



Your PassWORD is Outdated

An Upgrade to PassSENTENCE is advised!

Dale Stubbs - 14024149 | Computer Forensics and Security | 20/04/2017

Supervised by:
Dr Anthony Kleerekoper

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

A handwritten signature in black ink, consisting of a large, stylized 'O' followed by a series of loops and a final flourish.

Signed _____

Abstract

Passwords managers are becoming an important feature for the security of users online. Passwords are now and always will be a source of attacks from hackers. Once they have your password there is no limit to what they can accomplish with the data they manage to obtain about that user. The single point of failure of these password managers is the single 'Master Password'. Once this has been successfully obtained by a hacker they have access to all of the passwords contained within the manager. The PassSentence manager has been designed with this in mind. There is no login password to obtain access to the passwords stored. This would appear to be a large security flaw in the design, however, as the passwords themselves are stored in an unusual way, they are almost unhackable. The usability of PassSentence has also been addressed to ensure users will continue to use the manager for years to come.

Contents

List of Figures	5
List of Tables	5
1. Introduction	6
2. Literature Review	7
3. Probability	12
4. Design	13
4.1. Visual Overview	13
4.1.1. Welcome Screen	13
4.1.2. Login Process	14
4.1.2.1. Login Screen	14
4.1.2.2. Returning User Selection Screen	15
4.1.3. Sign Up Process	16
4.1.3.1. Sign Up Screen	16
4.1.3.2. New User Selection Screen	17
4.1.4. Create Password Screen	18
4.1.5. Password Display Screen	19
4.1.6. Password Viewing	20
4.1.6.1. Encoded Password View	20
4.1.6.2. Plaintext Password View	21
4.2. Software Operation Overview	22
4.2.1. Password Generation	22
4.2.2. Password Display	22
4.3. Coding	23
5. Implementation	24
5.1. Prototype Implementation	24
5.1.1. Prototype Testing – Test Cases	28
5.2. Product Implementation	29
5.2.1. Product Testing – Test Cases	35

5.2.2. Usability	36
6. Evaluation	37
7. Conclusion	38
7.1. Limitations.....	38
7.1.1. Single Browser Compatibility.....	38
7.1.2. No Ability to Change PassSentence	38
7.1.3. Limited Usability Testing	38
7.1.4. No browser Independent Storage.....	39
7.2. Future Work.....	39
References.....	40
Main Body References.....	40
Appendix 3 – ToR references	41
Appendices.....	42
Appendix 1 – Icons and Title Images	42
Appendix 2 – Usability Questionnaire.....	42
Appendix 3 – Ethics Form	43
Appendix 4 – Terms of Reference	46

List of Figures

Figure 4.i Potential Welcome Screen.....	13
Figure 4.ii Potential Login Screen	14
Figure 4.iii Potential Returning User Selection Screen	15
Figure 4.iv Potential Sign Up Screen.....	16
Figure 4.v Potential New User Selection Screen	17
Figure 4.vi Potential Create Password Screen	18
Figure 4.vii Potential Password Display Screen	19
Figure 4.viii Potential Encoded Password Viewing Screen.....	20
Figure 4.ix Potential Plaintext Password Viewing Screen.....	21
Figure 5.i: Browser button on display.....	29
Figure 5.ii: Welcome Page in landscape orientation	30
Figure 5.iii: Welcome Page in portrait orientation	31
Figure 0.ii: All images used for icons and product title.....	42

List of Tables

Table 5-1: Test case results for the Prototype.....	28
Table 5-2: Test case results for the Product.....	35

1. Introduction

As e-commerce increases the required number of passwords a user needs increases. As the human brain is limited in its memory capabilities the average user turns to password re-use. Password re-use is when a user uses the same password for multiple online accounts. This approach means the security of the passwords being used is only as strong as the weakest security of the accounts the password is being used on. This project aims to overcome the problem of password re-use via the use of a new password manager and the lack of user friendliness of current password managers. Current password managers are not considered user-friendly and all have a large security failing - the 'Master Password'. If a hacker obtains the master password then they obtain the ability to view all of your passwords. User-friendly interface is required to provide a useful and maintainable relationship between product and user. The author of this paper intends to create a user-friendly password manager that provides a more than reasonable form of security when storing the password. This is to be completed by creating a password manager called 'PassSentence'. PassSentence will use a single sentence to generate a user's passwords. When the passwords are then sent to the storage medium rather than the plaintext version of the passwords being stored, the placement of the characters from within the sentence will be stored. This method created a less than useful list of numbers for any attacker that may obtain the user's password list from the storage medium. The only way to decode these lists of numbers is through the PassSentence password manager itself or by having access to the sentence used to generate the passwords. PassSentence will be developed as a browser plugin. The following report will provide insight into the design, implementation and evaluation during the building phase and pre- and post-build testing of PassSentence - a password managing browser plugin.

2. Literature Review

Passwords have been used since ancient times. They were originally used in Ancient Rome and were called 'watchwords'. It is explained that during the night watch in Roman times a wooden tablet with a watchword inscribed upon it is passed to each of the watchmen before they are relieved of their duties (Polybius. and Shuckburgh, 2012). Passwords have come a long way since the dawn of the computer. They were first used in computing systems in 1961 when the first Unix system had a login page in which a user entered their secret password in order to log onto the system (Crisman, P.A., 1965). Since this time passwords and their uses have evolved almost as much as computers themselves and have become an everyday essential for anyone with a computer.

Passwords are the most frequent (Kuo, C., Romanosky, S. and Cranor, L.F., 2006 and Chiasson, Oorschot and Biddle, 2006) and the most relied upon a form of authentication practised to date. They are used to protect any and all information that users have online that they believe to be important and therefore needs protecting. The passwords and usernames required for access to this information are similar to locking a piece of paper into a cabinet using a lock and key. As such an important feature of protecting this confidential information, the security of the passwords needs to be paramount.

With the security being so important thoughts suggest that the requirements when creating new account passwords would encourage the creation of complex passwords in order to maintain the safety of the contained information. This is not the case. Current systems support poor practice construction methods (Gaw, S. and Felten, E.W., 2006). This leads to the passwords being created becoming easy to guess and therefore their security becoming far from what it needs to be (Chiasson, Oorschot and Biddle, 2006). Creating a 'not too easy to guess' password is far from difficult as there are many websites that offer guidance for the creation of secure passwords. The difficulty becomes when a user needs to remember multiple different passwords and where each password is used. Developers have attempted to aid the user in remembering the password by giving the user the ability to enter a 'password reminder' text box when creating a new account. This can be useful at times but if the password has been forgotten then no amount of reminders will allow the user to remember.

The growth of e-commerce has increased the number of accounts that any one user may have (Ives, B., Walsh, K.R. and Schneider, H., 2004). In turn, this has increased the number of passwords required and has forced the user to remember more and more passwords. Some have found this difficult and have turned to password reuse (Ives, B., Walsh, K.R.

and Schneider, H., 2004 and Gaw, S. and Felten, E.W., 2006). This concept has the benefit of distilling the number of passwords needing to be remembered down to only a few, however, the issue raised with this concept is that it is only as secure as the weakest system the passwords is being used on (Ives, B., Walsh, K.R. and Schneider, H., 2004). The failure to address and solve the issue of password reuse will only result in the problem becoming bigger and bigger as e-commerce continues to grow (Gaw, S. and Felten, E.W., 2006).

As technology advances, developers are looking into using biometrics as an alternative authentication system. On the face of it, this seems like the most logical path to take as fingerprints are unique to an individual as is their retinal scan. This could potentially eliminate an attacker's ability to gain access to the users' information and therefore ensuring its safety and confidentiality. This is not the case. 'Gummi' fingers that were made from the same ingredients that are found in gelatine sweets managed to fool fingerprint scanners 80% of the time (Matsumoto, 2002). Retinal scanners are a good form of authentication as a prosthetic eye with a matching retina could be very expensive and time-consuming to create, however, the drawback of this method is that whenever the scanner shines the light into the users' eye they were left feeling 'quite dazzled after multiple uses' which is far from ideal (Hall, 2016). These forms of authentication come with their own set of disadvantages too. A list of which can be found in a paper by Chien Le in 2006 on the advantages and disadvantages of technologies (Le, C. 2006).

As passwords seem irreplaceable several attempts have been made to solve the issue of password reuse. Mark Keith, Benjamin Shao, Paul John Steinbart have written a paper on the use of passphrases for authentication (Keith, Shao and Steinbart, 2007). This mitigates the vulnerability to Dictionary Password attacks but does nothing for the user in terms of remembering a lot of phrases and where each of them has been used. The next attempt solve was the use of Mnemonic Phrase-based passwords. One particular example from a study conducted in 2006 was the phrase: 'Four score and seven years ago, our Fathers' from the Abraham Lincoln Gettysburg address being converted into the password '4s&7yaoF'. On the face of it, this would seem to be a perfectly legitimate, complex password to have, however, when an attacker builds a 'dictionary' to use for their password attack they would place substitutions into for certain words. For example, the words four, for and fore would all be replaced with the number 4 thus increasing the chances of the attacker eventually obtaining the correct password. Upon the conclusion of this experiment, it was shown that method was no more secure than using a regular phrase. When using a 400,000-word dictionary 4% of the mnemonic passwords were obtained in comparison to the 1.2 million word 'standard' dictionary used for cracking passwords obtaining 11% of the control passwords(Kuo, C., Romanosky, S. and Cranor, L.F., 2006).

The security of the web-based password managers seems increasingly easier to defeat. In four of the five tested managers, an attacker could steal arbitrary credentials of the user (Li, Z., He, W., Akhawe, D. and Song, D., 2014). Some web-based managers store the passwords locally on the users' hard drives and in order to find these passwords developers have created a tool called Lupin. Lupin can explore passwords stored for 1,000 web pages in 35 seconds and 28% of the 45,000 most popular websites around to date are vulnerable to Lupin (Gonzalez, R., Chen, E.Y. and Jackson, C., 2013).

Another direction taken to solve the issue was the use of a Password Manager. This concept has various platforms available from stand-alone programs to web-based password managers. A password manager is a program that stores the users' passwords and keeps them under the protection of the master password. This distills the users need to remember passwords down to a single one, the master password (Stobert and Biddle, 2014). Some passwords managers actually generate and store passwords for the user which eliminates the users' problems even further as they would not even have to think of a password themselves. This turned out to be not as popular as predicted. People were not comfortable with the idea of handing over control of their passwords to a password manager (Chiasson, Oorschot and Biddle, 2006). Another flaw in this system was the single point of failure. A single password used to protect all other passwords meaning if the master password was acquired by an attacker, they would have access to all of the users' passwords. Most importantly, the usability of the password managers needs to be as user-friendly as possible (Chiasson, Oorschot and Biddle, 2006).

A study by Stobert and Biddle in 2014 propose that we overcome the issues associated with passwords by developing a new type of password manager that does not store the passwords directly but uses different approaches to trigger the users' memory of what the password for the account is. Their particular password manager had several issues and limitations including a lack of understanding of how it actually worked, security issues and incorrect interface elements (Stobert, E. and Biddle, R. 2014) concluding that there is a need to have a good balance between usability and security.

