UNIVERSITY OF HULL | DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

Researching and Implementing a Secure Password Manager using Multi-Factor Authentication

Final Report

Submitted for the BSc in

Computer Science with Industrial Experience

April 2021

by

Lachlan Craig Gourlay

Word count: 16,401

Abstract

Passwords make up the backbone of modern authentication, therefore the types and the way passwords are used should be taken very seriously. This paper explores the current state of the art in security and authentication. This research will then be used to design, implement and evaluate a functioning proof of concept secure password management system using industry standard cryptography techniques. The systems design will encourage users to follow proper password hygiene whilst allowing them to access this information from anywhere via the cloud.

Acknowledgements

I would like to thank Dr John Dixon for this continued patience and support through the whole process of my final year project.
I would also like to thank my friends and family for support and advice, especially Cait and Matthew who helped proof read and suggested improvements.

Contents

1 Introduction

1.1 Background to the project

The modern-day internet requires users to authenticate with many services, each demanding a long, unique, and complex password. Currently, the Ideal complex password comprises a 15-character string containing a random combination of numbers and letters (*How to Create a Strong Password and Beat the Hackers | Avast*, 2020), something that is not conducive for the human brain to remember. The NCSC (National Cyber Security Centre) analysed account information that third parties had accessed from cyber breaches and found the most commonly used password is "123456" , and simply the word "password" being in the top 5 (*Most hacked passwords revealed as UK cyber survey exposes gaps in online security*, 2019).

So, this leads to the big issue of remembering these strong and unique strings for every website you use. This project proposes a solution to these problems in the form of a password manager; this manager would allow the user to generate long, unique strings of saved characters in a secure database. The user would then only need to remember one strong password, which would gain them access to the password manager described.

In 2018 the Global Web Index performed a survey which found "more people own a smartphone, a pc and a laptop than people who own just a smartphone" (*How Device Usage Changed in 2018 and What it Means for 2019*, 2018). This system will allow a user to sync their saved credentials across devices via a server; this means a user will be able to access their login details from anywhere in the world.

This project is aimed at a highly privacy-conscious user. One factor affecting this is the use of a simple password protecting the user's credentials. This system will force the user to use a password as well as an RFID card to authenticate. The system will also avoid using an autofill mechanism as this opens the system up to a phishing attack (*Security flaws found in popular password managers*, 2020).

Similar systems exist already built into browsers; the main issue with this way of working is that you are stuck in the ecosystem of whichever browser you choose to use. Not only are you stuck in that single ecosystem, but if the data on the device is lost, you have lost all your saved passwords.

1.2 Aims and objectives

This project aims to solve the problem of poor password etiquette by taking the onus of remembering off the user, in the form of a secure password manager. The application should support the safe storage of website authentication credentials such as usernames and passwords.

The **main objectives** are as follows:

Objective 1 The solution will support a windows desktop application that will contain a local secure encrypted vault; this will store an infinite number of strong, unique strings of characters that form a user's passwords. All user data will be encrypted and hashed before it is stored.

Objective 2 The vault will support a search function and default and custom categories, allowing the user to filter their saved credentials easily.

Objective 3 The application will use MFA (Multi-Factor Authentication); when the user accesses their vault, they will need to use multiple forms of identification.

Objective 4 The project will also support a password checker and meter, informing the user of a password's strength and uniqueness.

Objective 5 The project will also contain a password generator, this will suggest passwords to the user using a set of rules outlined with knowledge gained from the research conducted earlier in the project.

If there is time, then these are the **secondary objectives** that could be achieved:

Secondary Objective 1 The solution could allow the local database containing encrypted user data to be synced across multiple devices via an API server.

Secondary Objective 2 The final project could also include a feature for securely syncing login details to another vault, and this would prevent somebody from sending passwords in plain text via messaging services.

Secondary Objective 3 The project could include a browser extension that could capture login details as they are used within the browser. This data will then be encrypted, hashed, and stored in the local database.

2 Literature review

2.1 Password Managers

2.1.1 Computer and Information Security

Information security is formed of the CIA Triad, defined by the ISO standard as "preservation of confidentiality, integrity and availability of information" (International Organisation for Standardisation, 2018). These three characteristics make up the cornerstones of Information security, and for a system to be secure, it must address all three. Confidentiality states information must not be accessible to unauthorised parties. Integrity, information, and data being stored must not have been tampered with or changed by an unauthorised individual during the storage process. Availability is defined as information being stored that should be made readily available and usable to authorized parties.

The CIA model has been criticized since its inception in the 1970s (Saltzer et al., 1975); these criticisms state the model does not include enough principles and thus isn't an accurate enough definition. In the last 50 years, there have been a variety of unique new definitions suggested by researchers. In 2000 Dhillon and Backhouse suggested 4 principles: responsibility, integrity, trust, and ethicality (Dhillon et al., 2000). Furthermore, the concept of CIA+ has been formulated by multiple researchers; this includes the addition of Authentication, Access Control, and Non-repudiation (Simmonds et al., 2004; Stallings, 2011). However, Samonas and Coss also suggest all of these new commonly proposed principles are too similar to the original 3 and can be classified under one or within an intersection of two of the original tenets (Samonas et al., 2020).

2.1.2 Types of authentication and Passwords

Authentication is defined by NIST 800-63-3 as "Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to a system's resources" (Grassi et al., 2017). Different forms of authentication can be used, which give different levels of confidence in authenticated users' identities. Some organisations may only need a low level of confidence in a user's identity for them to access system resources as information stored is not very sensitive. On the other end of the spectrum, an organisation may require an extremely high confidence level as information stored is extremely sensitive (Stallings, 2014).

There are three main ways existing systems authenticate users, they can be used alone or in combination with each other. First is some form of knowledge the user possesses, such as a password or a personal identification number (PIN). Next is a physical possession, such as a smart card or dongle. Finally, biometric can be static such as a physical characteristic of the user, or dynamic such as their signature (Brooks, 2013; Stallings, 2014).

Although each of these authentication methods has benefits, all three can be problematic. For a password to be secure, it needs to be complex, which means a user can struggle to remember it. This can lead to a user writing it down. Physical possession will always have the risk of theft or cloning (Brooks, 2013; Kasper, n.d.).

Biometric can be inconsistent; for example, if an individual wears a face mask, facial recognition fails (Brooks, 2013). An attacker can also spoof biometric, and this was proven by the hacker group Chaos Computer Club who published German Interior Minister Wolfgang Schauble's fingerprint ("Hackers Publish German Minister's Fingerprint", 2020).

Smith, 2015 suggests two-factor authentication, this requires two different authentication factors used in conjunction, such as a physical bank card and pin code stored separately. He mentions

this provides Defence in Depth, and this hinders the attackers as they would need both forms of authentication to gain access.

2.1.3 Password Psychology

Passwords continue to form the most common authentication method, even though researchers agree they are insecure and not user-friendly. The NCSC found the most commonly used password in 2019 was "123456" , with simply the word "password" being in the top 5 (*Most hacked passwords revealed as UK cyber survey exposes gaps in online security*, 2020). Research suggests that users continue to use bad practices because they favour easy-to-remember passwords (Tam et al., 2010).

Furthermore, it is estimated users have, on average, 25 distinct online accounts (Florencio et al., 2007). Human memory constraints make remembering complex passwords difficult, leading to reuse (Wang et al., 2015). Password reuse is an issue as leaked credentials that are not hashed securely can access other services. Password leaks become more common, that has been seen recently with the leak of 167 Million LinkedIn accounts (Burgess, 2016; Facebook et al., 2016; *LinkedIn Lost 167 Million Account Credentials in Data Breach*, 2020), 427 million MySpace accounts (*Hacker Tries To Sell 427 Milllion Stolen MySpace Passwords For $2,800*, 2020; Newman, 2016; Schroeder, 2020) and 450,000 Yahoo accounts (Arthur, 2012; Hamilton, 2020).

Even with their downfalls, it seems passwords will continue to be used as the de-facto option for authentication. This is due to several advantages, firstly they do not need custom hardware to be used, and they can also be incorporated into existing services make deployment easier (Das et al., 2014).

2.1.4 Password entropy and what makes a good password

The quality of a password can be challenging to define; researchers have found a way to calculate password strength. This theory is called password entropy and was first discovered by Claude Shannon in 1964 (Shannon, 1964), it was used as a measure of problems within communications theory. Entropy is defined differently depending on the researcher and the discipline within information theory entropy measures "randomness" and "uncertainty" that is contained within a password string (Komanduri et al., 2011; Taha et al., 2013) define entropy by taking the sum of the password length, the number and placement of each class of character, and the content of each character.

Another researcher by the name of Yan calculates entropy by using common patterns within 7-character alphanumeric passwords. This looked at passwords with patterns of repeating alphanumeric characters or a high number of numeric characters. By checking easy-to-crack passwords first in dictionary attacks, he proved that the entropy could be reduced (Yan, 2001).

The strongest possible password is made up of random characters of different classes such as letters, numbers, and special symbols (Brumen et al., 2014). The problem with a long, complex, randomly generated string of characters is that it is simply too difficult to remember, especially unique ones for each system used. Most modern services now offer a system to check password strength using password policies when users create their account (Wang et al., 2015). These password policies force the user to follow a set of requirements when creating their password, this can include password length and the type of characters such as letters, numbers, and symbols (Guo, 2020).

2.1.5 Alternate authentication, e.g., Multi-Factor Authentication and Biometrics

Multi-Factor Authentication is considered more secure than single-factor authentication as the user must possess multiple forms of unique factors for identification (Schneier, 2005). The speed at which passwords can be cracked (Exploitbytes, 2020) increases at an alarming rate, with some once considered secure encryption taking minutes (Güneysu et al., 2008). Password leaks from large companies are common news (Swinhoe, 2020), and an estimated 43-51% of users reuse passwords across multiple services.

The use of smart cards as a form of two-factor authentication was first proposed 30 years ago (Chang et al., 1991) and has since been used in various security-conscious systems. The user registers with the service using a username, password, and smart card. When the user logs in, they authenticate with the password and smart card, this is two-factor authentication (Wang et al., 2016). Smart cards are easy to use but come with their own set of issues. They require special hardware, which takes up space and is an expense to the client.

The second form of two-factor authentication, which also requires a peripheral device, is a One Time Password (OTP) token. An OTP token generates a randomised PIN which changes, and they can come in two formats, hardware or software (Aloul et al., 2009), and can be implemented in various unique ways. The hardware version involves a physical token with an LCD screen. The screen displays the PIN, which the user inputs when logging in (Liou et al., 2011). The software version generates the same OTP in the form of a PIN but does this via an existing computer or mobile device (Aloul et al., 2009).

Biometric authentication can be used to authenticate a user's identity. The individual's fingerprints, face, or iris can be used to identify them (Bhargav-Spantzel et al., 2007). These methods have a high level of security, and they extract features from a visual image which are then encoded using an algorithm (Lakshmanan et al., 2015). One issue with biometric authentication is dedicated hardware is required, which can be costly to the client (Liu et al., 2014).

RFID is another form of two-factor authentication, and it utilises an RFID token in conjunction with an RFID reader. The reader retrieves the information stored on the tag. That information is then fed into an authentication system. When used in combination with a text password, it forms a two-factor authentication system (Liou et al., 2011). Aldwairi et al., n.d. point out the RFID system has the same risks as the OTP token, in that it can be stolen as well, coming at a cost to the users.

2.1.6 Cyber-attacks and Password Cracking

In a generic cryptographic system, authentication is possession-based, in that the user can gain access to a system by supplying the decrypting key. These keys can be long and complex, such as advanced encryption standard (AES) which uses 256 bits (Technology, 2001), making it difficult for a human to remember. As a result, these keys are stored on a server to be used later for authentication (Jain et al., 2006).

Password cracking is used to recover the original password from the hashed value; two types of cracking can be employed. A cracker can interact with a live system and attempt to log in, this only works if the system does not limit the number of login attempts. Crackers can also steal databases of hashed passwords and then employ offline guessing to recover the original password (Yu et al., 2015).

A Brute-force attack is the simplest cracking method an attacker could employ to recover a plaintext password. It involves trying combinations of passwords by hashing them and comparing the

hash against the victim's password hash (Kazuhide Fujita et al., 2008). This is extremely time-consuming, and the problem space grows the larger the password and the more classes it uses. For example, if you had a password made up of 8 lower case characters, the problem space would be 26^8, equal to 208827064576. If we could solve 1000 combinations a second, it would still take 58007.52 hours (Raza et al., 2012).

Dictionary attacks are another form of cracking, they leverage the fact that many users tend to choose passwords from a small domain (Klein, 1992). This means an attacker can employ a dictionary of commonly used words, then they hash each word and common combinations in the list and compare this against the stolen hash of a user's password (Sander et al., 2002). Dictionary attacks are faster than a Brute-force attack, but they are limited by the number of words available to them in their list (Raza et al., 2012).

Rainbow tables can be used to improve the dictionary and brute force attacks. The word list used to compare against the stolen hash is pre hashed and stored. This saves time when the dictionary of words is searched as it only needs to be done once rather than every time the list is searched (Oechslin, 2003).

Key-loggers are a type of attack which can be installed on a victim's computer, this is usually done without a user knowing (Baig et al., 2007). The key-logger records every button the user presses, it then makes a record of these key presses, and they are sent to the attacker. An attacker can then use this information to access users' accounts (Ladakis et al., n.d.).

Phishing attacks involve redirecting an unsuspecting user to a website or system created to mimic another official system. The user is sent to the fake website, which is created to look like the original website, the individual then inputs their login information which the attacker intercepts. Sometimes the user doesn't even know they have visited the wrong site as the attacker will redirect them once he has received the stolen details (Jones, 2013; Uusitalo et al., 2009).

One phishing attack that password managers are susceptible to works by sending the user to a fake website or application, and the autofill mechanism built into the password manager fills the user's login details in. This is due to the password managers using weak matching criteria for identifying whether credentials should be auto-filled or not (*Security flaws found in popular password managers*, 2020).

2.1.7 Encryption and Hashing

Encryption is the process of encoding a plain text phrase into ciphertext, this allows sensitive data to be stored with less risk to the original data (Kazuhide Fujita et al., 2008). There are two types of encryption, Symmetric, and Asymmetric. Symmetric encryption uses the same key to encrypt as well as decrypt data. Whereas Asymmetric has two separate keys, one is used to encrypt data, the other is used to decrypt data. Symmetric key encryption is about 1000 times faster than asymmetric key encryption, making it better for larger data sets (Elminaam et al., 2009).

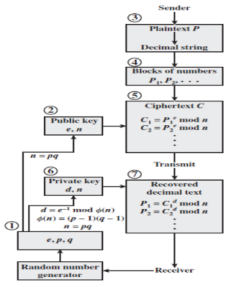


Figure 2.1: RSA processing of Multiple Blocks (Stallings, 2006)

RSA is a form of asymmetric encryption created by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. RSA uses variable-size blocks, as well as variable size keys. It is based on number theory and uses two prime numbers to generate public and private keys. Like any asymmetric encryption, the message is encrypted using the public key and decrypted by the receiver using the private key (Xin Zhou et al., 2011). RSA has some design flaws, if the public and private key are too small then it is easily cracked, if the keys are too large, then the encryption and decryption performance is negatively

affected. The two keys also have to be of similar length, increasing the time it takes for the algorithm to complete (Kakkar et al., 2012).

AES is a form of symmetric encryption recommended by NIST to take the place of DES in 2001. The AES block size can support any data size under 128 bits with three different key lengths, 128, 192 and 256 bits. These key lengths are where the names AES-128, AES-192 and AES-256 come from. The algorithm makes 10 passes for 128 bit, 12 passes for 192 bit and 14 passes for 256 bit, this is done when encrypting and decrypting (M. G. Singh et al., 2011). The 128-bit data is split into 4 smaller states which are each treated as a 4 Byte array. For each pass of the algorithm, the data undergoes 4 separate stages. ByteSub transformation, each byte within the data block is transformed into a new 8-bit substitution box, named an Sbox. ShiftRows transformation, the last 3 bytes of a state are cyclically shifted. MixColumns transformation, a fix matrix is multiplied to each column vector, the bytes are treated as polynomials rather than numbers. AddRoundkey transformation, an XOR bitwise operation is performed on the 128-bit current state and the 128-bit round key. Decryption is the inverse of each of the operations, performed the same number of passes (Mandal et al., 2012).

AES has different modes of operation, one of these modes is called Galois/Counter Mode. This mode allows for a higher performance level due to its ability to perform the block cipher operations in parallel (Lemsitzer et al., 2007). The main benefit of AES-GCM is its ability to authenticate the encryption using an authentication tag, this ensures the authenticity of the data (Computer Security Division, 2017).

Hashing is used to transform a plain text string into a fixed length output called a hash, the process is irreversible meaning it is not possible to get the original input from the hashed value. One way hashing algorithms create a unique hash for every message used, thus hashing is used to store plain text passwords in a database (Pal et al., 2009).

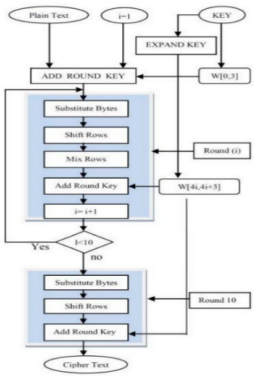


Figure 2.2: AES (Advanced Encryption Standard) process (Mandal et al., 2012)

Secure Hash Algorithm (SHA) and the MD5 message-digest algorithm are both cryptography hash functions. SHA was developed by the National Institute of Standards and Technology (NIST) in 1993, it has undergone many revisions to fix security weaknesses. SHA-1 produced a 160-bit hash, NIST revised this standard with SHA-2 with 256, 384, and 512 bits. It is the most widely used hashing function due to most alternatives being cryptographically insecure (Stallings, 2006). MD5 was extensively used as a cryptography hash function but has since been proven to have extensive vulnerabilities, making it unusable for the hashing of passwords (Gupta et al., 2014).

PBKDF2 (Password Based Key Derivation Function) is a key derivation function, it is a hashing algorithm which generates a secret key from a password and salt value using a hash-based message authentication code (HMAC) (Moriarty et al., 2017). The salt is random data produced as an additional input to the Password Based Key Derivation Function (*A salt-free diet is bad for your security | 1Password*, 0000). Before the password is fed into the HMAC, it is first hashed using a pseudorandom function (PRF), this is usually done using a Secure Hash Algorithm (SHA) (N. Li et al., 2015). The HMAC hashes the plain text password over several iterations, when the standard was first written, 2000 iterations was suggested, now companies who have implemented the algorithm use up to 100,000 (*LastPass Security Notification*, 2011; Turan et al., 2010).

Argon2 is also a key derivation function, it has two versions Argon2i and Argon2d. Argon2i is more resilient against side channel attacks, whereas Argon2d is has a greater resistance against GPU

cracking (*P-H-C/phc-winner-argon2*, 2021). Argon 2 won the Password Hashing Competition in 2015 (*Password Hashing Competition*, 2021). Currently it is regarded as one of the most secure hashing algorithms, however, there have been some successful attacks, one of which was fixed in version 1.3. The second attack currently has been shown to only be effective if, Argon2i uses less than 10 passes over memory (Alwen et al., 2017).

2.1.8 Managing multiple passwords

Password reuse among users is a common issue, this has been shown through many studies performed by researchers. The reason for this is the human brains memory is constrained so to reduce memory load we resort to repetition (Florencio et al., 2007; Ives et al., 2004; Shay, Komanduri et al., 2010). There is a lack of understanding by users regarding password security, in a study by (Grawemeyer et al., 2011) it was found that a user is nearly 18 times more likely to write a password down if it is unique.

Other studies have been conducted into managing multiple accounts and the number of unique passwords a user has. A study by Florencio et al., 2007, collected data on 250,000 Microsoft toolbar accounts. They found that users had an average of 6.5 passwords, each of those passwords was shared across 3.9 websites. The study also found users had on average 25 accounts and typed 8 passwords a day on average). A study by NIST found that users in an organisational setting needed to authenticate on average 23 times a day and found users suffered frequent disruptions (Steves et al., 2014). It is evident that if passwords are forgotten, it can negatively affect an organisation in employee downtime (Bonneau et al., 2012).

2.1.9 Password managers and security

If users are forced into using complex passwords by an organisation's strong password policy, they are more likely to write the password down (Shay and Bertino, 2009; Shay, Bhargav-Spantzel et al., 2007). Password managers are one solution to this security problem, they allow the users to save login details such as usernames and passwords in a database. This database of login information is encrypted using one master password. These managers free the user from the need to remember complex passwords, some managers even offer cloud-based solutions meaning passwords can be accessed anywhere by the user (Z. Li et al., 2014).

Some password managers attempt to reduce the risk when it comes to the action of users sharing passwords. They do this by offering a service within the manager, allowing a user to share a password to another user internally within the system. The password managers also combat poorly chosen passwords by allowing the user to generate a randomised password using a set of rules (David Silver et al., 2014).

Many password managers use a local-only security model. The user creates a master password usually hashed using a Password-Based Key Derivation Function such as PBKDF2. The encryption key produced is usually used for two things, first it is used as a login to the system to authenticate the user. The second is to encrypt the data stored within the password database, AES-256 is usually the preferred encryption method. Any data sent to the online server is first hashed or encrypted to ensure no

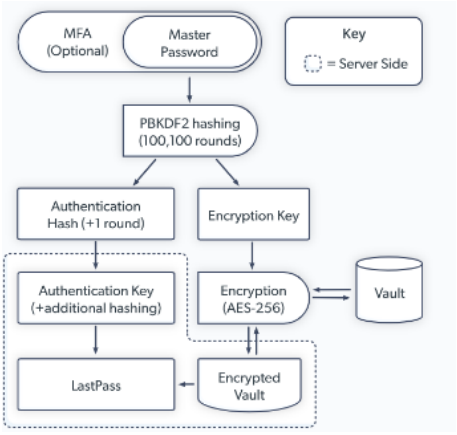


Figure 2.3: LastPass local-only security model (*Enterprise Security Model | LastPass*, 2020)

plain text is sent over the internet (*Enterprise Security Model | LastPass*, 2020).

Two-factor authentication is not required by any of the current commercial password manager options, this means a user can sign up and only use a simple easy to guess password to secure their details. Some password managers such as mSecure (*mSecure Password Manager and Digital Wallet*, 2021) use cloud file storage options such as Dropbox (*Dropbox*, 2021) to sync credentials across devices.

Dashlane (*Password Manager App for Home, Mobile, Business | Dashlane*, 2021) offers the ability like most password managers to sync across devices, as well as a built in VPN and document storage. Other options like LastPass (*#1 Password Manager & Vault App, Enterprise SSO & MFA | LastPass*, 2021) offer the ability to login to your vault via a phone, as well as autofill login details in a browser. Interestingly, Bitwarden (*Bitwarden Open Source Password Manager | Bitwarden*, 2021) is open source, and the free tier unlike most of the other offerings, allows unlimited item storage and syncing across all devices.

2.1.10 Alternative Password Management Methods

There are some alternatives to software-based password managers that have advantages as well as disadvantages. One alternative is hardware-based password keepers, such as a "Mooltipass" . These portable devices store passwords in a database but have the added feature of allowing the user to plug the device directly into their PC. The device emulates a USB keyboard and can autofill the stored login credentials (*The Mooltipass Hardware Password Keeper*, 2021).

Another alternative password management system is named the Single Sign-On (SSO) platform, this allows the user to authenticate once, which would give them access to multiple resources (R. K. Singh et al., 2009). This concept was then improved further with the "Social Login" , allowing a user to authenticate using a social network account (Sun et al., n.d.). However there are privacy concerns with the use of SSO outlined by Kontaxis et al., 2012, users could lose track of the number of sites which have access to their personal information, as these credentials are disclosed to multiple sites this ups the risk of loss, theft and accidental leakage.

2.2 Technologies

2.2.1 Architecture

When producing software, architectural patterns are used to solve common, recurring problems which may arise. These problems could include performance limitations, availability, and reduced business risk (Computing Machinery et al., 2010).

MVVM is a design pattern used for producing desktop applications, it creates a level of abstraction between the front-end GUI code and the domain logic back end. The Model represents the software's data. The ViewModels connects the View and Model, it does not know its View as it uses databinding. The View is like MVC in that it is the part the user interacts with; the View does not know the model. MVVM also lends itself to unit testing as components are completely independent, working on an observer pattern (*Introduction to Model/View/ViewModel pattern for building WPF apps | Microsoft Docs*, 2021). One benefit is that the GUI is decoupled from the domain logic making it easier and less costly to swap out the user interface whilst minimising changes that need to be made to the domain logic (BlogsNook, 2019).

Client Server model is a distributed architecture which involves a server host which delivers resources and services to a client. Usually, the two systems communicate via a computer network but can reside on the same hardware. One advantage of this design is that all of the resources are

stored in one place, this can also be a disadvantage if lots of users request resources at the same time overloading the server (*Client Server Architecture - CIO Wiki*, 2021).

2.2.2 SOAP vs REST

SOAP and REST are both used in the development of web services. SOAP is a communication protocol whereas REST is an architectural style. A SOAP client is tightly coupled with the server via a rigid contract, meaning if there are changes to the client or server the system will break. Whereas REST has a much weaker coupling and usually has stateless interactions, each request contains all the necessary information as the server forgets clients between calls. REST usually uses the HTTP protocol and cross-platform between devices and is very easy to implement (Fredrich, n.d.; Muehlen et al., 2005).

