Final project vision

I will create a simple password manager that allows a user to easily access and modify their passwords. A user will be able to load their passwords from an encrypted JSON file and the website will interpret it. Before accessing the `, the user must decrypt their JSON file using a master pair of username and password.
Once decrypted, the user can then search through their password file to find the details of the accounts they are looking for. Once the user has found an account, all of its details will be displayed.
The user can click a button to copy their password to the clipboard for ease of use, they can also edit the values and add notes to the account's record. Once the user has made their changes they can click "save" to save these changes to their password file.
The user will also be able to generate a new password based on specific parameters as well as view the strength of their current passwords.
The website will also generate QR codes for each of the user's passwords that can be scanned for use on a mobile device.
I want this password manager to be portable, the user will be able to take their JSON file with them and access their passwords anywhere through the webapp. The website is just an interface to better interact with their password file. This also means that the user is the sole owner of their passwords; they are not saved in any company's cloud storage.

Software development life cycle

In my initial Application Design Document (ADD), I planned to implement the Agile development methodology. For the most part I stuck with this throughout the development of the project.
I worked in 2 week sprints where I implemented new features to the application. All work was pushed to the main GitHub repository for this project. After setting up the core requirements for the app I published it to a separate repository for Netlify (a static site hosting service). With this I could see how the application would look when loaded by a new user on different browsers and devices without having to run the code locally. After each large feature was implemented, I would push the changes to the Netlify application. This allowed me to test the major releases from a published state.

After each major release was made, I carried out a series of tests to ensure that the core functionality of the application was still working as well as testing the new features. This made sure that I would catch any bugs from new features that may affect existing parts of the program.

I carried out the testing in a structure similar to the following (as planned in my ADD):

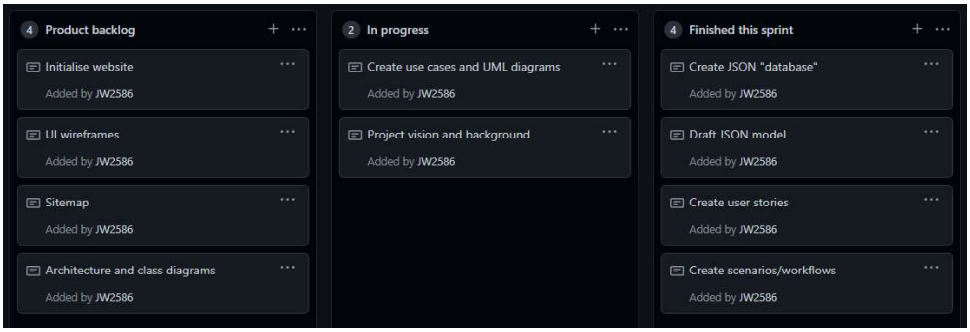
Test no.	Test summary	Test input	Input type	Expected output	Actual output	Pass/fail
1a	Uploading JSON file	Selecting a valid JSON file from dialog box	Normal	Display file name next to upload button		
1b	Uploading JSON file	Selecting a file that isn't JSON	Erroneous	Display an error message		
2a	Decrypting JSON file	Entering correct username and password	Normal	JSON file is decrypted		
2b	Decrypting JSON file	Entering incorrect username and password	Erroneous	JSON file is not decrypted, error message is displayed		
3a	Searching for account details	Inputting a valid account name	Normal	The account's details are displayed		
3b	Searching for account details	Inputting an invalid account name	Erroneous	Error message is displayed		
4	Generating QR codes	Generating the code and scanning it with smartphone	Normal	QR code successfully matches the password		
5	Copying passwords	Clicking the "Copy Password" button	Normal	Correct password is copied to the clipboard		
6a	Generating passwords	Select valid password parameters	Normal	A password is generated		
6b	Generating passwords	Select no parameters	Erroneous	No password is generated		
6c	Generating passwords	Select invalid length for password	Erroneous	Positive value of length is used to generate password		
7a	Saving and encrypting JSON file	Clicking "logout" and entering a password	Normal	JSON file is encrypted and saved to PC		
7b	Saving encrypting JSON file	Clicking "logout" and not entering a password	Erroneous	File not encrypted or downloaded, error message is displayed		

Sprint log

At the beginning of each sprint, I added cards to the “Product backlog” column for each task or feature I wanted to complete for that sprint. If a task had been started it was put in the “In progress” column, once it was completed it was moved to the “Finished this sprint” column.
At the end of each sprint, I evaluated my work and wrote a recap for the tasks I’d completed. I also took a screenshot of the kanban board in its current state at the end of the sprint. I then reset the kanban board for the beginning of the next sprint.
The cards inside the columns are ordered based on how recently the cards have been worked on. Cards are added to the top of each column like a stack.