In 1999 a study was conducted by Alma Whitten and J. D. Tygar on the usability of a software called PGP 5.0 (Whitten, A. and Tygar, J.D., 1999). PGP is an encryption programme that is used to encrypt and decrypt data communications such as text, emails, files and folders. The programme is believed to have a 'good' user interface by 'general standards'. They were able to define usability as:

Security software is usable if the people who are expected to use it:

1. Are reliably made aware of the security tasks they need to perform;
2. Are able to figure out how to successfully perform those tasks;
3. Don't make dangerous errors; and
4. Are sufficiently comfortable with the interface to continue using it.

The participants of this study were twelve people who were experienced users of email and had no previous knowledge of public and private key cryptography. Each participant was given the scenario that they were given the job of a political campaign coordinator and were required to send out updates of the campaign plan to the other members of the campaign team via email and using PGP 5.0 for privacy and authentication. The participants also had a 90-minute time limit in order to achieve this goal.

The experiment concluded with only one-third of the participants successfully signing and encrypting the email message within the 90 minutes time limit. Many of the participants showed unhappiness and frustration with the experience of using PGP 5.0 and confirmed that it was unlikely they would continue to use it in the real world. The conclusion of this experiment shows that the usability of any software is paramount to its success. If people find it easy to use then they will use it.

The book *Security and Usability*, written by Cranor and Garfinkel, states that “Increasingly, well-publicized security breaches are attributed to human errors that might have been prevented through more usable software.” (Cranor and Garfinkel, 2005). The book also suggests that in order for a password manager to be effective there needs to be a balance between the usability of the manager and the security that it provides. This is backed up by the findings of the study by Chiasson, Oorschot and Biddle.

This study was an examination to deduce if any progress had been made since the Whitten and Tygar experiment from 6 years earlier. Within their study, they examined the usability of two different password managers, PwdHash and Password Multiplier. Their study examined twenty-seven adults who were deemed to be ‘typical users of these systems’. All participants were given a list of instructions of how to use both of the managers. Each participant took part in a one-hour session in which they were required to complete five

tasks that had been designed to simulate ‘real life’ uses of the managers. These tasks were to log in, migrate password, remote log in, update password and finally a second log in.

Upon completion of this study, only one of the tasks set had a greater than 50% success rate (the second log in for PwDHash). Even after the instructions were given to them most of the participants still failed to complete more than half of the tasks. The conclusion drawn from this experiment was to add two additional criteria to the four points of defining a software as usable in the previous experiment. These are:

1. be able to tell when their task has been completed;
2. Have sufficient feedback to accurately determine the current state of the system.

The underlying message of this experiments conclusion is that that “Even the most technically secure system, if unusable, will fail in practice.” (Chiasson, Oorschot and Biddle, 2006).

In 2003 Deborah Abratt, Wayne Mallinson and Antonet Bekker wrote a paper on ‘Usability Testing: Recipe for Success’ for EuroStar (Abratt, D., Mallinson, W. and Bekker, A., 2003). In the paper, they compare multiple case studies and evaluated different aspects of each case including usability. The paper concludes that an increase in software usability testing, either with testers only or using testers and users, can offer significant benefits including user productivity.

After completing the analysis all of the papers within this literature review the main issue that has been highlighted is the usability of the software. If a piece of software has a complicated system of use then it is less likely that a user will use the software on a daily basis. Any future attempts to solve the issue of password reuse will need to address the issue of usability as the most important feature. It would also need to address any security shortcomings of its preceding competitors. There needs to be a good balance between usability and security tied into the new software.

3. Probability

When looking into probability we look for the likelihood of something occurring. It is measured on a scale of 0% to 100%, 0 to 1 or 0/1 to 1/1. In order to calculate the probability of an event you first need to find the total number of possible outcomes which is known as the set. When rolling a fair die the set would be the numbers appearing on the die [1, 2, 3, 4, 5 and 6]. As there are 6 entries in the set all calculations would need to be divided by 6. For example when calculating probability of rolling a six is $1/6$ or 0.16. The probability of rolling two sixes in a row is $1/6 * 1/6$ which is equal to $1/36$ or 0.027. Note we multiply the bottom part of the fraction, this is known as the 'rule of the product'.

In order to determine the minimum length of sentence required for the PassSentence program, the above principals need to be applied. When applying the same principals to selecting 12 possible letters for the PassSentence program, the number of possible combinations of different letters that can be selected needs to be determined. This is done to avoid the same characters being used in a different order for a different password. In order to calculate the different number of combinations of the 12 letters the following calculation was computed:

The 'n!' and 'r!' in the equation is notation for the factors of the numbers multiplied together e.g. $2! = 2*1 = 2$. The set (n) of 20 letters was chosen to begin with, as the program will be selecting 12 letters (r) from this set and 20 was a nice round number. The calculation went as follows:

$${}_{20}C_{12} = 20!/12!(20-12)! = 3,375,776,383$$

This meant that the probability of selecting any combination of the 20 characters from the password is $1/3,375,776,383$ which is equal to (approx.) 0.00000000002. The probability of selecting 2 exact passwords when using a set of 20 letters is equal to $1/3,375,776,383 * 1/3,375,776,383 = 1/11395866188020562689 =$ (approx.) 0.00000000000000000008

Therefore a minimum length of 20 at this stage was deemed sensible as the probability of the program selecting the same 12 letters on more than one occasion is so low. However, this may be increased when more probability aimed at the quantity of crossover of the selected numbers (how many times each letter appears in the total number of passwords) is calculated.

4. Design

The following section will look at the design process of how the author intends the software to both look and operate.

4.1. Visual Overview

This segment covers the visual overview of each of the screens that could be contained within the software.

4.1.1. Welcome Screen

Upon loading the browser plugin the user will be introduced to the product with the following screen:



Figure 4.i Potential Welcome Screen

The user is offered the option to either Login or to Sign Up. The user is to click on their desired action and will then be forwarded to the appropriate login or sign up screen.

4.1.2. [Login Process](#)

The next segment will focus on the screens displayed to a returning user when they log in.

4.1.2.1. [Login Screen](#)

If the user clicks on the Login button on the Welcome screen the following screen will be displayed:

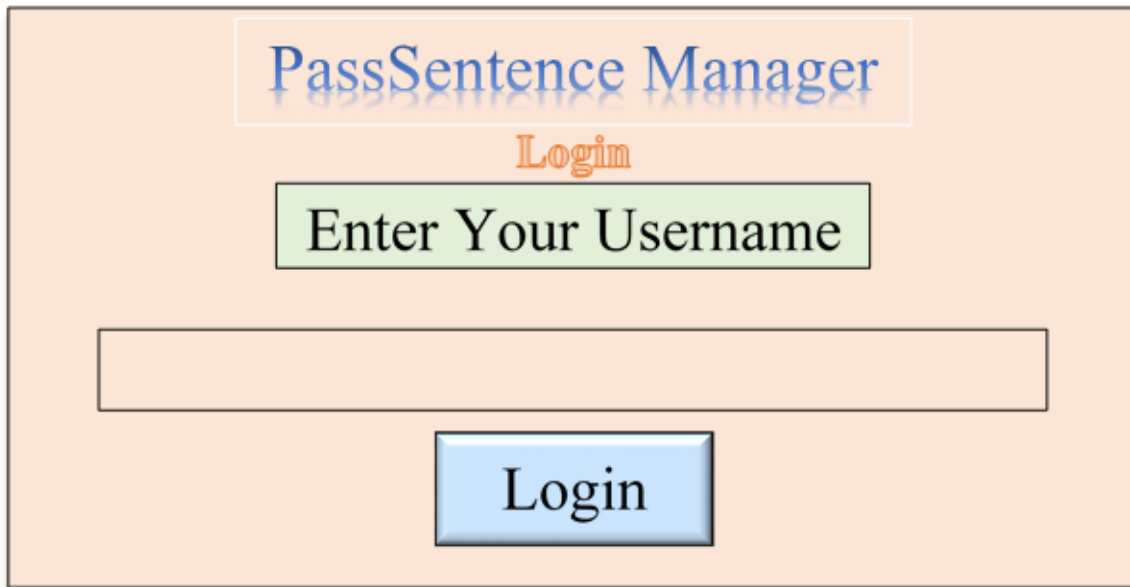
The image shows a login screen for 'PassSentence Manager'. At the top, the title 'PassSentence Manager' is displayed in a blue, serif font with a reflection effect. Below the title, the word 'Login' is written in an orange, serif font. Underneath 'Login', there is a green rectangular box containing the text 'Enter Your Username' in a black, serif font. Below this box is a long, empty rectangular input field for the username. At the bottom of the screen, there is a blue rectangular button with the word 'Login' in a black, serif font.

Figure 4.ii Potential Login Screen

This screen offers a returning user the opportunity to enter their username and then to login to the software to obtain access to their previously stored passwords.

4.1.2.2. [Returning User Selection Screen](#)

Upon entering the username and clicking on the Login button from the Login screen the user will be presented with the following screen:

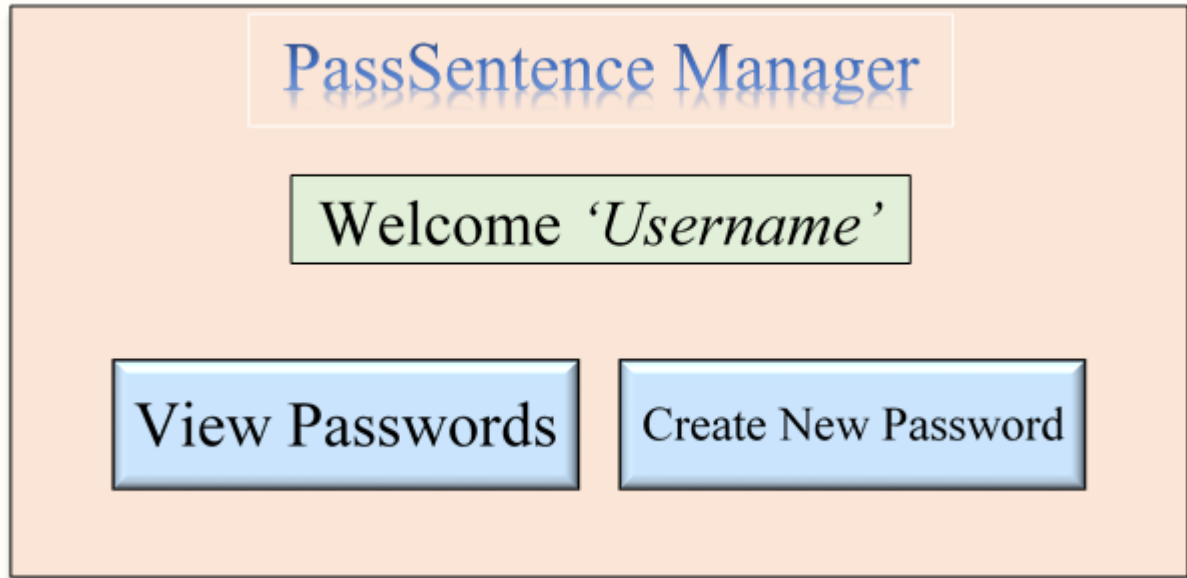


Figure 4.iii Potential Returning User Selection Screen

This screen offers the returning user to either create a new password or to view their existing passwords. This screen will also display the username at the top to show that they have successfully logged in and the correct user has been selected.

