Password Management Application

Thomas Hood

Student Number – 2847586

thomashood10@protonmail.com

Submitted in partial fulfillment for the degree of

B.ScH in Computing Science

Department of Computing Science, Griffith College Dublin

April, 2017

Under the supervision of Osama Abushama

**Disclaimer**

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Bachelor of Science in Computing at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

**Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Acknowledgements

I’d like to thank my mother and father for supporting me for the past four years of this degree and for putting up with me during that time. It’s been a stressful time for my family and they’ve always been there for me every step of the way.

A sincere thanks to my friends Julie-Anne Glennon, Alex Vodnev, Rostislav Panin, Anas Djebbari, Cesar L Velasco, Haider, Cintia Silva, Gareth Roper, Colm Corkery, Ross Cosgrave, Shane Downey, Alan Jones, Serghey Nita, Alex Radic, James Banna, Judy Ehiguese, John Lally, Eimear Brennan and everyone in the GCD Computer Science class of 2017 who has helped me over the years. I believe everyone in our year will go on to great things and will sincerely make this college proud of their students.

To all my lecturers and year heads, sincerely thank you for your patience and understanding. You do a thankless and vastly under-appreciated job which you should be given far more credit for doing. Like most things in life, things can be rocky and uncertain for people at times, but you consistently always communicated with us and kept striving for us to learn more. Many people would see college as an expensive test, but you have made it so much more than that with your attitude to teaching, your understanding, and care for the students under you. Sincerely thank you.

Table of Contents

[**Acknowledgements** iii](#_Toc479550706)

[**Abstract** vi](#_Toc479550707)

[**Chapter 1. Introduction** 1](#_Toc479550708)

[1.1 Conventional Password Systems in Web-Browsers 1](#_Toc479550709)

[1.2 Goals 2](#_Toc479550710)

[1.3 Approach 3](#_Toc479550711)

[1.4 Document Structure 3](#_Toc479550712)

[**Chapter 2. Background** 4](#_Toc479550713)

[2.1 Related Work 7](#_Toc479550716)

[**Chapter 3. Methodology** 8](#_Toc479550717)

[**Chapter 4. System Design and Specifications** 13](#_Toc479550717)

[**Chapter 5. Testing and Evaluation** 16](#_Toc479550718)

[**Chapter 6. Conclusions and Future Work** 18](#_Toc479550719)

[**References** 21](#_Toc479550721)

# Abstract

Passwords are security mechanisms that were traditionally committed to memory or potentially some form of physical document. It is by far the most common mechanism for maintaining secure access to an asset.

Over time with the rise in popularity of software and modern internet usage; the demand for password systems and improved security has also risen dramatically.

Storing passwords for convenience has always presented a security issue to varying degrees.[1] Mostly because any encryption implementation requires that the decryption method be immediately available to either the user or the software, thus rendering encryption an improvement in security but only through obfuscation of the credentials.

Software vendors and web-browser developers take different approaches. Microsoft now have a built-in credentials vault that manages all username and password combinations for not only Internet Explorer but almost all Microsoft applications and operating system security. Google use only a very basic encryption to obfuscate user credentials for their web-browser, and Mozilla use a complex system involving 3DES and numerous salts.[2]

Recently password managers have become quite popular as a way of improving the general level of security that currently isn’t readily available in most web-browsers and operating systems.

Here are some popular applications that are recently in use,

<http://lifehacker.com/5529133/five-best-password-managers>

The intention of this project is to develop an application that takes an alternative approach to password storage through utilizing MySQL.

# Chapter 1. Introduction

## 1.1 Conventional Password Systems in Web-Browsers

Majority of modern computing systems prioritize convenience of the user over security. Typically, operating systems are required to store the password locally for the purpose of the user logging into the system. There are already numerous existing solutions that allow users to remote login to a working computer system (e.g. SSH, RDP) so for the purpose of this project we will focus on the three most popular web-browsers in use today.

Google-Chrome

Google-Chrome has achieved a substantial amount of popularity since it was first publicly released Google has taken a stance of strongly discouraging the storage of passwords locally to aid a browser. They controversially decided to use almost no encryption for local password storage in order to discourage users from storing passwords locally.[3]

Mozilla Firefox

Firefox became very popular as an alternative to Internet Explorer and has enjoyed the benefits of being an open-source platform. Mozilla took a very different approach from Google in that they implemented very strong encryption for local browser password storage.[4]

Internet Explorer

Microsoft originally stored passwords for Internet Explorer in a credential file but more recently they’ve implemented their own form of password manager under the ‘Credential Manager’ from Windows 8 onwards.

