Password Manager

Introduction

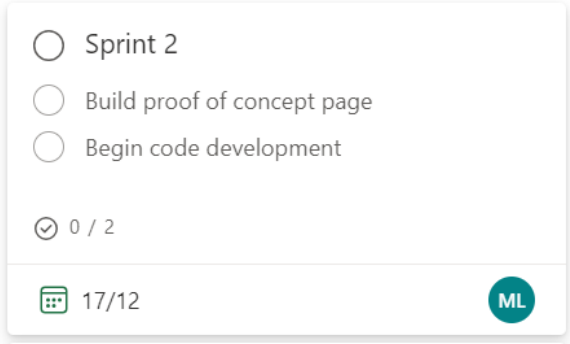
This report presents the COMP1004 coursework. The password manager created for this coursework focuses on data security.

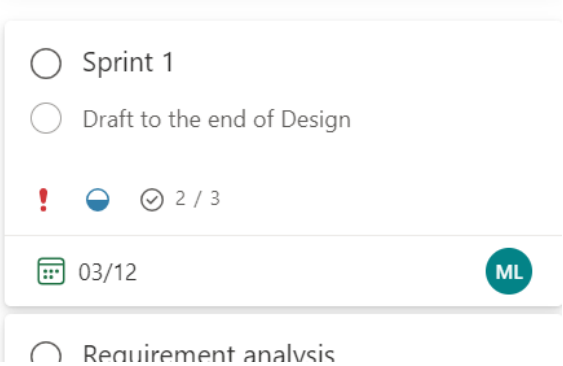
In addition, the Software Development Lifecycle (SDLC) has also been discussed. An agile driven approach has been used for this project.

Software Development Lifecycle

This section discusses the Software Development Lifecycle (SDLC) and describes how this is being used in this particular project.

This project was planned over a week, with other password managers -such as the chrome password manager- being investigated.
Once planning concluded, the rough period of development was established, with sprints being assigned tentative tasks.





In total, 10 sprints were planned, though the exact tasks assigned to these were significantly adjusted over the duration of the coursework, a result of tasks not being completed in the sprint period and new ones being discovered during development.

Sprints

Microsoft Planner was used for sprint planning, the task board containing the sprints being arranged into two columns: To Do, and In Progress. Completed tasks had their own section, provided by the application.
Below are the contents of the individual sprints:

Sprint 1

20/11/23 – 03/12/23
Contents:

- Research password managers (complete)
- Research encryption algorithms (complete)
- Draft to end of design (incomplete)

Though parts of the introduction and SDLC section were complete, they were not finished in their entirety, even at a draft level.

Sprint 2

04/12/23 – 17/12/23
Contents:

- Design page (complete)
- Build proof of concept page (incomplete)
- Begin code development (incomplete)

Minimal progress was made this sprint, though the wireframes were drafted, this can be found in the Wireframes section.

Sprint 3

18/12/23 – 07/01/24 (Abnormal duration to cover the Christmas break)
Contents:

- Make page draft (complete)
- Add Sign In functionality (incomplete)

Though not explicitly planned, initial coding was started this sprint, though this was minimal, and the webpage was in no way functional. Most of the new code was empty functions, planning out the eventual structure of the program.

Sprint 4

08/01/24 – 21/01/24
Contents:

- Add Sign In functionality (incomplete)
- Get initial Loading and Saving functional (incomplete)
- Password Encryption / Decryption added (incomplete)
- Make video (complete)

As a result of a focus on the video hand in, none of the other sprint tasks were completed. This said, progress had been made on the Sign In functionality.

Sprint 5

22/01/24 – 04/02/24
Contents:

- Add Sign In functionality (incomplete)
- Get initial Loading and Saving functional (incomplete)
- Add Show/Hide functionality (incomplete)
- Add Copy functionality (incomplete)
- Password Encryption / Decryption added (incomplete)
- Add initial Sign In functionality (incomplete)

As a result of an overestimation of efficiency and difficulties with JavaScript file management, no tasks were completed during this sprint. This said, further progress was made towards the loading and saving.

Sprint 6

05/02/24 – 18/02/24

Contents:

- Add initial Sign In functionality (incomplete)
- Get initial Loading and Saving functional (incomplete)

As with Sprint 5, limited progress was made because of difficulties with JavaScript file management.

Sprint 7

19/02/24 – 03/03/24

Contents:

- Add initial Sign In functionality (complete)
- Get initial Loading and Saving functional (incomplete)

Sign in was finally completed, and half of the second task -the loading- was also complete. As the export / save function had not been added yet, it could not be considered complete.