4.1.3. [Sign Up Process](#)

The next segment will focus on the screens displayed to a new user when they sign up.

4.1.3.1. [Sign Up Screen](#)

If the user clicks on the Sign Up button on the Welcome screen then the following screen will be displayed:

A screenshot of a web application interface for a sign-up process. The background is a light orange color. At the top center, the text "PassSentence Manager" is displayed in a blue, serif font with a reflection effect. Below this, the words "Sign Up" are written in an orange, cursive font. Underneath, a green rectangular box contains the text "Select Your Username" in a black, serif font. Below the green box is a long, empty white rectangular input field. At the bottom center, there is a blue rectangular button with a black border and the text "Sign Up" in a black, serif font.

Figure 4.iv Potential Sign Up Screen

This page offers a new user the opportunity to choose their desired username which would be required for future access to their stored passwords.

4.1.3.2. [New User Selection Screen](#)

Upon entering the username and clicking on the Sign Up button from the Sign Up screen the user will be presented with the following screen:

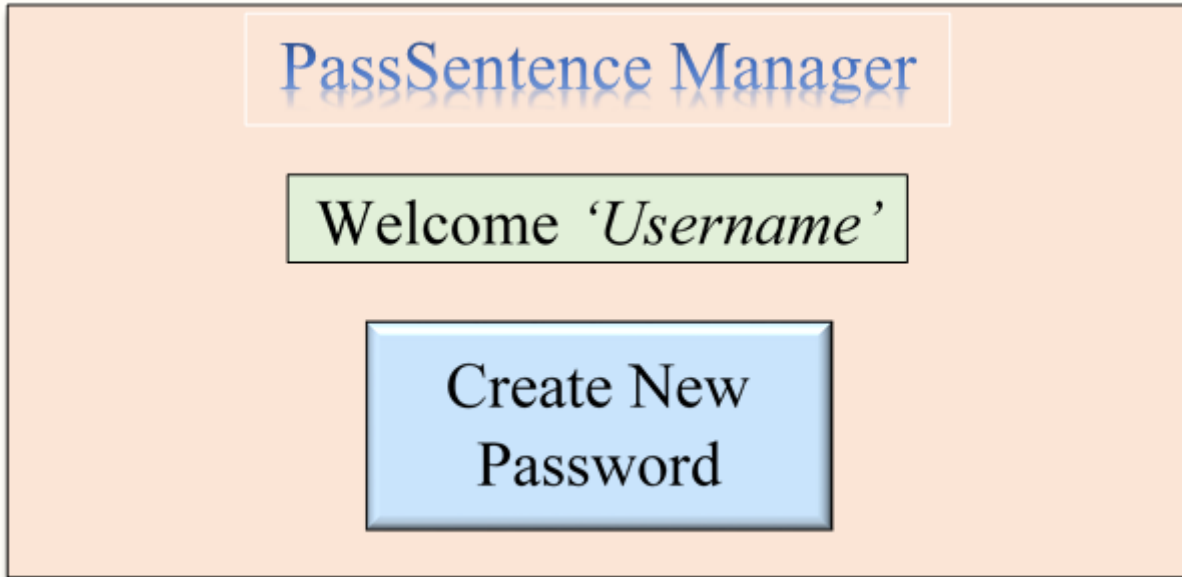


Figure 4.v Potential New User Selection Screen

Unlike the Login selection screen, the new user is only offered the opportunity to create a new password as they would not have any passwords stored in the database.

4.1.4. [Create Password Screen](#)

Upon clicking the 'Create New Password' button on either of the selection screens the following screen will be displayed:

PassSentence Manager

PassSentence: ManchesterMetropolitanUniversityIsGreat

New Password Generation

Website Name: LinkedIn

Are Numbers Required? ☐ Yes ☐ No

Are Special Characters Required? ☐ Yes ☐ No

Are Upper Case Characters Required? ☐ Yes ☐ No

Password Minimum Length? [] [v]

Password Maximum Length? [] [v]

Drop down list of numbers.

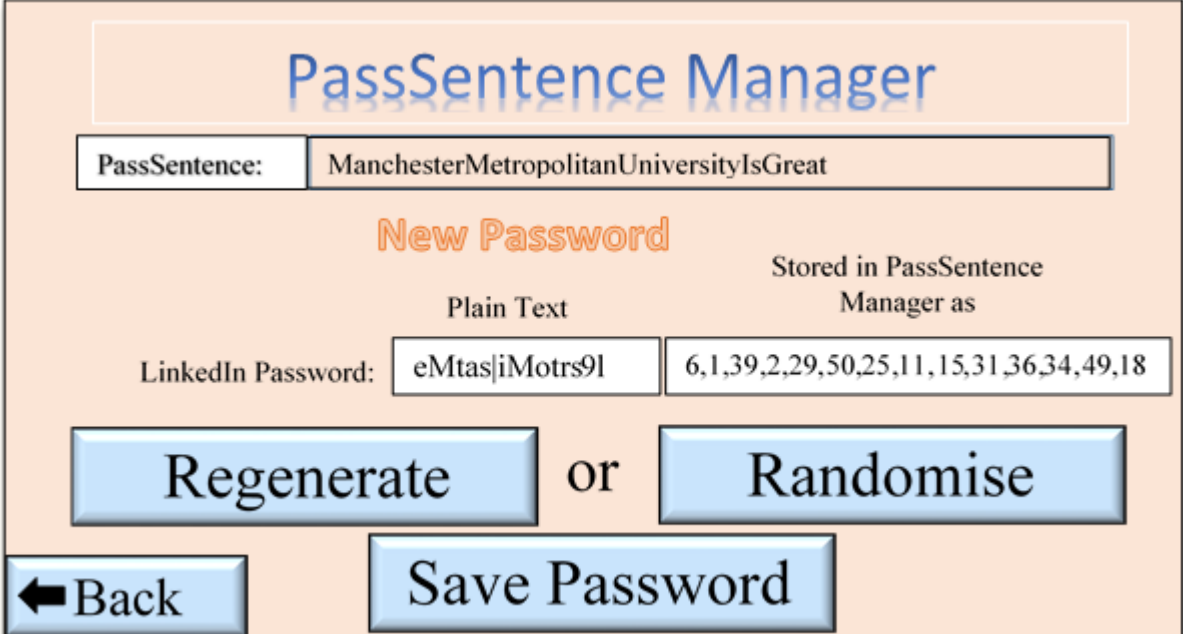
Generate Password

Figure 4.vi Potential Create Password Screen

This screen will offer the user their first opportunity to select a sentence that will be used to generate their passwords. On this page, the user will be required to input several options to determine what characteristics the new password would contain. The user enters the website name of the account that the new password would be associated with, whether or not numbers, special characters and capital letters are allowed or required. They also have the option of setting the minimum and maximum lengths of the password. The new password will be generated based upon the inputs on this page.

4.1.5. [Password Display Screen](#)

Once the 'Generate Password' button on the create screen has been clicked the following screen will be displayed to the user:



The image shows a web interface titled "PassSentence Manager". At the top, there is a label "PassSentence:" followed by a text box containing "ManchesterMetropolitanUniversityIsGreat". Below this, the text "New Password" is displayed in orange. Underneath, there are two columns: "Plain Text" and "Stored in PassSentence Manager as". The "Plain Text" column shows "LinkedIn Password: eMtas|iMotrs9l". The "Stored in PassSentence Manager as" column shows "6,1,39,2,29,50,25,11,15,31,36,34,49,18". At the bottom, there are four buttons: "Regenerate", "or", "Randomise", and "Save Password". A "Back" button with a left arrow is also present.

Figure 4.vii Potential Password Display Screen

This screen has several functions that are integral to the user-friendliness of the software. Here users are presented with the password in both plain text and encoded format. The user is offered the opportunity to generate a different password if they are not satisfied with the currently displayed password. If the user is happy with the characters that have been selected by the software but are unhappy with the order in which they are being used they have the ability to randomise the password. Once the user has generated an acceptable password and is happy with the order of the characters they can save the password to the database.

4.1.6. [Password Viewing](#)

The following segment will focus on the user's ability to view the passwords contained within their database.

4.1.6.1. [Encoded Password View](#)

Upon saving the password to the database of if the returning user clicks the 'View Passwords' button from the returning user selection screen they following screen will be displayed:

The screenshot shows a web application titled "PassSentence Manager". It features a "PassSentence:" input field. Below it, a green box displays "Passwords for 'Username'". A text prompt instructs the user to enter their PassSentence and click "Show Passwords" to view the plaintext version. A table lists passwords for "Facebook" and "Twitter" in an encoded format. At the bottom, there are "Back" and "Show Passwords" buttons.

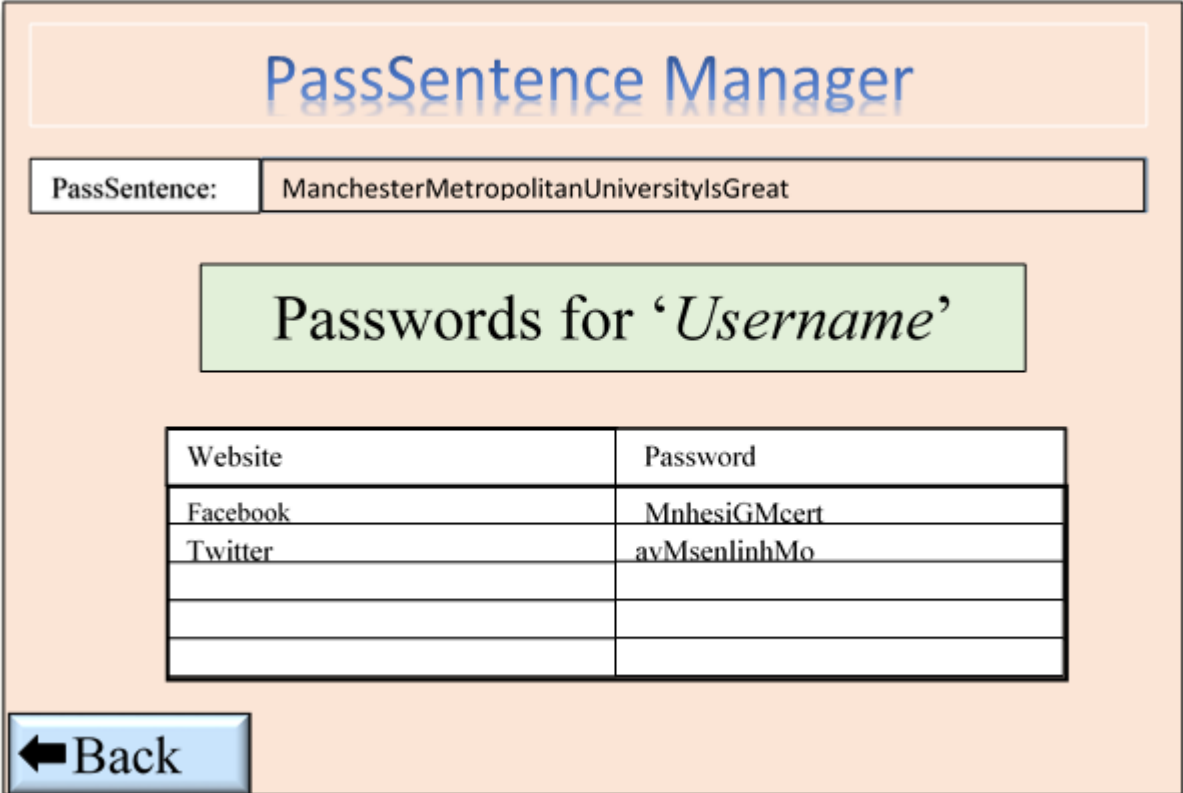
Website	Password
Facebook	1.3.5.9.7.25.35.11.4.9.28.13
Twitter	2.26.11.34.9.3.18.31.22. 5.1.15

Figure 4.viii Potential Encoded Password Viewing Screen

This screen is designed to allow the user to view all of the passwords stored in the database. When the user is first brought to this page they are shown the passwords contained within the database in their encoded format. This ensures that only the user who created the passwords will be able to generate the correct plaintext version of the encoded text as they alone know the sentence that was used to create that particular password.