## 1.2 Goals

The overall goal of this application will be to allow the user to manage their credentials for multiple browsers. By allowing the user to move locally stored passwords to a remote database, this will alleviate some of the risk of storing passwords on their local machine.



Ideally the application will allow the user to view his locally stored credentials, and transfer them to a remote database. If the user so chooses then they can remove their locally stored passwords.

The main concept we are working toward here is decentralized storage of password credentials, an alternative to storing them locally.

## Approach

Since we are creating a software application as the primary goal of this project, it follows that we need to select a programming language with which to build our application. For the purpose of this project we will be developing on a Microsoft operating system platform to implement our proof-of-concept application.

## Document Structure

This document will largely follow the template used by the college for project documentation. In essence it is the standard scientific ‘IMRAD’ format with a few modifications made to it.

<https://en.wikipedia.org/wiki/IMRAD>

Naturally with a project that requires substantial amounts of research and planning it is usually a good idea to break down methodology into a background chapter for research and discovery. While it is better to leave a methodology chapter for actual practical development of the project.

Since we are dealing with a software project and not a research project, there will also be a ‘System Design and Specifications’ chapter to show a use-case diagram for the project, and minimum specifications for the application.

There will also be a ‘Testing and Evaluation’ chapter that will detail some of the software unit testing and error detection built into the software.

The ‘Implementation’ chapter will cover the actual practical operation of the software application and detail the GUI as well as any other user functionality of the program.

# Chapter 2. Background

Password management applications are a relatively new field. It has only significantly taken off recently in the past 5-7 years due to the volume of credentials the average person has to remember on a somewhat regular basis. There are also other legitimate security concerns given that most web-browsers and other applications that store credentials typically do not implement a robust encryption system or security solution.

The first thing we need to understand is, why is modern credential security so low when there are readily available strong encryption methods out there that have been in use for a long time?

The major reason why little to no encryption takes place on the local machine for most web-browsers is that the credentials would need to be unencrypted to actually use them. As such almost any level of encryption can often be subverted using the web-browser itself or the operating system. Indeed, with Google-Chrome, Firefox, or the Credential Manager in Windows you can see your saved passwords in plain-text.

To start with let’s investigate into exactly how credentials are stored on a system.

Google-Chrome

Chrome uses the standard Windows API function call ‘CryptProtectData’ to provide a very basic level of encryption for credentials that are stored for the browser.

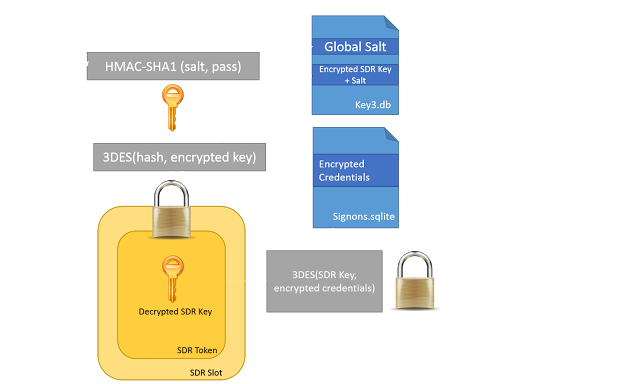
They are stored in the Windows application data folder for the local machine a SQLiteDB file. It is simply a matter of decrypting the password blob encrypted with the Windows API function to reveal the passwords.

*https://msdn.microsoft.com/en-us/library/windows/desktop/aa380261(v=vs.85).aspx*

Firefox

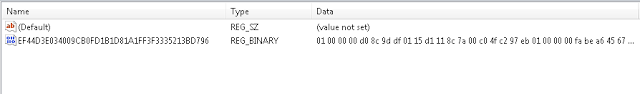
Firefox uses an open-source set of libraries called “Network Security Services” or NSS to provide developers with a framework to create security solutions for their own applications.

Firefox primarily uses what is known as a Secret Decoder Ring or SDR, a random salt, and the 3DES algorithm to encrypt all usernames and passwords. These are somewhat integrated into a master password too. The workings of these are very complex and as such the following diagram will give you an idea of how they interact with each other.