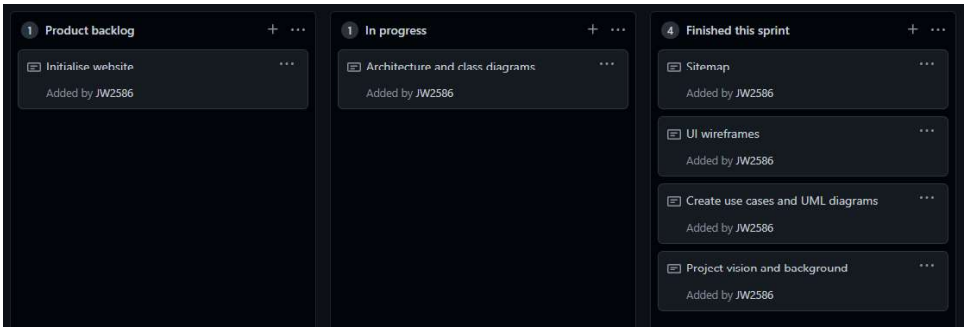
22/11/21 - 3/12/21

I completed researching the background of the project. I finished the user stories and scenarios. I also created some UML diagrams. I have designed how I’d like the JSON file to be structured and I have begun to implement it in JavaScript.



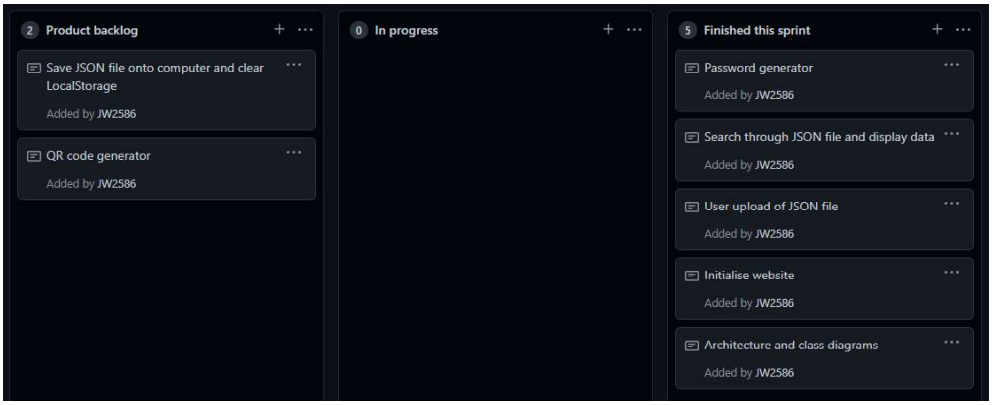
06/12/21 - 17/12/21

I first finished the project vision and background. I then worked on the uncompleted UML diagrams. I have created mockups for the user interface using Figma. I have also produced a visual sitemap that shows the main elements of the single page application. I have created one class diagram.



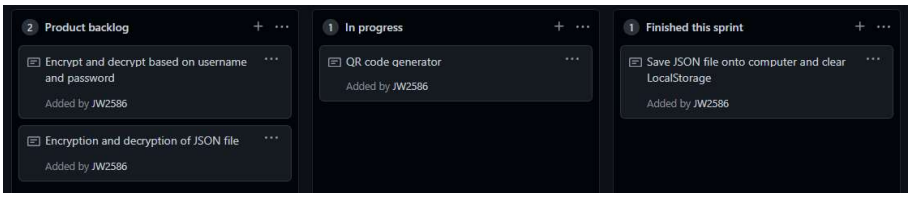
27/12/21 - 07/01/22

Added the main initial features of the website. My website can load the JSON file into LocalStorage and then I can search for a specific record in the JSON file. I also implemented the basics of the password generator.



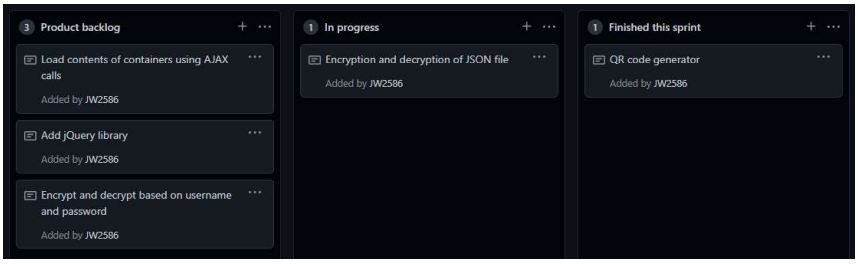
10/01/22 - 21/01/22

Added the ability to save the user's JSON file back onto their computer; this also clears LocalStorage.



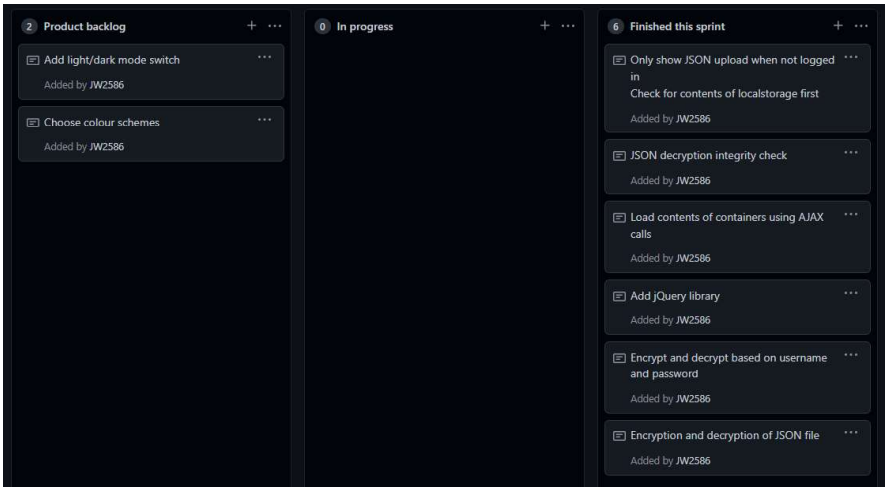
07/02/22 - 18/02/22

Added the QR code generator; generates a QR code every time the user generates a password, for use on mobile devices.