4.1.6.2. [Plaintext Password View](#)

If the user chooses to enter their sentence into the designated area on the password viewing screen and clicks the ‘Show Passwords’ button the following screen will be displayed:



The screenshot displays the 'PassSentence Manager' interface. At the top, the title 'PassSentence Manager' is shown in a blue, stylized font. Below the title, there is a label 'PassSentence:' followed by a text input field containing the value 'ManchesterMetropolitanUniversityIsGreat'. In the center, a green box contains the text 'Passwords for 'Username''. Below this, a table lists passwords for various websites. The table has two columns: 'Website' and 'Password'. The first two rows are filled with 'Facebook' and 'MnhesiGMcert', and 'Twitter' and 'avMsenlinhMo' respectively. There are three empty rows below. At the bottom left, there is a blue button with a left-pointing arrow and the text 'Back'.

Website	Password
Facebook	MnhesiGMcert
Twitter	avMsenlinhMo

Figure 4.ix Potential Plaintext Password Viewing Screen

This screen offers more to the user by way of usability. The plaintext versions of the passwords are displayed to the user who can then copy and paste the password and login to the account associated with that password.

4.2. Software Operation Overview

This segment covers the potential operation of the software and the processes it may use in conjunction with the screens displayed to the user. This will focus on the creation and display of the passwords to the user.

4.2.1. Password Generation

In order to generate a password for the user the following steps will be required based on the Password Creation screen as shown in *Figure 4.vi* and *Figure 4.vii*:

- Obtain website name from page;
- Obtain PassSentence from the page;
- Convert PassSentence to individual character list (Section A);
- Append the numbers 0-9 and the special characters to the character list (Section B);
- Select 12 digits from section A (original PassSentence) ensuring at least 1 upper case character has been selected;
- Select a number and a special character from section B;
- Randomize the selected characters to ensure the selection from both sections are completely mixed up.
- Store character positions used in a new list (C);
- Store characters used in a second new list (D);
- Display website name and contents of C and D in boxes on the screen;
- If the user selects 'Regenerate' on the page the program will loop back and select a different set of 14 characters.
- If the user selects 'Randomize' on the page the program will keep the selected 14 characters and mix them up. This is to allow the user to move the special character and number selected if they feel that they are too close together within the password.
- If the user selects 'Save Password' the website name and contents of C are committed into the database for the user;

4.2.2. Password Display

In order to display the passwords for the user the following steps will be required based on the Password Creation screen as shown in *Figure 4.viii*:

- Obtain the username from the page;
- Retrieve all passwords for user from database;
- Display stored website name and passwords in a table on screen;

For displaying the plaintext versions of the passwords as shown in *Figure 4.ix* the following steps are required:

- Retrieve the sentence input by the user;
- Decode the encoded passwords;
- Display the decoded plaintext format passwords;
- Allow user to copy passwords.

4.3.Coding

In order to create the software, the author will create a prototype version of the program in the form of a stand-alone program written in Python. This will run in the command line of a PC and will be used for testing and debugging purposes. Once all visible bugs are overcome the algorithm design will be converted to be run as a web browser plug-in.

The plugin will be written using JavaScript. An attempt will then be made to create the plugin using the Kango framework as this offers the best form of cross-browser compatibility. The Kango framework provides a single code base that has been proven to work on Google, Firefox and Safari. As Google Chrome, Firefox and Safari currently have 53%, 10% and 3% respectively of the market shares (Netmarketshare.com, 2016). The browser plugin being created will be useable by the majority share of the market. The algorithm will then need to be added to the framework using JavaScript and use JavaScript to connect to the database for the storage and retrieval of the passwords for any particular user. Kango also provides the ability to add button into the browser bar increasing usability for my users as well as providing the ability to create and modify pop-up windows in HTML format. This will be used to display the help/instructions page for the user to ensure they can operate the plugin correctly.

5. Implementation

As mentioned in the previous section, the implementation was done in two separate stages: prototype and product.

5.1. Prototype Implementation

Again, as previously stated the prototype was built as a stand-alone program. The language used was python as the author was most comfortable using this language and believed it offered a full range of functions and algorithms that could potentially aid in the development of the program.

The first stage of creating the prototype was to break the overall goal into smaller projects that were less complicated to develop. The first step in completing this was to create a list of required functions in order to have the program do as expected. The first list was as follows:

- Take in user input of a sentence,
- Generate a password using the sentence,
- Output the password to the user to ensure they are satisfied with it,
- Output the password to a database,

The next stage was to decide on how each of these steps would be completed. The first and third step were very basic as they only required receiving text and displaying text so these steps were demoted to the bottom of the list and the initial focal point was the generation of the password. Again, this step was broken down again as there were several steps required to complete the overall step. These were:

- Obtain the length of the sentence
- Generate a list of random numbers from zero to the length of the sentence
- Use the random numbers generated to select characters from within the sentence

The first step was, as with the previous step, very basic so the focal point here became the generation of the list of random numbers. When researching how to obtain a list of random numbers from a given set of numbers began, in this scenario it would be from zero to the length of the sentence, python has a function designed to accomplish this called *random.sample*. This function resolved the second step. The final step was to use the numbers obtained using *random.sample* to select the characters from the sentence and this process was placed inside a function called '*charPlacesList*'. As stated in the introduction, this list of random numbers is what is actually going to be stored within the database in order to achieve a higher level of security within the program. When the user enters their sentence into the program it is stored in the computer's memory as an array of characters as opposed to an actual sentence being stored. This makes selecting the characters from within the sentence much easier as the program could simply collect the characters from

the array positions rather than counting through each of the characters within the sentence. All of the selected characters are then joined together to create a password. This process was placed inside a function called '*generatePassword*'. This password is only used for displaying purposes to the user and is never stored. This process solved the two difficult steps in this series and obtaining the length of the sentence was a single line of code so this section of the algorithm was completed.

The next stage was to develop the algorithm required to output the password to the database. However, as this program was a simple prototype for the algorithm development it was decided that, rather than outputting to a database, the passwords would be saved to a file on the desktop. This would provide a proof of concept and would provide the desired functionality of the program. As python provides a list of functions designed to do this process the only requirement on the author was to implement them in the code. This was done inside of a single function called '*storePassword*'. The difficult steps of the original list have been solved and the foundation of the prototype was complete.

The next list involved attempting to implement the additional requirements of the program. The additional requirements were as follows:

- Input a Username
- Input the name of the website the password will be used
- Adding numbers to the password
- Adding special (non-alphanumeric) characters to the password
- Re-generating the password if the user is unhappy with the one generated
- Randomising the password if the user is happy with the characters of the password but not the order
- Creating additional passwords
- Viewing the user's stored passwords

In order to complete the first requirement, a function was created called '*createUser*' that's only purpose was to use the Username input to create the file on the desktop where the files would be stored. The file was named as the 'username.txt'.

The next requirement was added to enable the passwords being stored to have an associated website stored with them.

The next two requirements were added to provide stronger passwords to be generated by the program. A function called '*editSentence*' was created to accomplish this. The user is asked if the password is allowed or requires numbers and/or special characters. Their sentence is then edited by adding either the numbers zero to nine, five pre-determined special characters (|, {, _, @ and &) or both to the end.

The next two requirements were added to enable the user to regenerate or to randomise the generated password if they were unhappy with the original password that was generated. These additional requirements were initialised within the *'generatePassword'* function. The randomising of the password was a simple shuffle of the numbers within the chosen random numbers and the re-generation of the password was completed by simply re-initialising the *'generatePassword'* function. These requirements increased the usability of the program as it handed a degree of control of the passwords to the user themselves.

The creation of additional passwords was, again, a simple re-initialisation of the program. This caused a little issue at first as the user was continually presented with the original menu as opposed to just being directed to the *'generatePassword'* so a new function was created called *'createPassword'*. Inside this function, the input of the website name was placed along with a call to the *'generatePassword'* function. This allowed the user to attempt to generate a second password without the need to go through the entire program.

The final requirement was to allow the user to view the existing passwords that were stored within the file. This was broken down into two steps:

- Viewing the passwords in their encoded format
- Viewing the passwords in their plaintext format.

It was decided that the passwords would initially be displayed in their encoded format and that the user would be required to add their sentence to the program in order to view their plaintext passwords. This would provide an additional level of security as, if the user had left the program running and had left the computer, their passwords would not be visible to anyone else in their plaintext format. This was completed by simply opening the file and reading in each of the lines written to it and displaying them to the user inside a function called *'viewPasswords'*.

The plaintext format required some additional thinking and programming. Within the *'viewPasswords'* function, the user was asked to input their sentence to view their passwords in their plaintext format. Once this had been completed a new problem arose, how would the program know if numbers, special characters or both had been added to the end of the sentence before the password had been generated? This took some time to deliberate over but it was decided that the length of the sentence generated within the *'editSentence'* functions was added to the end of the random numbers that were to be stored. This allowed the program to run a test on how long the sentence was when the password was generated and compare it to the size of the sentence when the user entered it. If the password was five, ten or 15 characters shorter when the user entered it then the *'editSentence'* password had numbers, special characters or both respectively appended to the end of the original sentence. Once this check was complete and the sentence was the same length as when the password had been generated it was a simple case of doing the password generation algorithm of finding the characters within the sentence that

correspond to the numbers in the passwords encoded format and re-building the password. Once the passwords had been rebuilt they were displayed back to the user where they could be copied and input to the website account they were associated with.

