Nigerian Research Journal of Engineering and Environmental Sciences 7(1) 2022 pp. 257-265
p ISSN: 2635-3342; e ISSN: 2635-3350

Nigerian Research Journal of Engineering and Environmental Sciences
Journal homepage: www.rjees.com

Original Research Article

PASS-ME!: A Password Management Web Application with Chrome Extension

[*1]Shuwa, B.A., [2]Wakili, A. and [3]Liman, M.D.

[1]Department of Cyber Security, Faculty of Military Science and Interdisciplinary Studies, Nigerian Defence Academy, PMB 2109, Kaduna, Nigeria.
[2]Department of Computer Science, Faculty of Computing, Federal University Dutse, Dutse, Nigeria.
[3]Department of Computer Science, Faculty of Computing, Federal University Lafia, PMB 146, Lafia, Nigeria.
*bintashuwa@gmail.com

http://doi.org/10.5281/zenodo.6725520

ARTICLE INFORMATION

Article history:
Received 24 Jan, 2022
Revised 09 Apr, 2022
Accepted 14 Apr, 2022
Available online 30 Jun, 2022

Keywords:
Password management
Node js
Express server
Chrome extension
Hashing
Encryption

ABSTRACT

Password based authentication is still the most popular and dominant method of online authentication and has been the main target of malicious attacks. Password misuse due to improper management is the leading cause of data breaches leaving valuable information vulnerable. It is therefore crucial to devise systems that safely and effectively manage user online credentials. The purpose of this research is to build a password management web application with core considerations of security and usability. After survey of similar applications and tools, the system was successfully designed and implemented with express server framework of node js environment adopting standardized cryptographic functions of hashing and encryption to securely manage online credentials while enhancing usability through Google chrome extension. The implemented system was then subjected to functional and User evaluation tests to ascertain its usability and reliability. The test results showed that the system successfully achieved its set out objectives; however, with room for further improvement.

1. INTRODUCTION

Despite their well-established security and usability disadvantages, password-based authentication is still the prevailing method of user authentication online (Ciampa, 2013). In this age of technology, a single user can easily have hundreds of online accounts that requires authentication. On average, a survey by Lastpass in 2017 reported that individuals had to remember 191 different work passwords not to mention those used for personal accounts and social media. (LastPass, 2020).

Passwords are used to protect accounts with valuable assets and information on different websites, and thus, have continuously been the primary target by a plethora of malicious attackers. It becomes therefore paramount to secure online passwords which largely depend on creating unique and strong passwords for each account and protecting such passwords from been stolen or hijacked (Zhao and Yue, 2013). However, research illustrated that such strong password features of been adequately lengthy, random, unique and difficult to guess are more often than not difficult to remember by users causes password fatigue as a result of frustration and tiredness of having to remember multiple, complicated passwords at a time for online login accesses which leads to insecure practices of writing them down or reusing the same password for multiple accounts (Woods and Siponen, 2018). The consequences of password fatigue are the leading causes of website breaches by malicious attackers. A Verizon data breach investigations report of 2017 showed that weak, compromised and reused passwords constitutes about 80% of data breaches (Furnell, 2019). This was also reflected in a report by Google online security survey of 2019 stating more than half of the 3000 respondents reuses the same passwords across multiple accounts (Google.com, 2019). These challenges suggest a dire need for a centralized password management system that effectively and securely manages User logins into their online accounts.

With the stated impending problems of password-based authentications, the idea of this research is to build a password management system that securely manages online credentials. A password manager (PM) is simply a software application designed and implemented to securely store and manage online credentials (Gasti and Rasmussen, 2012). Most prominent PMs are browser-based password managers in which browser integration of the application enables it to save, manage passwords and seamlessly authenticate login accesses for the browser user. All the major browsers (Google chrome, Internet Explorer, Firefox, Safari and Opera) are found to have the password manager feature although with different structures and executions. However, researchers and security enthusiasts have argued about reliability of browser-based PMs pointing out loopholes in the design structure considered as vulnerabilities that could easily be exploited by attackers which could be because password managers are not the focus or priority of such browsers (Zhao and Yue, 2013). It is no wonder therefore that a plethora of third party-based PMs floods the market offering better alternatives that reported to have considered more, the security of saved passwords (Simoens et al., 2012).