2.2.3 Programming Languages and Frameworks

There is a wide selection of languages available when producing a windows desktop application, each with their advantages and disadvantages. 3 of the most popular programming languages for object-oriented windows development include, Java (*Java SE - Downloads | Oracle Technology Network | Oracle United Kingdom*, 2020), C++ (*cplusplus.com - The C++ Resources Network*, 2020) and C# (BillWagner, 2020; *index | TIOBE - The Software Quality Company*, 2021). C++ is a language which extends C and includes the concepts of classes and objects. It is extremely performant, requiring the developer to manage memory if objects are dynamically allocated (Stroustrup, 1996).

C# is another language which supports classes and objects. This language is more modern but is less performant due to more overheads such as its garbage collection. C# is also statically and strongly typed, and similar to C++ can be used on most platforms (*ECMA-334*, 2021). Java is also an object-orientated language which is statically and strongly typed like C#. Java runs on the Java Runtime Environment (*Java Downloads for All Operating Systems*, 2021) whereas C# runs on Common Language Runtime (*Common Language Runtime (CLR) overview - .NET | Microsoft Docs*, 2021).

2.2.4 Database and Mark-up

When it comes to storing data for the project there are multiple choices. The first method used traditionally is using RDBMS (relational database management systems), such as SQL Server (*SQL Server 2019 | Microsoft*, 2021) , MySQL (*MySQL*, 2021) and PostgreSQL (*PostgreSQL: The world's most advanced open source database*, 2021). Data is stored in tabular form such as a collection of tables, queries can then be used to manipulate said data (Sumathi et al., 2007). An alternative option for data storage is a NoSQL solution such as MongoDB (*The most popular database for modern apps*, 2020). NoSQL databases are non-tabular and non-relational, this mean they are more scalable, and do not require a full database rewrite for scheme changes (Lith et al., n.d.). NoSQL databases usually have higher searching performance as records are linked with pointers, but the trade-off is that they are harder to implement, maintain and design queries for (robvet, 2021)).

XML is another option for data storage, it is a mark-up language used for encoding documents in a standard format (*Extensible Markup Language (XML)*, 2021). XML is human readable and hierarchical, but some serialisation can be hard if there are complex relationships.

2.2.5 Version Control

Version control is the process of managing changes to files such as code and documents. The first system designed for source code control was called SCCS and was published in 1975. More recently the two most popular configuration control systems are Subversion (SVN) (*Apache Subversion*, 2021) and GIT (Spandel et al., n.d.). SVN is centralised, whereas GIT uses a decentralised model. This means a copy of the codebase and its history is stored on each developers computer, improving branching and merging, as well as allowing developers to work offline (*Intro to Distributed Version Control (Illustrated) – BetterExplained*, 2021).

2.2.6 Code Linting

Code linting uses tools to statically analyse code for programming errors, bugs, and stylistic errors. In Hansson's paper he mentions that by using code linting throughout the development of a product, bugs can be spotted earlier, saving time and money at the end of a project (Hansson, 2014).

Resharper is an example of a code linting tool for C#, it integrates with the Visual Studio IDE for "on-the-fly code inspections" (*ReSharper*, 2021). Resharper can be used to increase maintainability and decrease cyclomatic complexity and class coupling (Dibble et al., 2014).

2.2.7 Continuous Integration Pipeline

Continuous integration (CI) is automating the integration and unit testing of code when changes are merged to the source control system. Using CI highlights bugs and problems with code early on, meaning problems can be fixed quickly (Atlassian, 2021). GitHub Actions (*Features • GitHub Actions*, 2021) is built into GitHub and allows the easy automation of testing when code is merged with the main branch.

3 Requirements

The project's requirements outline the features and functionalities of the final software deliverable and the constraints that the software will be under. Requirements ensure that the final product meets the customers' expectations. The benefits of well-written requirements are the cost and time are kept low due to reduced integration and testing, as well as increased maintainability and readability of code (*Verifying and Validating Software Requirements and Design Specifications - ProQuest*, 2021).

3.1 Product requirements

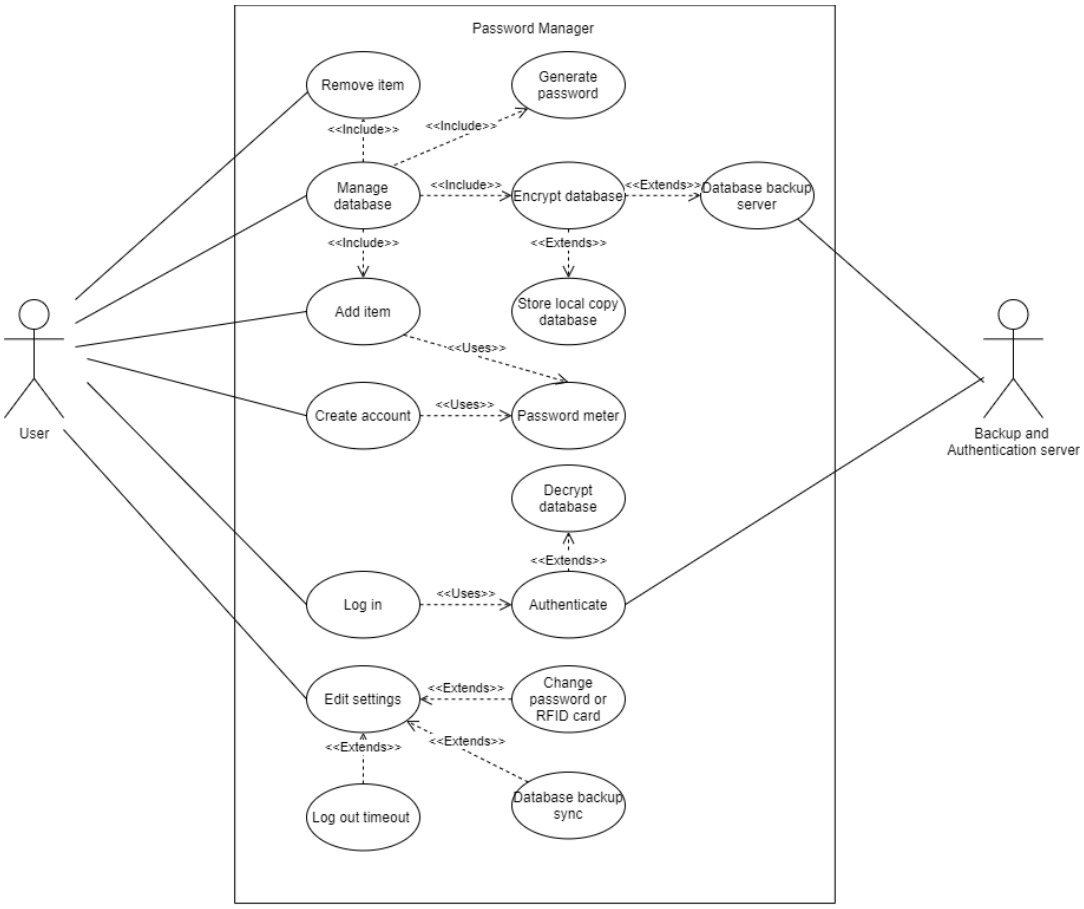


Figure 3.1: Use Case Diagram - Overview windows application use case diagram

The use case diagram in Figure 3.1 shows an overview of the windows client application. This use case diagram shows how the backend of the program will communicate between systems. It outlines a strong boundary in which the program in encapsulated, furthermore it suggests connections which will allow the windows client application to communicate with the server application.

The use case diagram defines ways the user can login, create an account, edit application settings, and manipulate the database. This diagram is a useful first step in the requirement modelling stage as it allows the author of this project to have a high-level overview of how the main part of the system will function and interconnect with different parts of the project.

3.2 External Interface Requirements

The external interface requirements outline how components or services will communicate with each other. Defining these requirements ensures different platforms and parts of the final software will interface effectively.

The client and server components of the project will interface using a Representational state transfer (REST) API over HTTPS, the data will be packaged as JSON. A REST API gives good performance (*Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)*, 2021) whilst also supporting large number of components and thus giving good scalability (*Fielding Dissertation: CHAPTER 2: Network-based Application Architectures*, 2021). A REST API is stateless making it a good fit for this use case as any number of users and lots of concurrent users could be using the authentication and sync service at one time. A stateless service frees up resources when a client is not connected.

The system will use an ACR122U NFC card reader ("ACR122U Application Programming Interface V2.02", n.d.), and MIFARE Classic EV1 1K ("MIFARE Classic EV1 1K - Mainstream contactless smart card IC for fast and easy solution development", 2018) because they are cheap, accessible and have free drivers for common operating systems.

3.3 Non-Functional requirements

Non-functional requirements make up the quality constraints with which the solution must be held, this is any requirement that doesn't fall under the umbrella of functional (*Functional vs Non Functional Requirements*, 2020).

3.3.1 Performance requirements

The system must encrypt the database in a reasonable amount of time, such that the user does not get frustrated. The system will retrieve and decrypt the database from the server in a reasonable amount of time. All database manipulation must happen within a few seconds of the user acting otherwise their attention is lost (Miller et al., 2001).

3.3.2 System Dependability

The system is a utility that needs be available on a 24-hour basis, this means it must be dependable and reliable. The system will keep an offline copy of the password database locally on the user's computer. If the user does not have access to the backup server, due to internet trouble or the server itself being down they would still have access to the most recent copy of their data. The system must also provide meaningful error messages to users to troubleshoot any issues with the system.

3.3.3 Usability

The system must have well written labels on all settings and buttons to make their use obvious. The user interface layout must be intuitive so that it is easy to use without a tutorial/ documentation. The system must also be laid out so that a given task can be done in as few button presses as possible.

3.3.4 Data Redundancy

The system will have the ability to synchronise and backup stored vault regularly to a server which means it can be retrieved if the client pc were to fail. When the user authenticates the latest copy of their vault will be retrieved from the server. The system must ensure that all data stored in databases is normalised, this ensures there is no repeating data. By ensuring there is no repeating data, the stored information will use storage space as efficiently as possible, this is important as we can storing files locally on a user's pc as well as on a server. Only allowing the user to connect to the backup server from one computer at a time allows for less conflicts. If the user accesses their database in offline mode the vault will be in read only mode, preventing data conflicts.

3.3.5 Scalability

The system must have the ability to scale well. This means it must be designed to sustain good performance during heavy load or have the ability to easily increase support more concurrent users (*The Importance of Scalability In Software Design Software Engineering*, 2021). As this is a university project with limited resources, the API must support at least 1000 concurrent users signing in and requesting their vaults, this must be completed in a reasonable amount of time while under load. The system must be designed to allow for easy scalability if the project was to be used in industry.

3.3.6 Constraints and Limitations

The main constraint and limitation which could affect this project is time, as this is a university project, the author is limited to two semesters to finish the project. The author also has other university responsibilities alongside this project which means they cannot dedicate all their time. The project is also constrained by the cryptographic options, as the author is not an expert in the field, off the shelf libraries must be used. These libraries must also be able to run on an average consumer's home computer.

The system will be constrained by the language used; this language will be C# . One requirement of the project is for the software to work on a windows PC, this limits the choice of languages which would apply to this task. C# is well suited to this project's objectives as it object orientated, allowing for large projects with multiple services. C# is also versatile in that it allows the software to be targeted to multiple platforms. The system will use .NET Core 5 for both the GUI application and the server application as they will both run on windows 10.

3.3.7 Error cases

Appendix 10.1 shows the error cases which the project must overcome, along with mitigation that can be used to reduce their likelihood.

3.4 Functional requirements

Functional requirements are requirements which outline the basic functionality of the system. Every defined functional requirement should be easily identified within the system, they are usually made up of an input, an operation performed and the output ("Functional vs Non Functional Requirements," 2020).

3.4.1 Interface/ GUI

REQ1 The system must have a create account page

- REQ1.1 The create account page must accept an Email Address, and a master password.
- REQ1.2 The create account page allow the user to re type the master password as confirmation.
- REQ1.3 The create account page must allow the user to associate an RFID card and 6-digit pin with the account.
- REQ1.4 When the user enters their password, a rating must be given in the form of a password meter.

REQ2 The system must have a login page

- REQ2.1 The login page must allow the user to enter login email and password.
- REQ2.2 The login page must also request the user to scan the RFID card and enter their 6-digit pin associated with the account.

REQ3 The system must display the current user entered login details from the vault

- REQ3.1 Login details must be made up of a Label/ Name, Username, Password, URL/Application name, Notes.
- REQ3.2 The password must be obscured with the ability to copy and a button to view in plain text.
- REQ3.3 The ability to add, delete and edit login detail entries in the vault must be present.
- REQ3.4 Entries must be able to be added to a category/ folder for organisation.
- REQ3.5 The system much allow the user to favourite certain login information.

REQ4 The system must display a page when the user attempts to add new login details into the vault.

- REQ4.1 The page must support text boxes for the Label/ Name, Username, Password, URL/Application name, Notes and Category.

REQ5 The system must allow a user to create and delete custom categories.

REQ6 The system must have a menu which allows the user to generate a random password.

- REQ6.1 The generated must have options to change the generated passwords attributes.

REQ6.2 The strength of the generated password must be indicated.

REQ7 The system must have a button which allows the user to sync their vault with a server.

REQ8 The system must have a settings page which allows the user to control certain aspects of the software.

REQ8.1 The settings page must allow the user to change the master password and RFID card and 6-digit pin associated with the account.

REQ8.2 The settings must have an option which allows the user to go online if they have logged in, in offline mode.

REQ8.3 The settings menu must contain an option which would allow the user to delete their account.

3.4.2 Functional Capabilities

REQ9 All the user's private information e.g., login details and vault must be hashed and salted or encrypted locally before being sent over the internet to the backup and sync server.

REQ9.1 The connection to the server must be HTTPS.

REQ10 The user will provide a master password when they create and account.

REQ10.1 The system will take the master password supplied by the user and generate a hash using Argon2i.

REQ10.2 The hashed password and the users email address will be stored in a database, on a server.

REQ10.3 The hashed password will also be used to encrypt the AES key used to encrypt the users sensitive credential information.

REQ10.4 The hashed password will also be used to encrypt the users vault file.

REQ11 The user supplies an NFC chip when creating their vault.

REQ11.1 A secret must be generated and stored on the RFID card when the user first associates the card.

REQ11.2 The secret stored on the RFID card must be concatenated with the master password before being hashed and salted locally on the users pc.

REQ11.3 The user must supply a 6-digit pin when the RFID card is added to the account, used to authenticate.

REQ12 The user will be able to view the vault of store credentials.

REQ12.1 The user will be prompted to scan an RFID card associated with their account, and the secret will be read.

REQ12.2 The user enters their master password, and the RFID Card secret is appended to the end.

REQ12.3 The concatenated text is hashed in two different ways, one uses the email as a salt and the other uses a randomly generated salt stored in the vault file.

REQ12.4 The "email" hash will authenticate the user by comparing the result with the stored hash in the database.

REQ12.5 When the user has successfully authenticated with the system, the latest copy of the vault will be fetched from the server and decrypted using the ”random” hash.

REQ12.6 The user will be able to access their vault when offline but only in read only mode and if they have a local copy of the vault file.

REQ13 The system must allow a copy of the encrypted vault to sync with the server.

REQ13.1 The system will do this each time the user manipulates the vault's data, either a Credential or Category.

REQ13.2 It will be possible for the user to manually sync the vault to the server.

REQ13.3 The system will only allow one computer to edit the database at a time, this will prevent conflicts.

REQ14 The system must support an authentication server for when the user is connected to the internet.

REQ14.1 The system must allow the user to authenticate with the backup server.

3.4.3 Error Handling

REQ15 Errors must be handled when user submits account creation form.

REQ15.1 When the user clicks submit a warning must show if the password strength scores under a certain value, asking the user to enter a more secure password.

REQ15.2 When the user clicks submit a warning must show if they did not enter a properly formatted email address.

REQ15.3 When the user clicks submit a warning must show if an account with that email address has been created in the past on this pc or that is synced with the server.

REQ15.4 When the user clicks submit a warning must show if they have not entered to matching passwords.

REQ16 When they user attempts to login, errors must be handled

REQ16.1 The system must warn the user if the card presented does not match with the login credentials provided.

REQ16.2 When the user clicks the log in button a warning must show if the email address has never been used to make an account.

REQ16.3 When the user clicks the login button a warning must show if the password and email address do not match with an account on the system.

REQ17 When the user attempts to add new login details to the vault errors must be handled

REQ17.1 The system must warn a user if the login details entered has a missing name.

REQ17.2 The system must warn the user if the login details entered have an attribute of the wrong datatype e.g., a number in a text box.

REQ17.3 The system must allow the user to enter duplicate entries of login details into the database.

REQ18 When a user attempts to add a new category, a warning must show is the value is invalid.

REQ19 When a user attempts to delete a category, a warning must show if it is a default category.

REQ20 When a user performs any database manipulation action while in offline mode, a warning must show and inform the user to go online.

3.4.4 Security/Privacy

REQ21 The system will hash and salt the master password supplied by the user when they create their account.

REQ21.1 Argon2i hashing to obscure the master password.

REQ22 The system must support the use of an RFID card as a second form of authentication.

REQ22.1 A secret stored on the RFID card will be combined with the master password before it is hashed using Argon2i.

REQ22.2 The Secret will be protected using a 6-digit pin supplied by the user.

REQ23 The system will store a vault of the user's private login information.

REQ23.1 Each credentials password will be encrypted using a random 256-bit AES key.

REQ23.2 All encryption in the system will be performed using AES-256-GCM.

REQ23.3 The whole vault will also be encrypted using the hashed master password.

3.5 Project Management

This project will be managed using the SCRUM framework (*What is Scrum?*, 2020), planning, testing, and reviewing the progress so far after 2 week "sprints" . In conjunction with this planning technique the project will use a Kanban board (Atlassian, 2020), this makes visualising tasks easier. Planning becomes easier as tasks can be moved to the completed section and the backlog shows what is left to be done. Both techniques will be used collectively to prioritise and deliver the requirements set out for the project.

4 Technical Considerations

4.1 Two-Factor authentication

The system will use two factor authentication, as described in section 2.1.5 of the literature review, this helps to ensure the user who is authenticating is who they claim to be. This project will use a user generated password and randomly generated secret phrase stored on an RFID chip to authenticate a user. The password and secret will be concatenated before being hashed and used to encrypt the user's vault. The other option for two factor authentication would be to use the password to encrypt the vault and the RFID secret during the login process, this would create a placebo effect or security theatre (Bruce Schneier, 2003) as an attacker may be able get a copy of the vault file if the server is compromised and the two factor authentication would not protect the file.

4.2 Encryption and Hashing

When it comes to encryption and hashing, the project will use libraries created by experts. This is because the developer of this project is not a cryptography expert and by using open-source libraries written by security experts that many others have also scrutinised, it helps to maintain a higher level of confidence in the implementations.

Hashing will be used as part of the authentication and encryption process. As mentioned in section 2.1.7 of the literature review, there are many hashing algorithms, the most secure of these being Key derivation functions (KDF). This project will use Argon2i as experts currently recognise it as the strongest hashing algorithm. The project will use the Konscious.Security.Cryptography.Argon2 (Aragon, 2021) implementation, the code for this is open source that has received multiple people's contributions.

Argon2i takes multiple parameters which affects the computational cost along with the security and cracking resistance (*Argon2 — argon2-cffi 20.1.0 documentation*, 2021). This project will use a degree of parallelism of 8. The memory size will be set to 8192KiB as the average user will have this amount spare. The iterations parameter will be set to 40, after some basic testing this ensures the user does not wait too long.

Encryption will also be used throughout the system so choosing a strong algorithm is important. As outlined in the literature review there are multiple available, currently AES-256 is highly regarded by experts making it a good choice for this project. Some versions of AES support authentication of the cipher text, ensuring the data's integrity.

GCM (Galois Counter Mode), OCB (Offset Codebook Mode), AEX and CCM (Counter Mode with CBC MAC) are all authenticated encryption modes for AES. GCM is currently one of the most popular modes of authentication encryption, this is due to the fact it is fast, feature rich and does not have a patent. OCB is also very fast, feature rich. EAX and CCM are much slower than the previous two modes, and with CCM you need to know the message length before you begin encrypting (*How to choose an Authenticated Encryption mode*, 2012). This project will use the .NET implementation of GCM (dotnet-bot, 2021) as .NET Core is open source meaning backdoors are less likely as the code can be audited by anyone.

4.3 Programming language and framework

Section 2.2.3 of the literature review looked at different options available for the programming language and frameworks which could be used to create the final piece of software. The project will be created using .NET 5 and C# . .NET 5 is the newest framework created by Microsoft, unifying all the previous versions of .NET into one single release, it is also open source. The latest version of .NET's performance is also very good, beating many other current popular languages and frameworks in a network test (*ixy-languages/ixy-languages*, 2021).

C# also has advantages over the competition and is best suited to this project. The main benefit of C# is that it integrates extremely well with windows, no special configurations are needed to a program working in a windows environment.

The GUI framework for client of this project will be WPF (TerryGLee, 2021), it includes advantages which make it a good choice. It uses XAML which makes creating and editing the GUI faster and easier. It also supports data binding, which helps separate the view model and view, helping to follow the MVVM architectural pattern. WPF also supports styles giving the design freedom to look however the developer of this project envisions (*XAML - Overview - Tutorialspoint*, 2021). This project will use the ModernWPF UI Library (Wu, 2021), allowing the project to follow the modern design patterns of UWP windows applications. These patterns outline universal; design features that can be included in modern windows applications, such as navigation and control layouts (jwmsft, 2021).

4.4 Architecture

Section 2.2.1 of the literature review mentioned different structural design patterns which could be used for this project. The pattern that has been chosen is MVVM, this pattern is commonly used for WPF application as it is suited well to the features available. The main advantage of MVVM is that the back-end logic is split up for the GUI, this makes swapping in a different user interface easier, with less refactoring needed.

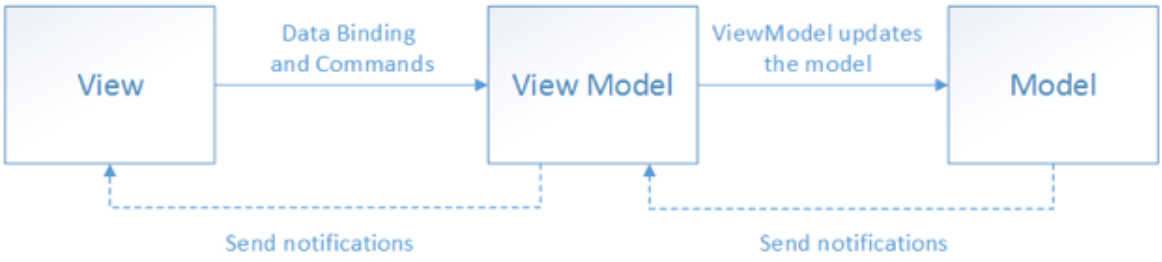


Figure 4.1: The MVVM pattern (davidbritch, 2021)

The project will also use the repository pattern (*Guide to app architecture*, 2021) for data access. The repository pattern separates the application from the data, this allows data from multiple sources to be used easily as the data is centralised and in its own data layer. By centralising data access this will provide better maintainability as the infrastructure is decoupled from the domain layer (nishanil, 2021).

This project will use a simple client server architecture. The server will be in the form of a REST API with RPC elements, taking HTTPS requests from the client. A client server architecture has the main benefit of having full control over the security as the system is centralised. Furthermore, as this project's server aspect is just handling backup and authentication, the server will not be under heavy load and thus will be quite cheap and scale easily.

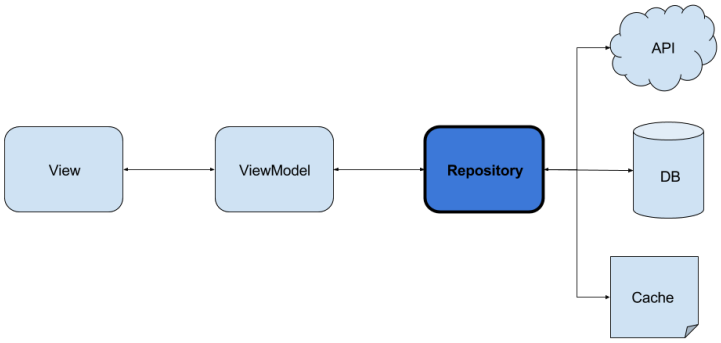


Figure 4.2: MVVM Repository pattern (Latcu, 2020)

4.5 Database and XML

There are multiple options available for the choice of database and mark-up language used to store the systems data, as outlined in the literature review in section 2.2.4. The project will use a relational database management system by the name of SQL Server as all the user data stored on the server will be structured and in a standardised format. Using a NoSQL option like MongoDB a poor choice as it favours unstructured data and with that has a fewer number of queries.

SQL Server offers great options, helping to meet the project's non-functional requirements such as data security, performance, and redundancy. It is also free making it a good choice for small projects such as this one.

There are multiple options for storing the user's actual credentials in the vault file as outlined in the literature review. SQLite (*SQLite Home Page*, 2021), a relational database solution and XML are the two-best fit for the use case of the vault file. This is because the vault is only small and does not need to store a large amount of data, it is also self-contained with simple data types, with the main requirement being performance. The vault files could also be stored on a remote server such as Amazon S3 (*Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3)*, 2021), as it offers secure and cheap data storage.

4.6 NFC Reader and Cards

The system will use an NFC reader and card for two factor authentication. The literature review identified multiple RFID/ NFC options, as this is a university project and a proof of concept rather than a fully-fledged industry ready solution the project will use a readily available card and reader. This project will use the ACR122u RFID Reader, this reader is compliant with ISO/IEC18092 (International Organisation for Standardisation, 2013) for Near Field Communication. This also means the reader will work with MIFARE cards and any cards which adhere to ISO 14443 A and B (*ACR122U USB NFC Reader*, 2021).