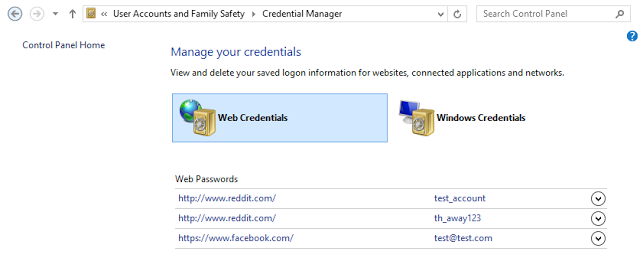
Internet Explorer / Microsoft Edge

## Microsoft have changed how they store credentials with most Internet Explorer and operating system changes. For Internet Explorer 7-9 credentials for form-based authentication were stored in the operating system registry as binary values. Other credentials and network logins were stored in a credentials file.



All credentials in this case were stored using the Windows API ‘CryptProtectData’ function. However, with the new Windows operating systems there is now a built-in credential manager that comes as part of the operating system.

Microsoft’s Credential Manager offers a very complete solution for recovering and storing credentials, but there is no option to remotely store credentials.



## Now that we’ve gone through the arguably three most popular browsers and how they store credentials, we have a good idea of where to begin looking for collect our data from as well as how this information is stored.

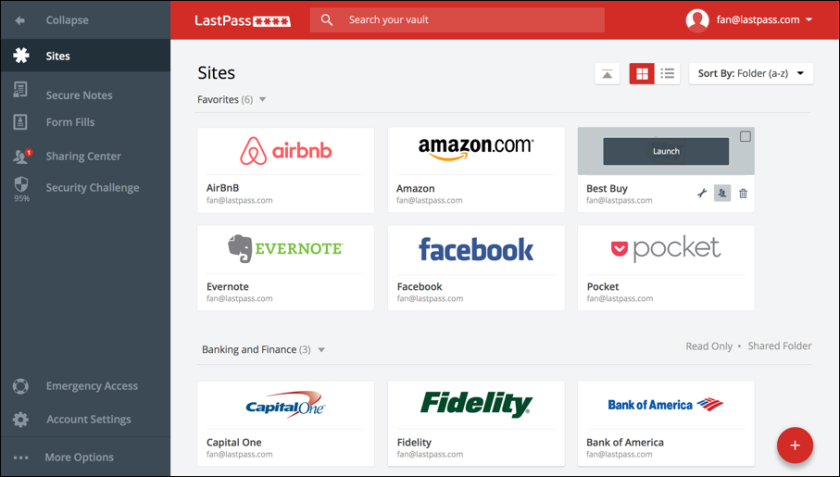
At this point I really began diving into the minutia of the technologies being used here with regards to manipulating them in a programming language. That will be covered extensively in methodology. Most of the background research I’ve done has been covered here in relation to what general information I gathered before I started working on the project.

Most of what I’ve discussed beyond known algorithms like 3DES is closed source except for Mozilla’s NSS framework, which is quite in-depth. However, I will explain in the methodology what parts of it we needed to utilize in our project.

## 2.1 Related Work

There are quite a few existing applications that are ‘Credential Managers’ of some kind, but the one that comes the closest to what this project has tried to achieve as well as arguably being the most prolific is an application called ‘LastPass’.





LastPass allows the user to store their credentials in an online account. Other functionality allows it to synchronize between web-browsers and integrate with live web-forms on a page.

While the application we are trying to make in this project differs substantiality it shows there is dedicated development being done for credential managers. There are many similar applications to LastPass but it certainly has a large amount of development and production value invested into it.

# Chapter 3. Methodology

Part of the specifications for this project was that it be a practical software application that could be used either by a technician, programmer, or everyday user. While background research was indeed encouraged, the primary goal of this project is to build a software application relating to what we have learned over the course of our computing degree.

As with any software project, regardless of the scale and scope of the project, the first step is selecting the tools with which to build our application. I think it’s fair to assume that it is not necessary to cover the finer points of programming languages in this document however I will outline the pros and cons of the languages I’ve used and considered in a general manner. It’s worth noting now that I decided to go for Java and Python based on their existing libraries and suitability for the task at hand over anything else.