For the purpose of this research, four PMs were surveyed and compared with the proposed application. Among the four surveyed, two are browser-based (Chrome and Explorer) while the other two are third-party based (1Password and Keepass). 1Password and Keepass were chosen from their numerous counterparts because they are found to have been among the top highly ranked PMs adopted by both enterprises and Governments (Arias-Cabarcos et al., 2016).

The browser-based PMs did not require a master password mechanism in managing its passwords. Instead, they adopted the data protection API (DPAPI) functions of *CryptoProtect* and *CryptoUnprotect* data to encrypt and decrypt saved passwords respectively. The only difference in executing the function by the two browsers is that Internet Explorer also uses the website URL as an additional entropy while encrypting whereas Google chrome encrypts passwords only. The encryption and decryption operations are not done by the browsers but rather, by the host computer's operating system and as such tied to the host computer's login credentials and user accounts thereby obfuscating the browsers from retaining the cryptographic keys (Zhao and Yue, 2013). Although this meant that a user cannot access his saved passwords with a different user account, which is good for security but lacking for interoperability with alternate browsers, it also implies that all a user needed to provide to have the unencrypted version of their saved passwords is their host device's login credentials. Although both browsers dynamically offer to save your passwords during signup and auto fills them on subsequent logins with minimum user interactions, only Chrome has the password auto generation feature. Both Chrome and Explorer are not found to have used any hashing algorithm for the sake of saving passwords at rest rather, both browsers hash the username and passwords with SHA (secure hashing algorithm) 256 and 512 algorithms respectively while auto filling to be compared with websites during logins (Nash, 2017).

With regards to the third party-based PMs, both applications have distinctively unique features in design structure, usability and security implementations. Whereas keepass works entirely offline and saves locally on your device with options of synching with user's cloud storage space like dropbox, 1Password on the other hand is a web application that manages user's online credentials in their cloud servers. Keepass hashes a User master password with a non-salted hash algorithm; SHA 256 and uses the hash value to encrypt the whole database (usernames and URLs inclusive) with AES (advanced encryption standard) Rijndael and Twofish algorithms. In contrast, 1Password adopts PBKDF2, a salted hashing algorithm to derive encryption keys which are then used alongside secret keys; a generated 34 pseudorandom characters saved locally on user's computer, to encrypt the password vault with AES 256 algorithm (Arias-Cabarcos et al., 2016). In terms of dynamic behavior, both applications require user actions to trigger password generation(sign-up) and autofill features (sign-in). Both applications do not set master password creation conditions, although, they also offer Users to synch with device biometric authentication options. This implies that a User need not to provide their master password anymore subsequently, but instead, use their biometric attributes and face scanning to access their password vaults. The security and privacy implications of such approach especially for highly sensitive system like PMs is highly debatable (Simoens et al., 2012).

PASS-ME! is more like 1Password as it is a web application integrated with a browser extension, uses a salted hashing algorithm, in this case Bcrypt, to derive encryption keys from user's master password and encrypts with the AES 256 algorithm. PASS-ME! also requires user actions to activate dynamic behaviors of password generation and autofill functions. The main distinctive feature with PASS-ME! is that the salted Bcrypt derived key is downloaded and saved locally on User's computer and subsequently, dynamically accessed by the application when its encryption need arises. This is to ensure that the key is not left in the browser or saved on the server to enhance security. PASS-ME! also imposes passphrase condition for master password creation instead of stricter password rules most likely to induce password fatigue. To ensure favorable memorability was achieved, a benchmark application (PASS-ME Benchmark) was implemented with similar features except for the password generation condition. The benchmark application follows the traditional rules of strict password generation whereas PASS-ME! implements a passphrase approach to solve password memorability issues.

PASS-ME! is a web application password management system integrated with a chrome browser extension. The system's main objectives are to achieve the following:

- A web user interface that can be utilized both as standalone and as a chrome browser extension for managing online credentials.
- Adoption of standardized cryptographic functions of hashing and encryption in securing a password vault.
- Setting favorable master password conditions that enhances both usability and security of the password manager.

These objectives were achieved with express server framework of node JS environment and mongo db, a no SQL database. The application is then subjected to appropriate testing and evaluations to ensure requirements were fulfilled.