The card chosen for this project is the MIFARE Classic 1k RFID card, this is due to its cheap price and availability. This card is not the most secure and newer, more secure, and more expensive options exist.

5 Design

5.1 System Design

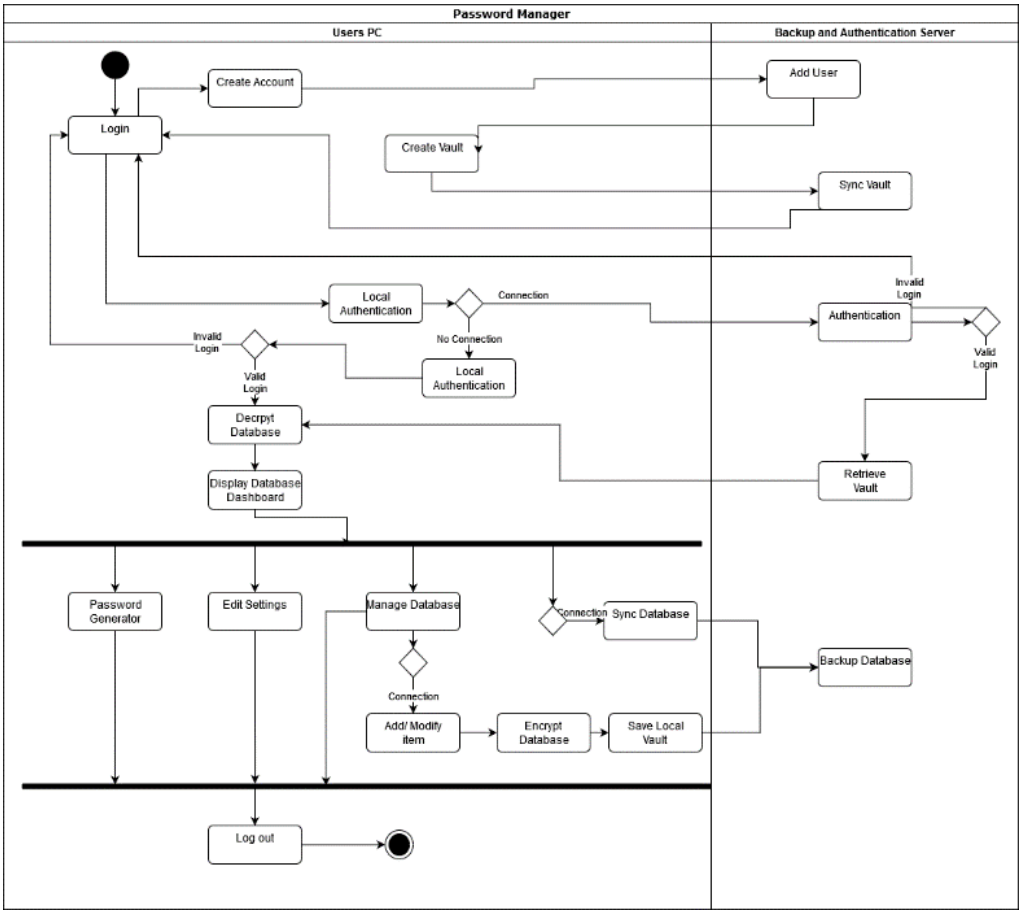


Figure 5.1: Activity diagram project overview (full-page version – Appendix 10.2)

Flow diagrams were created in conjunction with the use case diagram Figure 3.1 to give a high-level representation of the system's flow and dynamic relationships. Flow diagrams are used for business process modelling. Each action state in the diagram represents a process activity and the edges connecting them represent the flow between these actions. Activity diagrams can be used to show conditional structures, loops and concurrency (*UML Activity Diagram Tutorial*, 2021). This process of simplifying complicated use cases and breaking them down into step-by-step workflows ensures the developer has a strong understanding of what needs to be implemented and the flow between these behaviours.

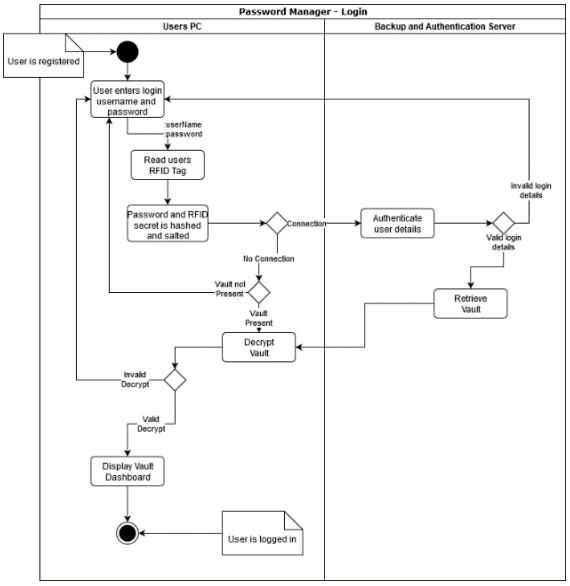
Other diagrams such as sequence diagrams would be poorly suited to displaying the flow from behaviour to behaviour as they are more appropriate for showing the interaction between two classes or services.

Figure 5.1 shows an overview flow diagram of the system's whole underlying structure, from starting up the software and logging in, to retrieving the vault and then performing actions to manipulate said vault. This diagram is best suited to modelling the underlying interactions between parts of

the system as it shows the interconnected activities with a beginning and end.

More detailed flow diagrams were created for each of the 5 main activities which it will be possible to complete using the system. These flows can be found in Figure 5.2 - Figure 5.2 and include, login, creating an account, managing the vault, and editing the system settings. Again, activity diagrams were suited to the task of showing the more detailed interconnected behaviours between processes.

5.1.1 Login



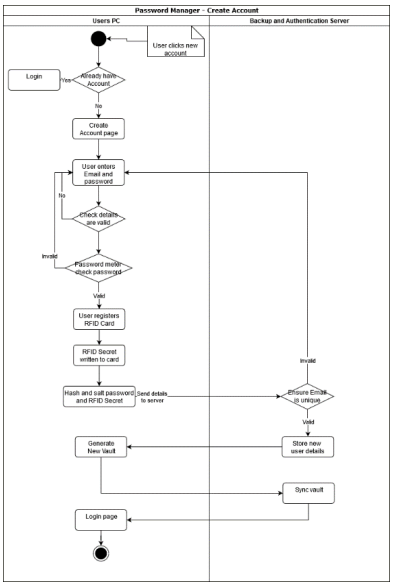
Use case name	Login to database dashboard	
Goal in Context	An existing user authenticates to access their vault.	
Preconditions	The user has an existing account with the system.	
Successful End Condition	The user gains access to their vault	
Failed End Condition	The user is rejected access to the system	
Actor	Password manager user	
Trigger	The user requests to login to the system	
Main flow	Step	Action
	1	The user enters their email and master password
	2	The RFID card is read, and the unique key is appended to the master password.
	3	The password and RFID ID is hashed and salted.
	4	The supplied login details are sent to the server and checked against the ones stored in the database.
	5	The users vault is retrieved from the server and stored locally.
	6	The system then attempts to decrypt the vault using master key.
	7	The data from the vault is displayed to the user's dashboard.
Extensions	Step	The user is offline
	4.1	The system checks for a local vault stored on the users pc.
	4.2	The system attempts to decrypt the vault using the supplied login details.
	4.3	The data from the vault is displayed to the user's dashboard.

Figure 5.2: Activity diagram login overview (full-page version - Appendix 10.3)

The flow diagram shown in Figure 5.2 shows the process of logging in to an existing account. The login page flow diagram adheres to the requirements set out under REQ1, REQ9, REQ12.1–REQ12.2, REQ14, and REQ16. The activity diagram shows the flow from beginning the login process to the end and how those two points are connected. The diagram shows how a user can enter login details, the process of hashing those details, authenticating with a server and the conditional statements which check that the details have been entered correctly.

When designing this flow, the non-functional requirements dependability, security, and usability also needed to be met. Dependability will be met as the user will be able to access the last local backup of their stored database using their login information if they cannot access authentication service. The security non-functional requirement is also met as the password will be hashed and salted.

5.1.2 Create Account



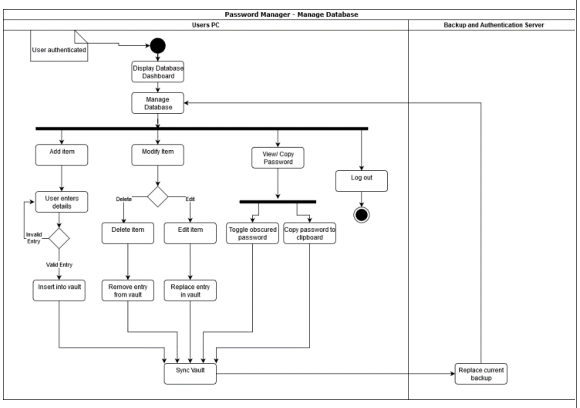
Use case name	**Create a new account**	
Goal in Context	A new or existing user creates a new account for a new password vault.	
Preconditions	The user has successfully installed and run the software.	
Successful End Condition	The user creates a new account with a new password vault	
Failed End Condition	The user fails to create and account	
Actor	Password manager user	
Trigger	The user requests to create an account	
Main flow	Step	Action
	1	The user enters their new email and master password
	2	The details are checked that they are valid and formatted correctly i.e., email address.
	3	The password meter returns a score on the entered password.
	4	The user scans their RFID card.
	5	The master password and RFID key is hashed and salted before the credentials are sent to the authentication server.
	6	The server checks if the email has been used before
	7	The user's details are stored in the authentication database.
	8	The user vault is generated and synced with the server
	9	The user is taken to the login page.
Extensions	Step	Branching Action
	2.1	The details entered are not formatted correctly
	2.2	The user is taken back to the previous page to try again
	3.1	If the password entered scores too low, then the user is asked to enter a new one
	6.1	If the email has been used before then the user is asked to enter a new one.

Figure 5.3: Activity diagram create account overview (full-page version - Appendix 10.4)

The create account flow diagram shown in Figure 5.3 outlines the process of creating an account. This workflow satisfies the functional requirements REQ1, REQ10, REQ11, REQ14, REQ15, REQ17, REQ21, REQ22, and REQ23, it does this by showing the flow behaviours from when a user begins creating an account to finishing the account creation. This process of modelling the application logic helps break down the larger more complex tasks, making complicated behaviour easier to understand and implement.

The flow diagram encapsulates the non-functional requirement of data security as all the hashing and salting will be done locally on the user computer before being sent over the internet to the authentication server.

5.1.3 Manage database

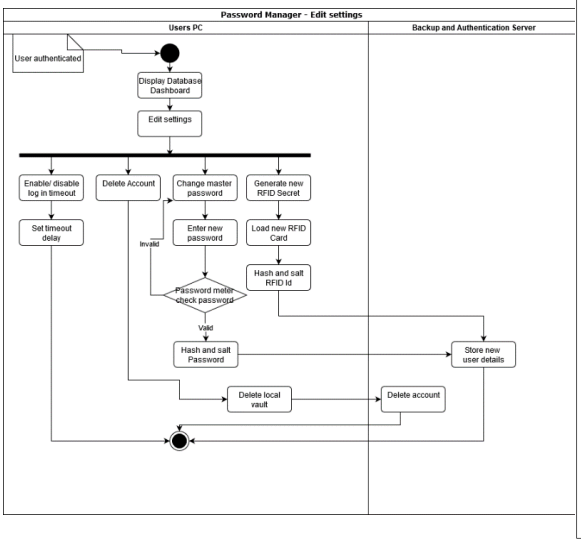


Use case name	**Manage database**	
Goal in Context	An existing user manages their vault of passwords	
Preconditions	The user has successfully installed and run the software, as well as having an existing account.	
Successful End Condition	The user can make all the changes they would like to their vault of passwords.	
Failed End Condition	The user fails to make changes to their database.	
Actor	Password manager user	
Trigger	The user enters the display database dashboard	
Main flow	Step	Action
	1	The user chooses to add an item to the vault
	2	The user enters the relevant details such as the username, password, and URL
	3	The details are checked for formatting
	4	The details are inserted into the existing database.
	5	The vault is synced and backed up to the server
Extensions	Step	Branching Action
	1.1.1	The user chooses to modify and existing item.
	1.1.2	The user chooses to edit the existing item
	1.1.3	The user edits the items information by replacing the necessary values and the new item replaces the old.
	1.1.4	The vault is synced and backed up to the server
	1.1.2.1	The user chooses to the delete the existing item
	1.1.2.2	The item is deleted from the database.
	1.1.2.3	The vault is synced and backed up to the server
	1.2.1	The user chooses to view or copy the password from an existing item.
	1.2.2	The user chooses to toggle the obscured password, so it is now visible.
	1.2.2.1	The user chooses to copy the obscured password to their clipboard.

Figure 5.4: Activity diagram manage database overview (full-page version - Appendix 10.5)

The manage database flow diagram shown in Figure 5.4 outlines the actions performed on the user's database. This encapsulates the functional requirements REQ3, REQ4, REQ5, REQ6, REQ13, and REQ17. This diagram shows the flow from displaying the dashboard to manipulating the vault in different ways to the end flow of the user logging out. This complex use case has been simplified making the interactions between activities easy to understand and easier to implement. This workflow also encapsulates some of the non-functional requirements such as data security, performance, and usability.

5.1.4 Edit settings



Use case name	**Edit account settings**	
Goal in Context	An existing user edits their account settings	
Preconditions	The user has successfully installed and run the software, as well as having an existing account.	
Successful End Condition	The user changes the settings which they set out to edit	
Failed End Condition	The user fails to edit the settings they needed to.	
Actor	Password manager user	
Trigger	The user enters the edit settings page.	
Main flow	Step	Action
	1	The user chooses to edit their master password.
	2	The user enters their new password.
	3	The new password is checked using the password meter and a score is given.
	4	The password is hashed and salted and then used to encrypt the vault
	5	The hashed password is sent to the authentication server to be stored.
Extensions	Step	Branching Action
	1.1.1	The user chooses to edit their RFID Id
	2.1.2	The user scans their new RFID chip
	2.1.2	The RFID key is hashed and salted and sent to the authentication server to be stored.
	1.2.1	The user chooses to edit the log in timeout settings
	1.2.2	The user can enable or disable the timeout setting
	1.2.3	The user can enter a new timeout values, and it is saved locally.
	3.1	If the score is too low, then the user is asked to enter a new password.

Figure 5.5: Activity diagram settings overview (full-page version – Appendix 10.6)

Figure 5.5 shows a flow diagram created to meet the relevant functional requirements for changing the system settings. This flow diagram meets the functional requirements set out under REQ7, REQ8, allowing the user to change the system's basic settings. This diagram breaks down the requirements into the interconnected relationships and shows how a user would navigate the system's settings. This is important to aid in the process of implementing the system as by simplifying the activities it makes implementing the logic easier.

5.2 Software design

5.2.1 Security Design

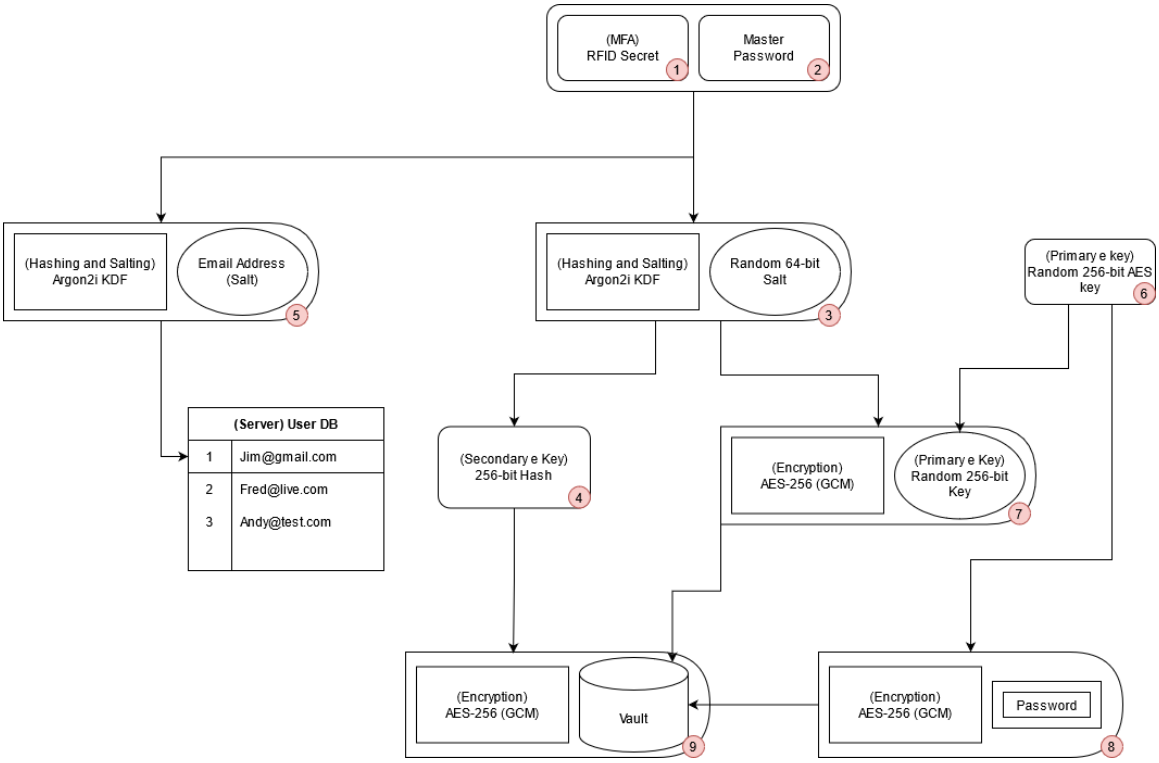


Figure 5.6: Overview of security model

Figure 5.6 shows an outline of the security model design, this diagram is a visual representation of how the users' sensitive data is protected using zero-knowledge encryption. The users master password (1), along with the secret stored on their RFID card (2) will be concatenated and then hashed and salted using Argon2i and a randomly generated 64-bit salt (3), this will form, the secondary encryption key (4). A separate hash will be calculated using the same password and RFID secret combination with the Argon2i hashing algorithm, but this time the users email address will be used as the salt (5). This second hash will be stored in the user database and will be used to authenticate the user.

When a credential is added to the vault, the password associated with it will be encrypted using AES-GCM (8). The key used for this encryption will be generated when the user first creates an account and will be formed of a randomly generated 256-bit AES key (6), this will be known as the primary encryption key. Before the primary encryption key is stored in the vault XML file's header, it will be encrypted using AES-GCM using the secondary encryption key as the symmetric key (7). The encrypted primary key will then be stored in the vault file. Finally, the whole vault file will be encrypted using the secondary encryption key (9) before being written to the disk or synced with the server.

Zero knowledge encryption means the data supplied by the user is obscured so that nobody else with admin privileges within the system can get access to it. When the vault file is sent to the

server, the server admin has no way to decrypt the file and view the stored credentials. This is important because if a hacker were to gain access to the server, they would have access to the decryption keys for every vault file without zero knowledge encryption.

5.2.2 Class design

As this project will be implemented using the object-oriented programming paradigm, it is important to first model the system's main class elements along with their relationships to other classes within the system. These diagrams also include basic method and attribute declarations, this gives the developer of the system an idea of what each class will know and a basic outline of what it will be able to do.

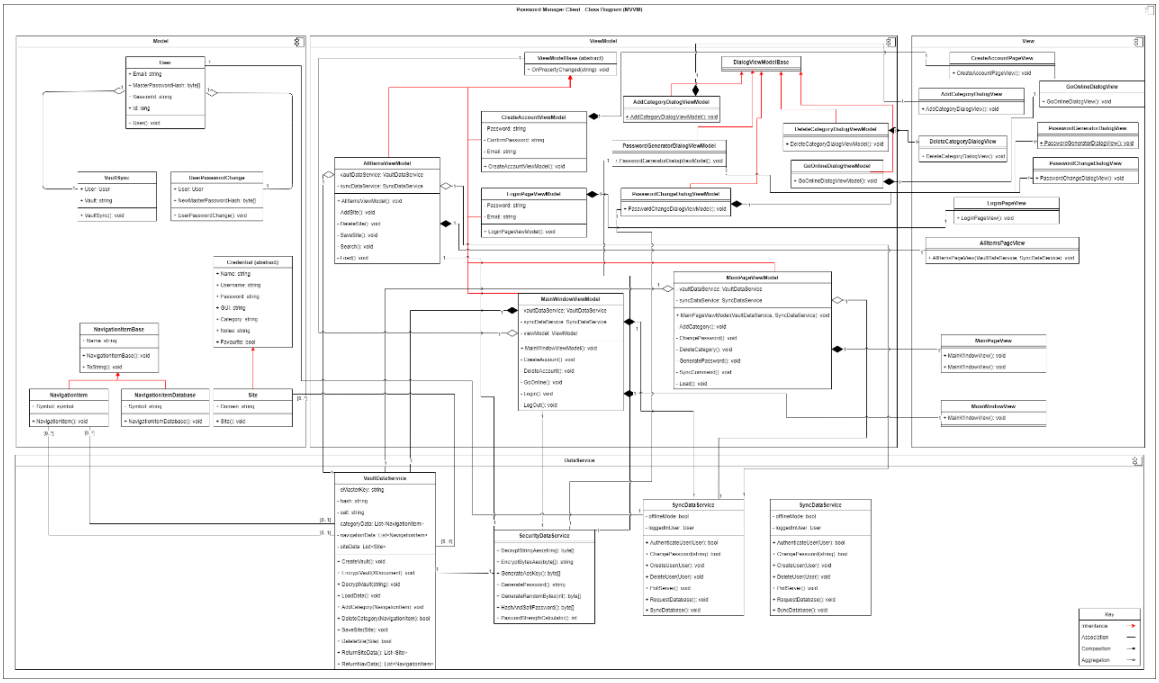


Figure 5.7: Overview System class diagram (full-page version – Appendix 10.7)

The diagram shown in Figure 5.7 shows the static relationships between the main classes within the system. It also shows an overview of the architectural design the system will follow, this is called MVVM (Model–View–Viewmodel). The diagram highlights that the graphical user interface is separated from the business and domain login back-end code.

The diagram shows many kinds of relationships. The red lines show inheritance within the system, these are child classes which are derived from a parent class, receiving some of their methods and attributes from this parent. Simple association between classes is also shown in the diagram, this shows the multiplicity between two objects. Aggregation and Composition are also shown, these define more complex associative relationships between objects.

5.2.3 Network Architecture Design

The system will support a server to allow the users local vault to be synchronised between devices. The server must also allow the user to authenticate before they are able to send or receive copies of the password vault. The server platform will be in the form of a ASP.NET REST Web API. This framework will provide a secure method of communication for multiple simultaneous clients due to its use of HTTPS.

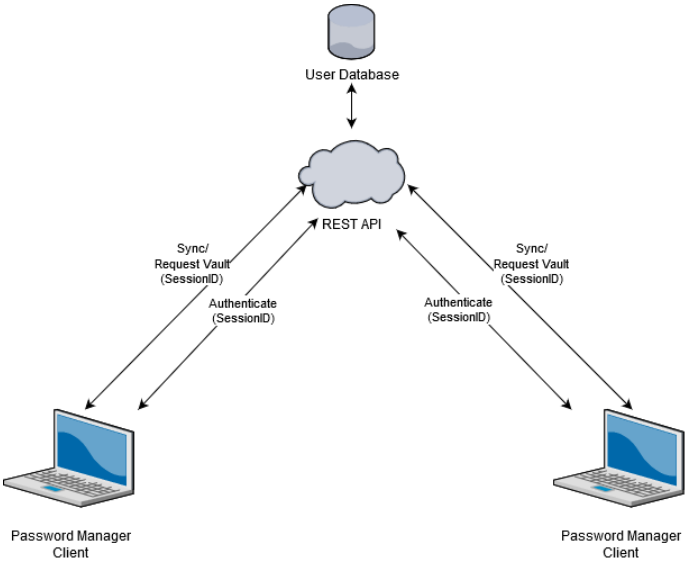


Figure 5.8: Network diagram

The diagram in Figure 5.8 shows an overview of the software network architecture. The client will communicate with the API using HTTP methods such as GET, POST and PUT requests. The diagram shows that the API will connect to multiple clients and deal with requests simultaneously. The API will have direct access to an SQL Server database containing the user's authentication details. The API will also store a copy of the users vault which will be used when syncing with the client.

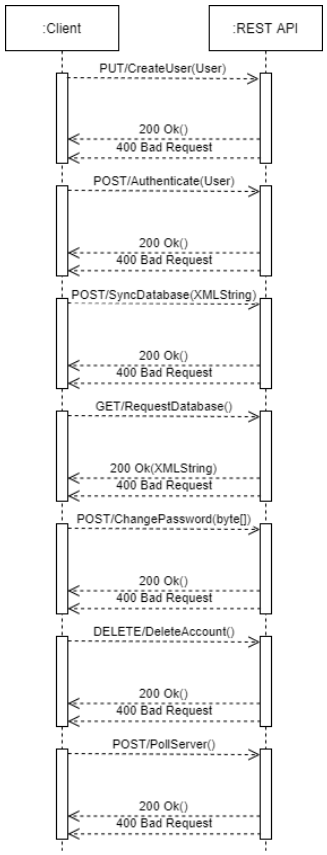


Figure 5.9: Sequence diagram REST API

The above sequence diagram Figure 5.9, shows an overview of the requests which can be made between the client and the API. The API design does not strictly follow the REST architecture and incorporates parts of RPC and REST. This is due to the fact the API will support procedure calls as well as return data to the client. When a user authenticates with the server a session Id will be generated. This session Id will then be added to the header of each API call, ensuring a user can only be authenticated from one client at a time, increasing security. Appendix 10.8 shows a break down for what each call to the API will do.