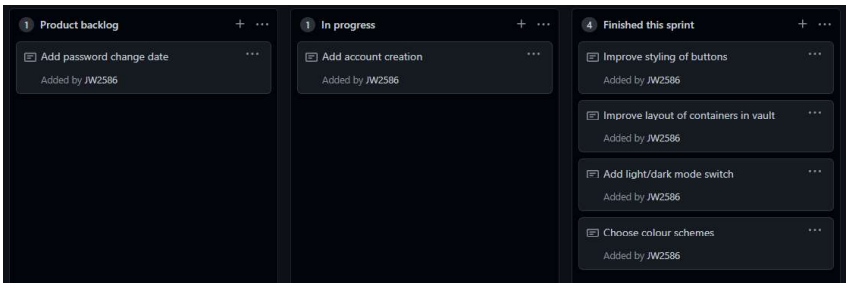
21/02/22 - 04/03/22

The user can now enter a username and password when they upload the JSON file. These credentials are used to decrypt the encrypted JSON file. I implemented the CryptoJS module to do this.
I also refactored the structure of the website to use jQuery and AJAX to implement the SPA. This debloated the site and separated the pages into different files.
The username was added to the beginning of the decrypted JSON file in order to verify that it has been decrypted successfully.
I also corrected the startup of the SPA so that the user was directed to the right page depending on whether or not they had already uploaded their password file.



07/03/22 - 18/03/2

Updated the user interface for the application. This involved deciding on light and dark colour schemes and implementing them in CSS. I also restructured the layout of the containers to display the information in a more aesthetically pleasing way. I created cohesive styles for all of the buttons used on the website.
I implemented the account creation feature allowing for the user to add new records to their password file.



21/03/22 - 01/04/22

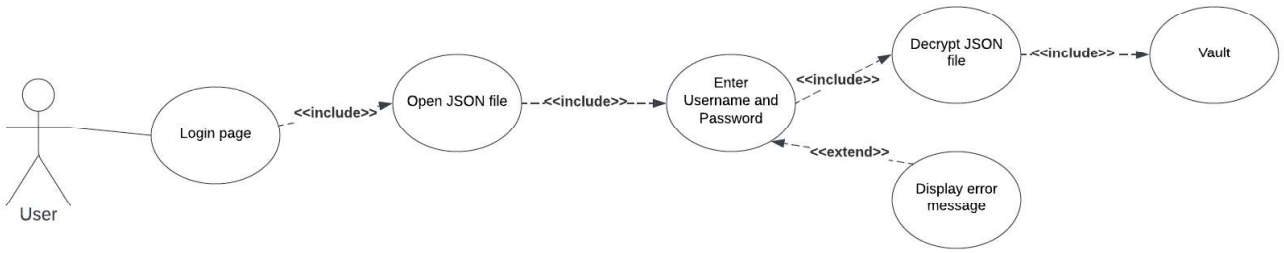
I added a new record to the account template that stores the date of the last time a password was updated. This allows for the user to keep track of how often they are changing their passwords.

18/04/22 - 30/04/22

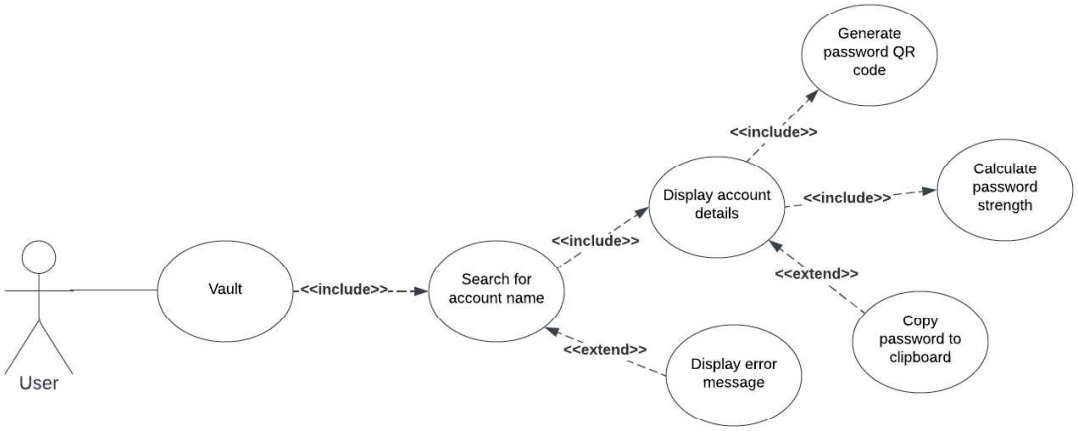
Added an error message for if the user enters the wrong username or password during the login phase.
Final tweaks to the user interface as well as fixing some bugs.
Cleaned up the code ready for submission e.g. removing console logs etc.

