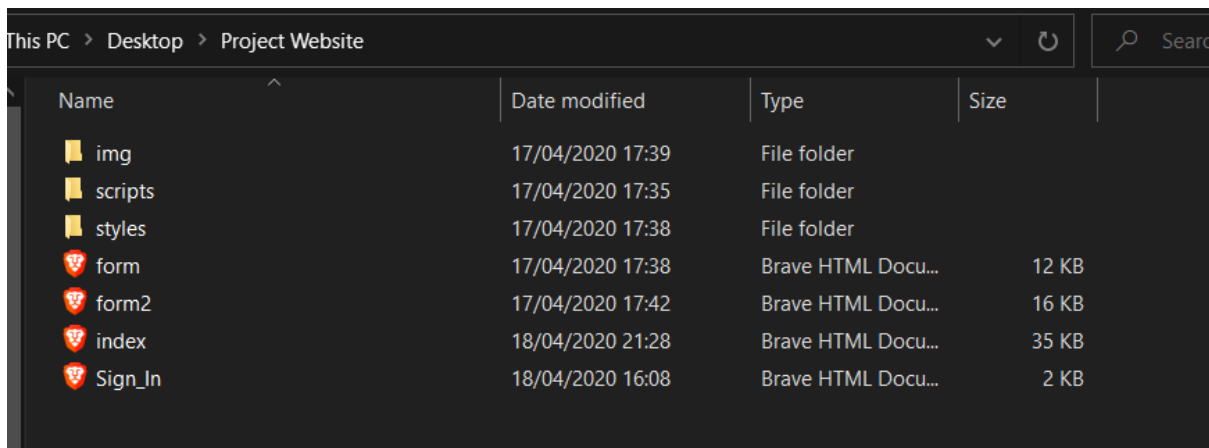
Project design & planning

The main area of personal learning that came from this project was what took up the most time, that being the planning and designing of the project. One of the reasons the design and planning took so long was because I had little experience planning a project of this size before. There are many tools out there specifically designed to help with projects such as these, but without knowing them, they cannot be used. Being able to use what is available to help with project like this, can be the difference between finishing and not finishing the project. For this reason, knowing what tools exists and what they should be used for is a crucial part of any technology project.

In this project much research was done into different technologies that were eventually found to not be suited for this project. Different programming languages and frameworks were tested in this phase, which gave me insight into how these work and what they are good for. However, they were not used in this project because they were not entirely suited for it. This knowledge will come in handy if ever another project is being designed.

Layout of the application

The website file structure has been kept simple and neat; all pages are stored in the lowest path, and their dependencies are stored higher up in the directories. The index page is the main page, which has links to the other pages on it. Images are kept in a separate folder titled “img” this is to keep them organised. Separate JavaScript and CSS files are kept in the directories “scripts” and “styles” respectively. This is to match standard website directory conventions.



Name	Date modified	Type	Size
img	17/04/2020 17:39	File folder	
scripts	17/04/2020 17:35	File folder	
styles	17/04/2020 17:38	File folder	
form	17/04/2020 17:38	Brave HTML Docu...	12 KB
form2	17/04/2020 17:42	Brave HTML Docu...	16 KB
index	18/04/2020 21:28	Brave HTML Docu...	35 KB
Sign_In	18/04/2020 16:08	Brave HTML Docu...	2 KB

Testing

When building a web application there are different areas of testing that need to be done. All these sections were manually tested during the development of the project and in the final stages dedicated manual testing of each section was also performed.

Frontend

The first area of testing was in the frontend as errors in this area are the most obvious to the user. This testing always involves checking website styling across devices, resolutions, and browsers as well as browser versions. This can be a difficult thing to do, and the best practise is to keep good coding standards when developing the frontend. This insures no outdated, or no longer supported code is used. Testing for this section involved using the developer tools, mainly in the element section, on the website within the browser. Using this, different resolutions were checked and the mobile device emulator was used to check styles on these devices. If everything was successful based on-screen resolution, the site was loaded on different browsers to test for flaws. These browsers include: Brave, Opera, Vivaldi, Edge, Chrome, and Firefox.

Encryption/Decryption

This section of testing was similar to the frontend testing, except for the resolution testing. The testing was done by performing the encryption and decryption on different devices and browsers. Like the frontend the main way to ensure no errors in the testing is to adhere to up to date coding practises during development. Features were tested during development regularly to check if they worked. After development testing of the full encryption and decryption features across browsers and devices was done. This was done by loading the site on a mobile device and testing it there. After this worked, multiple different browsers were tested to see if something would go wrong in anyone of them. The tests were successful, and the encryption/decryption worked on all devices and browsers. The browsers tested include: Brave, Opera, Vivaldi, Edge, Chrome, and Firefox.