Sprint 8

04/03/24 – 17/03/24

Contents:

- Add password copying (complete)
- Add the Show / Hide function (complete)

Both tasks for sprint 8 were completed, though these were smaller. Progress was made on unplanned tasks, but not sufficient to mark as complete.

Sprint 9

18/03/24 – 31/03/24

Contents:

- Add credential display functionality (complete)
- Allow for the creation of credential display elements (complete)
- Allow for the creation of credentials (complete)
- Allow for credential updates to be saved (complete)
- Implement Encryption / Decryption / Hashing (incomplete)
- Test (incomplete)

Significant progress was made in sprint 9, leaving two remaining tasks program-side. The password manager was now fully functional but lacked security.

Sprint 10

01/04/24 – 14/04/24

Contents:

- Hashing added (complete)
- Encryption / Decryption added (incomplete)
- UML diagrams created (complete)

Half-way through the sprint I realised that time for the portfolio writeup was running out, so my priorities switched to completing it. This means that the encryption and decryption functionalities will not be implemented before this portfolio is submitted.
The GitHub page was partially rolled back to ensure that the provided program was functional, even if missing the work in progress (and in fact broken) encryption and decryption features.

Design Document

This section provides the key elements of the design documentation.

Project Vision

This password manager will be for individuals to help store their passwords in a secure way. The application will store passwords in encrypted files, and these will only be decrypted when a successful sign-in occurs, preventing unauthorised access to passwords.

Background

Password managers are important for the secure, everyday use of computers. Provided they are competent in securing their accounts, a person will have several passwords, most of which are not memorable. A user can either store their passwords physically (which is less portable, and may be vulnerable to theft), in an unsafe plaintext file, or in a secure password manager.

Password managers are effectively databases which store usernames and passwords for different sites in a controlled form. A strong manager protected behind a strong access password means the user only needs to remember one password (Li, 2014).

A password manager must be secure, otherwise it can introduce a significant vulnerability by potentially exposing all of a users' passwords. Many managers have security flaws, with 4 out of 5 studied password managers in a study being vulnerable to attacks which reveal credentials (Li, 2014).

Regardless of where password manager data is stored: portable, cloud or local, there are dangers. This is because USB Memory Sticks and Smartphones can be stolen, and cloud storage can be accessed without permission by attackers. Though storing locally is more secure, requiring an attacker to either gain access to the computer, it is still not acceptable. Because of these threats, the password manager must use a secure format. (Gasti, 2012).

User stories and Associated Use Case Scenarios

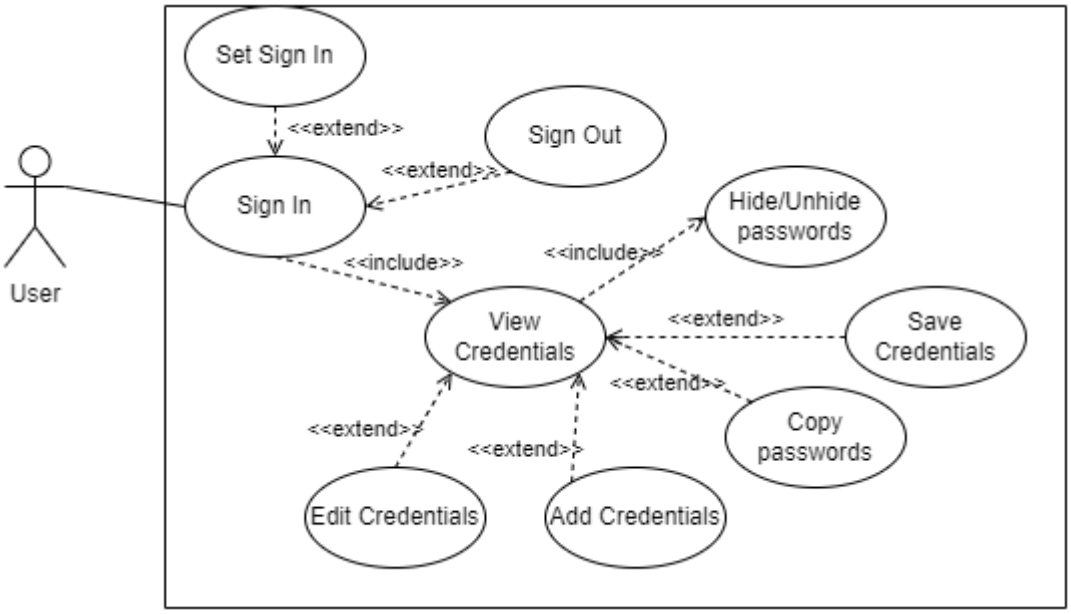
This section examines the user stories identified so far and provides the accompanying use case scenarios.