5.2.4 Interface Design

The tool used to design the basic layout of the user interface was draw.io (*Flowchart Maker & Online Diagram Software*, 2021). These early initial designs represent the arrangement of features and functionality outlined by the functional interface requirements in section 3.4.

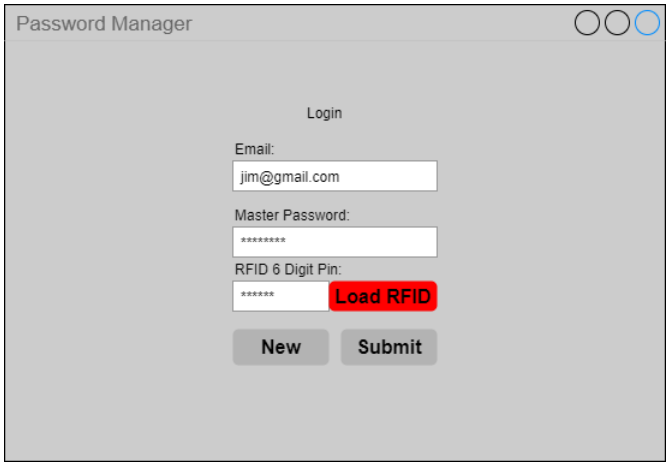


Figure 5.10: Login Page Wireframe Design

The design outlined in Figure 5.10 shows the login page. The page will have two text boxes for the users Email and Master Password, defined in requirement REQ2.1. There is also a box for the 6-digit pin, alongside a button to load the RFID card as outlined in REQ2.2.

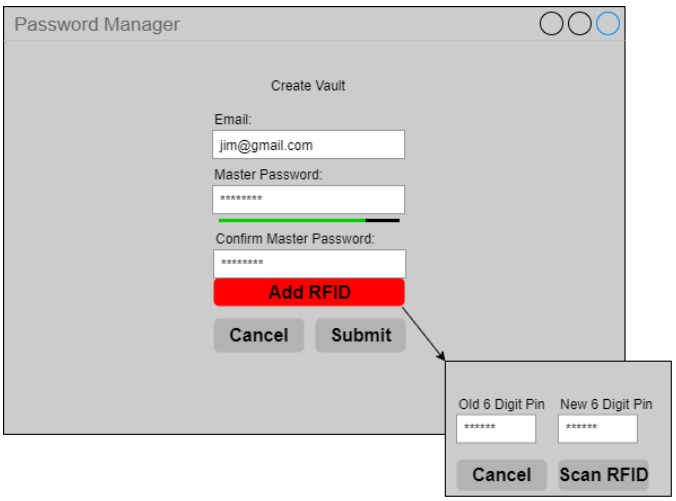


Figure 5.11: Create Account Wireframe Design

Non-Functional requirement REQ1 outlines sub requirements for the create vault page, these are represented in Figure 5.11. Figure 5.11 includes text boxes for the Email, Master Password and Master Password Confirmation. The design fulfils REQ1.4 as there is a bar which indicates a passwords strength. The user will also be able to associate an RFID card using the corresponding button.

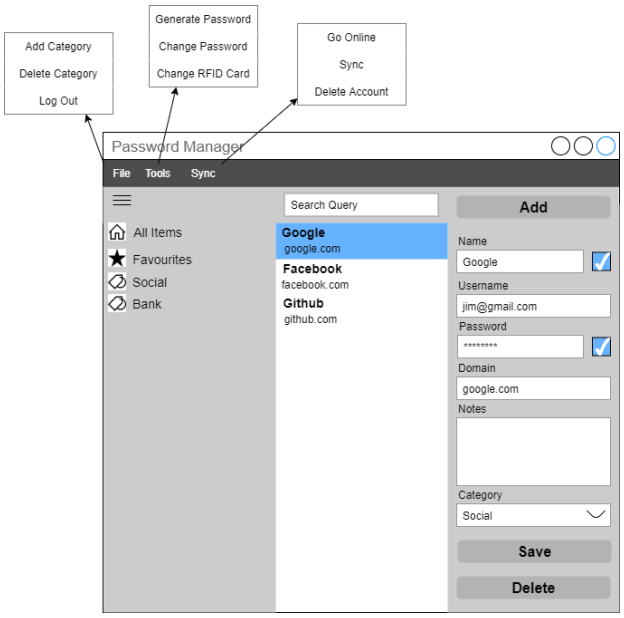


Figure 5.12: Main Page Wireframe Design

Figure 5.12 shows the wireframe design for the main program page. This design accomplishes the requirements set out in REQ3, this includes displaying the credentials saved in the system (REQ3.1), allowing the user to view associated credential passwords (REQ3.2), buttons to add, delete and save credentials (REQ3.3), the ability to associate categories with credentials (REQ3.4) and also a button to favourite a credential (REQ3.5).

Figure 5.12 also meets the requirements outlined in REQ4, this includes offering text boxes to be used for new credentials (RREQ4.1). Requirements REQ5, REQ7, REQ8.1 - REQ8.3 are also fulfilled in Figure 5.12, these can be seen at the top of the figure and are in the form of a menu bar.

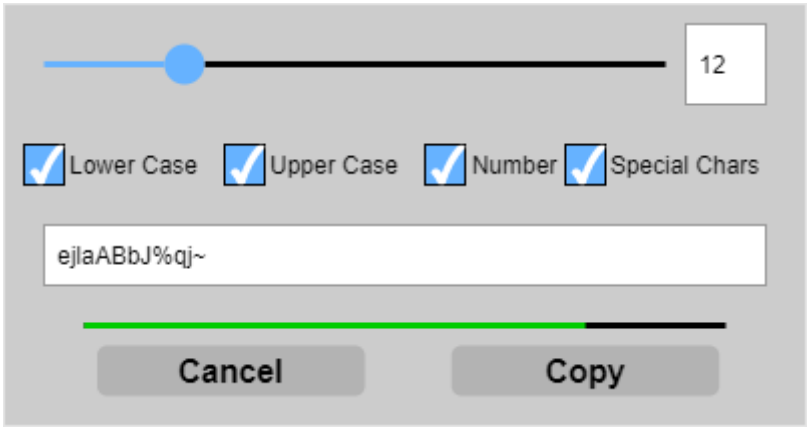


Figure 5.13: Password Generator Wireframe Design

Figure 5.13 shows the design for the password generator, the requirement for this design can be

found at REQ6. It contains option which allow the user to randomly generate a password with different characteristics.

6 Implementation

6.1 GitHub

As mentioned in the literature review, version control is essential to ensure a software project runs efficiently. This project has used GitHub as it has the best features out of all the current options. GitHub has worked well with SCRUM and Kanban, once the tasks have been implemented on the Kanban board, the developer of this project committed the change. If any new features break a part of the code, the change could have easily been rolled back. This worked well when the vault synchronisation feature was being implemented, different prototypes for its implementation were used and having the ability to roll back changes helped immensely.

6.2 API

The technical considerations section discussed that the authentication and vault backup server would be implemented using a .NET Web API. This API used REST with RPC elements to allow a user to authenticate using an SQL Server database and synchronise and retrieve a backup of the user's vault file. The REST architecture outlines a stateless operation, meaning the server does not keep knowledge of the client's state. As this project uses a session Id to ensure a user can only be logged in at one computer at a time, this means the implemented API is not stateless.

6.2.1 Middleware Authentication

.NET API uses middleware that creates a pipeline to handle requests and responses. This middleware decides whether to pass the request to the next component along the pipeline, it can also do some sort of action request before it is passed along (Rick-Anderson, 2020).

This project used this middleware to create an authentication layer in the pipeline. When a request comes in, which requires authentication, the token sent in the header is checked and if it is not valid, the request gets denied. This ensures only authenticated clients can request copies of the encrypted vault or upload copies of the vault to the server. This authentication layer also ensures only one instance of a single user can be authenticated at one time. This prevents concurrent changes occurring to a user's vault file and thus preventing synchronisation issues.

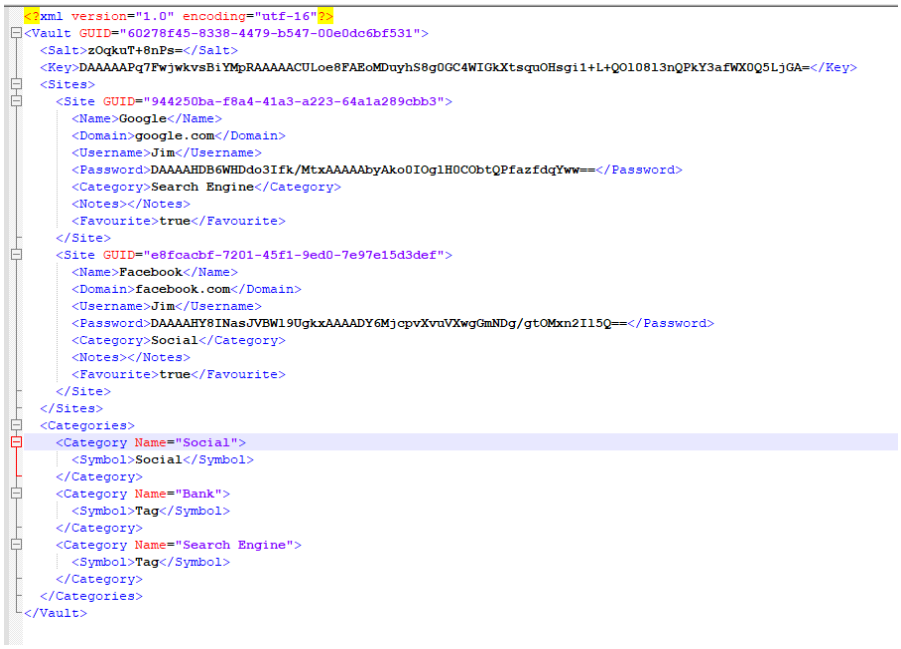
6.3 Data

6.3.1 XML

XML has been used to implement the user's vault; this file stores encrypted copies of the logged-in user's credentials. Before choosing XML, the developer of this project prototyped using SQLite, this ended up being a bad choice for this project as the file created was bloated, only certain elements of the file could be encrypted, and the header of the file could not be accessed. This led to the use of XML, it offers much more freedom as the document is created from scratch by the developer allowing full customization.

The XML vault file stores the users saved credentials, as well as their custom categories. The password associated with a credential is encrypted using AES-GCM, using the primary encryption key, which itself is encrypted and stored in the vault file's header, under the "Key" tag. Along with the Key, a randomly generated salt Is also stored in the header of the vault file, this salt is used to produce the secondary encryption key, which is a hash of the master password and the secret stored on the RFID card.

The whole vault file is then encrypted using a second randomly generated salt, this is stored along with the encrypted data inside the XML file. Figure 6.1 shows an example of the vault file before the whole thing is encrypted using AES-GCM.



```
<?xml version="1.0" encoding="utf-16"?>
<Vault GUID="60278f45-8338-4479-b547-00e0dc6bf531">
  <Salt>zOqkuT+8nPs=</Salt>
  <Key>DAAAAAPq7FwjwkvsBiYMpRAAAAACULoe8FAEoMDuyhS8g0GC4WIGkXtsquOHsgi1+L+QO10813nQPkY3afWX0Q5LjGA=</Key>
  <Sites>
    <Site GUID="944250ba-f8a4-41a3-a223-64a1a289cbb3">
      <Name>Google</Name>
      <Domain>google.com</Domain>
      <Username>Jim</Username>
      <Password>DAAAAHDB6WHDdo3Ifk/MtxAAAAAbyAko0IOg1H0CObtQPfazfdqYww==</Password>
      <Category>Search Engine</Category>
      <Notes></Notes>
      <Favourite>true</Favourite>
    </Site>
    <Site GUID="e8fcacbf-7201-45f1-9ed0-7e97e15d3def">
      <Name>Facebook</Name>
      <Domain>facebook.com</Domain>
      <Username>Jim</Username>
      <Password>DAAAAHY8INasJVBW19UgkxAAAADY6MjcpvXvuVXwgGmNDg/gtOMxn2I15Q==</Password>
      <Category>Social</Category>
      <Notes></Notes>
      <Favourite>true</Favourite>
    </Site>
  </Sites>
  <Categories>
    <Category Name="Social">
      <Symbol>Social</Symbol>
    </Category>
    <Category Name="Bank">
      <Symbol>Tag</Symbol>
    </Category>
    <Category Name="Search Engine">
      <Symbol>Tag</Symbol>
    </Category>
  </Categories>
</Vault>
```


Figure 6.1: Example Vault file before encryption

6.3.2 Database

As mentioned in the report's technical considerations section, this project has implemented a SQL Server database using Code First Entity Framework. Code First Entity Framework allows the developer to define the database tables from C# objects. This saves time producing SQL statements as these are pre-generated by the framework.

This database is used to authenticate users with the API server. It stores the user's email address, session Id and a master password hash, this hash is different from the one used to decrypt the vault file and uses the Users email address as the salt for the hash. This means the server owner cannot access the user's credentials, in turn protecting the credentials if an attacker gained access to the server.

6.4 Client

6.4.1 Secure String

As the project deals with sensitive information such as a user's passwords, the developer of this project initially planned to use Microsoft Secure String. Secure String represents some sensitive text and ensures when the text is no longer needed, it is removed from memory as soon as possible. After researching the use of this within a .NET project, the Microsoft documentation now advises against using it (*dotnet/platform-compat*, 2021). This is because Secure String does not exist as a concept within the Windows operating system. The class does not stop sensitise data appearing in memory and instead reduces the buffer's lifetime in which it is stored.

Microsoft suggests Secure String gives a false sense of security and instead suggests that the data should be treated as unencrypted and dealt with accordingly. To remedy this problem this project instead stores all sensitive user data such as passwords encrypted for as long as possible, the only time passwords are unencrypted is when the user attempts to view them within the GUI. This is unavoidable as there is no other way for the user to view the plain text password.

6.4.2 Hashing and Salting

Initially, the developer of this project planned to use PBKDF2 (Password-Based Key Derivation Function 2) () but after researching more options, found that it is now a little out of date and better options exist. As this project relies on security to protect the user's data, using the most up to date and secure methods is vital.

The project has now been implemented using Argon2i, winner of the Password Hashing Competition (*Password Hashing Competition*, 2021). This hashing algorithm offers very good ASIC and GPU resistance as well as better password cracking resistance when compared to PBKDF2, Bcrypt and Scrypt when using parameters for similar CPU and RAM usage (*P-H-C/phc-winner-argon2*, 2021).

6.4.3 Zero-knowledge Encryption

As mentioned in the technical considerations, ensuring that the project used a zero-knowledge encryption architecture was vital. This means that the server does not have the information available to decrypt a user's vault and view their credential passwords. Producing a secure implementation, allowed server authentication, offline authentication and followed this architecture through many iterations of prototypes and development.

The final implementation uses AES-GCM, this authenticates the encryption and decryption, providing data integrity. Initially, each credential password encrypted would use the hashed master password as the key, this was changed to make it easy to change the user's password or RFID card associated with the account. Using a randomly generated AES key that is then encrypted itself using the master password hash meant only one thing needed re-encrypting if any authentication details were changed.

6.4.4 Memory Leak

Through the development of the project, the project developer noticed that when the client was running for an extended period, the amount of memory it consumed would increase. This is a symptom of a memory leak when the program does not free up memory after it is done. The program JetBrains dotMemory (*dotMemory*, 2021) was used to find what was causing the memory not to free itself.

The memory used by the program would continue to climb every time the navigation pane was used to move between pages within the password manager. The page navigation was implemented using Frames and pages which were displayed within the frame, when the page changed, the frame would be updated. After some research, it became apparent that pages are not disposed when navigating using a Frame within .NET. The problem was fixed by replacing the Frame implementation with a more complex UserControl implementation. The ViewModels are associated with their corresponding UserControl page, when the global ViewModel is set, this triggers the corresponding UserControl to be displayed.

6.4.5 NFC Reader

The project uses an NFC Reader for multi-factor authentication, when designing the system, the developer of the project expected to use a library for communication between the Client, RFID Reader and RFID Card. After doing some research, it became apparent that the libraries currently existing are either out of date or would not work with the hardware being used for this project. This led the developer of this project to implement their own library, which could load custom keys to RFID reader memory, send Application Protocol Data Unit's between the card reader and card, authenticate using keys loaded into reader memory, as well as reading and writing to blocks of data on the cards.

The implementation requests the user to create a 6-digit pin when they create an account, the system then writes a randomly generated secret to a block on the RFID card, encrypting it using the 6-digit pin. The stored secret is concatenated with the master password before being hashed and used as the secondary encryption key.

6.4.6 Auto Log Out

The final solution implemented an auto log-out feature when a user is inactive. It utilises built-in windows functionality to detect when a user is idle, and each second checks the idle time information against the specified threshold for a timeout. This timeout is set to 10 minutes, after this time, the user will be required to log in again.

7 Testing

7.1 Testing Design

Testing is a vital part of the software development life cycle, the main advantage being bugs can be identified early on and fixed before software is released. This leads to lower costs and happier customers as the chance of problems with the software is reduced. Another advantage is that the project objectives and requirements can be achieved with more certainty as they can be proven to be met with acceptance testing (*What is Software Testing? Definition, Basics & Types*, 2021).

The project will use a layered testing approach to reduce the number of bugs which appear in the final solution. The first bugs will be caught by unit testing, these tests are performed in isolation meaning they ensure individual parts of the system work correctly. Integration testing will make up the second layer of testing, these tests ensure separate parts of the system function correctly. The final layer of testing will be made up of acceptance tests.

7.2 Unit Tests

Unit and integration testing has been used throughout the development process; this has ensured bugs were found early. As GitHub Actions (*Features • GitHub Actions*, 2021) in conjunction with the NUnit (*NUnit.org*, 2021) testing framework, has been used throughout the development process, build errors could be caught early. GitHub Actions emails the developer when a build fails, this gives an instant notification and makes sure broken code is not left broken. Figure 7.1 shows an example of one of these notifications.

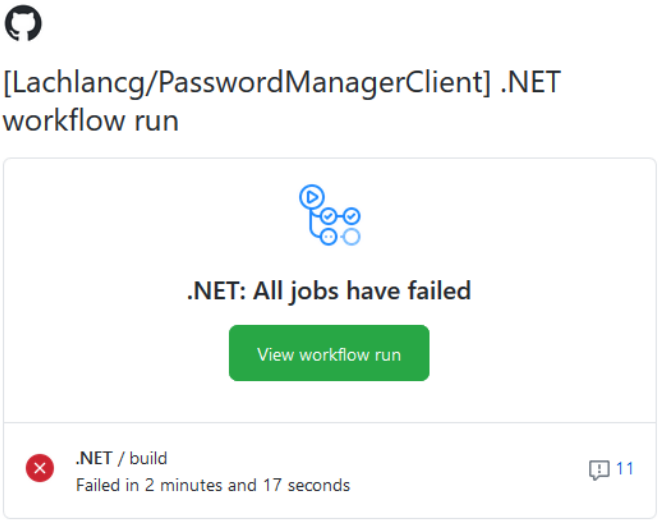


Figure 7.1: Example Email from failed GitHub Build Action

Unit tests can be run manually by the developer within the Visual Studio development environment. These unit tests can also be configured to use different test cases, this repeats the test multiple times, passing different attributes each time. Figure 7.2 shows some of the manual tests along with their results. Figure 7.2 also shows the unit test options, this is where the test has been

set to run after each build. The full list of unit tests can be found in the appendices at appendix 10.9.

Throughout development these tests were expanded as and when units were added to the code, the tests were run after each build and any modules which broke with the addition of features could be easily seen. If these problems were a sizable fix, then the issues would be added to the Kanban board and tackled later in the sprint or in the next sprint.

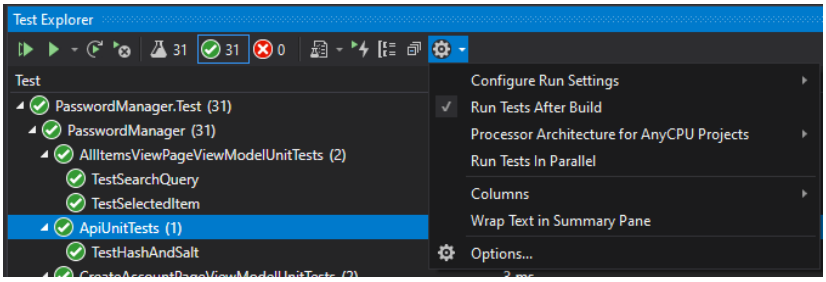


Figure 7.2: Unit Tests and settings menu

The method of using multiple test cases helped to find an issue with the password generator method. This method is given different attributes which effect the password which is generated, such as the length, whether it contains upper and lowercase numbers or if it contains numbers and special characters. The unit test found that even though certain attributes were enabled the output would not always meet all the requirements, this was fixed by checking the generated password met the requirements and if not, it would generate a new one.

7.3 Integration Tests

Integration testing was used to ensure modules continued to function after new features were added or changes were made. Like unit tests these can be manually created within Visual Studio using NUnit and the Microsoft MVC.Testing library, this library lets you create a HTTPClient which can communicate with an in-memory test server. Once the test plan has been created it can be set to run after each build.

The integration tests within the API are important as they make similar requests as the client. This mock client then sends mock requests to the API which accesses the database. If any of the different platforms have a problem communicating it would discover quickly and added to the Kanban board to be dealt with. This helps to manage time and give an accurate representation of how much work is left to complete. Figure 7.3 shows the mock API integration tests and their results.

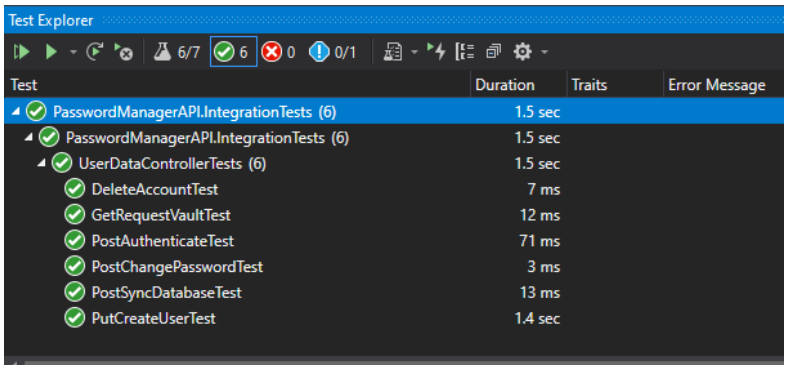


Figure 7.3: Integration Test Output

Integration testing brought a bug which would have been hard to find by just using the system through the GUI. This problem occurred when some of the REST methods checked the session Id timestamp. The logic was incorrect for calculating if the session Id was over 24 hours old, this problem appeared in the unit tests allowing it to be dealt with and fixed quickly.

7.4 Performance Tests

Performance testing was used to determine the maximum number of concurrent users the API server could handle. Apache JMeter (*Apache JMeter - Apache JMeter™*, 2021) was used to perform the tests, it allows the user to create tests plans which communicate with the web service.

JMeter was used to create a test plan with imitates users logging in, first it authenticates with the API and then request the latest copy of the vault. This test was not representative of a production version of the API as it was running on the developer's home computer, but it would help to give an idea of the worst-case scenario load limit.

Appendix 10.11 and 10.12 show the performance results for 1000 – 2000 concurrent users logging in and no errors occurred. Once the load tests reached over 2000 concurrent users logging in the API struggled to keep up and the error rate rose. Appendix 10.14 shows 10,000 users logging in, the authentication requests had nearly 50% error rate and the request database requests had over 95% error rate. The response times followed the same pattern, the more concurrent users the longer the average response time became. 1000 concurrent users saw an average response time of under 10 seconds whereas 2000 saw 20 second response times.

Appendix 10.10 shows the test for a single user, the response time was 23ms, fast enough that a user would not notice any delay. Miller (1968) and Card et al. (1991) said that anything over 1 second means the user would notice the system is slow and anything over 10 seconds and the user will stop paying attention, meaning they would need something on the screen to give feedback. From these results we can say that the current system hardware can comfortably support 1000 users.

7.5 Acceptance Tests

Acceptance testing represent the customers interests and ensures the final product meets the aim, objectives and requirements envisaged by the final user. These tests ensure the system behaves correctly, Miller et al., 2001 suggest that "when all acceptance tests pass the project is done" . Testing of this nature accomplishes three things, they capture user requirements in a verifiable and measurable way, they expose problems that the unit tests misses and they provide a definition for how close the project is to being "done".

Acceptance tests are used to ensure the system has met the functional requirements set out for the project. Figure 7.4 shows a snippet from the acceptance testing table, each functional requirement has been associated with a corresponding test, the expected result is given and then the actual result is recorded. A pass or fail value is then given for each test to indicate the result of the test.