2. METHODOLOGY

2.1. Web Application Frameworks

The whole system was developed on the express server of node js web application framework. It is an event-driven, asynchronous, single threaded environment designed to build scalable applications. It is a core JavaScript library that supports model-view-controller (MVC) architecture for seamless web development (Mardan, 2018). With express server, minimal coding is required as it is embedded with all basic building blocks and tools required to set a server running as well as its maintenance. Bootstrap, a front-end framework for JavaScript, HTML and CSS with aesthetically standardized and reusable view features which are easily

customized was adopted as the front end technology built on the express server. Mongo DB, a no SQL database which supports the JSON (JavaScript Object Notation) format in addition to been highly scalable, fast, flexible and does not require unified data structure formats was adopted for this study.

2.2. Hashing

In the password management world, it is a consensus that encryption alone is not enough to secure passwords or make it as hard as possible for brute force attacks (Sriramya and Karthika, 2015). This is because of the obvious reasons of reversibility which implies there is always a tendency a key might be intercepted, brute-forced, guessed or side-channeled. As hashing is a one-way function, it provides a better resistant option although researchers still argue collision, dictionary and rainbow table attack possibilities associated with conventional hashing like message digest (MD) and secure hashing algorithm (SHA) algorithms (Sriramya and Karthika, 2015). For these surveyed reasons, this research adopted salted hashing techniques which ensures the uniqueness of every hash value, as it is salted with a pseudorandom number for each hash round. This implies that exact same passwords will generate different hash values when hashed with a salt. Of the three salted hashing algorithms (scrypt, PBKDF2 and bcrypt) surveyed, bcrypt was adopted for this research. This is because bcrypt was found to have a scalable iteration count, more GPU-resilient due to its design structure of constantly accessing the memory, thereby rendering it more expensive to crack and, has the more standardized output encoding (Ertaul et al., 2016). Complemented by the salted hashing, a strong, NIST standardized encryption algorithm was adopted to encrypt the password vault. The AES 256 bits encryption is the go-to algorithm for secure data encryption. AES is a symmetric key encryption service that uses a single key to encrypt and decrypt information. The AES-256 has a length of 256 bits which is the longest bit size and practically unbreakable with brute force attack and is considered the strongest encryption standard. With AES-256 algorithm, an attacker will have to try a combination of 1.1 x10^77 in other to ensure the right key is included (Singh and Supriya, 2013).

2.3. Browser Extension

When it comes to the choice of browser to integrate with the application extension, Google chrome was the ultimate choice. Aside chrome been the most popular browser extension (Vaughan-Nichols, 2020), Safari and Explorer (now Edge) have very strict and controlled delivery system for extensions like reviewing developer codes before granting access (Ursell and Hayajneh, 2019) whereas Firefox does not set such permissions at all or even verify signatures. Google chrome allows developers to set extension permissions and enhance browsing safety through content security policies (Hoffman, 2013). Also, Google accounts are utilized to sign in to the application by Oauth verification, as such, adopting chrome to integrate the extension further becomes the obvious choice to facilitate flexible and secure User Logins.

PASS-ME! was developed using various installed node js packages that performs logic behind all functions executed by the application as seen in code snippet below.

```
const express = require('express')
var router = express.Router();
const googleStrategy = require('../googleStrategy');
const mongoose = require('mongoose');
const Password = mongoose.model('Password');
const User = require('../models/user.model');
var nodemailer = require('nodemailer');
const { response } = require('express');
const bcrypt = require('bcrypt');
const fs = require('fs');
const download = require('download');
const fileUpload = require('express-fileupload');
const CryptoJS = require("crypto-js");
const http = require('http');
```

A high level functional explanation on what the packages achieve in the system is numbered below:

1. First time User signs in with Google account using Google Oauth20 verification
2. First time Users create master password in form of a passphrase
3. System hashes master password with a salt using bcrypt function and saves result in a text file
4. User downloads hashed password value text file locally
5. User uploads saved hash value file to application; this is to help system save the file path for dynamic automatic next time access.
6. User integrates application as a chrome browser extension
7. User can now create, edit or delete web passwords in application interface or system generates strong randomized passwords for User when extension is clicked
8. System encrypts created or generated passwords with AES 256 bits algorithm using the locally accessible saved hash value as key before saving them.
9. Users click on extension to trigger autofill or copy saved passwords for subsequent website logins