Below are the User Stories identified for this project. These are shown in a Use Case Diagram and decomposed later in this section.

- As a user, I want to have easy access to my passwords so that I can spend less time signing in and won't lose them.

- As a user, I want my log in information to remain secure, so it is unlikely for my accounts to be breached.
- As a user, I want secure information (such as my passwords) to be behind a sign in, so malicious actors cannot access them even if they gain access to my computer.
- As a user, I want to be able to reset my login password, so if I forget it, I can still access my other passwords.
- As a user, I want the application to be easily accessible, preferably without requiring any prerequisites, so it is easier to install and use.
  Note: As this is a web-based project, the JavaScript is visible to anyone using inspect element. For security reasons it is necessary to find a way of hiding the encryption method. The best way of achieving this is through an API.
- As a user, I want to be able to add new passwords for when I create new accounts.

Below is the Use Case Diagram for these stories:



Decomposed Sign In process:

Usage:	Sign In
Description:	The user attempts to sign in to the password manager.
Precondition:	The user has a stored account and key, credentials are provided
Post-condition:	The user is signed in and credentials are displayed.
Error Situations:	There is no stored account / key. The stored account does not match the provided credentials.
Error State:	Waits for the stored account / key to be provided and for the user to try again. Error displayed stating incorrect credentials have been provided.
Actors:	User
Triggers:	User needs to gain access to the password manager.
Standard Process:	1. User enters their username and password into the respective textboxes. 2. User presses the Key button and provides a key (JSON) file.

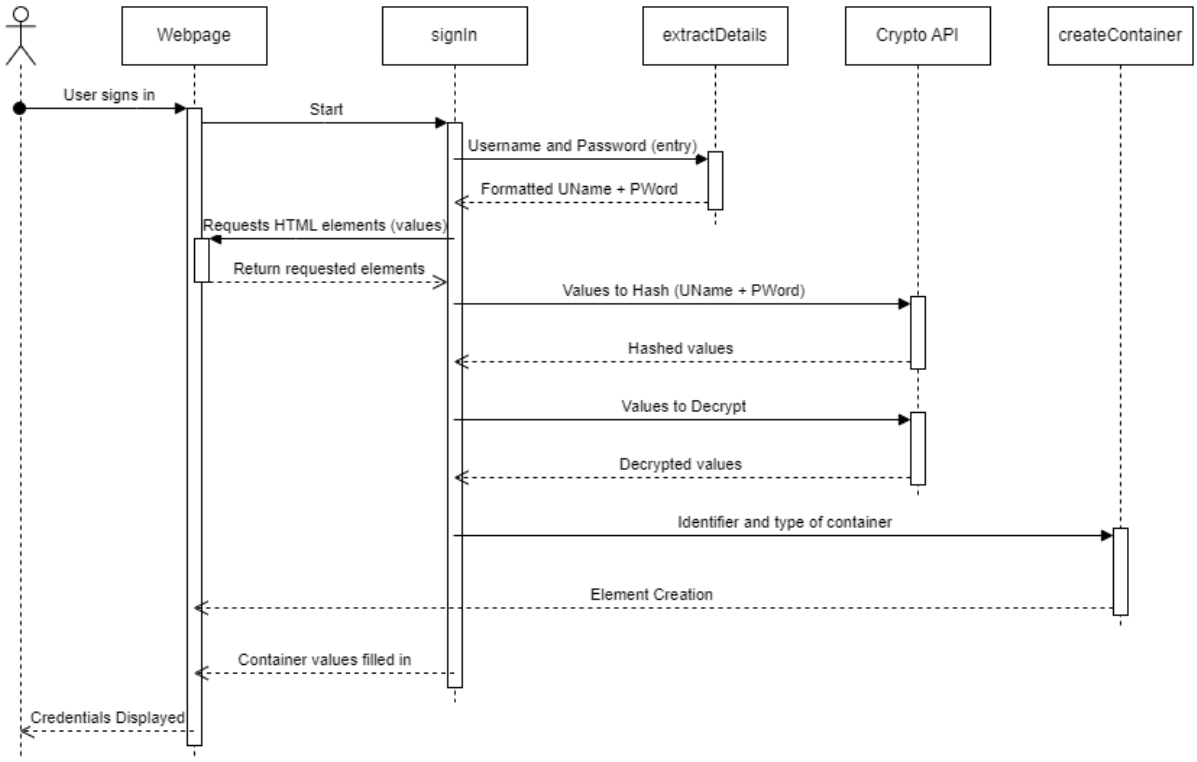
	3. User presses the Sign In button and provides the credential storing (JSON) file. 4. System verifies the username and password are identical to the stored ones. 5. System displays credentials, decrypted using the key file.
Alternative Process:	3'. User does not provide a file. 4'. System waits for the user to press the button again and provide a file. 4''. The username, password or both do not match. 5''. System displays an error message. 4'''. System detects that the credential storing file is incorrectly formatted. 5'''. System displays an error message. 5''''. System fails to decrypt the credential storing file as the key file is wrong or incorrectly formatted. 6''''. System displays an error message.