Java

Java has very good portability and Java applications will run on almost any platform with very little setup or unpackaging if any required. It is a natural choice in this case as it now supports the Windows API calls we need as well has having extensive existing libraries.

 Python

Python is an open-source based interpreted language. Since Python uses an interpreter instead of a compiler this marries it nicely to Java in the sense that portability will not be overly affected as long as we bundle a relatively portable Python deployment with our application. While Python is not usually readily available on most systems, it is quite easy to package it into a Java application with relative ease. Like Java there are extensive libraries to use and it is a very popular programming language for a variety of purposes.

****C / C++ / C#

While C is a very powerful language it doesn’t really suit what we want to achieve here. While we are dealing with somewhat lower-level system function calls, the higher-level languages we’ve selected are quite capable of dealing with what we require from them. As such there’s very little motivation for us to use C although I did briefly attempt to write a few of the simpler programming functions that are required like moving files around and in the end it was far easier to simply let Java handle that. There was no need to dive further into any kind of file encoding or otherwise deeper system operations.

Other Languages

There are many new JavaScript based and higher level languages coming out all the time. It’s reasonable if we had some kind of specialist need that the above languages did not provide a means of meeting that I might consider them but in this case there was none to be found. Most of our degree we have worked with the above three languages to an extensive amount, and venturing into learning a new language could provide a lot of difficulty.

The Platform

At this point I had decided on the programming language, what I was hoping to achieve and roughly how I was going to do it. The only other major high-level decision was what platform to do it for. Windows is an obvious choice but there was also Linux and other operating systems like OSX to consider.

While it could be said that other platforms would of course benefit from the application, the main reason for sticking to Windows solely is because developing the application for the other platform is not necessary to the completion of the project nor required. Windows can also somewhat reliably provide access to all the tools I need in this instance.

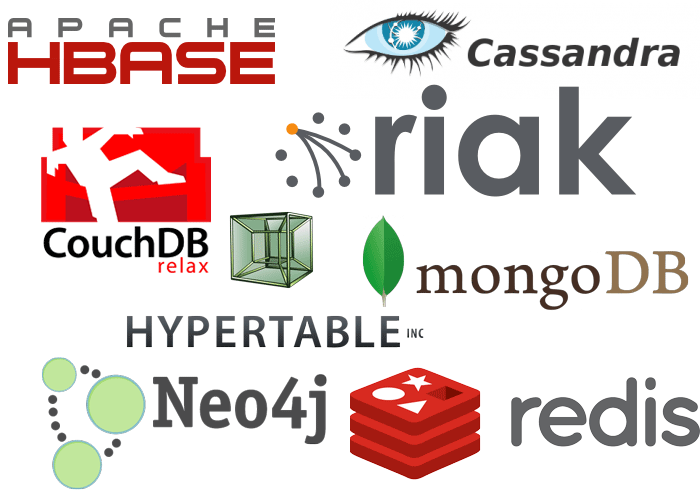
External Technologies

Part of our goals mentioned in the introduction is the storage of credentials in a local or remote database. The purpose being to remove vulnerable credentials on your local machine and put them in a more secure location.

The optimal technology for sending and retrieving data is somewhat obviously, a database. However, there are many to choose from.

MySQL

MySQL is a well-known relational database management system. It was released in 1995 and has been incredibly popular for a variety of uses since then and is still commonly used by major websites and other companies wanting to store data. MySQL offers us a simple and easy to use solution.



NoSQL

There are several specialist alternatives to MySQL that are primarily for document based database storage, big data, or other specialist needs. As such we do not have any extraordinary needs beyond normal data storage.

Naturally with this degree we have a good level of experience with MySQL and it fulfilled my needs for doing this project with ease in terms of flexibility and performance.

There would need to be substantial global usage of a server to warrant using anything other than MySQL in this case. The amount of data that is required per-user to be stored is incredibly small. As such it was reasonable to settle on MySQL barring any extraordinary circumstances at a later point in the project.

Methodology and Timeline

Initially I had started experimenting with a password recovery tool of some kind over the summer of 2016. I began to keep a diary of my progress, right up to the end of the project. The development of the application was very much a progress of working from the bottom-up. I started with experimentation for mostly fun and exploration purposes, then progressed into turning it into an application.

For this section, we’ll examine each diary entry and I’ll elaborate on what methods I was using to achieve what I was doing at the time.