Project Review

Successful components

The main goal of this project was to come up with a unique way of creating my own preferred digital inheritance system. This system had to be designed in a way that it matched the following criteria-

Criteria	Description
Free	If the service is free
Fast	If the service takes < 1 hour to setup
Decentralised	If the service is decentralised in nature
No single point of compromise	If there isn't a single point that can compromise schen
Open source	If the project is open source
Simple Setup	If the project can be considered easy to setup
Opt-in/opt-out	If the inheritance scheme can be ended at any time
Stores multiple types of data	If multiple types of data can be stored and transferred
Minimal beneficiary input	If the process doesn't involve > 1 beneficiary input
Doesn't need personal information	If the service doesn't require personal information

As it stands, the digital inheritance system meets all these criteria and functions. Although the encryptions section is done, there are a few more design elements that should be added for better usability. These include changes to how the information is displayed at the end of encryption and decryption steps and added tooltips for a better tactile feel when copying keys. Although these are not implemented yet, the inheritance system is usable.

In terms of the frontend design, the homepage is completed and has all the elements on the page to tell the user what the website is about. The frontend is dynamic and works across all browsers and screen resolutions. It has also been made to work on mobile, although that is not the type of device where most traffic should come from.

The technology choice was good, and the site has stayed true to its JAMstack design and is running for free on Netlify CDN. Overall, when it comes to the digital inheritance system and frontend components of the project, they have been successful.

Unsuccessful components

As discussed previously in the documents there were 3 large components of the project that were designed to be completed.

- Frontend
- Encryption/Decryption
- User login

Two of those components, the frontend and the encryption/decryption, were completed quite well. However, the user login which involved the database setup was not completed. This part of the project as of now is unsuccessful. This was due to more focus being placed on the other two elements throughout development and leaving this section far too late. With more time, this section may also have been completed. Although, for what this project aimed to do, it is considered the least important aspect. Even though it is not completed, it will not affect the main aspect, which is the encryption/decryption section.

This part of the project involved learning new technologies such as GraphQL and new services like FaunaDB. Learning this new technology was too much in between the development and implementation of the encryption with the frontend.

In industry jobs involving these 3 components are usually split between people who have expertise in each field. It may have been too much to try and manage full stack web development in the time period given while undergoing other studies.

Future development changes based on uncompleted components

If I was to do the project again, I would split my attention between the front end and user login more evenly. This is because these two components can be considered secondary to the encryption/decryption component and so should have gotten equal attention. While splitting my time between the front end and user login, the encryption/decryption should still receive the same amount of attention as it had gotten during the original development, as it is the primary objective. Outlining the components that need to be completed, ordering them by priority and scheduling time between them would have led to a more even spread of the work completed.

With the added experience I now have with front end development and the encryption/decryption functions, as well as an idea on how project components can be better

prioritised. I believe that if I were to do the project again, it would have been possible to get the user login component to the same completeness as the others.

Future developments

The 3rd component of the web application, user login, still needs to be completed. Its development will be the first to start. Although this won't be the only future development this project has features that were being worked on but didn't make the final cut, so will be brought in later. One such component is a QR code generator for decryption keys in the encryption sections and a QR reader in the decryption sections. By using QR codes, a user can more easily transport decryption keys to the physical world, such as on paper, and then back into the digital world, without having to type the decryption key back in.

During testing QR codes were found to be tricky, as they become less reliable when dealing with larger amounts of data and various camera qualities. I wasn't happy enough with the reliability of the implementation I had for QR codes. As they would rarely become unreadable when generating a code from a large character count key. For the task these codes need to do, it is not acceptable to have potential problems in a rare number of codes. Once better testing has been done to alleviate this problem, such as finding a maximum character count a code can be for real world use, this feature will be ported in.

Another feature that could have been added in but hasn't yet is an offline version of the encryption/decryption area. This offline version can work the same way as a desktop application, in that it can be used offline and transferred to other computers. To make this offline version, CDN based frameworks and assets will need to be moved locally. This is so that when using one of these pages offline, the styling will remain the same. When implementing this feature, other components of the site such as the homepage will also have CDN delivered content, supplied locally to increase load speeds and performance.

As mentioned previously, blockchain was originally intended to be a part of this project but was replaced after its design changed. Blockchain based database backup is a developing technology that is starting to be used in industry, it would be good to see if this project could benefit by backing up the database with an immutable ledger. However, this would first require the development of the database, so the development of the blockchain can only come after that.