UML diagrams

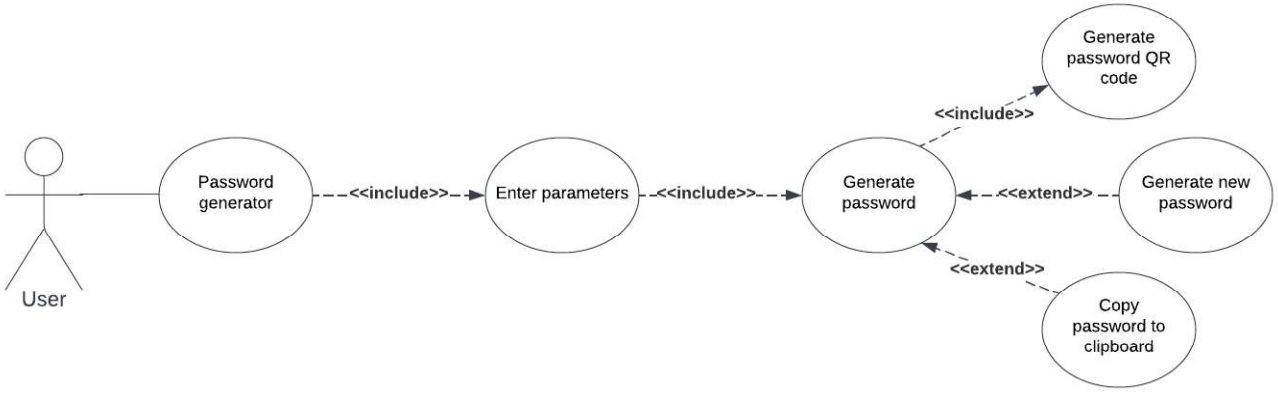
User login



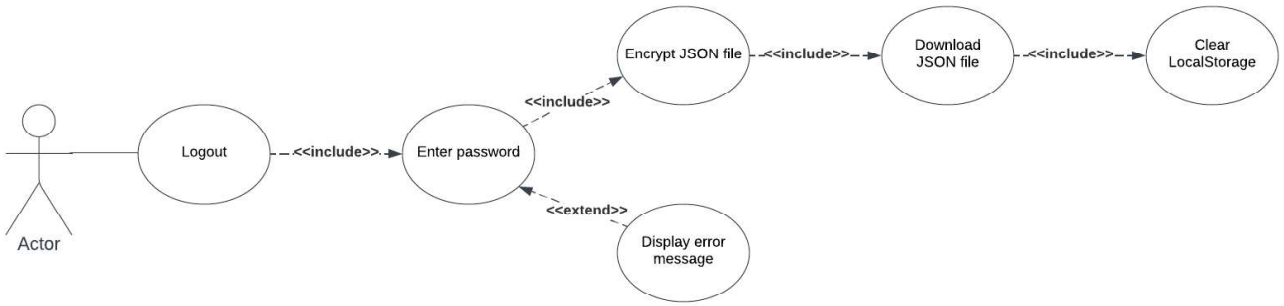
Vault page



Password generator page



Logout

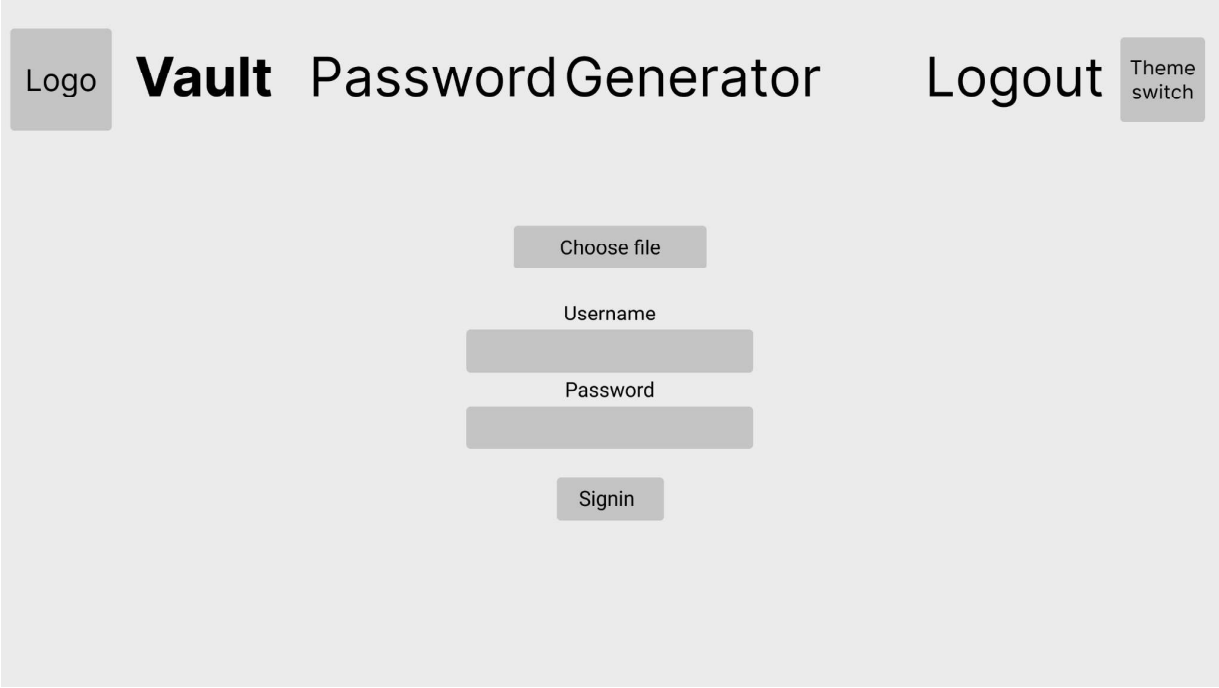


Design

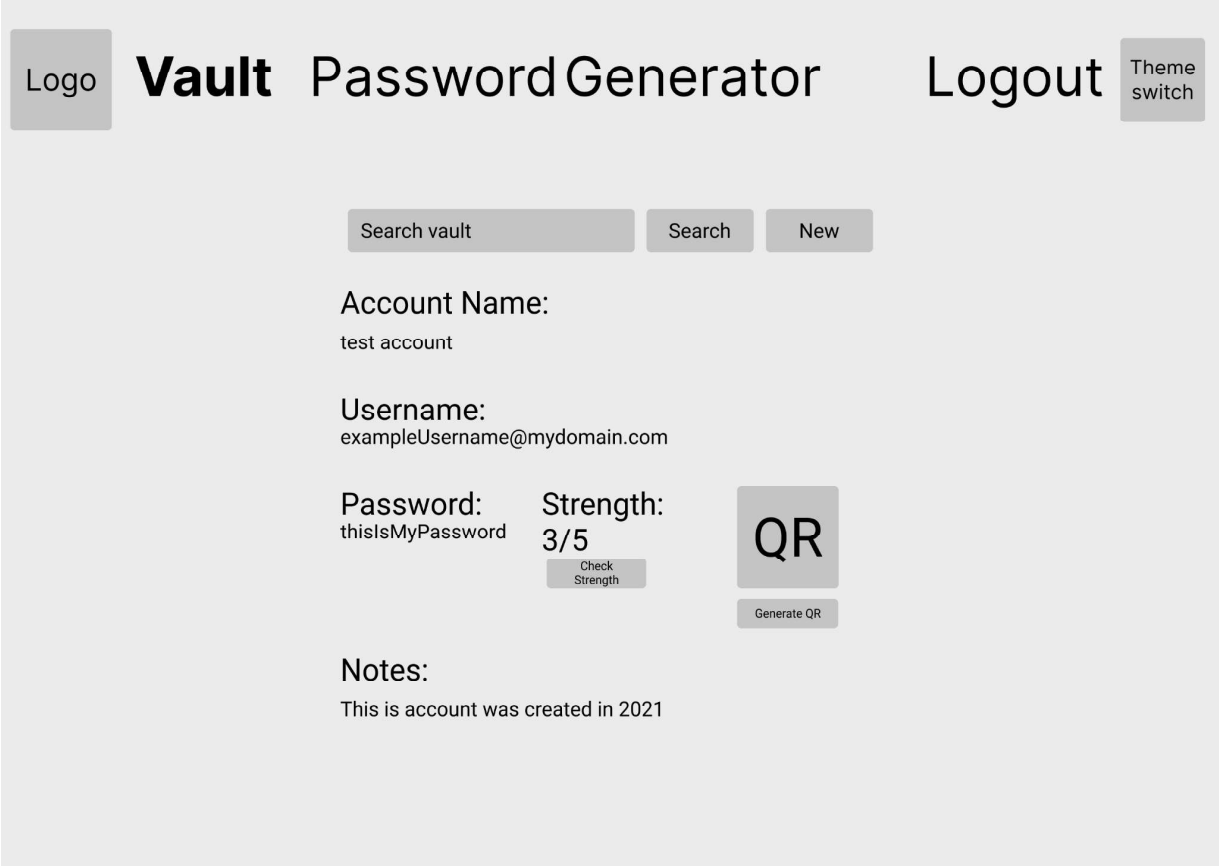
User interface

Since the initial prototype, I have updated the user interface. Notably restructuring the navigation bar and adding a light/dark theme switcher.
The navbar features the NewPass logo on the far left hand side. The 2 main page selectors are also on the left to signify their importance. The Logout button sits on the far right hand side, this is to make it seem less significant. Websites typically place logout buttons at the end of menus to suggest that it is something to only be pressed at the end of your session and to discourage users from clicking it. This will keep users on your website for longer, often generating companies more money.