The other processes would also have been decomposed; however, these do not have alternative processes, and regardless of user input will always run the same.

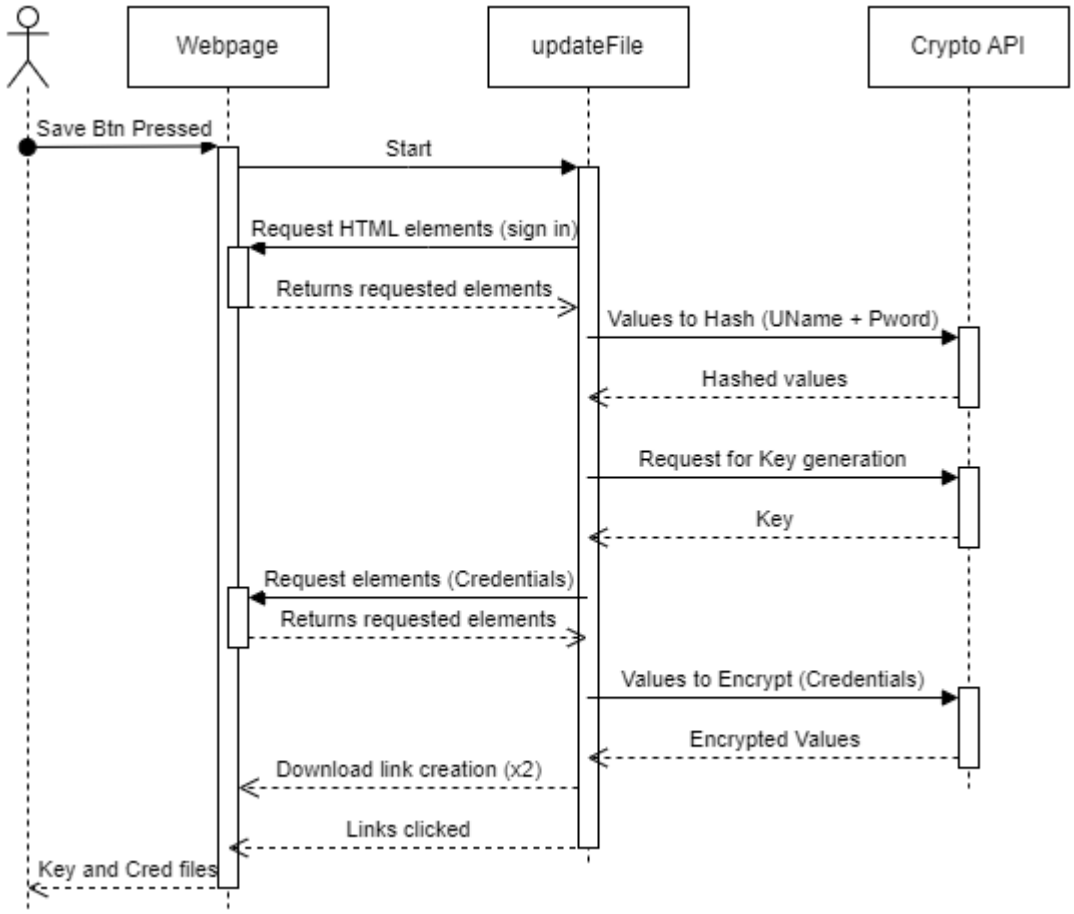
Architecture

This section discusses how the architecture for the single page application is envisaged.