2.4. Testing and Evaluation

The software was test run by the authors on several occasions during development. Every function implemented in the software was subjected to tests to ensure they achieved desired outcomes specified in the requirements. Test cases where set and executed for each function of google Oauth login, master password creation and hashing, master password download and upload, creating, editing and deleting passwords, generation of random passwords, encryption and decryption of saved passwords, and extension features. To further test the functional requirements and ascertain how memorable the master password experience is, a user evaluation was carried out on both PASS-ME! and the benchmark applications. This was to conform with core research motivation of building a secure and usable password manager with a memorable master password. The feedback from participants helped the authors greatly in recognizing the achievements and drawbacks of the application as utilized by Users.

A total of 16 between subject participants were asked to conduct the experiment with 8 for each independent variable conditions of PASS-ME! and PASS-ME Benchmark among which 12 of the participants were fellow students from computing science equally divided for each condition. The remaining 4 participants were friends such as civil servants, that frequently use computers for their daily activities. They were also equally divided for each condition. Participants age fell between 24 and 37 among which 11 were males and 5 females. Each participant was sent a folder containing the extension file, a 2 min video, a questionnaire and a generated local-tunnel global link to access the application. The video contained a screenshot with a voice over that illustrates how to load and unpack the extension and start the application. Participants utilized the application for ten days and were later asked to fill in the attached questionnaire. The questionnaire contains a standard system usability scale (SUS) questions coupled separately with two specific memorability assessment questions.

3. RESULTS AND DISCUSSION

The discussion of evaluation results is done in two forms; the system usability scale feedback from participants and the memorability scale feedback. This is done separately because although SUS is a standardized system usability technique, it is more general and does not capture specific memorability markers.

3.1. System Usability Scale

All 16 Participants filled in the SUS questionnaire after testing the application for 10 days. The SUS offers a reasonably precise usability score to the application according to agreed standards (Lewis, 2018). Tables 1 and 2 show the responses from participants for both applications; PASS-ME! and PASS-ME! Benchmark.

Table 1: PASS-ME! SUS feedback

Participants	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS raw score	SUS final score
1	4	2	4	1	4	2	4	1	4	2	32	80
2	5	2	5	1	4	2	4	2	3	2	32	80
3	4	2	4	2	3	2	4	2	3	2	28	70
4	2	3	4	1	4	2	2	1	2	2	25	62.5
5	3	2	4	2	4	2	4	1	4	2	30	75
6	4	2	4	2	4	2	4	2	4	2	30	75
7	4	2	4	1	4	2	4	1	4	2	32	80
8	5	1	5	1	5	1	4	1	4	1	38	95
Average												87.5

Table 2: PASS-ME! benchmark SUS feedback

Participants	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS raw score	SUS final score
1	4	4	4	1	4	1	4	2	4	1	31	77.5
2	2	5	3	2	4	2	4	2	4	2	24	60
3	3	2	4	2	4	2	2	1	3	1	28	70
4	3	3	4	2	3	2	3	2	2	2	24	60
5	4	5	4	2	4	2	4	1	3	2	27	67.5
6	2	3	4	2	4	1	2	1	3	1	27	67.5
7	2	5	4	1	3	2	2	2	4	2	23	57.5
8	2	4	3	2	4	2	4	1	4	1	27	67.5
Average												72.5

The scored gathered from Tables 1 and 2 were converted into percentile ranks to interpret the results using the SUS standardized manner adopted from Equations (1) and (2). The SUS score for each participant from the standard was computed in such a way that for every odd numbered question, 1 is subtracted from the score and from every even numbered question, the score is subtracted from 5 as follows (Grier et al, 2013).

$$\text{SUS raw score} = (Q1-1)+(5-Q2)+(Q3-1)+(5-Q4)+(Q5-1)+(5-Q6)+(Q7-1)+(5-Q8)+(Q9-1)+(5-Q10) \quad (1)$$

$$\text{SUS final score} = \text{SUS raw score} * 2.5 \quad (2)$$

Where Q is the Likert scale score responses from a participant for each question (Grier et al, 2013).