Once this requirement had been completed, the prototype was complete. The next phase of development was to include data validation, error checking and exception handling within the program. As there were only four instances of the user inputting data (username entry, sentence entry twice and the website entry), this section was quite small in the prototype.

The username provided a new issue to combat: had the username ever been used before? This was a simple issue to answer: check to see if a file had the correct filename on the desktop. This then provided a second issue: what should the program do if the file doesn't exist? Again, this was a simple solve: if there isn't a file, create one. However, this didn't solve the problem as the question 'What if the user misspelt their username?' the program would make a new file and the user wouldn't have access to their passwords. It was at this juncture that a '*Login*' or '*Sign Up*' option was added to the program. Here if the user selected '*Sign Up*' the username was taken from the user and a new file was created on the desktop with that username as the filename. This raised the next issue: what if a file existed with that same username? The fix here was to check to see if a file existed with that username and if not create the file if the file did exist the user was informed that their specified username was unavailable and another was to be input. Once this had been implemented the '*Sign Up*' section was completed. The '*Login*' was easier to implement as it required only one thing: check the file named the same as the username exists, if it existed: proceed, if not: tell the user that the username is invalid and to re-enter the username. This was done to ensure they enter the correct username.

The next area of data validation was the entry of the sentence. It had been decided by this point that, in order to create stronger passwords, the password would require at least one upper case letter. This added some complexity into the data validation of the sentence input as the program would be required to cycle through each letter of the sentence, once it had been input by the user, and checked for any upper case letters. If none were found then the user was informed that no upper case letters had been found and that they were to re-enter their password with a minimum of one upper case letter. As the sentence is not stored within the program or stored within the password file there was no way for the program to determine if the sentence had been entered correctly. The user had the sole responsibility of ensuring the sentence is input correctly every time.

The final entry that required data validation was the website name entry. As there was no way of knowing which website the user was creating a password for the only form of data validation required here was to ensure that the field had not been left blank. Once this had been completed the program was complete and a set of test cases was completed to ensure that all areas of the program worked appropriately.

5.1.1.1. Prototype Testing – Test Cases

The only stage of testing required at this stage is to test the code within the prototype in order to ensure the program does as expected when presented with a specific scenario. The following table is the results generated from the four cases that would fully test the programs password creation functionality and to attempt to find the bugs within the code. The sentence used was ‘ManchesterMetropolitanUniversity’.

Scenario	Test Case	Expected Output	Actual Output
The password requires a number adding to the text	<ol style="list-style-type: none"> 1. Login 2. Enter PassSentence 3. Request a new password 4. Indicate that a number is required 5. Indicate that a special character is not required 	12 letters and one number	<p>Letters: ‘tnvitserolMo’ Number: ‘4’ No bug found.</p>
The password requires a special character adding to the text	<ol style="list-style-type: none"> 1. Login 2. Enter PassSentence 3. Request a new password 4. Indicate that a number is not required 5. Indicate that a special character is required 	12 letters and one special character	<p>Letters: ‘Uossitpeeae&M’ Special Character: ‘&’ No bug found.</p>
The password requires a special character and a number adding to the text	<ol style="list-style-type: none"> 1. Login 2. Enter PassSentence 3. Request a new password 4. Indicate that a number is required 5. Indicate that a special character is required 	12 letters, one special character and one number	<p>Letters: ‘nMeetotMitcl’ Special Character: ‘_’ Number: ‘1’ No bug found.</p>
The password doesn’t require a special character or a number adding to the text	<ol style="list-style-type: none"> 1. Login 2. Enter PassSentence 3. Request a new password 4. Indicate that a number is required 5. Indicate that a special character is required 	12 letters only	<p>Letters: ‘rtltanciMnpo’ No bug found.</p>

Table 5-1: Test case results for the Prototype

5.2. Product Implementation

Once the implementation and testing of the prototype were completed the next phase of operation was to begin the development of the final product. As mentioned in the Coding section of the design chapter, the author intended to create a cross-platform browser plugin using the Kango frameworks. When the investigation into the supporting documentation for these frameworks it became apparent that this specific set of frameworks was not designed with novice developers in mind. These frameworks are aimed at people who actually know and understand how to develop browser plugins and, as this was the authors first ever attempt to create a browser plugin, the option of using these frameworks was considered no longer viable and, as such, the possibility of creating a cross-platform browser plugin was no longer attainable. A decision then had to be made as to which platform would become the focal point for the browser plugin. The decision was relatively simple as, as stated in the design chapter, Google Chrome occupies 53% of the market shares. Creating a browser plugin on the Chrome platform would give the largest coverage and, as an added incentive, Chrome offers its own storage API that can be used to store data locally or can be synchronised across multiple machines. It was therefore decided that the browser plugin would be a Chrome plugin.

The first phase was to create a folder that contained the manifest required to create a browser plugin. This manifest is written in the JSON format and is required by all browser plugins in order to be able to use the plugin. The supporting documentation for manifest gave an in-depth knowledge of what was required within the manifest and it was concluded that the only required 'permission' within the manifest was the 'storage' as the plugin would be required to store the passwords.

Once this phase was completed the plugin was then implemented in the browser and the 'button' for the plugin was visible to the user. When this button was clicked nothing happened as there were no additional features within the manifest. In order for anything to happen when this button is clicked a '*browser_action*' needs to be declared within the manifest. Also, an icon needed to be added to the folder and referenced within the manifest in order to alter the display of the button to the user and the author decided that, with the aid of Photoshop, he would create his own logo and browser icons for the product.

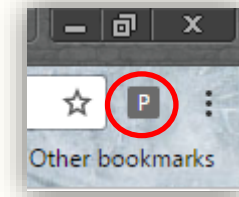


Figure 5.i: Browser button on display

Upon completing some research into different text effects available online a decision was made to create the logo and icons in Photoshop using the clean, glossy plastic text effect available at <https://design.tutsplus.com/tutorials/how-to-create-a-clean-glossy-plastic-text-effect-in-adobe-photoshop--cms-22962>. Once the instructions within this website had been completed the logo and icons were completed the finished items can be seen on the product itself and in appendix 1. In order to display the icons on the browser button, the

manifest required an *'icons'* section to be declared and the icons to be set to the created images. It was required that the author created three different sized icons as Chrome uses three different sized images when displaying the information about the plugin. These sizes were 16x16, 48x48 and 128x128 pixels square images. The 16x16 pixel image is used on the browser button itself, the 48x48 pixel image is used on the extensions page in Chrome and the 128x128 pixel images is used when the user clicks on the 'Details' link on the extensions page within Chrome. The *'icons'* section of the manifest was edited and the icons were then being displayed in all areas.

The next phase was to set an action that occurs when the user clicks on the button in the browser. This was completed by creating a Hypertext Markup Language (HTML) page and setting the *'default_popup'* within the *'browser_action'* section of the manifest to be the created HTML page. Once this was completed whenever the user clicks on the button in the browser window the created HTML page would be displayed to the user.

Once the *'default_popup'* page had been written and was being displayed the author chose to determine exactly how many HTML pages were going to be required in order to display all of the program to the user. There were eight pages that needed to be designed based on the design chapter of this report so, with the *'default_popup'* being used as the first page, there were seven more that needed to be created. The list of pages are as follows:

- Welcome Page
- Login Page
- Sign Up Page
- Returning User Page
- New User Page
- Create Password Page
- Display Password Page and
- View Passwords Page

This seemed a little excessive to the author and a decision was made to combine the Welcome page and Login page into a single page (herein referred to as the Login page) that welcomes the user and offers them the opportunity to either 'Login' or 'Sign Up'. This was deemed a better function of the program and if the user has fewer pages to go through in order to obtain their objective they would be more likely, in the author's opinion, to use the product more often. Upon completing this new combined page the author then decided to look at the orientation of the page itself. In the design phase the orientation of the page was landscape, however, when displaying this to the user the orientation



Figure 5.ii: Welcome Page in landscape orientation

didn't seem right. It looked like there was a lot of wasted space that the user may not like and the addition of a scroll bar on the right of the page made it feel like the orientation was wrong. The page was then changed to portrait orientation and the page looked better. The layout was more pleasing to the eye and the style seemed more presentable so the decision was made to create the plugin using a portrait window for displaying purposes.



Figure 5.iii: Welcome Page in portrait orientation

Once the welcome page frame had been completed the additional six pages were designed using the same frame. Upon completion of the HTML pages, the next step was to style them using Cascading Style Sheets (CSS). The decisions for what CSS styles would be added were kept to a minimum to avoid overcomplicating the design and it also provided a uniformity across all of the pages which aid in the user-friendliness of the program. The CSS selections were as follows:

- Background Colour: Set to a pale orange colour,
- Text Font: set to 15 pixels in size and Arial font.

These two items were the base for all of the pages with aspects to all of the writing on each of the pages. Only the size and text effects (bold, italic and underlined) would differ from page to page if it was deemed necessary.

Upon completing the visual aspect of the browser plugin, the next phase was to allow the pages to communicate and work together in order to achieve the desired results of the program. This was completed by implementing JavaScript pages that were referenced in the HTML pages.

The original plan of operation was to have the HTML pages talk directly to each other in the background of the plugin, however, browser plugins cannot communicate in this way. Any form of communication between pages of a browser plugin needs to be completed via the use of a background script file, therefore, the next step was to create this background script and to initialise within the manifest.

Once this was completed the HTML pages within the plugin were able to communicate via the mediation of the background script. The communication was completed using the Chrome messaging API. This process is completed by the current window sending a

message to the background script which then processes this message and then sends a message from the background script to the next page, which will have loaded in the window while the messages were being sent. This whole process added an extra layer of complication to the build but was necessary in order to have the program provide the desired service.

Each of the seven pages required its own JavaScript each of which had different processes to complete. The workflow of each of these processes is quite complex but a simple overview of the steps taken is as follows:

Welcome Page

- If Username entered and Login button clicked – Username sent to background script. Returning User page displayed.
- If Sign Up button clicked – No message sent to background script. Sign Up page displayed.

Sign Up Page

- Username entered and Sign Up button clicked - Username sent to background script. New User page displayed

Returning/New User Page

- When page loads a message is sent to the background script to tell it that that page is now loaded
- Background script sends Returning/New User page the entered Username.
- Username received from background script – ‘Welcome Username’ displayed.
- Returning User page displays Create New Password button and View Existing Passwords Button
- New User page displays Create New Password button
- If Create New Password button is clicked the Create New Password page is loaded
- If View Existing Passwords button is clicked the View Existing Passwords page is loaded

Create New Password Page

- User inputs sentence
- Sentence is checked for uppercase letter – if none found an error message is displayed to the user and they are asked to enter a valid sentence
- Sentence is hashed and the hash is sent to the background script to be stored.
- User enters the website name associated with this password

- User states if numbers are allowed
- User states if special characters are allowed
- (If required) New sentence is created with numbers and special characters
- New sentence, original sentence and website are sent to the background script
- Display Passwords page loads.