## 19/09/2016

Status:

jar file will decrypt Chrome passwords and output to text file in working directory



During the previous semester of our degree we learned a substantial amount in regard to information security and encryption. As part of my research into encryption I had started reading more and more about password encryption and encryption’s practical uses in modern operating systems. September of last year I had begun experimenting a bit with web-browser password decryption for fun as well as reading up about password manager applications in the industry.

The final function that handles recovery of Google-Chrome credentials has not changed too much since I wrote it as you can see above. Chrome stores the credentials in an SQLite database which of course Java has a very nice and easy-to-use library for. Once I was able to open the database in Java it was simply a matter of using the Windows API ‘cryptUnprotectData’ to decrypt the password blobs into plain-text. From there it’s simply a case of moving the data from one place to another.

## 05/10/2016

Status:

Progress with Firefox is slow, NSS is a complete nightmare to use in Java. Struggling to initialize it

and get it up and running.

Ended up using nss3.dll which was in the 64-bit Firefox install thankfully.

Still struggling to set the config file for the Java NSS library,

At this point it would just be easier to simply use a Python script and simply execute it in Java.

Goal: Get NSS initialized

At this point I was confident enough to proceed to the next browser which was Firefox. Mozilla use an open-source security library called NSS which is a very complex token based system. Firefox uses PKCS#11 which is one of many public-key cryptography standards. The nss3.dll is the NSS library file that ultimately, I was unable to get access to in a reliable way.

## 21/11/2016

Status:

Switching to Jython to execute Python script to decrypt Firefox username and passwords.

As far as I can tell Mozilla do not have NSS dlls readily available for my platform so I'm

just going to cut my losses and use one of the many Python scripts that are out there. Going

to begin work on it tomorrow. Briefly attempted to migrate to C++ with little success, typical of

C++ the code has become bloated already and takes probably 20 lines to do what Java or Python can do in 3.

At this point I was not having much luck with getting NSS working with Java and I had begun looking towards Python scripts for what I needed to do. I briefly attempted to migrate the code to C++ but ultimately found it to be very challenging and in the end quite unnecessary.

## 12/12/2016

Status:

Using https://github.com/Unode/firefox\_decrypt to decrypt Firefox passwords. Made executable using py2exe,

and I'm simply going to execute that and the store the results by redirecting the output to a Java structure.

In the end, I found a very good python script that was available under the GNU Public License, with some minor changes to the script I was able to get it working the way I wanted it to. Effectively this required that I bundle the application with a python deployment. It greatly increased the overall size of the application in terms of size in MB but it worked flawlessly. It uses the PK11SDR\_Decrypt function in the NSS library to decrypt the Firefox credentials and push the results to the output. Naturally I had to program around the output and collect it into Java.

## 22/02/2017

Status:

Going to drop support for Internet Explorer, Microsoft actually have a built-in tool in Windows for recovering credentials from

the vault which makes doing password recovery for Windows credentials a little redundant.

At this point I'm going to refine existing code further and begin documentation.

Around December there was a break for exams and Christmas, after which I began working towards getting Internet Explorer and Microsoft Edge credentials decrypted and into Java. However, I learned later in February that Microsoft already had a credential manager built into all of their Windows and server platforms, so in the interest of not expending unnecessary energy I opted to drop their browsers and credentials from the application.

Towards the start of February, I had begun preparing this documentation as well as creating a GUI for the application with JavaFX. Mostly this involved assigning application functions to JavaFX modules which I will detail under the design specification in this document.

**Chapter 4. System Design and Specifications**

The overall objective of the application is to make it easy and convenient for the user to transfer his locally stored credentials to a remote database.

The core components of the program are,

* The GUI (JavaFX)
* Credentials Class
* Decryption Functions
* Remote Database Functions
* Output Redirection Functions

The GUI

The graphics user interface is how the user can interact with the application. It has five buttons, each of which execute functions of the application. Another purpose of the GUI is to convey information to the user. In the context of the application it needs to show local credentials that the user is storing as well as credentials the user has stored on the remote database. This is conveyed through JavaFX’s ‘TableView’ functionality and classes. There are also a number of labels to direct the user, as well as three text-fields to allow the user to input credentials they want to store remotely.

The Credential Class

The purpose of this class is to store credentials retrieved from the local machine in their own Java class, this allows other functions within the Java application to easily access credentials without having to do too much iterative style programming or using unnecessary loops and counting.