Finally, the average of the SUS final scores for all participants was determined. This average should normally be a number from 1-100. The averagely good percentile score for SUS is 68, this implies that scores less than 68 indicates serious usability issues in an application (Lewis, 2018).

As seen from the Table 3, the average percentile obtained (87.5) was well above 68 which portrays how usable the application is according to participants. This is in contrast from the benchmark's respondents that has an average percentile of 72.5 which was just above the required average of 68. Going through the responses, the low scores can be attributed to SUS question 1 and 2 which asks about intention of frequent use and whether the application is unnecessarily complex. As the only difference between the two applications are the master key creation conditions, the distinctive results interpreted that a simpler, memorable password goes a long way in improving system's usability. Although the confidence marker scores obtained from questionnaire for the PASSME! Application was low, of which according to participants is because they have reservations with the copy password function integrated in the extension which renders the application more usable, could also be a vulnerable loophole to having access to one's credentials, also coupled with the non-recovery of master passwords. Overall, PASS-ME! can be considered as usable, easily adaptable, consistent and well-integrated.

3.2. Master Password Memorability Scale

The SUS questionnaire was followed by two simple questions that specifically addressed the memorability issues with the master password used in the application. The two questions captured whether a user needed to write down their passwords to remember and if its complexity/length serves as a necessary nuisance that might hinder their use of the application. Table 4 shows the responses to the two memorability questions by participants.

Table 3: Memorability scale response across all participants

	PASS-ME!		PASS-ME! Benchmark	
	YES	NO	YES	NO
Q1	7 (87.5%)	1 (12.5%)	8 (100%)	0 (0%)
Q2	2 (25%)	6 (75%)	2 (25%)	6 (75%)

From the responses, it can be deduced that most participants testing PASS-ME! did not write the passphrase down to remember, and the few ones that did added that they are just been extra safe as there is no password recovery mechanism in the application. This is in contrast with the benchmark testers' responses in which 100% of participants admitted to writing down their passwords in other to remember. The second question has a more positive response across all participants, this is largely because users are seldom prompted to provide their master passwords unless they needed to see the plain text versions of their saved passwords as all other functionalities are executed by default.

Consequently, it is shown through the user experiments that the implemented application has a good usability scale and that master password's memorability plays an important role in implementing a usable password manager which also strengthens its security. This is because a memorable password helps alleviate password fatigue practices that causes serious vulnerability issues to the application. Although the passphrase creation condition of PASS-ME! is lengthy (20-40 characters), it is a more memorable password with increased entropy which increases password best practices and decreases the guessability of a password thereby enhancing both usability and security.

4. CONCLUSION

The research intended to create an interactive web interface with core minimal functions of online User credential's secure management. After survey of similar applications, PASS-ME! was implemented with express server framework of node js environment integrated with standardized cryptographic functions of hashing and encryption. The main feature of PASS-ME! that stood out is the passphrase-based master password creation for a memorable password experience and the downloadable hashed value of the master password on User's local device to obscure the server and database from having access to it. These measures arguably enhance the usability and security of the application. The finished application was then subjected to functional testing to ensure it successfully achieves required functions. Also, User evaluation of the application was conducted to ascertain master password memorability and overall usability of the system. For the evaluation, a benchmark application was mocked to test the master password's memorability alongside a standardized System Usability Scale for overall system function. The responses from the evaluation shows that the implemented application is usable, password memorable, and efficient system. However, there is room for improvement to further enhance both usability and security of the system. These improvements are mostly deduced from the non-implemented features as prioritized by requirements gathered.

5. CONFLICT OF INTEREST

There is no conflict of interest associated with this work.

REFERENCES

Arias-Cabarcos, P., Marin, A., Palacios, D., Almenarez, F. and Diaz-Sanchez, D. (2016). Comparing Password Management Software: Toward Usable and Secure Enterprise Authentication. *IT Professional*, 18(5), pp. 34-40.

Ciampa, M. (2013). A Comparison of User Preferences for Browser Password Managers. *Journal of Applied Security Research*, 8(4), pp. 455-466.

Ertaul, L., Kaur, M. and Gudise, V. (2016). Implementation and performance analysis of PBKDF2, Bcrypt, Scrypt algorithms. *International Conference on Wireless Networks* (ICWN), CA, USA, pp. 66–72.