Display Password Page

- When page loads a message is sent to the background script to tell it that that page is now loaded.
- Background script sends Display Password page new sentence, original sentence and website.
- Display Password page displays 'Password for Website'
- Display Password then creates the password using the new sentence
- Password is displayed in plain text and encoded format to the user
- User has option to 'Generate Another' or 'Save Password to Database'*
- If 'Generate Another' is chosen then another password is generated and displayed.
- If/when 'Save Password to Database' is chosen then the encoded password is sent to the background script. Saved Password page** is loaded.

Saved Password Page

- No JavaScript is associated with this page as no communication with the background script is required. Saved Password page offers the user the opportunity to 'Create Another Password' or to 'View Existing Passwords'
- If 'Create Another' is chosen then the 'Display Password' page is loaded
- If 'View Existing Passwords' is chosen then View Existing Passwords Page is loaded.

View Existing Passwords Page

- When page loads a message is sent to the background script to tell it that that page is now loaded.
- Background script retrieves the stored websites and associated passwords from the database and sends them to the View Existing Passwords page.
- View Existing Passwords page then builds a table containing websites in column one and their associated passwords in their encoded format in column two and displays it to the user.
- User then enters the original sentence
- The View Existing Passwords page sends a message to the background script requesting the stored hash value.

- Background script replies to the message with the hashed value stored for the user.
- The two hashes are then compared against each other to ensure the correct sentence has been entered.
- View Existing Passwords page re-created the passwords in their plaintext format and creates a new table containing the websites in column one and the plaintext passwords in column two.
- User offered the opportunity to 'Create Another Password'
- If 'Create Another Password' button is clicked the 'Create Password' page is loaded.

* Randomise function was determined to be unnecessary and was removed

** Additional page added to allow users to see the password has been stored. This was deemed a better approach than simply display 'Password Saved' to the user.

Once this stage had been completed, the production of the browser plugin was complete. The next phase was the testing phase using the same set of test cases that were used for the testing of the prototype.

5.2.1. Product Testing – Test Cases

As with the testing conducted on the prototype, the code would require testing in order to ensure that the programs password creating function operated as expected. The same set of test cases will be used to test the program as this was determined to have the best chance of finding a bug in the code. However, unlike the prototype, there is a second phase of testing as the final product requires a usability test in order to ensure the program is as user-friendly as possible. This will be completed by distributing the product to a select few and documenting their findings by use of a questionnaire. A copy of the questionnaire can be found in appendix 2.

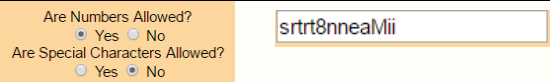

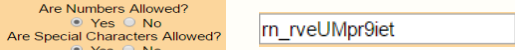
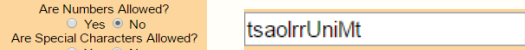
Scenario	Test Case	Expected Output	Actual Output
The password requires a number adding to the text	6. Login 7. Enter PassSentence 8. Request a new password 9. Indicate that a number is required 10. Indicate that a special character is not required	12 letters and one number	 Letters: 'srtrtneaMii' Number: '8' No bug found.
The password requires a special character adding to the text	6. Login 7. Enter PassSentence 8. Request a new password 9. Indicate that a number is not required 10. Indicate that a special character is required	12 letters and one special character	 Letters: 'enrMnntssiyv' Special Character: '&' No bug found.
The password requires a special character and a number adding to the text	6. Login 7. Enter PassSentence 8. Request a new password 9. Indicate that a number is required 10. Indicate that a special character is required	12 letters, one special character and one number	 Letters: 'rnrvеUMpr9iet' Special Character: '_' Number: '9' No bug found.
The password doesn't require a special character or a number adding to the text	6. Login 7. Enter PassSentence 8. Request a new password 9. Indicate that a number is required 10. Indicate that a special character is required	12 letters only	 Letters: 'tsaolrrUniMt' No bug found.

Table 5-2: Test case results for the Product

5.2.2. Usability

A copy of the completed product was issued to a select group of people. Their task was to create three passwords and view them once complete. They were then asked to complete a short questionnaire based on their experience of the product. These questions were:

- On a scale of one to five, one being very difficult and five being very easy, how would you rate the difficulty in creating a password using this product?
- On a scale of one to five, one being very displeasing and five being very pleasing, how would you rate the level of control you had over the creation of the passwords?
- On a scale of one to five, one being very displeasing and five being very pleasing, how would you rate the look of the product?
- On a scale of one to five, one being very unlikely and five being very likely, how likely are you to continue using the product after testing it?
- On a scale of one to five, one being very unlikely and five being very likely, how likely are you to recommend the product to a friend or family member?
- Do you have any other comments on the usability of the program?

The product appeared to have a positive impact on these individuals as almost all of the subjects gave rating of four and above for the first three questions with only one person giving it a rating of three. All but one of the subjects stated they would be 'likely' or 'very likely' to continue using the product after testing with only two being less than 'likely' or 'very likely' to recommend the product. Only two subjects offered additional comments. These were:

- 'The overall ease of use and simplistic design is what is going to make PassSentence my new password manager!'
- 'The fact that it is so simple to create such strong passwords while still not having to worry about remembering them makes PassSentence just brilliant.'

It would appear to the author that the design of the final product and the password selection algorithm have both exceeded the expectations set out at the beginning of this report. The passwords being generated are almost impossible to hack and the overall user experience has received incredibly positive feedback from the people questioned.

6. Evaluation

When this project was begun there were two points raised with regards to existing password managers: their usability and single point of failure. PassSentence has solved both of these issues.

The single point of failure has been solved by storing the passwords in such a unique way that it would take an extraordinary amount of time to obtain each of the passwords making any attempt to obtain them seem less appealing to an attacker. The key to successfully protecting your accounts from an attacker is to make the effort to obtain access less beneficial than what can be obtained. When the plaintext passwords were entered into the website <https://howsecureismypassword.net/> the average length of time to hack a single password was in the range of hundreds of millions of years. As the plaintext version of these passwords seems almost impossible to hack, the fact that the passwords are stored by way of using the character placements from within a sentence makes them all the more secure. As mentioned earlier this has solved the issue of having a single point of failure. PassSentence requires no master passwords as even if an attacker was to obtain the list of passwords within the database there would be no way for them to decode the passwords without access to the sentence as well.

The usability of the product was deemed almost as important when creating a password manager as the security it provided. If a user had difficulty in operating the program then they would be much less likely to continue using it. The author believes that by having only three items to enter when creating a password combined with handing some level of control back to the user when generating the passwords was the key to success. The look and ‘feel’ of the product was kept to a comfortable standard as to not overpower the users senses with a multitude of colours and fonts on the different pages and also having a sense of uniformity allows the user to develop a sense of comfort throughout the process as on each page only a few items change.

7. Conclusion

Passwords are the most secure form of protection as, outside of cyberspace, they should only exist in your head. As they are such a secure form of protection they will, in the opinion of the author, never be replaced. Therefore the need to make a better form of storage for them will never dissipate. Users are obtaining more online accounts each requiring a password and the users are expected to not only remember them all but also remember which password is for which account. This increase has led to a security issue known as password reuse which decreases the security of password with each new account it becomes used for. The solution to this issue was to create a password manager but, again, these came with their own set of problems: a single master password used to protect all of your other passwords and the need to make them user-friendly. Some password managers forego the usability for security while others forego the security for usability. This is where PassSentence comes in. PassSentence has been designed with a healthy balance of usability combined with security in order to allow even a novice user the ability to create secure passwords and not have to worry about how they're going to remember all of them. This issue of recalling all of a user's passwords has been abolished and replaced with only having to remember a single sentence. Combine this reduced complexity of remembering a multitude of passwords and their associated accounts with passwords that are stored in an encoded format that would take hackers several millions of years to break a single password and you have a recipe for success.

7.1. Limitations

There are, however, limitations to this particular version of PassSentence.

7.1.1. Single Browser Compatibility

It currently only works on the Google Chrome browser and, as such, requires users to have a Google account in order for the browser plugin to access the storage API as well as having Google Chrome installed on their machine. Without these two items, PassSentence will not work.

7.1.2. No Ability to Change PassSentence

There is a significant limitation in the inability of the user to change their PassSentence once the first password has been stored. Currently the only way to change a PassSentence is to completely clear the database by clicking the 'Sign Up' button on the welcome page. This is not ideal as the user will lose all of the stored passwords within the database.

7.1.3. Limited Usability Testing

The usability could be improved by adding a function in that would allow the user to fill the password field of the web page by clicking a button as opposed to having to copy and

paste the passwords in, assuming that the website will allow the pasting of passwords in the password field.

7.1.4. No browser Independent Storage

A final idea that the author discovered was that having the ability to communicate with a database not provided by a single browser but rather by a third party, such as SQL, would aid in the development of a cross-platform browser that could have a larger presence on the market.

7.1.5. Password Length

During the design phase of the project a ‘Minimum’ and ‘Maximum’ password length had been specified on the ‘Create Password’ page. This appears to have been overlooked during the development phase and has been missed out of the final product build.

7.2. Future Work

The author would like to suggest that a more in depth analysis of the usability of the product be considered for future works as the original testing the product on only a handful of people could have led to biased results. With regards to the limitation of an inability to change a user’s PassSentence the author would suggest a simple new ‘Change PassSentence’ page that would require the user to enter their current PassSentence, have the hash value checked to ensure this is the correct original PassSentence and then enter a new PassSentence. The hash value within the database could then be replaced with the hash value of the new PassSentence. The passwords within the database would become obsolete as they would never be decoded correctly again and it would be up to the user to ensure all of the passwords are replaced with passwords generated using the new PassSentence. The inclusion of a simple dropdown menu or slider with predetermined numbers entered onto would have been easily implemented correct this limitation. It would allow the user to specify how long (number and special character not included) the password could be.

References

Main Body References

Polybius and Shuckburgh, E. (2012). The Histories of Polybius. 1st ed. Cambridge: Cambridge University Press.

Crisman, P.A. ed., (1965). The compatible time-sharing system: A programmer's guide. Mit Press.

Chiasson, S., Oorschot, P. and Biddle, R. (2006). A Usability Study and Critique of Two Password Managers. In Usenix Security (Vol. 6). Available at: https://www.usenix.org/legacy/events/sec06/tech/full_papers/chiasson/chiasson.pdf [Accessed 11 Nov. 2016].

Gonzalez, R., Chen, E.Y. and Jackson, C., (2013). Automated password extraction attack on modern password managers. arXiv preprint arXiv:1309.1416.

Kuo, C., Romanosky, S. and Cranor, L.F., (2006). Human selection of mnemonic phrase-based passwords. In Proceedings of the second symposium on Usable privacy and security (pp. 67-78). ACM.

Ives, B., Walsh, K.R. and Schneider, H., (2004). The domino effect of password reuse. Communications of the ACM, 47(4), pp.75-78.

Gaw, S. and Felten, E.W., (2006). Password management strategies for online accounts. In Proceedings of the second symposium on Usable privacy and security (pp. 44-55). ACM.

Li, Z., He, W., Akhawe, D. and Song, D., (2014). The emperor's new password manager: Security analysis of web-based password managers. In 23rd USENIX Security Symposium (USENIX Security 14) (pp. 465-479).