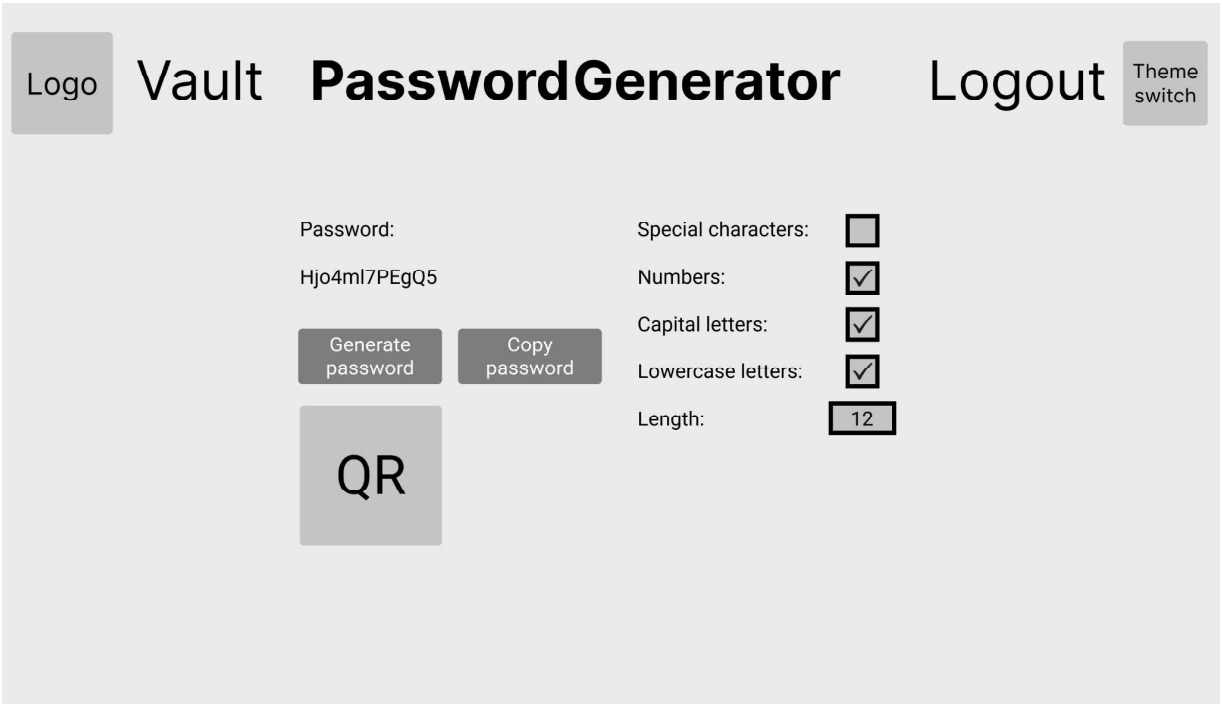
Login page



Vault



Password Generator

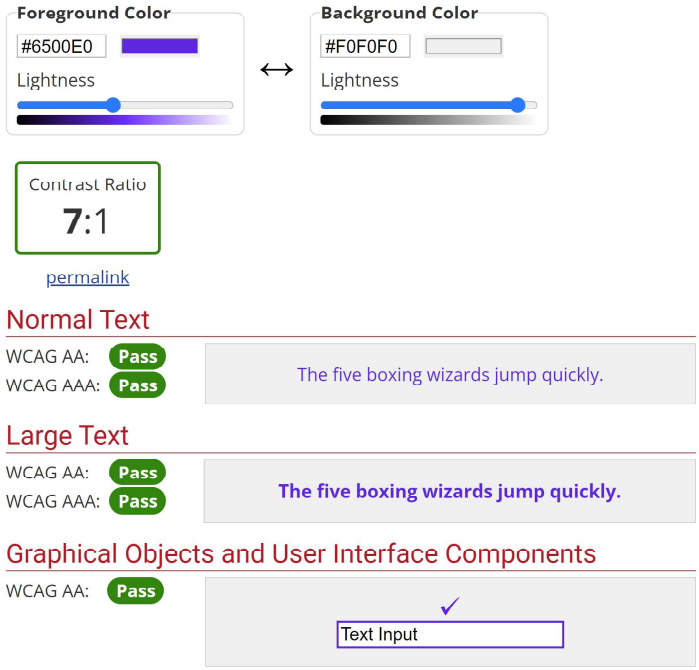


Colour themes

I created both light and dark colour themes for the application, this gives users the option to choose their desired look.
When choosing the colours for the themes I used the WebAIM contrast checker to ensure that the colours worked well and that text would be readable on the screen. Both themes have a minimum contrast ratio of 7:1 which passes WCAG Level AAA.