A sequence diagram for the Sign-In process:

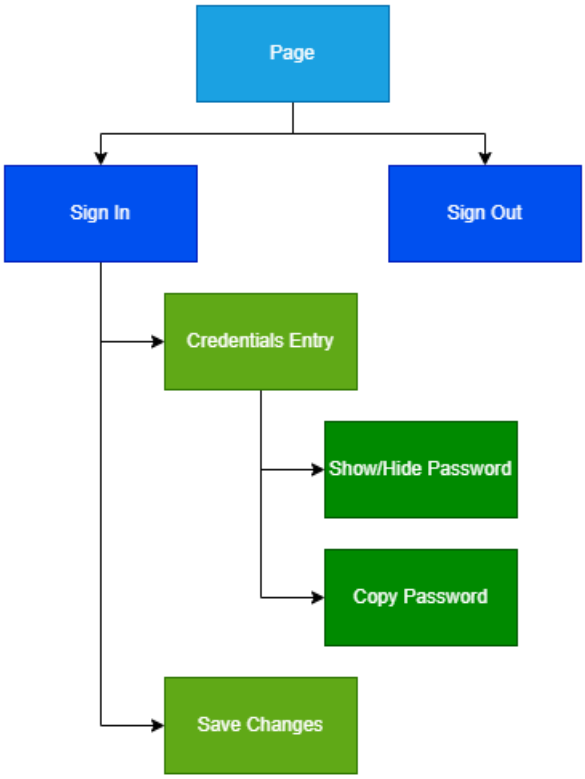


A sequence diagram for the Saving process:



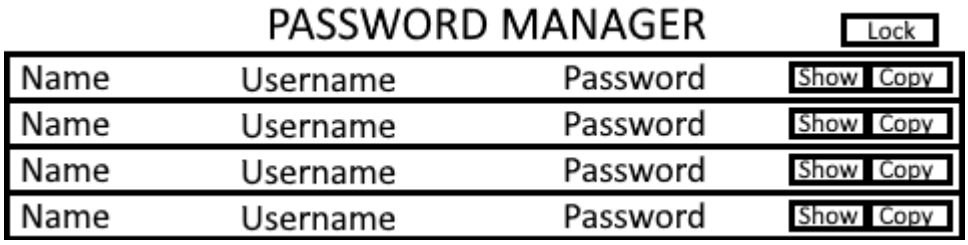
Sitemap

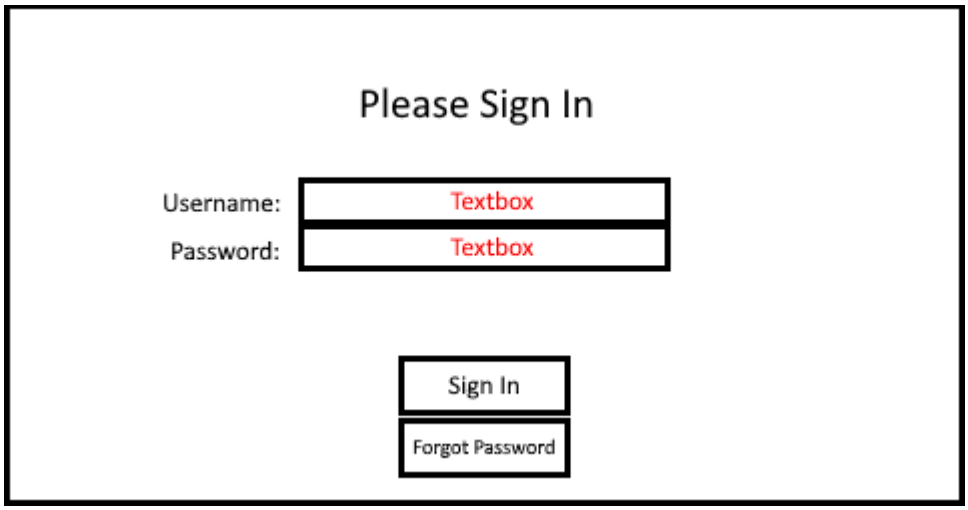
This section provides an outline of how the application is designed. Whilst this is a single page application, the sitemap indicates how the user will navigate through the topics.