Matsumoto, T. (2002). *Importance of Open Discussion on Adversarial Analyses for Mobile Security Technologies---A Case Study for User Identification---*. 14th ed. [ebook] Seoul. Available at: <http://perso.telecom-paristech.fr/~chollet/Biblio/Articles/Domaines/BIOMET/spoof-fp5p4.pdf> [Accessed 11 Nov. 2016].

Hall, C. (2016). *Microsoft Lumia 950 review: The dawn of Windows 10 Mobile - Pocket-lint*. [online] Pocket-lint.com. Available at: <http://www.pocket-lint.com/review/136018-microsoft-lumia-950-review-the-dawn-of-windows-10-mobile> [Accessed 11 Nov. 2016].

Cranor, L. and Garfinkel, S. (2005). Security and usability. 1st ed. Beijing: O'Reilly.

Adams, A. and Sasse, M. (1999). USERS ARE NOT THE ENEMY. COMMUNICATIONS OF THE ACM, [online] (Volume 42 Issue 12), pp.40-46. Available at: <http://cacm.acm.org.ezproxy.mmu.ac.uk/magazines/1999/12/7773-users-are-not-the-enemy/fulltext> [Accessed 18 Nov. 2016].

Whitten, A. and Tygar, J.D., (1999). Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In Usenix Security (Vol. 1999).

Abratt, D., Mallinson, W. and Bekker, A., (2003). Usability Testing: Recipe for Success. STARWEST: International Conference on Software Testing Analysis and Review.

Le, C. (2006). A Survey of Biometrics Security Systems [online]. Available at: <http://www.cse.wustl.edu/~jain/cse571-11/ftp/biomet/index.html> [Accessed 18 Nov. 2016].

Keith, M., Shao, B. and Steinbart, P. (2007). The usability of passphrases for authentication: An empirical field study. International Journal of Human-Computer Studies, [online] 65(1), pp.17-28. Available at: <http://www.sciencedirect.com.ezproxy.mmu.ac.uk/science/article/pii/S1071581906001236> [Accessed 18 Nov. 2016].

Stobert, E. and Biddle, R. (2014). A Password Manager that Doesn't Remember Passwords. [online] pp.39-52. Available at: http://delivery.acm.org.ezproxy.mmu.ac.uk/10.1145/2690000/2683471/p39-stobert.pdf?ip=149.170.3.14&id=2683471&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E8670C7C518068584%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=694905259&CFTOKEN=42396938&__acm__=1479486129_0cf6d461849a9a6386a636d68c65f51e [Accessed 18 Nov. 2016].

Appendix 3 – ToR references

SearchSecurity. (2016). *What is password? - Definition from WhatIs.com*. [online] Available at: <http://searchsecurity.techtarget.com/definition/password> [Accessed 14 Oct. 2016].

Passrequirements.com. (2016). *Password requirements for itunes | passrequirements.com*. [online] Available at: <http://passrequirements.com/passwordrequirements/itunes> [Accessed 30 Sep. 2016].

Passrequirements.com. (2016). *Password requirements for microsoft | passrequirements.com*. [online] Available at: <http://passrequirements.com/passwordrequirements/hotmail> [Accessed 30 Sep. 2016].

Matsumoto, T. (2002). *Importance of Open Discussion on Adversarial Analyses for Mobile Security Technologies---A Case Study for User Identification---*. 14th ed. [ebook] Seoul. Available at: <http://perso.telecom-paristech.fr/~chollet/Biblio/Articles/Domaines/BIOMET/spoof-fp5p4.pdf> [Accessed 14 Oct. 2016].

Murgia, M. (2016). *Biometrics will replace passwords, but it's a bad idea*. [online] The Telegraph. Available at: <http://www.telegraph.co.uk/technology/2016/05/26/biometrics-will-replace-passwords-but-its-a-bad-idea/> [Accessed 14 Oct. 2016].

Hall, C. (2016). *Microsoft Lumia 950 review: The dawn of Windows 10 Mobile - Pocket-lint*. [online] Pocket-lint.com. Available at: <http://www.pocket-lint.com/review/136018-microsoft-lumia-950-review-the-dawn-of-windows-10-mobile> [Accessed 14 Oct. 2016].

Ofcom.org.uk. (2016). *UK adults taking online password security risks - Ofcom*. [online] Available at: <https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2013/uk-adults-taking-online-password-security-risks> [Accessed 14 Oct. 2016].

Appendices

Appendix 1 – Icons and Title Images



Figure 0.i: All images used for icons and product title

Appendix 2 – Usability Questionnaire

https://docs.google.com/forms/d/e/1FAIpQLSce3W6Cj6PnXUcGarKcnNUUGeCvUZXB7thHJoBpEvbx40U9hA/viewform?usp=sf_link

Appendix 3 – Ethics Form

ETHICS CHECKLIST

This checklist must be completed **before** commencement of **any** research project. This includes projects undertaken by **staff and by students as part of a UG, PGT or PGR programme**. Please attach a Risk Assessment.



Please also refer to the [University's Academic Ethics Procedures](#); [Standard Operating Procedures](#) and the [University's Guidelines on Good Research Practice](#)

Full name and title of applicant:	Dale Stubbs	
University Telephone Number:	N/A	
University Email address:	14024149@stu.mmu.ac.uk	
Status:	Undergraduate Student <input checked="" type="checkbox"/> Postgraduate Student: Taught <input type="checkbox"/> Postgraduate Student: Research <input type="checkbox"/> Staff <input type="checkbox"/>	
Department/School/Other Unit:	Science and Engineering	
Programme of study (if applicable):	Computer Forensics and Security	
Name of DoS/Supervisor/Line manager:	Anthony Kleerekoper	
Project Title:	Your PassWORD is Outdated An Upgrade to PassSENTENCE is advised!	
Start & End date (cannot be retrospective):	18/06/2016 to 24/04/2017	
Number of participants (if applicable):	1	
Funding Source:		
Brief description of research project activities (300 words max):		
The aim of this project is to create a new type of password manager that has no single point of failure. I will then implement this PassSENTENCE manager as a browser plug-in.		
	YES	NO
Does the project involve NHS patients or resources? If 'yes' please note that your project may need NHS National Research Ethics Service (NRES) approval. Be aware that research carried out in a NHS trust also requires governance approval. Click here to find out if your research requires NRES approval Click here to visit the National Research Ethics Service website To find out more about Governance Approval in the NHS click here	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Does the project require NRES approval?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
If yes, has approval been granted by NRES? Attach copy of letter of approval. Approval cannot be granted without a copy of the letter.	<input type="checkbox"/>	<input type="checkbox"/>

NB Question 2 should only be answered if you have answered YES to Question 1. All other questions are mandatory.	YES	NO
1. Are you are gathering data from people?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
For information on why you need informed consent from your participants please click here		
2. If you are gathering data from people, have you:	<input type="checkbox"/>	<input type="checkbox"/>
a. attached a participant information sheet explaining your approach to their involvement in your research and maintaining confidentiality of their data?	<input type="checkbox"/>	<input type="checkbox"/>
b. attached a consent form? (not required for questionnaires)	<input type="checkbox"/>	<input type="checkbox"/>
Click here to see an example of a participant information sheet and consent form		
3. Are you gathering data from secondary sources such as websites, archive material, and research datasets?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to find out what ethical issues may exist with secondary data		
4. Have you read the guidance on data protection issues?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
a. Have you considered and addressed data protection issues – relating to storing and disposing of data?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
b. Is this in an auditable form? (can you trace use of the data from collection to disposal)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5. Have you read the guidance on appropriate research and consent procedures for participants who may be perceived to be vulnerable?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
a. Does your study involve participants who are particularly vulnerable or unable to give informed consent (e.g. children, people with learning disabilities, your own students)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6. Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited (e.g. students at school, members of self-help group, nursing home residents)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click for an example of a PIS and information about gatekeepers		
7. Will the study involve the use of participants' images or sensitive data (e.g. participants personal details stored electronically, image capture techniques)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here for guidance on images and sensitive data		
8. Will the study involve discussion of sensitive topics (e.g. sexual activity, drug use)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here for an advisory distress protocol		
9. Could the study induce psychological stress or anxiety in participants or those associated with the research, however unlikely you think that risk is?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read about how to deal with stress and anxiety caused by research procedures		
10. Will blood or tissue samples be obtained from participants?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read how the Human Tissue Act might affect your work		
11. Is your research governed by the Ionising Radiation (Medical Exposure) Regulations (IRMER) 2000?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to learn more about IRMER		
12. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read about how participants need to be warned of potential risks in this kind of research		
13. Is pain or more than mild discomfort likely to result from the study? Please attach the pain assessment tool you will be using.	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Click here to read how participants need to be warned of pain or mild discomfort resulting from the study and what do about it.		
14. Will the study involve prolonged or repetitive testing or does it include a physical intervention?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to discover what constitutes a physical intervention and here to read how any prolonged or repetitive testing needs to managed for participant wellbeing and safety		
15. Will participants to take part in the study without their knowledge and informed consent? If yes, please include a justification.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read about situations where research may be carried out without informed consent		
16. Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read guidance on payment for participants		
17. Is there an existing relationship between the researcher(s) and the participant(s) that needs to be considered? For instance, a lecturer researching his/her students, or a manager interviewing her/his staff?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Click here to read guidance on how existing power relationships need to be dealt with in research procedures		
18. Have you undertaken Risk Assessments for each of the procedures that you are undertaking?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
19. Is any of the research activity taking place outside of the UK?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
20. Does your research fit into any of the following security sensitive categories: <ul style="list-style-type: none"> • commissioned by the military • commissioned under an EU security call • involve the acquisition of security clearances • concerns terrorist or extreme groups If Yes, please complete a Security Sensitive Information Form	<input type="checkbox"/>	<input checked="" type="checkbox"/>

I understand that if granted, this approval will apply to the current project protocol and timeframe stated. If there are any changes I will be required to review the ethical consideration(s) and this will include completion of a 'Request for Amendment' form.

- ☒ have attached a Risk Assessment
☒ have attached an Insurance Checklist

If the applicant has answered YES to ANY of the questions 14 – 17 then they must complete the [MMU Application for Ethical Approval](#).

Signature of Applicant: **Dale Stubbs** Digitally signed by Dale Stubbs
Date: 2016.10.26 14:11:00
+01'00' Date: **26/10/2016** (DD/MM/YY)

Independent Approval for the above project is (please check the appropriate box):

Granted

☐ I confirm that there are no ethical issues requiring further consideration and the project can commence.

Not Granted

☐ I confirm that there are ethical issues requiring further consideration and will refer the project protocol to the Faculty Research Group Officer.