The class also has getter and setters for the ‘TableView’ modules, which are required to assign the data to cell values in the tables.

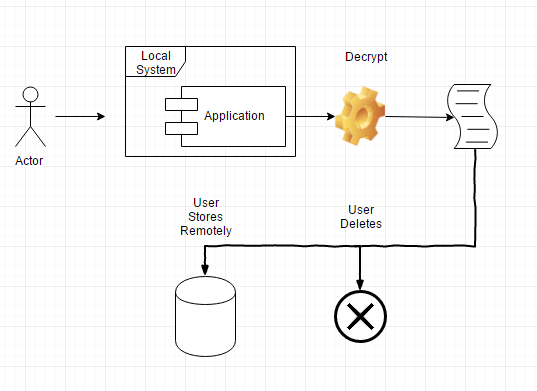
Decryption Functions

This class will contain all functions necessary to decrypt any local credentials and then take them from an sqlite database or JSON format and put them into a Java datatype. Part of these functions may need to capture output from the system and move it to a datatype too.

Remote Database Functions

This class will contain all functions for sending credentials to the remote database and also all functions for retrieving the credentials from the remote database. These are effectively SQL queries that are passed from the Java connection driver to the remote MySQL database.

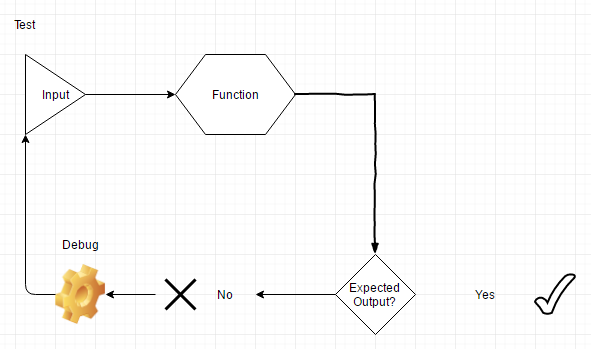
For proof of concept, I’ve used very basic schemas, tables, and configurations for the MySQL server, however the specifications of these functions would need to change to incorporate any additional database types or expected outputs from a given database.



# Chapter 5. Testing and Evaluation

Testing for the project was largely results based testing. Each function had an expected input and output as is the case with all programming functions.

Most of the error detection and testing is built into the program itself, there is a status label to alert the user to any potential problems or issues with the application. From a development point of view, the few bugs the project ran into were resolved with standard breakpoint debugging methods that have been used for years now.



The above diagram is a general workflow for how I approached each function. Once the function was written to the specifications I needed, I would then test it with some input and then check the output. If the output data was as expected, then I would move on and try another test input to ensure all kinds of inputs were tested and checked. If the output was not as expected I would then return to the function and debug it to determine why the output was not as expected. Once debugging was complete the same test input was used again and tested. I repeated the process with all functions until they were working correctly and producing expected output.

At one point, I did ask some friends to test the application to see if it was intuitive to use. I asked a handful of close friends to see if they could figure out how to use the basic functions of the application. All of them had no problem interacting with the tables and buttons of the application and understanding what the application was doing. Explaining the objective of the application was relatively well understood once they understood that credentials stored on their system were not very secure. For most people, they were quite surprised to learn that the credentials stored on their machine were not better protected in some fashion.

I did not get a chance to really pitch the idea for small to medium size businesses with public facing systems like say an internet café. As such I was not able to conduct any testing on a public system to see if there was any natural adoption of the application although I highly doubt there would be given general public awareness around information security issues.

The application is highly bespoke, so any future additions to the codebase or the functions within it would need to be tested using the model above. The technologies are all integrated at this point to an acceptable level, but that could change depending on any new additions to the codebase.

The project testing took about a period of two months or so and by the end I was satisfied that the core functionality of the application was working correctly. Some minor bugs still remain with removing the local credentials from a system but they can usually be resolved by restarting the application. I believe the issue stems from a file access conflict between the application and either main memory or the web-browsers themselves. Further testing will be required to pin down the nature of the issue, but at this point the user can work around it by either restarting the application or deleting the credentials from their browser.