Wireframes

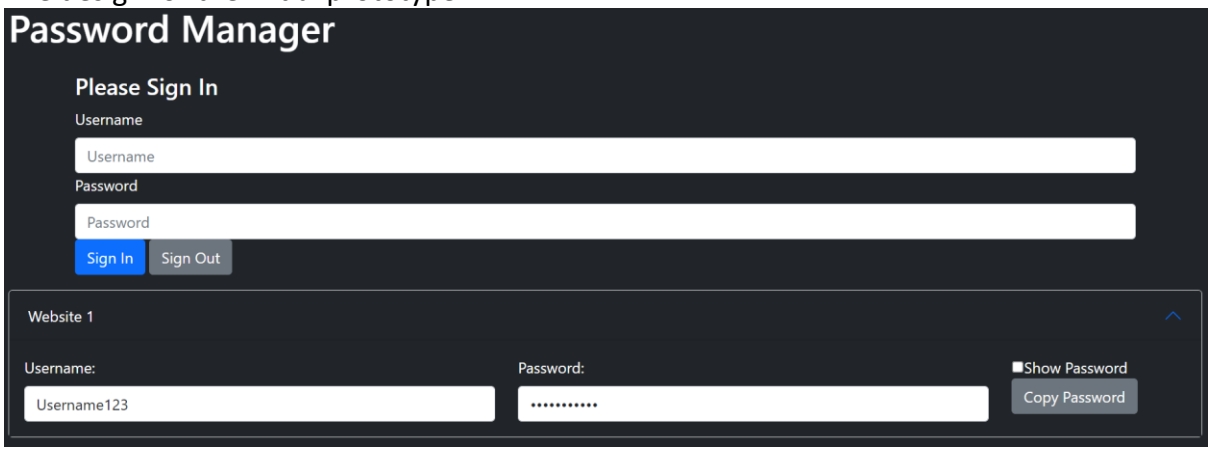
This section provides an illustration of the wireframes, these were adjusted when implemented.





The sign-in box was to be displayed on top of the main password manager page, which would not have any information present, and might have been blurred.

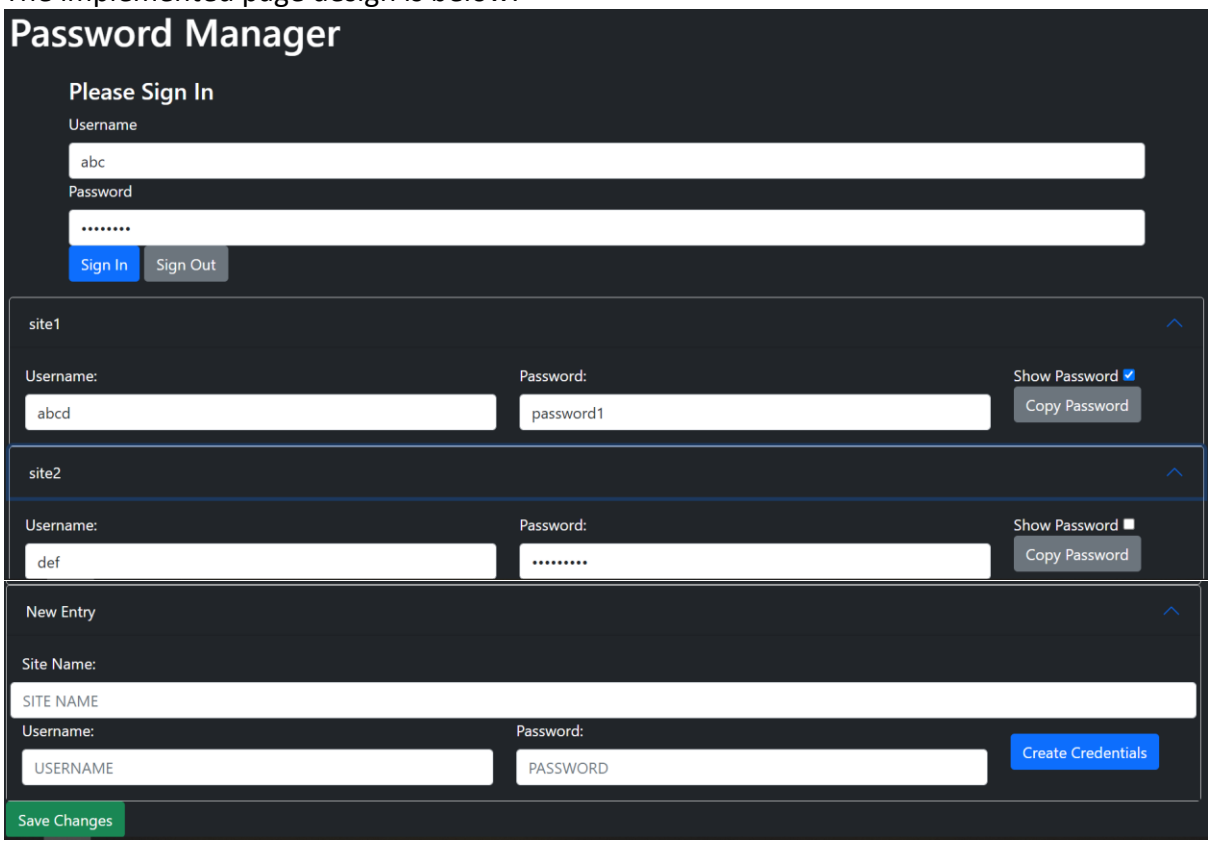
The design for the initial prototype:



This is an initial prototype page.
When functional, no passwords will be shown until the user signs in, at which time they will be decrypted and the containers filled.
The contents of Website 1 can be shown and hidden.
Show Password sets the password box from the “password” type to “text”, making it visible and able to be directly copied.
Copy Password copies the contents of the password box to clipboard without showing it.