Signature: _____ Date: _____ (DD/MM/YY)

Print Name: _____ Position: _____

**Approver: Independent Scrutiniser for UG and PG Taught/ PGRs RD1 Scrutiniser/
Faculty Head of Ethics for staff.**

Appendix 4 – Terms of Reference

Course Specific Learning Outcomes

Upon completion of this project the student will:

- Develop an understanding of the scope and theoretical underpinnings of forensic computing including its technical, professional, legal and ethical aspects;
- Develop a critical and analytical approach to problem-solving in the forensic domain.
- Independently plan, manage and successfully complete a project of substantial size in an area that is relevant to your Degree programme;
- Demonstrate that you have the capacity to gain new skills and knowledge independently of teaching;
- Critically reflect and evaluate existing work and your own work;
- Integrate the learning obtained from other units taken on your Degree programme.

Project Background:

Passwords are everywhere and used on a daily basis by most, if not all, of today's population that have access to a computer, phone tablet or Smart Technology. SearchSecurity defines a password as:

"A password is an unspaced sequence of characters used to determine that a computer user requesting access to a computer system is really that particular user. Typically, users of a multiuser or securely protected single-user system claim a unique name (called a user ID) that can be generally known. In order to verify that someone entering that user ID really is that person, a second identification, the password, known only to that person and to the system itself, is entered by the user." (SearchSecurity, 2016)

Anyone that has to create an online account will most likely be asked to provide a password as a way of securing the data held within the account being created by only allowing those who can provide the password with access to the data being held. The concept of the password system is fundamentally valid, however, the variety of rules regarding password creation from different sources creates a frustrating situation when creating passwords. Moreover, it increases the difficulty in memorising the multitude of passwords that any one person will need to create throughout their lifetime. For example when a user creates a new Apple account they need to follow these rules:

"Passwords must be at least 8 characters, including a number, an uppercase letter, and a lowercase letter. Don't use spaces, the same character 3 times in a row, your Apple ID, or a password you've used in the last year." (Passrequirements.com, 2016)

Whereas someone making a new Microsoft account have these rules to follow:

"Passwords must have at least 8 characters and contain at least two of the following: uppercase letters, lowercase letters, numbers, and symbols." (Passrequirements.com, 2016)

An attempt has already been introduced to overcome the need to create and remember multiple passwords for the multitude of accounts that a single person may require in their lifetime. This 'solution' is the use of a password manager. There are several password managers in existence to date and they range in price from free to \$40 (£31.45) per year for the premium services provided.

The theory behind these passwords managers is, again, a fundamentally valid theory but it appears to be, for the most part, a double edged sword. These programs generate and store your unique passwords for all of the accounts you create which is a valuable asset to have but all of them require a 'master password' to keep your passwords safe and this is their single biggest problem. It is the one single point of failure on each of the password managers; if someone 'cracks' your master password then they have all of your passwords.

This has led to an attempt to keep the master password safe by replacing it with biometric security features. Fingerprint scanners are one such feature. On the face of it, this would appear to solve the issue, however, in reality, the cost of installing or distributing fingerprint scanners could be expensive. They are also not as infallible as most people believe as they can give an increased amount of false negatives (denying the legitimate user access) for reasons such as wet/damp or dirty fingers. Also, research at Yokohama National University in 2002 found that 'gelatin (or "Gummi") fingers were successful around 80% of the time' (Matsumoto, 2002).

Another biometric feature I will discuss the 'futuristic' Retinal Scanner. Microsoft has announced that the Microsoft Lumia 950 will have a feature called "Windows Hello" which will grant fast access to the user's phone after scanning the user's iris. The scanner shines a red light into your eye that reviewers of the product found to be 'quite dazzling after multiple uses which is far from ideal if you are a frequent user of the phone.' (Hall, 2016)

This leads to a lack of trust lack of proficiency when using biometric features in security. Some people don't trust the master password system or biometric tools whereas most just don't like the idea of not knowing what their passwords. Users are looking for easier ways to overcome these issues and have done so by turning to a less secure but easier way to create passwords for various accounts – Password Reuse. A study by Ofcom, the UK communications watchdog, revealed that 55% of the 1805 adult internet users questioned used password reuse (Ofcom.org.uk, 2013).

Password reuse, as the name suggests, is where rather than making a brand new password and attempting to memorise it and what account it is used for, a user, will use the same password with slight variances to it in order to meet the specified criteria. This creates a potentially dangerous situation for those who wish to protect their personal information as if one password is cracked by a computer criminal then there's a higher risk of them cracking all of the user's passwords.

Project Overview:

I intend to create a program that, on the surface, will be a password manager but one that does not generate random passwords in the way that most others do. My program will ask the user to provide a 'PassSentence' (an easy to remember sentence) that only the user will ever have access to. My program will then generate a series of numbers that will represent the placement of characters within the sentence and which order they are to be used.

For example, if I were to use the sentence

'Manchester_Metropolitan-University!Is&Great'

My program would produce an outcome that would look like '6, 21, 42, 11, 25' which would mean the actual password for the account would be 'eta_U'.

This eliminates the requirement to remember a lot of passwords and replaces it with the need to only remember your selected sentence. The passwords generated from this sentence will be randomly generated allowing for highly secure passwords. However, unlike current password managers, the password characters will never be stored, only the character placements and order will be stored (6, 21,42,11,25 from the example above). This means that even if a hacker acquires access to your password database they would only see a database of numbers thus removing the single point of failure.

The single biggest issue I will come up against is the lack of a single generic set of rules for creating a new password for a specific account. I will need to research the password requirements on as many sites as deemed necessary in order to obtain an average set of requirements and use this information to create a set of guidelines for my passwords generator to follow.

Based on the information obtained I will then need to grant a certain level of control to the user in order to generate a suitable password for certain accounts (numeric characters and/or non-alphanumeric characters).

This will not affect the security of the passwords created or stored but it will pose a challenge when attempting to write the code for the program and it will also add in extra areas where human error can occur.

I am also unsure of which language to write the program in as I am most comfortable when using Java but I do not believe this will provide me with a good enough program. As a result of this, I will be attempting to enhance my understanding of the Python programming language and creating the program using Python.

I will be creating a stand-alone program to use as my prototype which will be written in Python and then using this prototype as the basis for creating my browser plug-in (extension). I will attempt to create the extension using the Kango platform which itself uses JavaScript for the creation of browser extensions.

Aim:

The aim of this project is to create a new type of password manager that has no single point of failure. If a hacker was to obtain your passwords from the password manager they would only have a set of numbers and would be deemed useless by the hacker.

Objectives:

In order to achieve the required aims, I will:

- Produce a fully functioning prototype of the program to begin my testing of the product in order to determine how many of the passwords need to be cracked before the entire sentence becomes compromised. This will require a deeper understanding of Probability Theory.
- Determine a minimum character length for the PassSentence itself to ensure the amount of passwords that may be required can be generated with a minimal amount of crossover of characters. This too will require a deeper understanding of Probability Theory.
- Generate a minimum requirement for each of the generated passwords in order to ensure compatibility of the password to the specified account. I will need to give the user the ability to request a password that contains (or does not contain) capital letters, numbers and/or non-alphanumeric characters depending on the accounts password rules.

Problems:

Incorrect algorithm – The inability to create the necessary algorithm for the creation of every password requirement is the most important problem I may face. If this occurs I will need to give the user the option of entering their own password into the program and storing it appropriately.

Cross-Platform Compatibility – There will be a high probability that I will not be able to ensure my product will work with all currently available web browsers. I will need to aim my product at a selection of the most popular browsers on the market today to achieve the largest scope of coverage possible.

Timetable and Deliverables:

Week Number	Week Commencing	Work To Be Done	Deliverable
Week 0	19/09/2016	<u>Python Programming</u> Code Academy Tutorial Videos	

Week 1	26/09/2016	<u>Python Programming</u> Lynda.com Tutorial Videos	
Week 2	03/10/2016	<u>Draft 1 of ToR</u> Complete first draft of ToR.	Draft 1 of ToR
Week 3	10/10/2016	<u>Draft 2 of ToR</u> Make necessary modifications to ToR	Draft 2 of ToR
Week 4	17/10/2016	<u>ToR and Ethics Form</u> Final Draft of ToR and Ethics Form	Completed ToR Uploaded
Week 5	24/10/2016	<u>Literature Review (LR)</u> Find and evaluate 8 – 10 papers on passwords and password managers	Draft 1 of LR and Ethics Form Uploaded
Week 6	31/10/2016	<u>Second Draft of LR</u> Make necessary amendments to LR	Completed LR Uploaded
Week 7	07/11/2016	<u>Probability Theory</u> <u>Research</u> Calculate the number of possible passwords that can be generated from a specific password length.	Minimum Length of PassSentence Confirmation
Week 8	14/11/2016	<u>Probability Theory</u> <u>Research</u> Calculate the rate of character crossover within the password generator	Quantity of compromised passwords needed to crack PassSentence Determined
Week 9	21/11/2016	<u>Algorithm design</u> Construct the pseudocode for the password generation algorithm along with the user interface	Product Pseudocode
Week 10	28/11/2016	<u>Algorithm design and</u> <u>Construction of Prototype</u> Convert the pseudocode into Python code and construct the prototype.	Prototype Constructed

Week 11	05/12/2016	<u>Product Design</u> Create a detailed document of how my product will work and instructions of use.	Product Design and Guidelines of Use
Week 12	12/12/2016	<u>Prototype Debugging</u> Use the PassSentence Prototype for debugging purposes.	Debugging Complete
Week 13	09/01/2017	<u>JavaScript Pseudocode</u> Create pseudocode of the Python program for conversion into JavaScript	JavaScript Pseudocode
Week 14	16/01/2017	<u>JavaScript Code</u> Convert the pseudocode into JavaScript code	JavaScript Code
Week 15	23/01/2017	<u>ED</u> Complete the evaluation design of the product	First Draft of ED
Week 16	30/01/2017	<u>ED Draft 2</u> Make the necessary amendments to the ED	Complete ED
Week 17	06/02/2017	<u>Product Construction</u> Build the PassSentence browser plugin and begin testing.	Final Product and Test Results
Week 18	13/02/2017	<u>Report Outline (RO)</u> Complete the first draft of the report.	First draft of RO
Week 19	20/02/2017	<u>RO Draft 2</u> Make the necessary amendments to the RO	Complete RO
Week 20	27/02/2017	<u>Presentation Slides</u> Produce the necessary slides for the presentation	Presentation Slides Draft 1
Week 21	06/03/2017	<u>Presentation Slides</u> Complete any and all amendments to the slides ready for the practice presentation	Presentation Slides Draft 2

		<u>Presentation Practise</u> Practice the presentation and achieve the correct timing and materials (slides) required	Practice Presentation
Week 22	13/03/2017		
Week 23	20/03/2017	<u>Presentation Modification</u> Make necessary modifications to the presentation materials/script	Final Presentation Slides and Script
Week 24	27/03/2017	<u>Project Completion</u> Ensure all work has been completed and product is fully operational	All work Completed.
Week 25	24/04/2017	Prepare Report, Product and Presentation slides for submission	Complete Presentation and Submit Competed Project

Required Resources:

As the project only requires the use of either Java or Python in order to create the working model I will require no additional resources when creating this program. I will only require the use of a notepad variety program (such as Sublime Text or Notepad++) to write the code and the computer terminal to compile the program.

Anthony Kleerekoper