Test No.	REQ No.	Input	Expected Result	Actual Result	Pass/ Fail
1	REQ01, REQ10	Register a new valid account and submit account details.	The user is taken to the login page to use their new account.	Result as expected	P
2	REQ01, REQ10, REQ15.1	Register an account with a low password strength.	The user is warned that their password is insecure.	Result as expected	P
3	REQ01, REQ10, REQ15.2, REQ15.3	Register an account with an invalid email address.	The user is asked to enter a valid email address.	Result as expected	P
4	REQ01, REQ10, REQ15.4	Register an account with non-matching passwords.	The user is asked to enter matching passwords.	Result as expected	P

Figure 7.4: Example acceptance tests (full-page version - Appendix 10.15)

The full table of acceptance tests and their results can be found at appendix 10.15. These tests show that all the tests passed, giving us a string indication that the requirements have been met.

8 Evaluation

8.1 Risk Evaluation

The project risks were planned for during the project's initial stages as part of the project initiation document (PID). This risk analysis can be seen in appendix 10.16 and covered a wide range of possible scenarios, along with mitigation and a method of recovery. The project did suffer from some of the projected risks such as the developer suffering from Covid 19 (*Coronavirus (COVID-19)*, 2020), however due to there being buffer to account for sickness in in the project Gannt chart the delivery was not delayed.

Another risk the project suffered from was computer failure, the developers operating system became corrupted during an update meaning it had to be reinstalled. But as per the PID, regular backups were taken using Google Drive, GitHub, and an external hard drive. This meant that there was zero data loss and the project continued from the backup files. As both risks were planned for, they had little impact on the project deadline, and the project was delivered on time and to specification.

8.2 Project Management Evaluation

An agile approach has been used during the development of this project using the SCRUM framework. This has been a very successful approach as there were significant design changes at the beginning of the project. The most significant change was the decision to use MVVM, meaning some of the code had to be rewritten to allow for this.

The use of the Kanban board has enabled the project to stay Agile, changing the order tasks could be completed has helped to ensure the project has kept on track. 8.1 shows the Kanban board near the end of development.

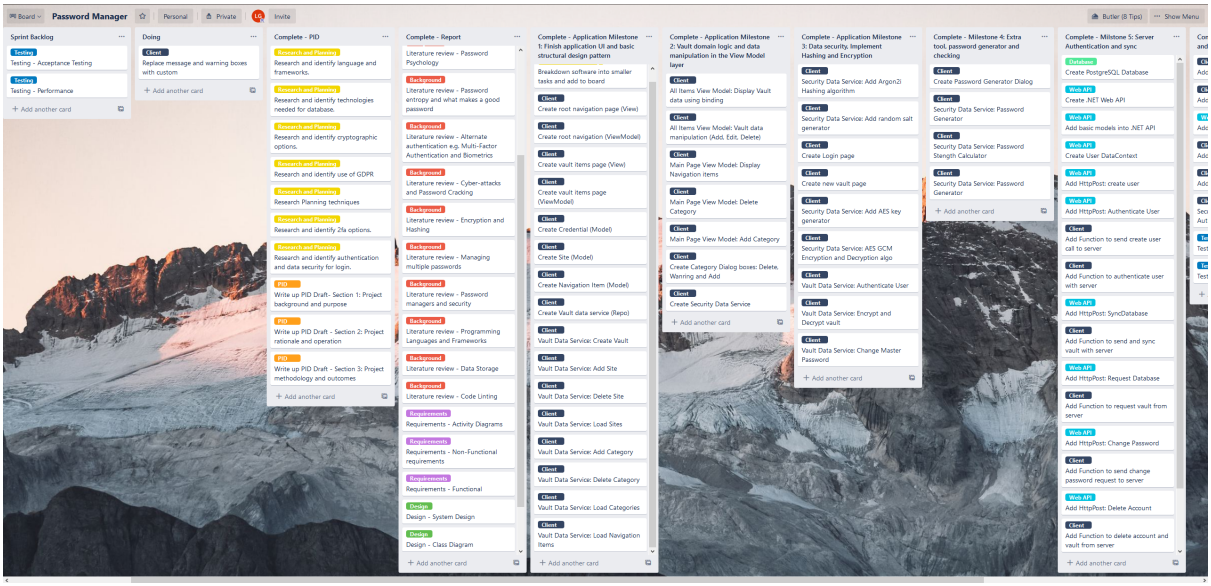


Figure 8.1: Kanban Board

Appendix 10.17 shows the PowerPoint slides from the Mid Project Review (MPR), this presentation

occurred part way through the project and served as an opportunity to discuss the project's progress with the developer's supervisors. Having the Kanban board helped present the MPR as it made showing the current objective progress easy and being able to incorporate advice and feedback from supervisors easily into the project plan to be completed at a later date.

8.3 Ethics Evaluation

This project did not require full ethical approval as it did not involve any external participants. The project could however, contain sensitive user information as the system stores users' passwords. As this is a proof of concept, I chose only to use made-up test data and no rea world data, this means the project did not break the ethical guidelines set out by the university.

8.4 Completed Objective Evaluation

8.4.1 Objective 1 – Windows application with a secure vault

The solution which has been produced meets this objective as the client application runs on windows and has a desktop user interface displaying a vault of credentials. The client also allows a user to create credentials entries that can be added to the vault. These credentials can also be edited and completely removed from the vault file. The passwords are encrypted along with the whole file before being saved to the user's hard drive.

When producing this feature, the main challenge was creating an easy-to-use user interface and login system whilst ensuring that the user's information was sufficient. This involved many design iterations to produce a security model that would allow for server authentication and a vault file that could be sent and stored securely. This meant producing a solution that would be secure even if a hacker had a copy of the vault file, whilst only needing the user to remember one password and the pin for their RFID card. The vault file went through multiple iterations, from a MySQL database file to XML, allowing for much more customization in terms of the file's encrpytion and header.

8.4.2 Objective 2 – Search and Categories

The second objective was also met successfully as the system allows users to search their whole database of credentials. The system also allows a user to add a credential to a category, the system has two default categories, "All Items" and "Favourites" . "All Items" displays all the stored credentials with no filters applied. The "Favourites" filter is unique, each credential has a tick box that allows it to be added to this category, overriding any other category associated with the item.

The system also has a feature to allow for custom categories, the user from the settings menu adds these. They can then associate a credential with a custom category in the credential's creation/ edit page. When a user clicks on the custom filter, only the credentials with this category are displayed, making finding, and filtering easy and straightforward.

This feature also went through multiple iterations of development as the program needed to make it easy to find credentials even if the user had many stored.

8.4.3 Objective 3 – Multi-Factor Authentication

This feature was essential to the project as it is one of the main characteristics which differentiated this project from similar implementations. This objective was satisfied by the user of an RFID card, when a user logs in to the system, they are required to use a password and a physical RFID card.

One aspect of implementing this feature that brought challenges was executing it in such a way that it protected the users' account and the vault file used to secure their credentials. The project ended up using a system of storing a secret on the sectors of the RFID card, this secret is then concatenated with the user's password before being hashed and used as an encryption key.

Due to communication varying between different RFID cards, the system was designed so that adding support for another card wouldn't be too difficult.

8.4.4 Objective 4 – Password checker

A critical aspect of the system is its ability to validate passwords strength when they are used with the system. As the project is heavily focused on the insecurity of passwords, users must be warned of poor security passwords. This objective was met, and the password "meter" was implemented throughout the system, from signing up to when a user uses the password generator.

One challenging aspect of this password checker was getting the balance of the score correct. When a user inputs a password, the checker looks at the string and gives a point for each positive attribute. If it uses a capital letter, then it receives a point, if it uses a number, it would receive another point. This score is then calculated, and the user is informed of the score, this project displays the score on a sliding bar which is also colour coded.

8.4.5 Objective 5 – Password Generator

The fifth objective was successfully implemented into the project, it supplies a tool that allows the user to create a randomly generated password. This tool also allows the user to set different settings which affect the generated string. These settings include numbers, special characters, upper-case and lower-case letters, and the length.

One challenge that was overcome when implementing this feature involved the generated password, not including all the required attributes. This bug was uncovered via unit tests and allowed the developer to improve the algorithm used to generate the passwords.

8.4.6 Secondary Objective 1 – Vault Sync

The final objective that the project achieved was a secondary objective, this objective was to implement a vault backup service. This vault backup service uses an API to communicate with a server, allowing the client to send and receive vault files and authenticate before any other communication happens with the API.

Producing a secure API was a challenge with the original implementation of the vault database. The system used a local SQL database, but this came with lots of overhead and large file sizes that would be unsuitable for a server with many clients. The system was redesigned to use XML for vault storage, this also gave more control over encryption.

The API was designed in such a way so that the domain logic and the data access layers were de coupled. Decoupling these aspects of the API makes the project minitablet as well, making expanding it easier. The API was also made to have loose coupling with the client, making it easy

to swap a different client, such as a website or mobile app, without making any changes to the API.

8.5 Uncompleted Objective Evaluation

8.5.1 Secondary Objective 2 - Peer to Peer credential sync

This secondary objective fell out of scope for this project, this is due to the time constraints that come with a university project. One solution for implementing this feature would involve using RPC (Srinivasan, 1995), two clients would create a secure direct peer to peer connection where the credential could then be shared.

Another method for implementing this solution could involve using an asymmetric encryption algorithm, one client would encrypt the credential using the receivers public key, the receiver could then decrypt the message using their private key.

8.5.2 Secondary Objective 3 – Browser Extension

This secondary objective was also out of the scope of this project due to time constraints. This feature would be a good improvement for a future addition, it would also allow a user to access their vault using any operating system rather than them having to use windows. The feature could also offer an autofill feature, this would have to be implemented with strict rules for identifying websites/applications as there are known phishing attacks that prey on this feature of other password managers.

8.6 Future Improvements

Due to the constraints of the project, what could be accomplished for this project was limited. If this proof-of-concept project was developed into a commercial software piece, these improvements could be made.

8.6.1 Other Credential types

The project could be extended to support more credential types other than just websites/ application. This could include credit/ debit cards, bank accounts and addresses. The software has been written in such a way to make these additions easy to do, and it would be trivial to add this improvement in the future.

8.6.2 Mobile app for RFID

Most mobile phones have support for RFID, a good addition to the system would be to add support to use the phone as the second factor for authentication. This could come in the form of a mobile application that would emulate an RFID card. This would make the users' lives easier as the risk of losing the physical RFID card and then losing access to their vault would be reduced.

The phone would also add another layer of security as a hacker would need to authenticate with the user's phone and physically get a hold of the device.

8.6.3 Website or Mobile app for accessing the vault

Another good future feature would be a mobile app for accessing the vault. This would allow users to access their credentials even if they do not have access to their computer at home or on their laptops. The vault sync feature would allow them to access the same credentials on all their devices.

A website would give similar benefits, users would be able to access their credentials on any devices without the need for the device to be directly supported. A website could also be integrated with the API making it possible to sync the vault file across all platforms.

8.6.4 Amazon/ Azure web hosting of API

If this project was to be made into a commercial application, it might be a good idea to host the database and API using a cloud-based solution such Amazon AWS. Using a cloud-based provider such as Amazon would allow the product to scale without worrying about purchasing server hardware to host the services. This can be more cost-effective and offer better up time than hosting the server at home. Amazon AWS has also been used by other security-conscious companies ("Egis Technology Case Study," n.d.), making it a viable option for future development.

8.6.5 Only allow a subset of the passwords to be viewed on certain devices (such as a laptop)

Another feature that could be developed in the future is the ability only to allow a subset of passwords to be synchronised to specific devices. This means somebody could have a vault with all their credentials on their home computer and then only allow their work laptop to access a subset of these credentials.

This improvement would add quality of life benefit for the user and increase security as credentials such as online banking login details may not need to be shared with a user's laptop or phone.

8.6.6 Audit logs

The final suggested improvement is audit logs, these logs could be added to track every transaction the user makes with the system. This gives a paper trail to see if somebody who is not the user is accessing the system. These event logs could serve as evidence that cybercrime has been committed or simply log faults with the system.

9 Conclusion

The aim of this project was to implement a piece of software to address the problem of poor password etiquette by allowing a user to store well-formulated, randomly generated password strings in an encrypted well-protected software solution. This project looked to find a solution to the inadequacy of a single memorable password for security by forcing users to use two-factor authentication to protect their credentials.

Current password managers only require the user to protect their account with a single form of authentication, this is ill-advised, according to security researchers. This final product has been created to be aimed at security-conscious users who understand the need for multiple forms of authentication. Furthermore, the system has an auto-logout feature, further increasing security against session-based attacks.

The project, on the whole, has been a success, as it supplies a user with a secure method for storing their credentials. Furthermore, it forces the user to use two authentication forms, something that the user knows, which is the password and something that the user owns the RFID card. The project also forces a user to logout after 10 minutes of inactivity

The project also implemented distributed solution allowing a user to synchronise their credentials with across all their devices. This solution uses zero-knowledge encryption, ensuring only the person with the password and RFID card can access the vault file, even when it is sent to the backup server.

All the projects primary objectives were met fully, further proving that the overall project was a success. These were SMART objectives making them measurable, meaning they can be quantifiably measured towards accomplishing the goal. Therefore, the testing undergone during the development process proves these objectives were met.

Only one of the secondary objectives was met due to time constraints, meaning two secondary objectives were unfinished. The one secondary objective that was implemented was implemented to a high standard and again can be measured by its accomplishment of the goal, which is that it is fully working.

The use of SMART measurable objectives also helped with the development process, being able to give a time frame for task made planning with the Gantt chart and Kanban Board much easier. Using SCRUM worked well for this project as there were some unexpected changes throughout the development cycle. SCRUM allowed these changes to be made and changes to the scope of the project to be easily accounted for. In conjunction with the Kanban board, SCRUM ensured the project did not end with a “crunch time” as all tasks were planned accordingly.

Test-driven development has been a critical development pattern used throughout the whole process. It has ensured features and systems have continued to function even after new functionality has been incorporated into the project. The use of linting and automated tests has ensured the code is of high quality and works together correctly.

The next stage in the development process for this project would be to implement the browser extension. This would need to include strict matching criteria so that the system isn’t vulnerable to a common phishing attack.

Even though this project has been created as a proof of concept, the code has been written in a highly maintainable way. This ensures it can be easily expanded to include different credential types as well as other improvements to make it ready for production.

Bibliography

#1 Password Manager & Vault App, Enterprise SSO & MFA | LastPass (2021). URL: https://www.lastpass.com/ (visited on 24/03/2021).

262588213843476 (2021). *Generate a number of random ascii characters in C#*. Gist. URL: https://gist.github.com/khalilovcmd/494988c43b6c205f9d76 (visited on 22/02/2021).

A salt-free diet is bad for your security | 1Password (0). 1Password Blog. Section: 1Password. URL: https://blog.1password.com/a-salt-free-diet-is-bad-for-your-security/ (visited on 11/11/2020).

"ACR122U Application Programming Interface V2.02" (n.d.). In: (), p. 47.

ACR122U USB NFC Reader (2021). Advanced Card Systems Ltd. URL: https://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/ (visited on 16/02/2021).

adegeo (2021). *How to verify that strings are in valid email format*. URL: https://docs.microsoft.com/en-us/dotnet/standard/base-types/how-to-verify-that-strings-are-in-valid-email-format (visited on 13/03/2021).

Aldwairi, Monther et al. (n.d.). "Multi-Factor Authentication System". In: (), p. 8.

Aloul, Fadi et al. (2009). "Two factor authentication using mobile phones". In: *2009 IEEE/ACS International Conference on Computer Systems and Applications*. 2009 IEEE/ACS International Conference on Computer Systems and Applications. Rabat, Morocco: IEEE, pp. 641–644. ISBN: 978-1-4244-3807-5. DOI: 10.1109/AICCSA.2009.5069395. URL: http://ieeexplore.ieee.org/document/5069395/ (visited on 11/11/2020).

Alwen, Joel et al. (Apr. 2017). "Towards Practical Attacks on Argon2i and Balloon Hashing". In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2017 IEEE European Symposium on Security and Privacy (EuroS&P). Paris: IEEE, pp. 142–157. ISBN: 978-1-5090-5762-7. DOI: 10.1109/EuroSP.2017.47. URL: https://ieeexplore.ieee.org/document/7961977/ (visited on 24/03/2021).

Apache JMeter - Apache JMeter™ (2021). URL: https://jmeter.apache.org/ (visited on 04/03/2021).

Apache Subversion (2021). URL: https://subversion.apache.org/ (visited on 02/03/2021).

Aragon, Keef (27th Jan. 2021). *kmaragon/Konscious.Security.Cryptography*. original-date: 2016-06-30T06:32:01Z. URL: https://github.com/kmaragon/Konscious.Security.Cryptography (visited on 10/02/2021).

Argon2 — argon2-cffi 20.1.0 documentation (2021). URL: https://argon2-cffi.readthedocs.io/en/stable/argon2.html (visited on 18/03/2021).

Arthur, Charles (12th July 2012). "Yahoo Voice hack leaks 450,000 passwords". In: *The Guardian*. ISSN: 0261-3077. URL: https://www.theguardian.com/technology/2012/jul/12/yahoo-voice-hack-attack-passwords-stolen (visited on 10/11/2020).

Atlassian (2020). *Kanban - A brief introduction*. Atlassian. URL: https://www.atlassian.com/agile/kanban (visited on 20/10/2020).

– (2021). *What is Continuous Integration*. Atlassian. URL: https://www.atlassian.com/continuous-delivery/continuous-integration (visited on 25/03/2021).

Baig, M. M. et al. (Feb. 2007). "A Robust Technique of Anti Key-Logging using Key-Logging Mechanism". In: *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*. 2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference. ISSN: 2150-4946, pp. 314–318. DOI: 10.1109/DEST.2007.371990.

Bhargav-Spantzel, Abhilasha et al. (23rd July 2007). "Privacy preserving multi-factor authentication with biometrics". In: *Journal of Computer Security* 15.5. Ed. by A. Goto, pp. 529–560. ISSN: 18758924, 0926227X. DOI: 10.3233/JCS-2007-15503. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JCS-2007-15503 (visited on 11/11/2020).

BillWagner (2020). *C# docs - get started, tutorials, reference.* URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (visited on 20/10/2020).

Bitwarden Open Source Password Manager | Bitwarden (2021). URL: https://bitwarden.com/ (visited on 24/03/2021).

BlogsNook (8th May 2019). *MVVM Pattern Advantages – Benefits of Using MVVM Model*. BlogsNook. URL: https://blogsnook.com/mvvm-pattern-advantages/ (visited on 27/02/2021).

Bonneau, J. et al. (May 2012). "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes". In: *2012 IEEE Symposium on Security and Privacy*. 2012 IEEE Symposium on Security and Privacy. ISSN: 2375-1207, pp. 553–567. DOI: 10.1109/SP.2012.44.

Brooks, Richard R. (2013). *Introduction to Computer and Network Security: Navigating Shades of Gray*. London, UNITED STATES: CRC. ISBN: 978-1-4822-1412-3. (Visited on 10/11/2020).

Bruce Schneier (2003). *Beyond fear*. In collab. with Internet Archive. Copernicus Books. 314 pp. ISBN: 978-0-387-02620-6. URL: http://archive.org/details/beyondfearthinki00schn_0 (visited on 18/03/2021).

Brumen, B. et al. (May 2014). "Brute force analysis of PsychoPass-generated Passwords". In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1366–1371. DOI: 10.1109/MIPRO.2014.6859780.

Burgess, Matt (24th May 2016). "How to check if your LinkedIn account was hacked". In: *Wired UK*. Section: Technology. ISSN: 1357-0978. URL: https://www.wired.co.uk/article/linkedin-data-breach-find-out-included (visited on 10/11/2020).

c# - How can I convert a hex string to a byte array? (2021). Stack Overflow. URL: https://stackoverflow.com/questions/321370/how-can-i-convert-a-hex-string-to-a-byte-array (visited on 03/03/2021).

c# - How can I make a TextBox be a "password box" and display stars when using MVVM? (2021). Stack Overflow. URL: https://stackoverflow.com/questions/1119605/how-can-i-make-a-textbox-be-a-password-box-and-display-stars-when-using-mvvm (visited on 01/02/2021).

c# - Unit testing the Viewmodel (2021). Stack Overflow. URL: https://stackoverflow.com/questions/4845332/unit-testing-the-viewmodel (visited on 03/03/2021).

c# - Using the AesGcm class (2021). Stack Overflow. URL: https://stackoverflow.com/questions/60889345/using-the-aesgcm-class (visited on 24/01/2021).

c# - WPF inactivity and activity (2021). Stack Overflow. URL: https://stackoverflow.com/questions/4963135/wpf-inactivity-and-activity (visited on 25/01/2021).

Card, Stuart K. et al. (1st Mar. 1991). "The information visualizer, an information workspace". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '91. New York, NY, USA: Association for Computing Machinery, pp. 181–186. ISBN: 978-0-89791-383-6. DOI: 10.1145/108844.108874. URL: https://doi.org/10.1145/108844.108874 (visited on 04/03/2021).

Chang, C. -C et al. (1991). "Remote password authentication with smart cards". In: *IEE proceedings. Part E, Computers and digital techniques* 138.3. Publisher: Institution of Electrical Engineers (IEE) (UK), p. 165. ISSN: 0143-7062. DOI: 10.1049/ip-e.1991.0022.

Client Server Architecture - CIO Wiki (2021). URL: https://cio-wiki.org/wiki/Client_Server_Architecture (visited on 07/04/2021).

Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3) (2021). Amazon Web Services, Inc. URL: https://aws.amazon.com/s3/ (visited on 22/02/2021).

Common Language Runtime (CLR) overview - .NET | Microsoft Docs (2021). URL: https://docs.microsoft.com/en-us/dotnet/standard/clr (visited on 01/03/2021).

Computer Security Division, Information Technology Laboratory (4th Jan. 2017). *Block Cipher Modes - Block Cipher Techniques | CSRC | CSRC*. CSRC | NIST. URL: https://csrc.nist.gov/projects/block-cipher-techniques/bcm (visited on 24/03/2021).

Computing Machinery, Association for et al., eds. (2010). *2010 ACM/IEEE 32nd International Conference on Software Engineering (ICSE 2010): Cape Town, South Africa, 2 - 8 May 2010*. Meeting Name: ACM/IEEE International Conference on Software Engineering. Piscataway, NJ: IEEE. ISBN: 978-1-60558-719-6.

Coronavirus (COVID-19) (2nd June 2020). nhs.uk. URL: https://www.nhs.uk/conditions/coronavirus-covid-19/ (visited on 25/03/2021).

cplusplus.com - The C++ Resources Network (2020). URL: https://www.cplusplus.com/ (visited on 01/11/2020).

Das, Anupam et al. (2014). "The Tangled Web of Password Reuse". In: *Proceedings 2014 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium. San Diego, CA: Internet Society. ISBN: 978-1-891562-35-8. DOI: 10.14722/ndss.2014.23357. URL: https://www.ndss-symposium.org/ndss2014/programme/tangled-web-password-reuse/ (visited on 10/11/2020).

David Silver et al., eds. (2014). *Password Managers: Attacks and Defenses*. Meeting Name: Large Installation Systems Administration Conference OCLC: 254320948. Berkeley, Calif: USENIX Association. 242 pp. ISBN: 978-1-931971-15-7.

davidbritch (2021). *The Model-View-ViewModel Pattern - Xamarin*. URL: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm (visited on 15/02/2021).

Dhillon, Gurpreet et al. (July 2000). "Technical opinion: Information system security management in the new millennium". In: *Communications of the ACM* 43.7, pp. 125–128. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/341852.341877. URL: https://dl.acm.org/doi/10.1145/341852.341877 (visited on 09/11/2020).

Dialogs In WPF (MVVM) (2021). URL: https://www.c-sharpcorner.com/article/dialogs-in-wpf-mvvm/ (visited on 01/02/2021).

Dibble, Christopher et al. (1st Oct. 2014). "Refactoring code to increase readability and maintainability: a case study". In: *Journal of Computing Sciences in Colleges* 30.1, pp. 41–51. ISSN: 1937-4771.

dotMemory (2021). *dotMemory: a Memory Profiler & Unit-Testing Framework for .NET by JetBrains*. JetBrains. URL: https://www.jetbrains.com/dotmemory/ (visited on 04/03/2021).

dotnet-bot (2021). *AesGcm Class (System.Security.Cryptography)*. URL: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aesgcm (visited on 10/02/2021).

dotnet/platform-compat (2021). GitHub. URL: https://github.com/dotnet/platform-compat (visited on 04/03/2021).

Dropbox (2021). URL: https://www.dropbox.com/en_GB/ (visited on 24/03/2021).

ECMA-334 (2021). Ecma International. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-334/ (visited on 01/03/2021).

Elminaam, D S Abdul et al. (2009).
"Performance Evaluation of Symmetric Encryption Algorithms".
In: *Communications of the IBIMA* 8, p. 7.

Enterprise Security Model | LastPass (2020).
URL: https://www.lastpass.com/enterprise/security (visited on 12/11/2020).

Exploitbytes (24th Mar. 2020).
What Is Password Cracking? And Types Of Attacks? — ExploitByte. Medium.
URL: https://medium.com/@shrinathdhormare/what-is-password-cracking-and-types-of-attacks-exploitbyte-e984876be6ad (visited on 07/12/2020).

Extensible Markup Language (XML) (2021).
URL: https://www.w3.org/XML/ (visited on 02/03/2021).

Facebook et al. (18th May 2016). *117M LinkedIn Passwords Leaked*. PCMag UK.
Section: Encryption.
URL: https://uk.pcmag.com/encryption/77436/117m-linkedin-passwords-leaked (visited on 10/11/2020).

Features • GitHub Actions (2021). GitHub.
URL: https://github.com/features/actions (visited on 04/03/2021).