As can be noted, there is a notable difference between the initial prototype and initial designs, this is because of the discovery of the options available with Bootstrap, which allowed for the collapsable password containers. In addition, the “Forgot Password” button was removed due to the realisation of the difficulty of implementing it, and the lock (or Sign Out” button was moved to take its’ place.
Though originally intended to display over the manager page, the Sign In section was instead placed at the top of it, this was due to a lack of knowledge regarding HTML.

The implemented page design is below:



The only notable changes with the implemented design are that there is now a “New Entry” container, and “Save Changes” button. The former is used to create new credential entries, while the latter saves all entries to file.

Implementation

File format:

```
{
	"username": "192264086126130732522295155242931211122225921413817810152 36",
	"password": "226169358410138211953219210299120023720270157114194121 14",
	"site1":{"username":"abcd","password":"password1"},
	"site2":{"username":"def","password":"password2"},
	"Google":{"username":"Max","password":"123"},
	"Plymouth University":{"username":"Max.Lowther","password":"123"}
}
```

As can be seen, the username and password values associated with sign-in are hashed. Due to space constraints, the values have been shortened.
The hashing algorithm used is SHA-256, a well known hashing algorithm which was produced by the USAs’ National Security Agency (Griškėnas, 2023). This is considered one of the standards for all forms of hashing, but is still fairly secure, making it suitable for the hashing of sign-in data.

Every set of credentials is identified by the site name, given when creating a new entry (see the page design section). This said, there is nothing preventing names being duplicated, which has both positive and negative implications – a user could accidentally create a

duplicate or wrong entry, becoming confused, but can also store multiple sign-ins for the same site. The values of the username and password would be encrypted, had this feature been implemented.

This file is downloaded when either the “Save Changes” or “Sign Out” buttons are pressed, ensuring it remains up to date. Due to the limitations of the system, however, the file which the values were loaded from cannot be updated, meaning an updated copy is created on every save.

The site names ("site2", for example), alongside username and password values were intended to be encrypted, but this feature was not implemented before the Portfolio was completed.
Had encryption been implemented, there would be a second JSON file, Key, storing the output of the crypto.subtle.exportKey function, this would have allowed for the input to be decrypted.

Poster

Password Manager

Max Lowther
max.lowther@students.plymouth.ac.uk

Introduction

Project Vision:

This password manager will be for individuals to help store their passwords in a secure way. The application will store passwords in encrypted files, and these will only be decrypted when a successful sign-in occurs, preventing unauthorised access to passwords.

Description:

Many password managers are either reliant on the browser running them, such as the Chrome Password Manager, or store their data in the cloud, such as NordPass (1). This means that there is a lack of fully local password managers which are not constrained to one browser or set of browsers.
This password manager fills the gap by storing user credentials locally with minimal internet use – in fact, it is fully functional without ever connecting to the internet at all.

Technology

Features:

- Users can sign in securely, with their credentials being hashed
- The page will reset, and all credentials will be saved, when the user signs out
- Only the necessary credentials can be opened for viewing
- Passwords can be copied to clipboard without ever being displayed
- Passwords can be made visible, if the user cannot copy them to clipboard
- Edits can be saved, in the event of a mistake or accidental shutdown

Technology:

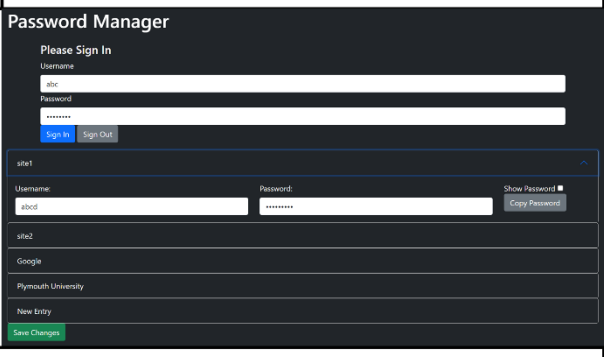
Bootstrap, a HTML toolkit, has been used to create and stylise the password manager, providing the coloured buttons and collapsable credential containers.
To achieve hashing, SubtleCrypto has been used. This cryptographic system is integrated into many web browsers, such as Chrome, Microsoft Edge and Firefox (2), and is far superior to anything which could have been implemented as part of the password manager.