# Chapter 6. Conclusions and Future Work

# My work on the project spanned about half a year in total or slightly over one college semester. The one overwhelming conclusion I came to during my work is that the storage of any credentials on a public facing system is a very real security issue. We’ve reached the point now where the average individual has multiple online accounts that require a username and password to access, making it very difficult for someone to keep track of all the different usernames and passwords they might use. There is of course the option of using the same username and password for all accounts but that would not be a recommended course of action for anyone unless they didn’t care about security whatsoever.

I believe LastPass as an application has the right idea but I think should offer the user an alternative to storing them in an online account. Much like the way an organization can choose to run an email server, I think it’s a relatively good idea to offer a sort of credential server application too.

There should probably also be some sort of warning by the major browser distributors. In all cases the user is simply given the option to ‘Save the Password’ or simply click ‘No Thanks’. I think a warning of some kind to the tune of “Saving your Passwords is Not Securing Them”.

As far as expanding on the existing application, the natural progression from this stage is to integrate the application with web-browser credential storage so that credentials can be restored to the local machine when desired and then removed at the end of a browsing session.

Naturally for the GUI a lot of work could be done to make it look a lot more modern and slicker. Pretty much everything is stock JavaFX and standard fonts.

More security improvements could be made in regard to the transfer of credentials. However, using SSL or TLS would probably be the recommended way to go. Using standard encryption algorithms with those protocols has provenly over time been the safest way to transfer credentials.

I’m quite happy with the overall progression of the application. For the given amount of time for the project, the amount of experience I have, and the amount of people working on it. I feel good progress was made to get the application to a proof of concept state from nothing.

It does perhaps come across a relatively simplistic given that it’s simply transferring credentials, but the hardest part by far was decrypting the stored credentials, getting them into Java, and finally capturing the output from Python. The general concept that the mainstream languages have everything you need as well as the internet having every possible resource you might want is quite unfounded in my opinion.

Dealing with Firefox was indeed incredibly difficult as documentation on NSS and the PKCS#11 standard was incredibly sparse and often not kept updated beyond a broken link on a website somewhere.

All things aside, I really enjoyed working on the project and I think it demonstrates a good fundamental understanding of software-engineering, file manipulation, data management, and core programming skills. As a demonstration of programming ability, it perhaps lacks any functional programming or anything significantly mathematical, but it is quite rare for practical applications not involved in physics modelling or intense calculations to call for such styles of programming.

I believe the above could be quite achievable within the space of about 2 years or so if I continue at this pace. I believe a team of about three to four people with good experience in putting together software applications could probably get it done in half that time.

Either way at this point I do intend to continue working on the project after the submission of this documentation and the demonstration. There is definitely a growing demand for better management of user credentials for both web-browsers and other software application credentials.

If I could change anything I probably would have liked to be more clear about my objectives with the project at the beginning, but once I sat down and thought about the objectives of what the application needed to do it became much clearer.

Java and Python both performed remarkably well, the brief migration of the code to C++ really made me appreciate the amount of effort that has gone into both Python and Java to make things much easier for the programmer. At no point during the project did I feel either language was holding me back from achieving what I wanted to do, and it was remarkably straightforward to create a working build of the Java application in comparison to say creating a working build of a C++ project of similar size and scope.

Both Mozilla and Google were naturally great resources for learning more about how credentials are stored on the local machine and I really learned a substantial amount about how that is done over the course of the project.

On a personal note, I really enjoyed it and would strongly encourage anyone to put as much passion into their own project as they possibly can. There are definitely times where what you’re doing may seem pointless or arbitrary but ultimately all software applications are about convenience or doing things in a slightly different way to see if it’s more efficient or easier for people.

Thank you for reading this far if you have, if anything requires clarification don’t hesitate to contact me at the email address listed at the start of the documentation. I am always available if anyone would like to talk about software, programming, or college in general.

# References

I’d like for you to keep in mind while reviewing these resource that there is pretty much zero actual academic material in regard to password security or password management. Almost all data and statistics are provided from independent 3rd parties done for the purposes of research or indeed for product testing. At all times I tried to get the most academic non-biased source possible for all information presented here.

[1] - <https://www.wired.com/2016/08/browser-password-manager-probably-isnt-enough/>

[2] - <http://raidersec.blogspot.ie/2013/06/how-browsers-store-your-passwords-and.html>

[3] - <https://www.engadget.com/2013/08/07/chrome-saved-passwords/>

[4] - https://luxsci.com/blog/master-password-encryption-in-firefox-and-thunderbird.html