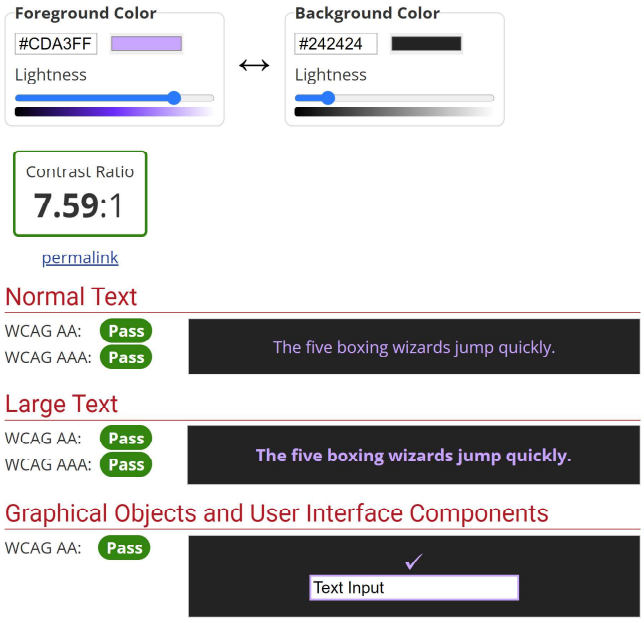
Light theme

Background colour: #F0F0F0 - This is a light grey, less harsh than a complete white.
Foreground colour: # 6500E0 - A dark shade of purple that has good contrast with the light grey background.
Foreground active colour: #46009c - Used when buttons are pressed.



Dark theme

Background colour: #242424 - A dark and modern grey, less harsh than a full black.
Foreground colour: #CDA3FF - A light purple that has good contrast with the dark grey background.
Foreground active colour: #B06EFF - Used when buttons are pressed.



Final JSON file structure


```
{
  "Master Username": {
      "Account Name": {
          "Username": "johnsmith@mydomain.com",
          "password": "password123",
          "extraInfo": "lorem ipsum",
          "passwordStrength": 3,
          "Application": "Account Name",
          "PlayMins": 25,
          "Score": 4,
          "Level": 2,
          "passwordChangeDate": "2000-01-01T00:00:00.000Z"
      }
  }
}
```


Encrypted JSON file example


```
umfYut4adJbTqJlK919LDbvMsLY3jmngJdsLn8V2XpUz3nPF6+du1vSvXqnDx4XRXT44yn+T9oKWUBX3dpvXZoo
E+RsgfcTjIMx80ZsM4kpfWB4mi2jNBH395WDDxwR0u2SwBx4zg5untUBuJ4aSm1kesoybcYndHOFCupZK1KUAoI
7icdIyfpiBrQR0a5fpPjz9UIM5CpZWaU+ksdTAzN5elxTBzhY1M2PbO+5xmGYXOxz4c18Dnpl2Hmh4a7fwJ5Xba
wHhTelsb1EEncfrur1LFHexhZGv78bNdU/JTp4cHOlRfdvYj+OIqzB7mcpPPetPUgHmfqX/r8Phww2zgPJ/0WPs
MAE+vf+GxPZ9z/4Q+fPQRxYDYWtkSh7bTikWKzm8vxbIY8o+ndVxPLHzxGs28TjLxoen/gQ0DnTifELA2slC2XX
9LkU7nTNdjcVTkxFc0cQgUd0lG96WKyDGtE1zwyvUSO5LV/mhU8rDhUOzPzg3tKl110SSCrI0fn6Ygaj6LEJ3hU
tds/0spISz+7YYVeW1wq5zibGwx2iCv6A6Zg0zgc2Y34ZxGqBxXCZfwHa7pgbgYv2yJEcp2iEqxIsxB/X7LRjTx
Q7aJH2A04VGzAyJIo5rgyJYDVls2rqQ7IYtI4esAGznE33W2K/R4sByivxMJk3zq0RP4o7YcW15fP6Sk6mvHa67
ipbcYDOChvXpTd21Sm92M0YHWdWgphwrgrx4XHu2oCpE2qbTre5T2RVoK5acurgNukeGUME0SqV4dWp3mxgQvcV
/waGsIO1EOatEir2ytUntDFmvfi20gFCrwWYX13Eg8P4y3Ql4dIk/VhS66XGy2JJrUq4ihJ/lNxQl3uqW8+DAcm
ufmVpwVzBzdlEqjH7tk8f39/f57H1M4tx7eptPvsyeRM+WZjLtoKJsj39/Fn9VjoCI70jZfqoYfzWHfe7RGZgq3
NgTzzSuT11Aq2lwW9OP4u/v8ZN1Wr2+TNPtqUbyD5wpIvmcQ+NFTs3b4cEqPj+BMCKQ
```