Features • GitHub Actions (2021). GitHub.
URL: https://github.com/features/actions (visited on 25/03/2021).

Fielding Dissertation: CHAPTER 2: Network-based Application Architectures (2021).
URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/net_app_arch.htm (visited on 02/02/2021).

Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST) (2021).
URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (visited on 02/02/2021).

Florencio, Dinei et al. (2007). "A large-scale study of web password habits".
In: *Proceedings of the 16th international conference on World Wide Web - WWW '07*.
the 16th international conference. Banff, Alberta, Canada: ACM Press, p. 657.
ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242661. URL:
http://portal.acm.org/citation.cfm?doid=1242572.1242661 (visited on 12/11/2020).

Flowchart Maker & Online Diagram Software (2021).
URL: https://app.diagrams.net/ (visited on 22/02/2021).

Fredrich, Todd (n.d.). "RESTful Service Best Practices". In: (), p. 40.

Functional vs Non Functional Requirements (28th Apr. 2020). GeeksforGeeks.
Section: Software Engineering.
URL: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (visited on 07/01/2021).

Grassi, Paul A et al. (22nd June 2017). *Digital identity guidelines: revision 3*. NIST SP 800-63-3.
Gaithersburg, MD: National Institute of Standards and Technology, NIST SP 800–63–3.
DOI: 10.6028/NIST.SP.800-63-3.
URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf (visited on 10/11/2020).

Grawemeyer, Beate et al. (May 2011).
"Using and managing multiple passwords: A week to a view".
In: *Interacting with Computers* 23.3, pp. 256–267. ISSN: 09535438.
DOI: 10.1016/j.intcom.2011.03.007. URL:
https://academic.oup.com/iwc/article-lookup/doi/10.1016/j.intcom.2011.03.007
(visited on 12/11/2020).

Guide to app architecture (2021). Android Developers.
URL: https://developer.android.com/jetpack/guide (visited on 15/02/2021).

Güneysu, T. et al. (Nov. 2008). "Cryptanalysis with COPACOBANA". In: *IEEE Transactions on Computers* 57.11. Conference Name: IEEE Transactions on Computers, pp. 1498–1513.
ISSN: 1557-9956. DOI: 10.1109/TC.2008.80.

Guo, Y. Guo {and} Z. Zhang {and} Y. (July 2020). *Optiwords: A new password policy for creating memorable and strong passwords | EndNote Click*.
URL: https://kopernio.com/viewer?doi=10.1016%2Fj.cose.2019.05.015&token=
WzI3MDg5MjQsIjEwLjEwMTYvai5jb3NlLjIwMTkuMDUuMDE1Il0.IvMf0MkeasvfQ9DGXq6QdPSD_HE
(visited on 10/11/2020).

Gupta, Piyush et al. (2014). "A Comparative Analysis of SHA and MD5 Algorithm". In: 5, p. 4.

Hacker Tries To Sell 427 Milllion Stolen MySpace Passwords For $2,800 (2020).
URL: https://www.vice.com/en/article/pgkk8v/427-million-myspace-passwords-
emails-data-breach (visited on 10/11/2020).

"Hackers Publish German Minister's Fingerprint" (2020). In: *Wired* (). ISSN: 1059-1028.
URL: https://www.wired.com/2008/03/hackers-publish/ (visited on 10/11/2020).

Hamilton, David (2020). *Yahoo's password leak: What you need to know (FAQ)*. CNET.
URL: https://www.cnet.com/news/yahoos-password-leak-what-you-need-to-know-faq/
(visited on 10/11/2020).

Hansson, Daniel (Dec. 2014). "Continuous Linting with Automatic Debug".
In: *2014 15th International Microprocessor Test and Verification Workshop*.
2014 15th International Microprocessor Test and Verification Workshop (MTV).
Austin, TX, USA: IEEE, pp. 70–72. ISBN: 978-1-4673-6858-2. DOI: 10.1109/MTV.2014.25.
URL: http://ieeexplore.ieee.org/document/7087237/ (visited on 02/03/2021).

How Device Usage Changed in 2018 and What it Means for 2019 (20th Nov. 2018). GWI. URL:
https://blog.globalwebindex.com/trends/device-usage-2019/ (visited on 24/03/2021).

How to choose an Authenticated Encryption mode (19th May 2012).
A Few Thoughts on Cryptographic Engineering.
URL: https://blog.cryptographyengineering.com/2012/05/19/how-to-choose-
authenticated-encryption/ (visited on 10/02/2021).

How to Create a Strong Password and Beat the Hackers | Avast (2020).
URL: https://blog.avast.com/strong-password-ideas (visited on 01/11/2020).

index | TIOBE - The Software Quality Company (2021).
URL: https://www.tiobe.com/tiobe-index/ (visited on 01/03/2021).

International Organisation for Standardisation (2013). *ISO/IEC 18092:2013*.
URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/
standard/05/66/56692.html (visited on 16/02/2021).

International Organisation for Standardisation (Feb. 2018). *ISO/IEC 27000:2018*.
URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/39/73906.html (visited on 09/11/2020).

Intro to Distributed Version Control (Illustrated) – BetterExplained (2021).
URL: https://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/ (visited on 02/03/2021).

Introduction to Model/View/ViewModel pattern for building WPF apps | Microsoft Docs (2021).
URL: https://docs.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps (visited on 26/02/2021).

Ives, Blake et al. (Apr. 2004). "The domino effect of password reuse".
In: *Communications of the ACM* 47.4, pp. 75–78. ISSN: 0001-0782, 1557-7317.
DOI: 10.1145/975817.975820.
URL: https://dl.acm.org/doi/10.1145/975817.975820 (visited on 12/11/2020).

ixy-languages/ixy-languages (14th Feb. 2021). original-date: 2018-10-14T15:06:00Z.
URL: https://github.com/ixy-languages/ixy-languages (visited on 15/02/2021).

Jain, A.K. et al. (June 2006). "Biometrics: A Tool for Information Security".
In: *IEEE Transactions on Information Forensics and Security* 1.2, pp. 125–143.
ISSN: 1556-6013. DOI: 10.1109/TIFS.2006.873653.
URL: http://ieeexplore.ieee.org/document/1634356/ (visited on 11/11/2020).

Java Downloads for All Operating Systems (2021).
URL: https://www.java.com/en/download/manual.jsp (visited on 01/03/2021).

Java SE - Downloads | Oracle Technology Network | Oracle United Kingdom (2020).
URL: https://www.oracle.com/uk/java/technologies/javase-downloads.html (visited on 01/11/2020).

Jayasekara, Irantha (2021a). *ACR122u NFC Card reader application by Irantha Jayasekara.*
URL: https://iranthajayasekara.com/blog/nfc-card-reading-system-with-acr122u-reader.html (visited on 06/02/2021).

– (2021b). *Read and Write to NFC Card with ACR122u Reader in C# by Irantha Jayasekara.*
URL: https://iranthajayasekara.com/blog/read-and-write-to-nfc-card-with-acr122u-reader.html (visited on 06/02/2021).

Jones, M. Khonji {and} Y. Iraqi {and} A. (2013).
Phishing Detection: A Literature Survey | EndNote Click.
URL: https://kopernio.com/viewer?doi=10.1109%2Fsurv.2013.032213.00009&token=WzI3MDg5MjQsIjEwLjExMDkvc3Vydi4yMDEzLjAzMjIxMy4wMDAwOSJd.x1ymW5Xk6GmSXwwnWiOqhOLoxMY (visited on 11/11/2020).

jwmsft (2021). *Introduction to Windows app design (Windows apps) - UWP applications.* URL: https://docs.microsoft.com/en-us/windows/uwp/design/basics/design-and-ui-intro (visited on 18/03/2021).

Kakkar, Ajay et al. (2012). "Comparison of Various Encryption Algorithms and Techniques for Secured Data Communication in Multinode Network".
In: *International Journal of Engineering and Technology* 2.1, p. 6.

Kasper, Timo (n.d.). "Cloning Cryptographic RFID Cards for 25$". In: (), p. 15.

Kazuhide Fujita et al. (Sept. 2008).
"A study of password authentication method against observing attacks".
In: *2008 6th International Symposium on Intelligent Systems and Informatics*.
2008 6th International Symposium on Intelligent Systems and Informatics. ISSN: 1949-0488, pp. 1–6. DOI: 10.1109/SISY.2008.4664927.

Klein, Daniel V (1992).
"'Foiling the Cracker'': A Survey of, and Improvements to, Password Security". In: p. 11.

Komanduri, Saranga et al. (7th May 2011).
"Of passwords and people: measuring the effect of password-composition policies".
In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11.
New York, NY, USA: Association for Computing Machinery, pp. 2595–2604.
ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979321.
URL: https://doi.org/10.1145/1978942.1979321 (visited on 10/11/2020).

Kontaxis, Georgios et al. (Oct. 2012).
"Minimizing information disclosure to third parties in social login platforms".
In: *International Journal of Information Security* 11.5, pp. 321–332.
ISSN: 1615-5262, 1615-5270. DOI: 10.1007/s10207-012-0173-6.
URL: http://link.springer.com/10.1007/s10207-012-0173-6 (visited on 08/01/2021).

Ladakis, Evangelos et al. (n.d.).
"You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger". In: (), p. 6.

Lakshmanan, Ravi et al. (1st Jan. 2015). "Integrated Multi-Stage Biometric System Design".
In: *International Journal of Applied Engineering Research* 10, pp. 9611–9629.

LastPass Security Notification (5th May 2011). The LastPass Blog.
URL: http://blog.lastpass.com/2011/05/lastpass-security-notification/ (visited on 11/11/2020).

Latcu, Ovidiu (24th Feb. 2020). *Android Repository Pattern using RX & Room*. Medium.
URL: https://medium.com/corebuild-software/android-repository-pattern-using-rx-room-bac6c65d7385 (visited on 15/02/2021).

Lemsitzer, Stefan et al. (2007). "Multi-gigabit GCM-AES Architecture Optimized for FPGAs".
In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier et al.
Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 227–238.
ISBN: 978-3-540-74735-2. DOI: 10.1007/978-3-540-74735-2_16.

Li, N. et al. (Dec. 2015). "Realizing High-Speed PBKDF2 Based on FPGA".
In: *2015 International Conference on Intelligent Transportation, Big Data and Smart City*.
2015 International Conference on Intelligent Transportation, Big Data and Smart City, pp. 580–583. DOI: 10.1109/ICITBS.2015.148.

Li, Zhiwei et al. (7th July 2014).
The Emperor's New Password Manager: Security Analysis of Web-based Password Managers:
Fort Belvoir, VA: Defense Technical Information Center. DOI: 10.21236/ADA614474.
URL: http://www.dtic.mil/docs/citations/ADA614474 (visited on 12/11/2020).

LinkedIn Lost 167 Million Account Credentials in Data Breach (2020). Fortune.
URL: https://fortune.com/2016/05/18/linkedin-data-breach-email-password/ (visited on 10/11/2020).

Liou, Jing-Chiou et al. (Apr. 2011).
"A Sophisticated RFID Application on Multi-Factor Authentication".
In: *2011 Eighth International Conference on Information Technology: New Generations*.
2011 Eighth International Conference on Information Technology: New Generations (ITNG).
Las Vegas, NV, USA: IEEE, pp. 180–185. ISBN: 978-1-61284-427-5.
DOI: 10.1109/ITNG.2011.38.
URL: http://ieeexplore.ieee.org/document/5945229/ (visited on 11/11/2020).

Lith, Adam et al. (n.d.). "A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data". In: (), p. 71.

Liu, Wenyi et al. (Apr. 2014).
"MACA: A privacy-preserving multi-factor cloud authentication system utilizing big data".
In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*.
IEEE INFOCOM 2014 - IEEE Conference on Computer Communications Workshops
(INFOCOM WKSHPS). Toronto, ON, Canada: IEEE, pp. 518–523. ISBN: 978-1-4799-3088-3.
DOI: 10.1109/INFCOMW.2014.6849285.
URL: http://ieeexplore.ieee.org/document/6849285/ (visited on 11/11/2020).

Mandal, A. K. et al. (Mar. 2012).
"Performance evaluation of cryptographic algorithms: DES and AES".
In: *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*.
2012 IEEE Students' Conference on Electrical, Electronics and Computer Science, pp. 1–5.
DOI: 10.1109/SCEECS.2012.6184991.

"MIFARE Classic EV1 1K - Mainstream contactless smart card IC for fast and easy solution development" (2018). In: 2018, p. 36.

Miller, Roy W et al. (2001). "Acceptance Testing". In: p. 7.

Moriarty, Kathleen et al. (2017).
PKCS #5: Password-Based Cryptography Specification Version 2.1.
URL: https://tools.ietf.org/html/rfc8018 (visited on 11/11/2020).

Most hacked passwords revealed as UK cyber survey exposes gaps in online security (21st Apr. 2019). URL: https://www.ncsc.gov.uk/news/most-hacked-passwords-revealed-as-uk-cyber-survey-exposes-gaps-in-online-security (visited on 10/11/2020).

Most hacked passwords revealed as UK cyber survey exposes gaps in online security (2020).
URL: https://www.ncsc.gov.uk/news/most-hacked-passwords-revealed-as-uk-cyber-survey-exposes-gaps-in-online-security (visited on 01/11/2020).

mSecure Password Manager and Digital Wallet (2021).
URL: https://www.msecure.com/ (visited on 24/03/2021).

Muehlen, Michael zur et al. (July 2005).
"Developing web services choreography standards—the case of REST vs. SOAP".
In: *Decision Support Systems* 40.1, pp. 9–29. ISSN: 01679236.
DOI: 10.1016/j.dss.2004.04.008.
URL: https://linkinghub.elsevier.com/retrieve/pii/S0167923604000612 (visited on 27/02/2021).

MySQL (2021). URL: https://www.mysql.com/ (visited on 02/03/2021).

Newman, Lily Hay (31st May 2016).
Myspace (Which Still Exists) Suffers Major Data Breach. Slate Magazine.
URL: https://slate.com/technology/2016/05/myspace-leaks-hundreds-of-millions-of-login-credentials.html (visited on 10/11/2020).

nishanil (2021). *Designing the infrastructure persistence layer*. URL: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design (visited on 18/03/2021).

NUnit.org (2021). URL: https://nunit.org/ (visited on 04/03/2021).

Oechslin, Philippe (2003). "Making a Faster Cryptanalytic Time-Memory Trade-Off".
In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Red. by Gerhard Goos et al. Vol. 2729. Series Title: Lecture Notes in Computer Science.
Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 617–630.
ISBN: 978-3-540-40674-7 978-3-540-45146-4. DOI: 10.1007/978-3-540-45146-4_36.
URL: http://link.springer.com/10.1007/978-3-540-45146-4_36 (visited on 11/11/2020).

P-H-C/phc-winner-argon2 (2021). GitHub.
URL: https://github.com/P-H-C/phc-winner-argon2 (visited on 24/03/2021).

Pal, Saibal K. et al. (Mar. 2009). "A New Cryptographic Hash Function based on Latin Squares and Non-linear Transformations".
In: *2009 IEEE International Advance Computing Conference*.
2009 IEEE International Advance Computing Conference (IACC 2009). Patiala, India: IEEE, pp. 862–867. ISBN: 978-1-4244-2927-1. DOI: 10.1109/IADCC.2009.4809128.
URL: http://ieeexplore.ieee.org/document/4809128/ (visited on 11/11/2020).

Password Hashing Competition (2021).
URL: https://www.password-hashing.net/ (visited on 04/03/2021).

Password Hashing Competition (2021).
URL: https://www.password-hashing.net/ (visited on 24/03/2021).

Password Manager App for Home, Mobile, Business | Dashlane (2021).
URL: https://www.dashlane.com/ (visited on 24/03/2021).

PostgreSQL: The world's most advanced open source database (2021).
URL: https://www.postgresql.org/ (visited on 02/03/2021).

Raza, Mudassar et al. (2012). "A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication". In: p. 6.

ReSharper (2021).
ReSharper: The Visual Studio Extension for .NET Developers by JetBrains. JetBrains.
URL: https://www.jetbrains.com/resharper/ (visited on 02/03/2021).

Rick-Anderson (2020). *ASP.NET Core Middleware*.
URL: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/ (visited on 25/03/2021).

robvet (2021). *Relational vs. NoSQL data*.
URL: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data (visited on 02/03/2021).

Saltzer, J.H. et al. (1975). "The protection of information in computer systems". In: *Proceedings of the IEEE* 63.9, pp. 1278–1308. ISSN: 0018-9219. DOI: 10.1109/PROC.1975.9939. URL: http://ieeexplore.ieee.org/document/1451869/ (visited on 09/11/2020).

Samonas, Spyridon et al. (2020). "THE CIA STRIKES BACK: REDEFINING CONFIDENTIALITY, INTEGRITY AND AVAILABILITY IN SECURITY". In: p. 25.

Sander, Tomas et al. (2002). "Securing Passwords Against Dictionary Attacks". In: p. 10.

Schneier, Bruce (1st Apr. 2005). "Two-factor authentication: too little, too late". In: *Communications of the ACM* 48.4, p. 136. ISSN: 0001-0782. DOI: 10.1145/1053291.1053327. URL: https://doi.org/10.1145/1053291.1053327 (visited on 11/11/2020).

Schroeder, Stan (2020). *You can now browse through 427 million stolen MySpace passwords*. Mashable. URL: https://mashable.com/2016/07/01/myspace-password-database/ (visited on 10/11/2020).

Security flaws found in popular password managers (19th Mar. 2020). WeLiveSecurity. Section: Password. URL: https://www.welivesecurity.com/2020/03/19/security-flaws-found-in-popular-password-managers/ (visited on 24/03/2021).

Shannon, C. E. (1964). *A Mathematical Theory of Communication | EndNote Click*. URL: https://kopernio.com/viewer?doi=10.1002%2Fj.1538-7305.1948.tb01338.x&token=WzI3MDg5MjQsIjEwLjEwMDIvai4xNTM4LTczMDUuMTk0OC50YjAxMzM4LngiXQ.Qt5Uioox_-1DQ8t7kdP-OKyENm4 (visited on 10/11/2020).

Shay, Richard and Elisa Bertino (Aug. 2009). "A comprehensive simulation tool for the analysis of password policies". In: *International Journal of Information Security* 8.4, pp. 275–289. ISSN: 1615-5262, 1615-5270. DOI: 10.1007/s10207-009-0084-3. URL: http://link.springer.com/10.1007/s10207-009-0084-3 (visited on 12/11/2020).

Shay, Richard, Abhilasha Bhargav-Spantzel et al. (1st Jan. 2007). *Password policy simulation and analysis*. Journal Abbreviation: DIM'07 - Proceedings of the 2007 ACM Workshop on Digital Identity Management Pages: 10 Publication Title: DIM'07 - Proceedings of the 2007 ACM Workshop on Digital Identity Management. 1 p. DOI: 10.1145/1314403.1314405.

Shay, Richard, Saranga Komanduri et al. (2010). "Encountering Stronger Password Requirements: User Attitudes and Behaviors". In: p. 20.

Simmonds, Andrew et al. (2004). "An Ontology for Network Security Attacks". In: *Applied Computing*. Ed. by Suresh Manandhar et al. Red. by David Hutchison et al. Vol. 3285. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 317–323. ISBN: 978-3-540-23659-7 978-3-540-30176-9. DOI: 10.1007/978-3-540-30176-9_41. URL: http://link.springer.com/10.1007/978-3-540-30176-9_41 (visited on 09/11/2020).

Singh, Mr Gurjeevan et al. (2011). "CRYPTOGRAPHY ALGORITHM COMPARISON FOR SECURITY ENHANCEMENT IN WIRELESS INTRUSION DETECTION SYSTEM". In: 4, p. 9.

Singh, Rajesh Kumar et al. (2009). "Secure Web Based Single Sign-On (SSO) Framework Using Identity Based Encryption System". In: *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. 2009 International Conference on Advances in Recent Technologies in Communication and Computing. Kottayam, Kerala, India: IEEE, pp. 430–432. ISBN: 978-1-4244-5104-3. DOI: 10.1109/ARTCom.2009.82. URL: http://ieeexplore.ieee.org/document/5329361/ (visited on 08/01/2021).

Smith, Richard E. (2015). *Elementary Information Security: With Navigate Premier Package*. Sudbury, UNITED STATES: Jones & Bartlett Learning, LLC. ISBN: 978-1-284-05594-8. (Visited on 10/11/2020).

Spandel, Daniel et al. (n.d.). "Choosing between Git and Subversion". In: (), p. 39.

SQL Server 2019 | Microsoft (2021). URL: https://www.microsoft.com/en-gb/sql-server/sql-server-2019 (visited on 02/03/2021).

SQLite Home Page (2021). URL: https://www.sqlite.org/index.html (visited on 02/03/2021).

Stallings, William (2006). *Cryptography and network security: principles and practice*. 4th. Book, Whole. Upper Saddle River, N.J: Pearson/Prentice Hall. ISBN: 0131873164;9780131873162;

– (2011). *Network security essentials: applications and standards*. 4th ed. OCLC: ocn504276173. Boston: Prentice Hall. 417 pp. ISBN: 978-0-13-610805-4.

– (24th Oct. 2014). *Computer Security: Principles and Practice, Global Edition*. Pearson Education. ISBN: 978-1-292-06620-2.

Steves, Michelle et al. (Feb. 2014). *Report: Authentication Diary Study*. NIST IR 7983. National Institute of Standards and Technology, NIST IR 7983. DOI: 10.6028/NIST.IR.7983. URL: https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7983.pdf (visited on 12/11/2020).

Stroustrup, Bjarne (1st Jan. 1996). "A history of C++: 1979–1991". In: *History of programming languages—II*. New York, NY, USA: Association for Computing Machinery, pp. 699–769. ISBN: 978-0-201-89502-5. URL: https://doi.org/10.1145/234286.1057836 (visited on 01/03/2021).

Sumathi, S. et al. (2007). *Fundamentals of Relational Database Management Systems*. Red. by Janusz Kacprzyk. Vol. 47. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-48397-7 978-3-540-48399-1. DOI: 10.1007/978-3-540-48399-1. URL: http://link.springer.com/10.1007/978-3-540-48399-1 (visited on 02/03/2021).

Sun, San-Tsai et al. (n.d.). "The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems". In: (), p. 13.

Swinhoe, Dan (17th Apr. 2020). *The 15 biggest data breaches of the 21st century*. CSO Online. URL: https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html (visited on 11/11/2020).

Taha, M. M. et al. (Aug. 2013).
"On password strength measurements: Password entropy and password quality".
In: *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*. 2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE), pp. 497–501. DOI: 10.1109/ICCEEE.2013.6633989.

Tam, L. et al. (May 2010).
"The psychology of password management: a tradeoff between security and convenience".
In: *Behaviour & Information Technology* 29.3, pp. 233–244. ISSN: 0144-929X, 1362-3001.
DOI: 10.1080/01449290903121386.
URL: http://www.tandfonline.com/doi/abs/10.1080/01449290903121386 (visited on 10/11/2020).

Technology, National Institute of Standards and (26th Nov. 2001).
Advanced Encryption Standard (AES). Federal Information Processing Standard (FIPS) 197. U.S. Department of Commerce. DOI: https://doi.org/10.6028/NIST.FIPS.197. URL: https://csrc.nist.gov/publications/detail/fips/197/final (visited on 11/11/2020).

TerryGLee (2021). *What is WPF? - Visual Studio*. URL: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf (visited on 18/03/2021).

The Importance of Scalability In Software Design Software Engineering (2021). URL: https://www.conceptatech.com/blog/importance-of-scalability-in-software-design (visited on 22/02/2021).

The Mooltipass Hardware Password Keeper (2021).
URL: https://www.themooltipass.com/ (visited on 08/01/2021).

The most popular database for modern apps (2020). MongoDB.
URL: https://www.mongodb.com (visited on 01/11/2020).

Turan, Meltem Sönmez et al. (2010). "Recommendation for Password-Based Key Derivation".
In: p. 18.

UML Activity Diagram Tutorial (2021). Lucidchart.
URL: https://www.lucidchart.com/pages/uml-activity-diagram (visited on 05/03/2021).

Uusitalo, I. et al. (June 2009). "Phishing and Countermeasures in Spanish Online Banking".
In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. 2009 Third International Conference on Emerging Security Information, Systems and Technologies. ISSN: 2162-2116, pp. 167–172. DOI: 10.1109/SECURWARE.2009.33.

Verifying and Validating Software Requirements and Design Specifications - ProQuest (2021).
URL:
https://search.proquest.com/openview/354318cb8aa696975144cbcb65db5f86/1?pq-origsite=gscholar&cbl=37787 (visited on 02/02/2021).

Wang, Ding et al. (2015). "The Emperor's New Password Creation Policies:"
in: *Computer Security – ESORICS 2015*. Ed. by Günther Pernul et al.
Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 456–477.
ISBN: 978-3-319-24177-7. DOI: 10.1007/978-3-319-24177-7_23.

Wang, Ding et al. (2016). "Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound".
In: *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1. ISSN: 1545-5971.
DOI: 10.1109/TDSC.2016.2605087.
URL: http://ieeexplore.ieee.org/document/7558124/ (visited on 11/11/2020).

What is Scrum? (2020). Scrum.org.
URL: https://www.scrum.org/resources/what-is-scrum (visited on 20/10/2020).

What is Software Testing? Definition, Basics & Types (2021).
URL: https://www.guru99.com/software-testing-introduction-importance.html (visited on 04/03/2021).