Furnell, S. (2019). Password meters: inaccurate advice offered inconsistently?. *Computer Fraud & Security*, 2019(11), pp. 6-14.

Gasti, P. and Rasmussen, K. (2012). On the Security of Password Manager Database Formats. *Computer Security – ESORICS 2012*, pp. 770-787.

Google.com. 2019. *Online Security Survey*. [online] Available at: <https://services.google.com/fh/files/blogs/google_security_infographic.pdf> [Accessed 13 August 2020].

Grier, R. A., Bangor, A., Kortum, P. and Peres, S. C. (2013). The System Usability Scale: Beyond Standard Usability Testing. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* 57(1), pp. 187–191.

Hoffman, C. (2013). *Beginner Geek: Everything You Need To Know About Browser Extensions*. [online] How-To Geek. Available at: <https://www.howtogeek.com/169080/beginner-geek-everything-you-need-to-know-about-browser-extensions/> [Accessed 21 August 2020].

Lastpass. (2020). *The Password Expose*. [online] Available at: <https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LastPass-Enterprise-The-Password-Expose-Ebook-v2.pdf> [Accessed 15 July 2020].

Lewis, J. (2018). The System Usability Scale: Past, Present, and Future. *International Journal of Human–Computer Interaction*, 34(7), pp. 577-590.

Mardan, A. (2018). *Full Stack JavaScript*. Berkeley, CA: Apress.

Nash, M. (2017). *Mobile & web browser credential management: Security implications, attack cases & mitigation.*. [online] Nccgroup.trust. Available at: <https://www.nccgroup.trust/globalassets/our-research/uk/whitepapers/2017/mobile-and-web-browser-credential-management-security-implications-attack-cases-and-mitigations.pdf> [Accessed 10 July 2020].

Simoens, K., Bringer, J., Chabanne, H. and Seys, S. (2012). A Framework for Analyzing Template Security and Privacy in Biometric Authentication Systems. *IEEE Transactions on Information Forensics and Security*, 7(2), pp. 833-841.

Singh, G. and Supriya, S. (2013). A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security. *International Journal of Computer Applications*, 67(19), pp. 33-38.

Sriramya, P. and Karthika, R. A. (2015). Providing Password Security By Salted Password Hashing Using Bcrypt Algorithm. *ARPN Journal of Engineering and Applied Sciences*, 10(13), pp. 5551-5556.

Ursell, S. and Hayajneh, T. (2019). Desktop Browser Extension Security and Privacy Issues. *Lecture Notes in Networks and Systems*, pp.868-880.

Vaughan-Nichols, S. (2020). *What's the most popular web browser in 2020? | ZDNet*. [online] ZDNet. Available at: <https://www.zdnet.com/article/whats-the-most-popular-web-browser-in-2020> [Accessed 20 June 2020].

Woods, N. and Siponen, M. (2018). Too many passwords? How understanding our memory can increase password memorability. *International Journal of Human-Computer Studies*, 111, pp.36-48.

Zhao R. and Yue C. (2013). All your browser-saved passwords could belong to us. *Proceedings of the third ACM conference on Data and application security and privacy – CODASPY,* pp. 33-40.

Appendix: SUS and password memorability questionnaire.

SUS	Strongly Disagree (1)	Disagree (2)	Neutral (3)	Agree (4)	Strongly Agree (5)
1. I think that I would like to use the password manager frequently	☐	☐	☐	☐	☐
2. I found the password manager unnecessarily complex	☐	☐	☐	☐	☐
3. I thought the password manager was easy.	☐	☐	☐	☐	☐
4. I think that I would need the support of a technical person to be able to use this password manager.	☐	☐	☐	☐	☐
5. I found the various functions in the password manager were well integrated.	☐	☐	☐	☐	☐
6. I thought there was too much inconsistency in this password manager.	☐	☐	☐	☐	☐
7. I would imagine that most people would learn to use this password manager very quickly.	☐	☐	☐	☐	☐
8. I found the password manager very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9. I felt very confident using the password manager	☐	☐	☐	☐	☐
10. I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐
MEMORABILTY MARKERS (ANSWER YES OR NO)					
1. Do you need to write down your master password in other to remember it?					
2. Is your master password complexity/lenght a hindrance to you while using the application?					