Security

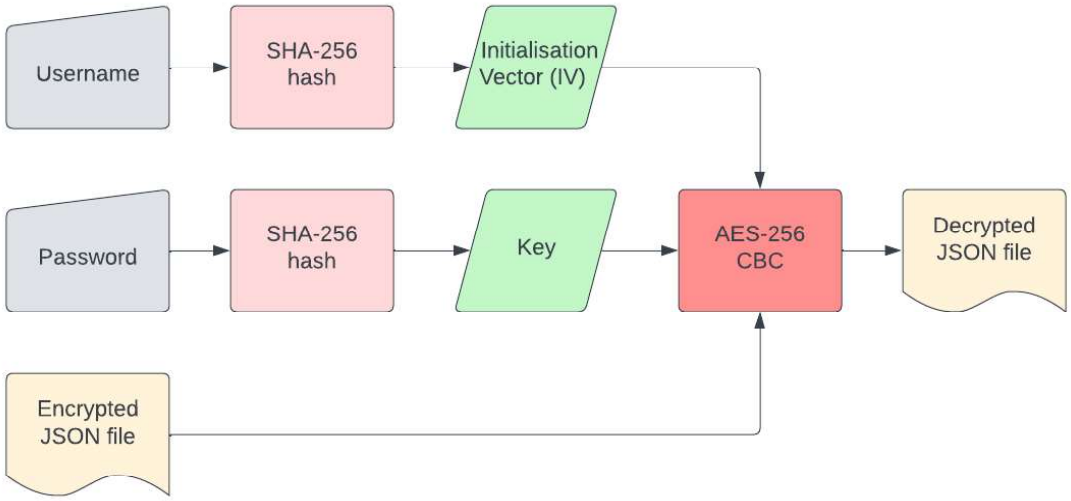
The main concept of my application is for the user to be in complete control of their data, I've achieved this by storing the user's passwords on a local JSON file that they can carry around themselves. The issue with this is that standard JSON data is stored in plaintext and is very easy for humans to read. An attacker could steal the file and simply open it in a text editor to gain access to the passwords. This is one of the pieces of feedback I received in the initial marketplace demo.

To combat this, I have decided to encrypt the JSON file so that it cannot be easily read by a human and only used inside the application. I am using AES-256 encryption as it is the gold standard method and is practically uncrackable by brute force methods using current computing power in a reasonable amount of time. I am using the CryptoJS module to do this which implements AES-256 CBC encryption.

AES-256 CBC requires a Key and an Initialisation Vector (IV) which both need to be 256 bits in length. Using the same module, the application creates a SHA-256 hash of the username for the IV and a SHA-256 hash of the password for the Key. This hashing algorithm outputs 256-bit hashes for use with the encryption.
This means that the username and password can be any length or complexity and the program will always create 256-bit hashes for the encryption method.

The user's master username is stored in the beginning of the JSON file. After decrypting the file, the program searches for the username in the decrypted data; if the username is found then the file must have been decrypted successfully. This verifies the integrity of the file after being decrypted.

Decryption Flowchart



Once the file is decrypted it is stored in LocalStorage in the browser along with the user's username. It is then freely available for use by the application. When the user wants to logout, they simply enter their password and the file is re-encrypted using the same method and is downloaded to their computer.

Reflection

Overall I am happy with the final result of the application, I am most proud of the number of quality features I have been able to implement with no prior JavaScript knowledge. I have successfully met the requirements set out by my project vision. I believe it also meets all of the minimum requirements set by the assignment as well as including some innovative ideas that have pushed the functionality of the project further.
At its core, I have created a single page application that allows for a user to upload a JSON file, read and modify its contents and then download the file when finished with. The JSON file stores usernames, passwords and other fields deemed useful for a password manager and the password strength is indicated to the user.
The extra functionality all builds upon and stays relevant to the core features of the project. The main standout feature for my application is that the user's JSON file is encrypted so that it cannot be read as plain text outside of the application. This heavily increases the security of the user's data.
An additional field I have added to the JSON file is the date a password is last modified. This allows for the user to keep track of how often they are changing their passwords. Seeing the date of an old password may remind them to change their passwords more frequently.
I have also added a password generation page that creates pseudorandom passwords based on a number of user specified parameters. The passwords can then be copied to the clipboard for use when creating a new account on a website.
The user can generate QR codes for their passwords that can be scanned and copied to the clipboard of their smartphone. This allows for the user to easily use their passwords on a mobile device.

The main challenges I faced were due to my lack of experience with JavaScript, I had issues with importing libraries using a CDN and allowing for the different JavaScript files to communicate with each other. I used the Mozilla MDN Web Docs to work through these issues.

If I had to develop this application further, I would take the time to learn a modern JavaScript "framework" like React instead of using Vanilla JS and JQuery. Using React would allow me to make a more responsive user interface using newer techniques like pop-up boxes for example. It would also make creating a mobile-friendly version of the site easier as it is very simple to build scalable user interfaces with it.
Another improvement to the user interface that I would like to make is cleaning up the alert boxes. I think they should integrate better with the rest of the application and not look like separate windows popping up. Along with that I would like to improve the password entry box when logging out from the website.
A feature I would have liked to add is the ability to detect duplicate passwords. This would alert the user if they have used a password multiple times and encourage them to use unique one's for each account (this is best practice for security).

Resources

Programming

https://github.com/Plymouth-University/comp1004---main-assignment-JW2586 - Final repository

https://github.com/davidshimjs/qrcodejs - Library used for QR code generator
https://github.com/brix/crypto-js - CryptoJS - Library used for encryption and hashing
https://cdnjs.com/libraries/crypto-js - CDN used to access CryptoJS
GitHub project boards - For tracking sprints and user stories

Design

https://webaim.org/resources/contrastchecker/ - For finding the contrast ratio of user interface colours
https://fontawesome.com/ - Icons used in user interface
https://www.figma.com/ - User interface design tool
https://undraw.co/ - Illustrations used in poster