Wu, Yimeng (14th Feb. 2021). *Kinnara/ModernWpf*. original-date: 2019-10-18T15:08:32Z.
URL: https://github.com/Kinnara/ModernWpf (visited on 15/02/2021).

XAML - Overview - Tutorialspoint (2021).
URL: https://www.tutorialspoint.com/xaml/xaml_overview.htm (visited on 18/03/2021).

Xin Zhou et al. (Aug. 2011).
"Research and implementation of RSA algorithm for encryption and decryption".
In: *Proceedings of 2011 6th International Forum on Strategic Technology*.
2011 6th International Forum on Strategic Technology (IFOST).
Harbin, Heilongjiang, China: IEEE, pp. 1118–1121. ISBN: 978-1-4577-0398-0.
DOI: 10.1109/IFOST.2011.6021216.
URL: http://ieeexplore.ieee.org/document/6021216/ (visited on 11/11/2020).

Yan, Jianxin Jeff (10th Sept. 2001). "A note on proactive password checking".
In: *Proceedings of the 2001 workshop on New security paradigms*. NSPW '01.
New York, NY, USA: Association for Computing Machinery, pp. 127–135.
ISBN: 978-1-58113-457-5. DOI: 10.1145/508171.508194.
URL: https://doi.org/10.1145/508171.508194 (visited on 10/11/2020).

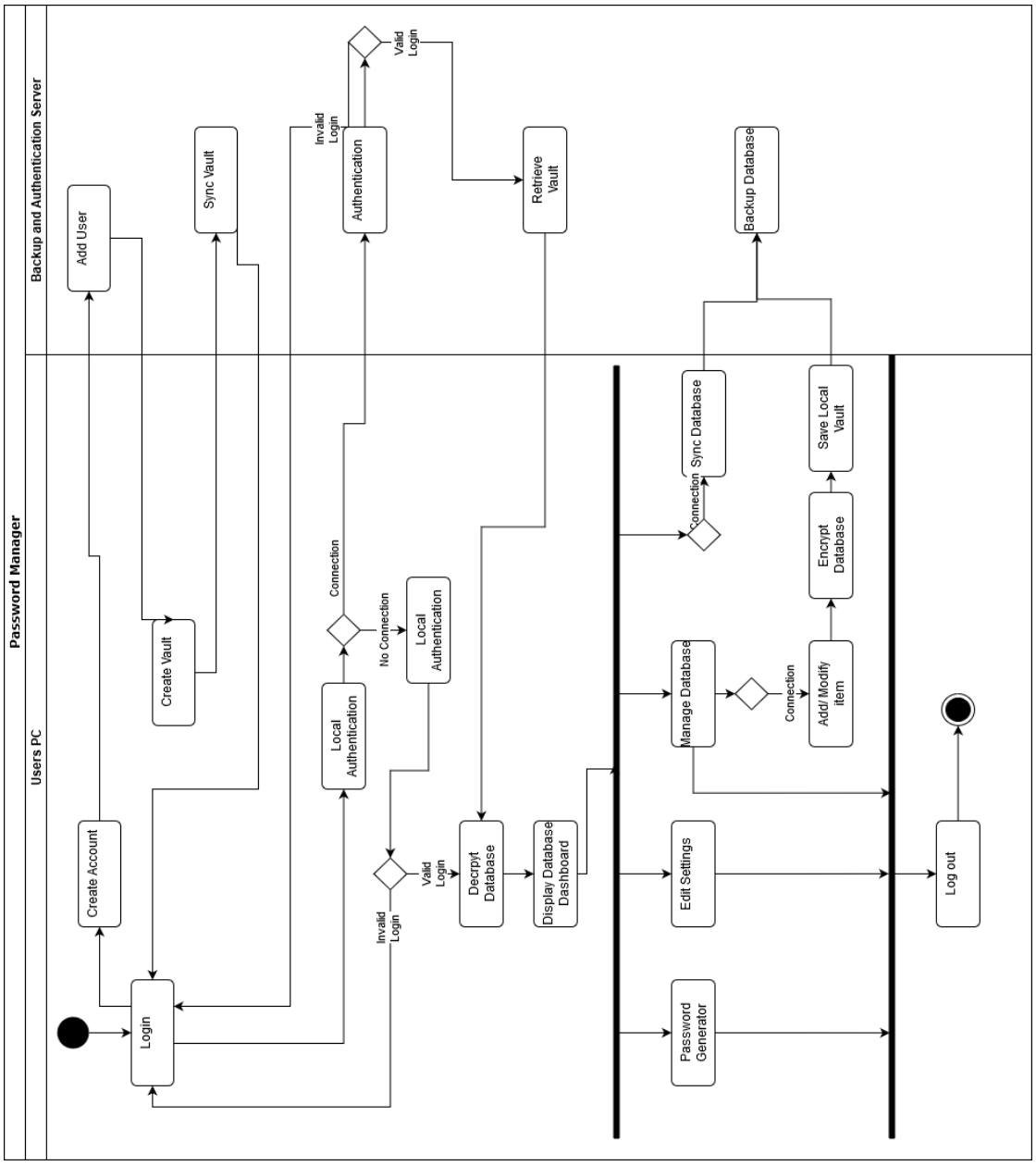
Yu, Fei et al. (Oct. 2015). "An Overview of Study of Passowrd Cracking".
In: *2015 International Conference on Computer Science and Mechanical Automation (CSMA)*.
2015 International Conference on Computer Science and Mechanical Automation (CSMA).
Hangzhou, China: IEEE, pp. 25–29. ISBN: 978-1-4673-9166-5. DOI: 10.1109/CSMA.2015.12.
URL: http://ieeexplore.ieee.org/document/7371616/ (visited on 11/11/2020).

10 Appendix

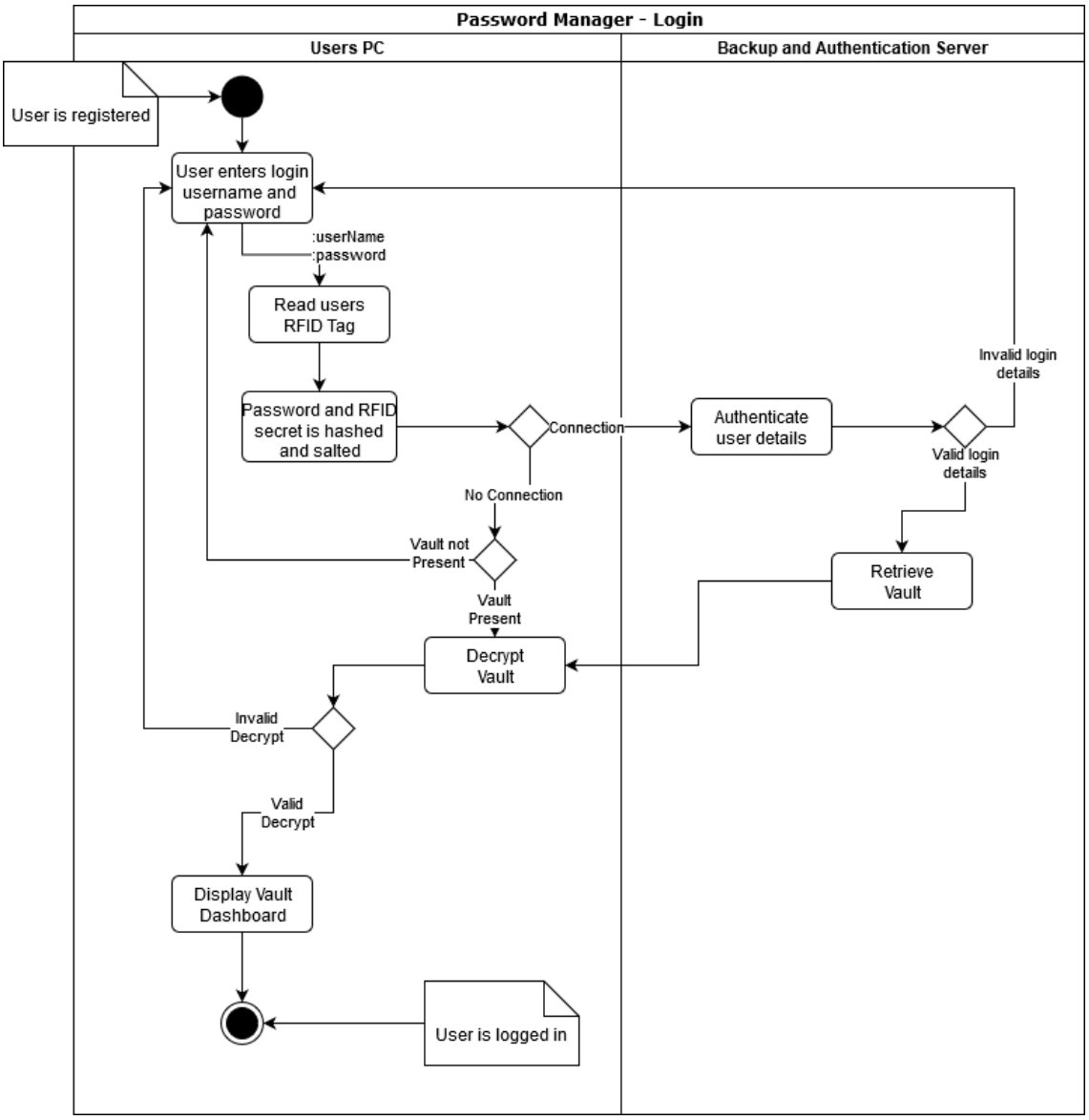
10.1 Error Cases

Error	Mitigation
The user enters a poorly formatted email address.	The system will inform the user by displaying a warning message until the issue is addressed.
The user attempts to create an account with an email address already in use.	The system will inform the user by displaying a warning message until the issue is addressed.
The user attempts to login to an account which is already logged in on another PC.	The system will inform the user of the issue and request they log out of the other computer.
The user enters an email and password which does not match.	The user will be informed of the problem and they will be able to try again.
The user attempts to submit any form within the system which contains empty/ null values.	The user will be informed of the problem and they will be asked to try again.
The user attempts to enter numerical values into a text box which only accepts letters.	The user will be informed of the problem and the system will allow then to try again.
The user attempts to enter letters into a text box which only accepts numerical values.	The user will be informed of the problem and the system will allow then to try again.
The system attempts to retrieve the user's database from the server, but it is corrupt/ non-existent.	The system will look locally on the user's PC for the local backup, if this is unsuccessful then the system will create a new database.

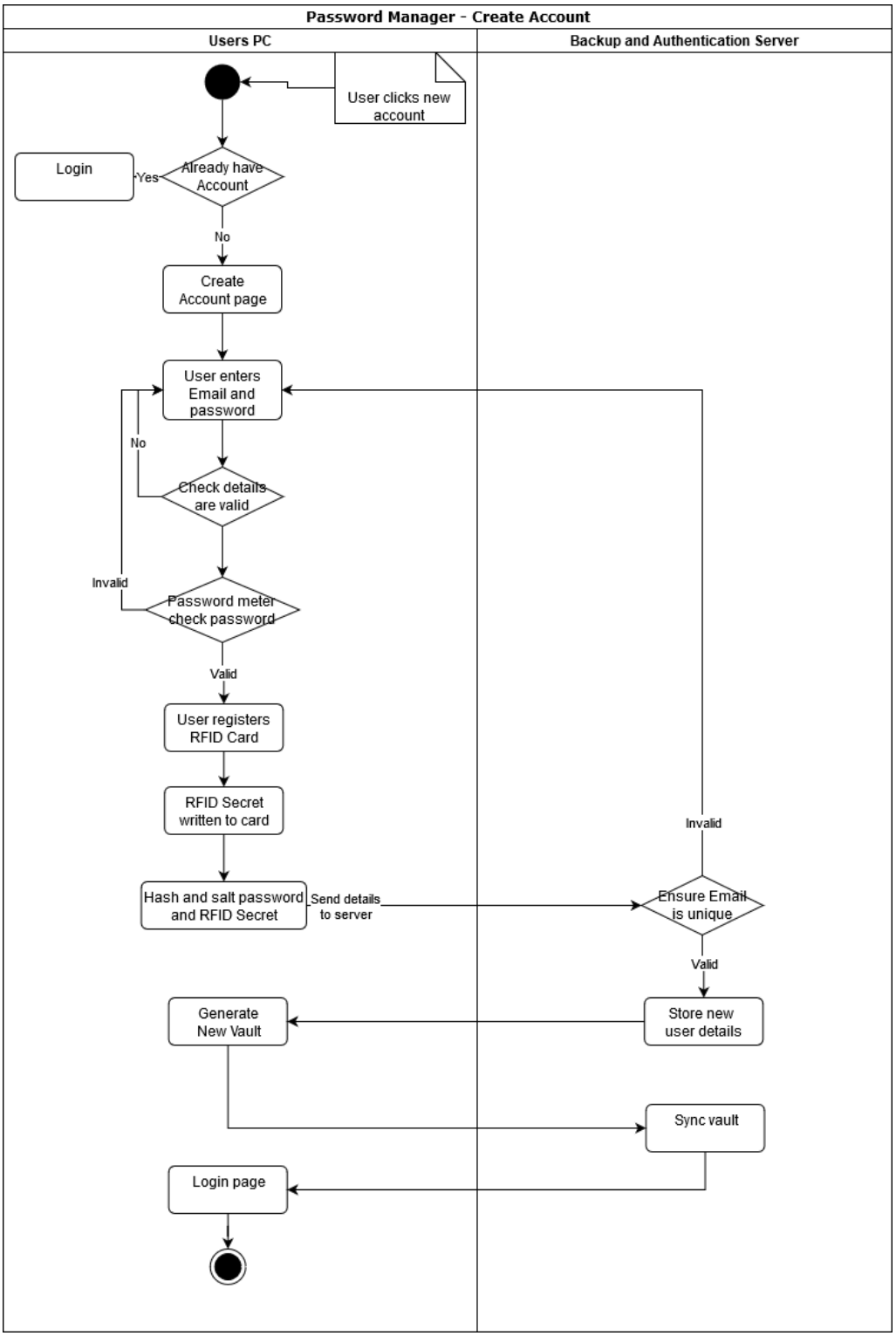
10.2 Password Manager Flow Diagram Overview



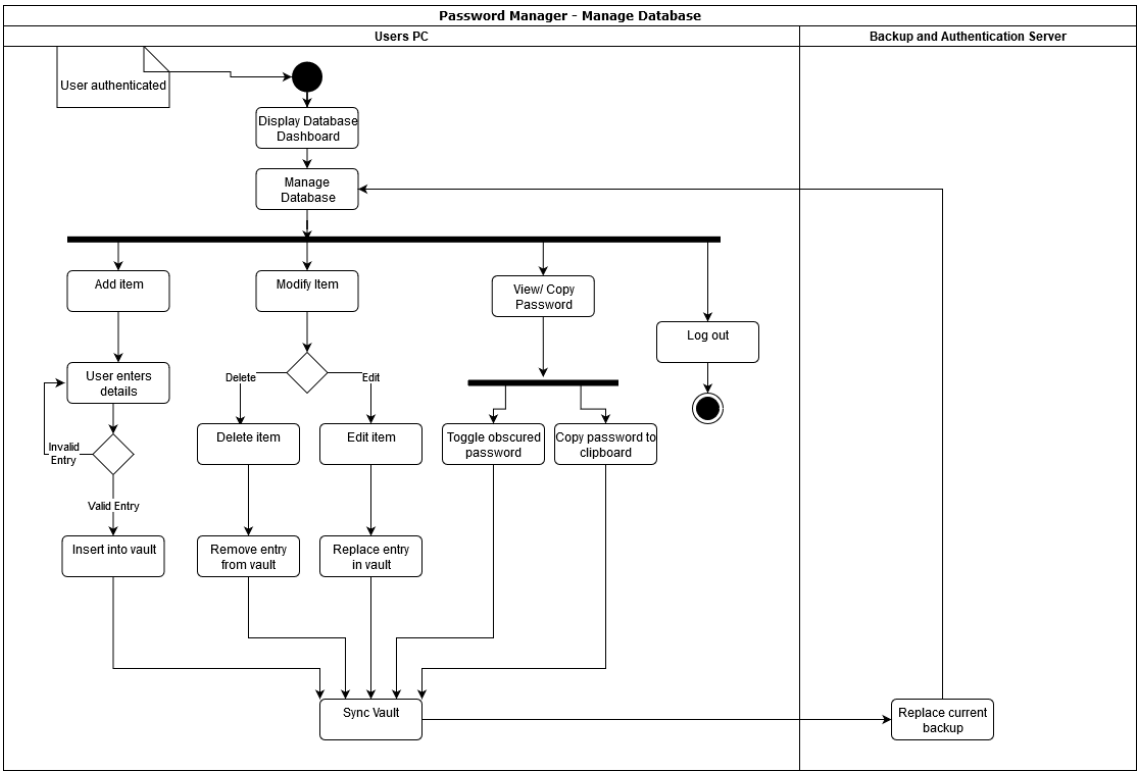
10.3 Activity Diagram – Login



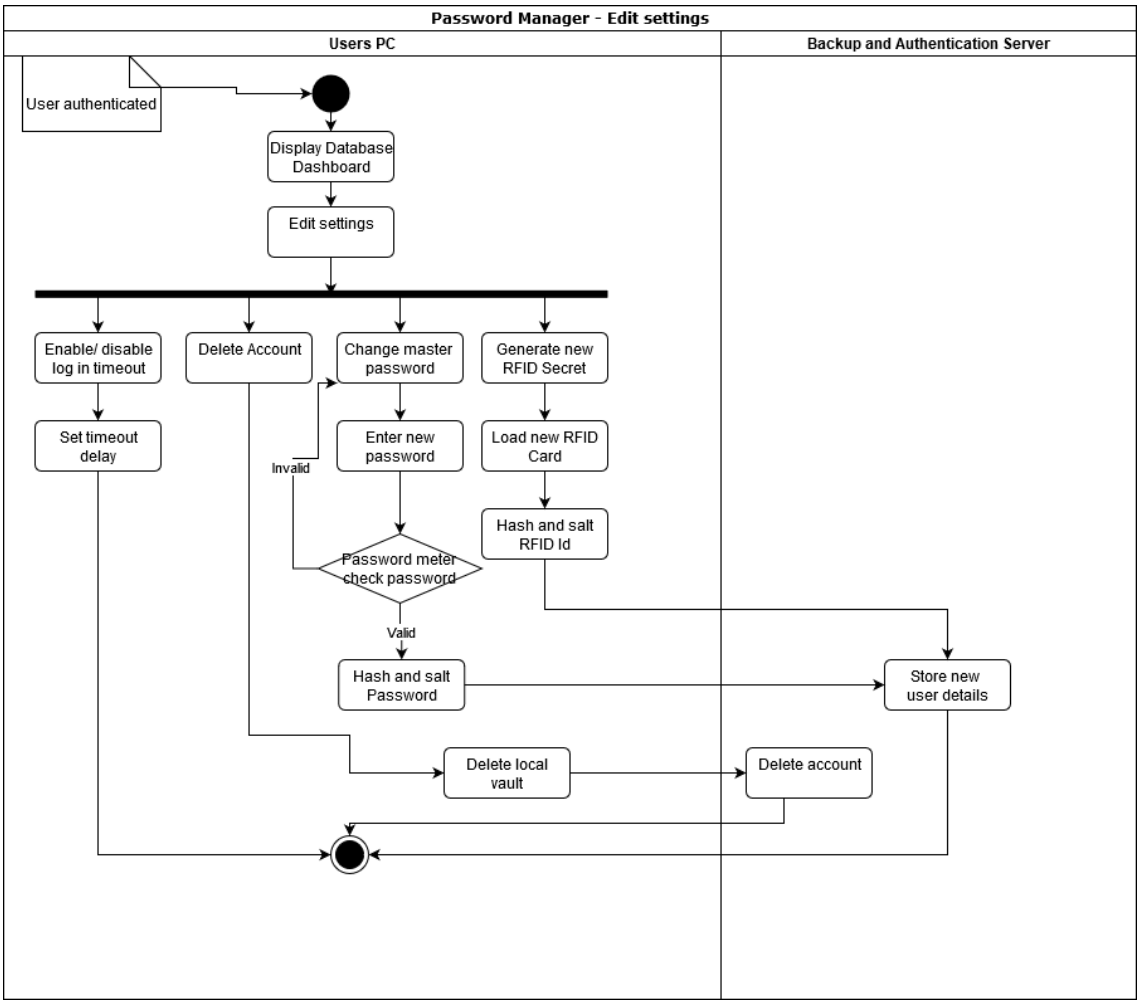
10.4 Activity Diagram – Create Account



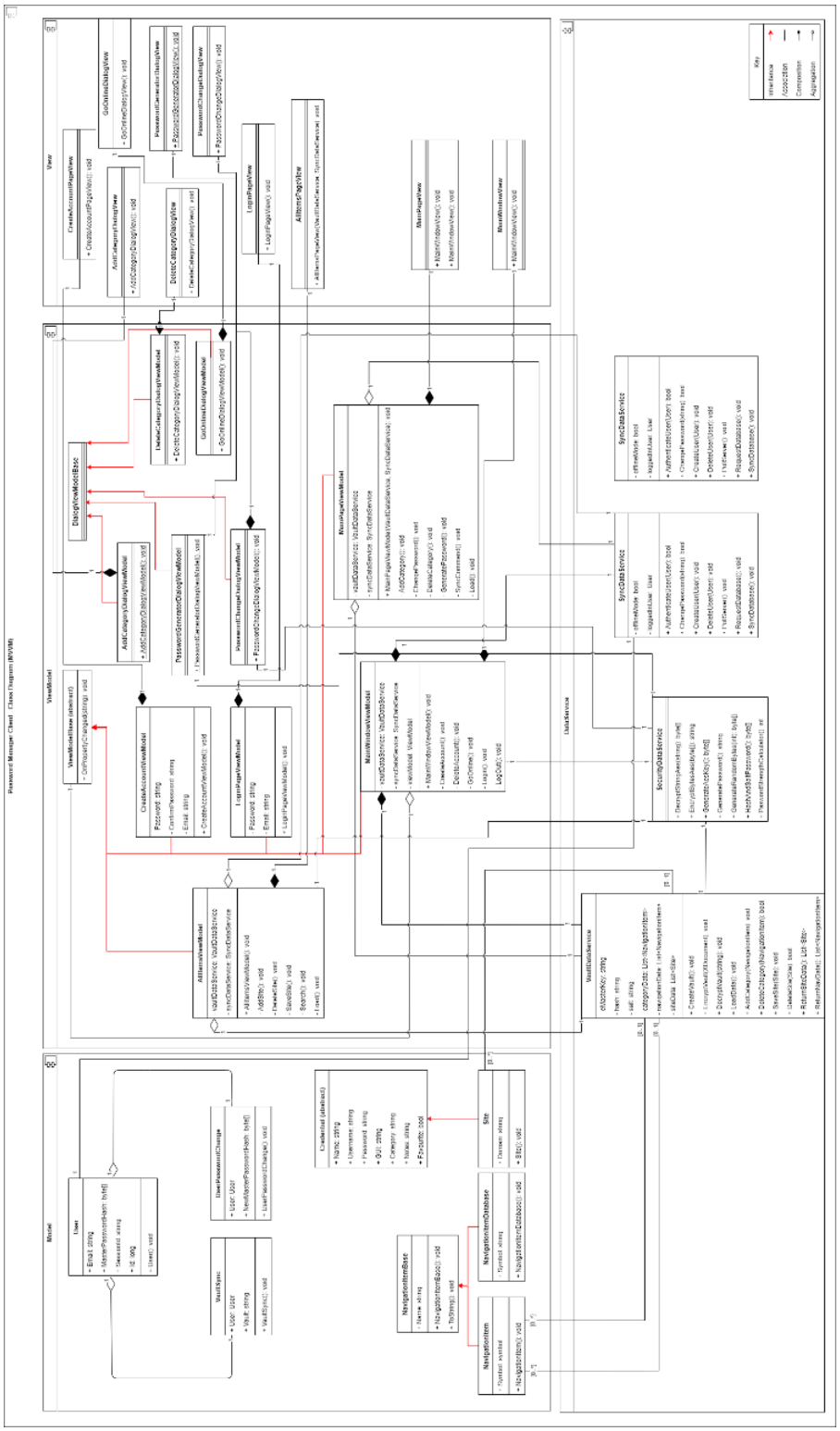
10.5 Activity Diagram – Manage Database



10.6 Activity Diagram – Edit Settings



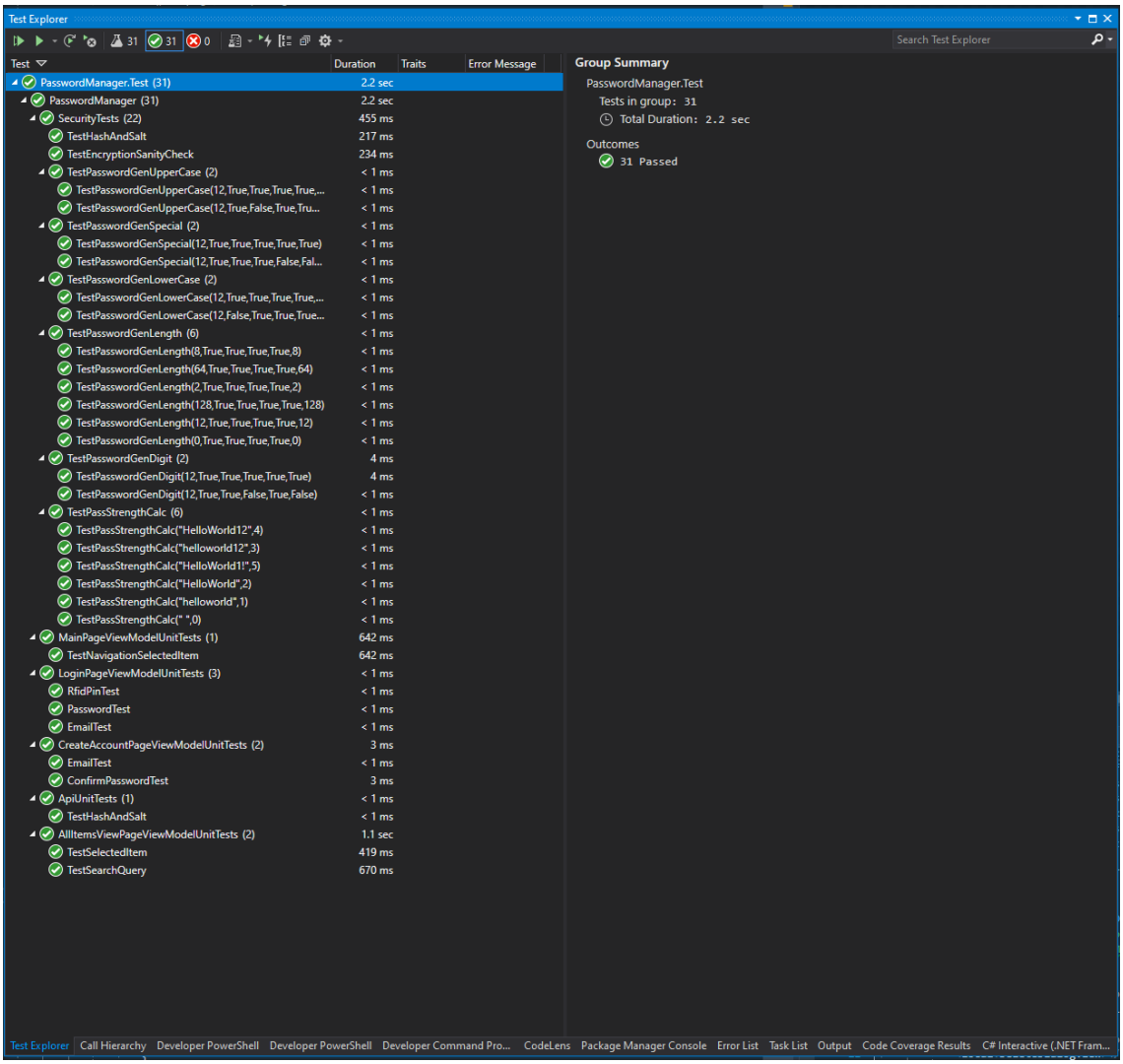
10.7 Password Manager Class Diagram



10.8 API Calls

API Call	Description
CreateUser	Takes a user object and adds it to the database if the user does not already exist.
Authenticate	Takes a user object and uses it to authenticate a user and returning Ok if the details are correct.
SyncDatabase	Takes the vault as an XMLString and replaces the copy saved on the server.
RequestDatabase	Returns a copy of the vault file as an XMLString.
ChangePassword	Sends a new password as a byte[] and replaces the currently stored password hash.
DeleteAccount	Deletes all the stored corresponding user's data from the server.
PollServer	Returns an Ok response to check the connection to the server is healthy.