Conclusion

Though an incomplete implementation, missing the ability to encrypt or decrypt the credentials being stored, the password manager could easily be improved to provide this functionality.

Beyond this, future developments could include:

- Improvements to the file system, which currently requires the user to select a file, and creates new files rather than updating them when saved.
- Using templates to increase the speed at which credentials are displayed, while decreasing the size of the JavaScript file.
- The inclusion of a search feature, to allow for users to quickly find the credentials they are searching for when using large sets.
- Making the implemented Bootstrap functionality local, so that the Password Manager is no different when being run for the first time online or offline.



References

(1) https://nordpass.com/features/zero-knowledge-architecture/
(2) https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto#browser_compatibility

Noted issues and constraints

This project is limited in terms of what can be used. Ideally, an encryption algorithm, if not most of the program, would be in C#, C or C++. Unfortunately, it is required that the scripting side of the project is exclusively done in JavaScript. This means that the application is less secure than alternatives, and many open-source encryption / hashing algorithms cannot be used.

As the program required a number of HTML element creations, namely the creation of the Credential containers, and these elements needed to be uniquely identified, the function for creating containers is incredibly large, almost certainly being slower than using HTML templates to create non-uniquely identified elements. The only potential improvement that could be made would be to use templates, and then modify IDs, though this has not been implemented.

Web security measures prevented the manipulation of the saved credentials in files. This resulted in the need to have the user select the credentials when signing in. Though it could be argued that this is beneficial, as it allows for multiple users to use the app on the same device, it requires the user to do more to sign in. As files seemingly cannot be updated, a new file must be downloaded whenever the file is saved, occurring when the “Save File” and “Sign Out” buttons are pressed. If multiple edits are made and saved, this can result in file duplication, and if a user saves before signing out, the program will end up giving them two identical files.

The encryption and decryption of credentials was not implemented into the password manager because of the deadline approaching, alongside issues regarding the way the SubtleCrypto API handled the process. Though implementing hashing was fairly simple, requiring two asynchronous operations, the encryption process required significantly more, and experienced difficulties with the file loading process, this reduced the readability of the code and generated numerous errors, the cause of many of which were not immediately apparent.

This said, the program is functional in that credentials can be loaded, saved, and modified, in addition to the less important functionalities such as showing and hiding passwords, alongside copying them. As I had minimal experience with JavaScript, and the functions needed were not exactly simple to implement, I would consider this program flawed, but not terrible. With more time, the missing features could certainly be properly implemented, and existing features could be improved, such as the file management mentioned previously.

Considering the User Stories, the Password Manager has succeeded in accomplishing most of the goals. Users can easily access their passwords, must sign in to display them, and will run so long as the user has a compatible internet browser – they do not even necessarily have to be online, as the CSS is cached, and the program functions even without the Bootstrap visuals.

However, two of the User Stories have not been met, specifically “As a user, I want my log in information to remain secure, so it is unlikely for my accounts to be breached” and “As a user, I want to be able to reset my login password, so if I forget it, I can still access my other passwords”. The reason as to why the first story was not met has already been covered, with the second story having not been met because of a lack of possible solutions. To allow for a user to reset their login password, either advanced functionality such as the ability to send an email would have to be added, or the security of the sign in would be compromised. It is possible for a user to change their username and password, but only once they have signed in. In addition, due to an oversight, mixed with time constraints, the ability to create

a fresh file was not added, meaning that they cannot create a new file if they forget their sign in.

In conclusion, the solution is not great, but can be improved with time.

GitHub repo link

https://github.com/CentralC0re/COMP1004

References

Li, Z et al (2014), *The Emperor's New Password Manager: Security Analysis of Web-based Password Managers*, Page 3, Section 2.1, Source: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-li-zhiwei.pdf (Accessed 28/11/23)

Gasti, P and Rasmussen, K (2012), *On The Security of Password Manager Database Formats*, Section 1, Source: https://ora.ox.ac.uk/objects/uuid:926086ea-180b-4f11-a599-2522a80837f4/download_file?file_format=application%2Fpdf&safe_filename=pwvault.pdf&type_of_work=Conference+item (Accessed 28/11/23)

Griškėnas, S (2023), *What is the SHA-256 algorithm, and how does it work?*, Source: https://nordvpn.com/blog/sha-256/
(Accessed 16/04/24)