10.9 Unit Test Output



10.10 JMeter Load Test Summary Output - 1 User

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Authentication - HTTP...	1	29	29	29	0.00	0.00%	34.5/sec
RequestVault - HTTP R...	1	18	18	18	0.00	0.00%	55.6/sec
TOTAL	2	23	18	29	5.50	0.00%	42.6/sec

10.11 JMeter Load Test Summary Output - 1,000 User

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Authentication - HTTP...	1000	9872	2180	18240	1109.38	0.00%	51.7/sec
RequestVault - HTTP R...	1000	8761	225	17155	1088.10	0.00%	57.5/sec
TOTAL	2000	9316	225	18240	1231.37	0.00%	102.2/sec

10.12 JMeter Load Test Summary Output - 2,000 User

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Authentication - HTTP...	2000	21602	20243	39317	912.28	0.00%	47.4/sec
RequestVault - HTTP R...	2000	19174	328	19323	429.68	0.00%	103.0/sec
TOTAL	4000	20388	328	39317	1407.67	0.00%	94.1/sec

10.13 JMeter Load Test Summary Output - 5,000 User

Label ↑	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Authentication - HTTP Request	5000	30433	52	45648	18318.13	25.82%	104.0/sec
RequestVault - HTTP Request	5000	25241	0	51519	15123.93	26.62%	64.4/sec
TOTAL	10000	27837	0	51519	16996.53	26.22%	124.3/sec

10.14 JMeter Load Test Summary Output - 10,000 User

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Authentication - HTTP Request	10000	29900	53	63857	29717.31	49.84%	138.0/sec
RequestVault - HTTP Request	5222	1973	0	61254	9358.17	95.81%	48.0/sec
TOTAL	15222	20320	0	63857	28035.25	65.61%	135.4/sec

10.15 Acceptance Testing

Test No.	REQ No.	Input	Expected Result	Actual Result	Pass/ Fail
1	REQ01, REQ10	Register a new valid account and submit account details.	The user is taken to the login page to use their new account.	Result as expected	P
2	REQ01, REQ10, REQ15.1	Register an account with a low password strength.	The user is warned that their password is insecure.	Result as expected	P
3	REQ01, REQ10, REQ15.2, REQ15.3	Register an account with an invalid email address.	The user is asked to enter a valid email address.	Result as expected	P
4	REQ01, REQ10, REQ15.4	Register an account with non-matching passwords.	The user is asked to enter matching passwords.	Result as expected	P
5	REQ01, REQ10, REQ11, REQ22	Register an account with an invalid RFID card.	The user is asked to add a valid RFID card.	Result as expected	P
6	REQ02, REQ12	Login to a valid account.	The user is taken to their vault management page.	Result as expected	P
7	REQ02, REQ12, REQ16.1	Login to an account with the wrong RFID card.	The user is warned that the login details are incorrect.	Result as expected	P
8	REQ02, REQ12, REQ16.2	Login to an account with invalid email address.	The user is warned that the login details are incorrect.	Result as expected	P
9	REQ02, REQ12, REQ16.3	Login to an account with invalid email and password combination.	The user is warned that the login details are incorrect.	Result as expected	P
10	REQ02, REQ12, REQ22	Login to a valid account, with an invalid RFID Card	The user is warned that the login details are incorrect.	Result as expected	P
11	REQ03.1, REQ12, REQ03.2	View previously added credential details.	The user can view each attribute of a credential, the password is obscured unless the user presses a button.	Result as expected	P
12	REQ03.3, REQ04	Add new credential.	The user can add a new credential to the vault.	Result as expected	P
13	REQ03.3, REQ04, REQ20	Add new credential, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
14	REQ03.3, REQ04, REQ17.1	Add a new credential with a missing name.	The user is warned that the details entered are not valid.	Result as expected	P

15	REQ03.3, REQ04, REQ17.2	Add a new credential with an invalid type of value.	The user is warned that the details entered are not valid.	Result as expected	P
16	REQ03.3, REQ04, REQ17.3	Add a new credential which has the same values as another.	The user can add a new credential to the vault.	Result as expected	P
17	REQ03.4	Add a credential to an existing category.	The credential is added to the custom category.	Result as expected	P
18	REQ03.4, REQ20	Add a credential to an existing category, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
19	REQ03.4	Try to add a credential to a non-existent category.	It is not possible as the rentals are a drop-down list.	Result as expected	P
20	REQ03.5	Favourite a credential.	The credential is added to the favourite category.	Result as expected	P
21	REQ03.5, REQ20	Favourite a credential, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
22	REQ05	Add a valid new custom category.	The category is created and made available to the credential page.	Result as expected	P
23	REQ05, REQ18	Add an invalid or empty new custom category.	The user is warned that the category must be valid.	Result as expected	P
24	REQ05, REQ20	Add a valid new custom category, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
25	REQ05	Delete a custom category.	The category is deleted and disassociated from credentials.	Result as expected	P
26	REQ05, REQ19	Delete a default category.	The user is warned that they cannot delete a default category.	Result as expected	P
27	REQ05, REQ20	Delete a custom category, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
28	REQ06	Generate a password.	The user is given a randomly generated password using their required attributes.	Result as expected	P

29	REQ07, REQ13.2	Manually sync vault.	The latest copy of the user's vault file is successfully synced to the server.	Result as expected	P
30	REQ07	Manually sync vault, whilst offline.	The user is warned that they need to be online to perform this action.	Result as expected	P
31	REQ08.1	Change master password and RFID card.	The user can change their authentication details.	Result as expected	P
32	REQ08.1	Change master password and RFID card, whilst in offline mode.	The user is warned that they need to be online to perform this action.	Result as expected	P
33	REQ08.2	Go online with valid login details and a connection to the server.	The user can change the system state to online.	Result as expected	P
34	REQ08.2	Go online with invalid login details and a connection to the server.	The user is warned that their login details are incorrect.	Result as expected	P
35	REQ08.2, REQ20	Go online with valid login details and no connection to the server.	The user is warned that they need to connect to the server.	Result as expected	P
36	REQ08.4	Delete the currently logged in account.	The users account is deleted, and they are taken to the login page.	Result as expected	P
37	REQ08.4, REQ20	Delete the currently logged in account, whilst not online.	The user is warned that they need to be online to perform this action.	Result as expected	P
38	REQ21	Password hashed using Argon2i	The user's password is hashed using Argon2i	Result as expected	P
39	REQ23	Data is encrypted using AES-GCM	The user's data is encrypted using AES-GCM	Result as expected	P
40	REQ13	Vault is synced after each vault change.	The user's vault is automatically synced after each database manipulation action.	Result as expected	P
41	REQ14	System authenticates user with server on login.	The user is authenticated with the server when they login.	Result as expected	P

10.16 Project Initiation Document

Project Initiation Document

Researching and Implementing a Secure Password Manager using MFA

October 2020

By

Lachlan Craig Gourlay

Student number 201700039

Word count: 3300

Project Initiation Document

Contents

Project Initiation Document

1. Project background and purpose

1.1. Background

Password management is a modern-day problem many of use face. Anybody who uses an internet-based service which requires authentication, has most likely come across a password as a way of gaining access to some form of private information. The Cambridge dictionary defines a password as *"a secret word or combination of letters or numbers, used for communicating with another person or with a computer to prove who you are"*("PASSWORD | meaning in the Cambridge English Dictionary," n.d.) , the key phrase within that definition being secret. This is where the problems arise, keeping your password secret, as well as memorable is not as straight forward as it may seem. The NCSC (National Cyber Security Centre) analysed account information that had been accessed by third parties from cyber breaches, and found that the most commonly used password is "123456", and simply the word "password" being in the top 5("Most hacked passwords revealed as UK cyber survey exposes gaps in online security," n.d.). A strong password is important, and security company Avast suggest a strong password needs to be at minimum 15 characters long, with a mix of letters, numbers, and special characters.("How to Create a Strong Password and Beat the Hackers | Avast," n.d.) Not only this, but the passwords you use should be unique to every place you need to login. So, this leads to the big issue of remembering these strong and unique strings for every website you use. This project proposes a solution to these problems in the form of a password manager, this manager would allow the user to generate long, unique strings of characters which are saved in a secure database. The user would then only need to remember one strong password which would gain them access to the password manager described. Similar systems exist already built into browsers, the main issue with this way of working is that you are stuck in the ecosystem of whichever browser you choose to use. Not only are you stuck in that single ecosystem but if the data on the device is lost you have lost all your saved passwords.

1.2. Aims and objectives

This project aims to solve the problem of poor password etiquette by taking the onus of remembering off the user, in the form of a secure password manager. The application should support the safe storage of website authentication credentials such as usernames and passwords.

The main objectives are as follows:

Objective 1.	The solution will support a windows desktop application that will contain a local secure encrypted vault; this will store an infinite number of strong, unique strings of characters that form a user's passwords. All user data will be encrypted and hashed before it is stored.
Objective 2.	The vault will support a search function and default and custom categories, allowing the user to filter their saved credentials easily.
Objective 3.	The application will use MFA (Multi-Factor Authentication); when the user accesses their vault, they will need to use multiple forms of identification.
Objective 4.	The project will also support a password checker and meter, informing the user of a password's strength and uniqueness.
Objective 5.	The project will also contain a password generator, this will suggest passwords to the user using a set of rules outlined with knowledge gained from the research conducted earlier in the project.

If there is time, then these are the secondary objectives that could be achieved:

Secondary Objective 1.	The solution could allow the local database containing encrypted user data to be synced across multiple devices via an API server.
Secondary Objective 2.	The final project could also include a feature for securely syncing login details to another vault, and this would prevent somebody from sending passwords in plain text via messaging services.
Secondary Objective 3.	The project could include a browser extension that could capture login details as they are used within the browser. This data will then be encrypted, hashed, and stored in the local database.

1.3. Scope

This project will solve the issue of password security and management of said passwords. This dissertation will only be looking at what makes a good password and how we can measure the entropy of a password. The project will not be trying to invent any new radical ideas for password security but will instead be using knowledge other researchers have produced. The project will also investigate the most common forms of password cracking, the project will not be attempting to create a new method of cracking passwords, it will simply be putting any knowledge gained into practice.

The project will not be collecting any actual user passwords as the project is a proof of concept to show good practice for a viable product which would use similar techniques. The project will not be collecting any first-hand data to be used for analysis, all data used will be found from existing data sets.

The software artifact will be formed of a secure database with the necessary security in place to give users confidence in its ability to store sensitive data. The software artifact will not be using any ground-breaking new technology and will instead use techniques and libraries produced by researchers with more knowledge and understanding than the creator of this project. The system will use a strong master password supplied by the user in conjunction with a username which will be used to generate a unique encryption key. This is then used to encrypt the user data which is stored locally on the device. The encrypted file is then shared to a server where it can be synced to other devices.

1.4. Deliverables

The project will be formed of five main deliverables. The first deliverable will be in the form of a software artifact, this artifact will be one executable file which will be compatible with a windows machine. The software will cover objectives 2, 4 and 5 outlined in section 1.1 The second deliverable will be a server executable file, it will run on windows. This deliverable will satisfy objective 3 from section 1.1. The third and fourth deliverables will be in the form an NFC reader and token, as well as a google pay pass which can be loaded onto any android phone, this will be formed of an Arduino and relevant circuitry. These deliverables will satisfy objective 6 from section 1.1.

The final deliverable will come in the form of a demonstration, this will used as an opportunity for me to show off the other two deliverables and give an in-depth explanation to how the software is used and the features it contains.

1.5. Constraints and Assumptions

The project is constrained by the secondary data, as the project will rely on data collected from journals and other online sources there is an assumption that the information supplied is correct and up to date. Some academic journals are also kept behind pay wall, this could cause constraints on the project as the creator of the project is a student and the budget attached to the project is very low. As the project will most likely rely on libraries created by other people it is assumed that these libraries will not have any bugs or issues.

Due to the fact the project is going to be a desktop application it is imperative users of the software artifact are using a windows desking computer. Time is also a constraint, to ensure the project is completed on time the planning and scheduling techniques outlined in section 3 will be used. Scope is another constraint, well defined requirements ensure scope creep will not occur, along with the use of Kanban.

The assumptions made surrounding this project are that the project that has been outlined is possible with the constraints mentioned, this includes the time and scope of the project. Assumptions have also been made that the law surrounding GDPR (“What is GDPR, the EU’s new data protection law?,” 2018) will not change and thus as long as no personal information of users is collected, kept or processed then there should be no issue.

2. Project rationale and operation

2.1. Project benefits

The project that has been outlined so far is aimed at helping people who use any online service which requires authentication in the form of a text-based string. In general, most people's attitudes to passwords are quite poor in terms of the quality and the reuse of said passwords. This project will help these people avoid reuse of passwords in form of a storage database. The user will only need to remember one password whilst still being able to have unique passwords to the many services they use. One issue with this solution is the added security risk of having all your passwords written down in one place. This risk is mitigated using encryption, the passwords stored will done so using modern security techniques. The system will also use MFA (Multi-Factor Authentication), this will create an extra layer of security which will protect the users' data.

The project will also aim to solve the issue of poor-quality passwords, this is defined as a password with poor entropy. The solution will offer the ability for the user to input their passwords into a database, at this point the software will give the inputted password a rating which will inform the user if the password is of a high enough quality. Sometimes thinking of a password with a high enough quality can be difficult, to help with this the solution will offer the ability to generate a password using a set of rules.

The project will also synchronise all your login data across devices, this has the added benefit of acting as a backup system so if the data on your local device is lost it can still be retrieved using the master password. Using an external password manager also means the user is not stuck in one ecosystem and can store login details for a multitude of login systems including desktop applications and even physical devices like a padlock.

2.2. Project operation

Kanban (Atlassian, n.d.) will be used to organise and plan the project, the Kanban board will be hosted on Trello ("What is Trello? - Trello Help," 2020). Kanban is used to visualise the software that will be created, making it easier to envision the plan and what will be completed. The project must first be broken down into smaller more manageable work tasks, each task gets its own card on the board and starts on the farthest left column which in this case is Backlog. As tasks get completed, they move across the board until they reach the furthest right column, this is called workflow. This method works well for many people as it is an agile way of working, this allows them to continue to add tasks to the backlog as the project progresses, meaning any ideas which are thought of as more information is learnt can be added easily to the plan.

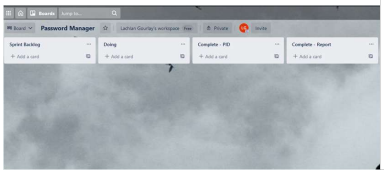


Figure 1 – Kanban Board

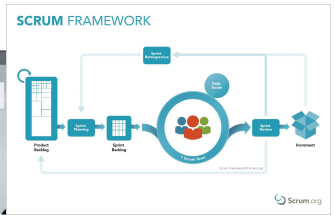


Figure 2 – ("The Scrum Framework Poster," n.d.)

Along with using Kanban to visualise what tasks need to be completed the project will also be using SCRUM ("What is Scrum?," n.d.) to organise and to ensure what is produced at the end of the project meets the objectives and aims which were set. The project will be completed in 2-week

sprints, at the end of a sprint, what has been completed thus far will be reviewed, this information can then be used to identify what can be improved in the next sprint.

Furthermore, the project will use a Gantt chart produced in Microsoft Project, this is a type of bar chart which presents the tasks to be completed along with their dependency relationships. Whereas the Kanban board shows me where the project is up to and what needs to be completed, the Gantt chart shows me the best path to the end. One main feature of the Gantt chart is the critical path, this shows me the sequence of tasks which must be completed in time otherwise the project could be delayed.

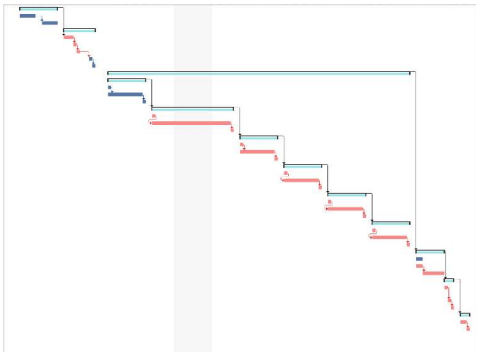


Figure 3

It will be known whether the use of Kanban and a Gantt chart alongside SCRUM was successful or not when all of the tasks which were in the backlog, have been moved to the done column, after this fact there should be a final working solution which is ready for production.

2.3. Options

The main deliverable will be in the form of a desktop application. This means there are lot of options when it comes to the language and frameworks available to the project. These languages include C# (BillWagner, n.d.), Java ("Java Programming Language," n.d.), C++ ("cplusplus.com - The C++ Resources Network," n.d.) and any other application centred, object-oriented language. When it comes to performance C++ is better as its overheads are much lower, this is also why the binary files compiled by C++ are smaller. In terms of platforms supported C++, Java and C#, can all be compiled for almost any desktop operating system including Windows ("Explore Windows 10 OS, Computers, Apps, & More | Microsoft," n.d.), Mac OS ("macOS Catalina - Apple (UK)," n.d.) and Linux ("What is Linux?," n.d.).

When it comes to ease of use of the three languages, C# has its own inbuilt garbage collector, and so does Java. C++ does not offer garbage collection, meaning the programmer needs to handle memory allocation and deallocation. C++ does offer error messages, but they can only be seen after the code has been compiled. C# on the other hand tells the programmer as they write about issues with the code, making it easy and faster to spot mistakes.

When it comes to storing the data within the project there are a few options. The first traditional method is using RDBMS (relational database management systems) ("What is a relational database?," n.d.) such as MSSQL ("SQL Server 2019 | Microsoft," n.d.) and MySQL ("MySQL," n.d.),

this method stores the data in tabular form in rows and columns. To interact with the data SQL (Structured Query Language) (“SQL Introduction,” n.d.) is used. MongoDB (“The most popular database for modern apps,” n.d.) on the other hand is a NoSQL, document orientated database. It uses BSON to store the data in the form of documents meaning no SQL is used to access the data. This also means MongoDB cannot be attacked using SQL injection (“SQL Injection,” n.d.) which is a large security vulnerability with SQL based storage solutions.

There are a few options when it comes to the NFC reader, the first being an Arduino(“What is Arduino?,” n.d.) with the relevant circuitry and modules such as the RC522 RFID reader module (“RC522 RFID Module,” n.d., p. 522). The Neuftech ID Card Reader and writer (“RFID -Software,” n.d.)is another option but would not offer as much customisability as the Arduino option. The writer of this report also already has access to the Arduino and parts so there is a cost benefit to the Arduino.

2.4. Risk analysis

Risk	Mitigation	Recovery	Likelihood	Severity	Impact
Supervisor departure	As the supervisor is only offering advice, this would not affect the outcome of the project heavily.	Speak to the university.	1	1	2
Illness for long duration	Ensure contingency is in place within the plan.	Contingency time will account for this.	1	1	2
Computer Failure	Backup data and ensure development environment is available in multiple places.	I would have a backup of my work which could be recovered.	2	1	3
Task overruns	Ensure contingency is in place if tasks take more time than planned for.	Use planned contingency time.	2	1	3
File corruption	Ensure all files are backed up to multiple places.	Recover a backup of the file.	2	1	3
Scope change	SCRUM will mean scope changes can be caught early and accounted for.	Incorporate new ideas into the Kanban plan.	2	1	3
Poor quality code	Thorough design such as class and object diagrams will help to mitigate this.	Refactoring could be used if certain areas of code are poorly written.	1	2	3
Poor management	Use management techniques such as SCRUM and Kanban.	Refer to plans created at beginning of project	2	2	4

2.5. Resources required

The only resources that will be needed to complete the project are an integrated development environment (IDE) and configuration control, an IDE is used to write and compile code and configuration control is used to back up versions of the code which is written. A desktop computer will be needed for development and testing. Furthermore Visual Studio(“Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio,” n.d.) and the Arduino IDE (“What is Arduino?,” n.d.) will be used as IDE’s and Github (“What Exactly Is GitHub Anyway?,” 2012) for config control.

3. Project methodology and outcomes

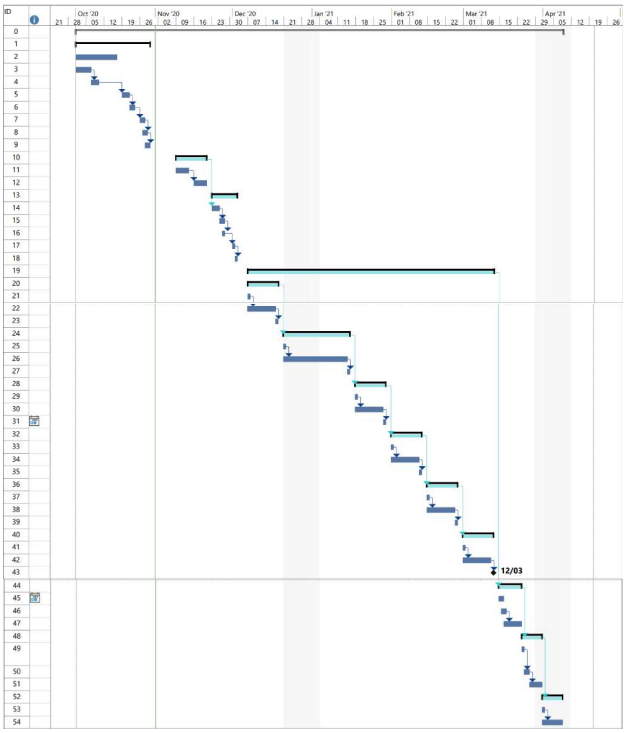
3.1. Initial project plan

3.1.1. Tasks and milestones

Research
• Research and identify language and frameworks.
• Research and identify best database solution.
• Research password quality and entropy.
• Research forms of password cracking e.g Brute force and dictionary.
• Research encryption techniques.
• Research hashing algorithms.
• Research rules for strong passwords.
• Research cyber-attack vectors.
Literature
• Literature search.
• Literature review.
Initial Design
• Design class and object diagrams.
• Design database.
• Design UI layout.
• Design hashing algorithm.
• Add information and designs created to report.
Milestone 1: Finish first half of report and design.
Sprint 1:
Develop basic application layout
• Create template for application layout.
• Create login system.
Sprint 2:
Objective 3 – Database
• Create database for user logins and passwords.
• Create form for adding login details to database.
Milestone 2: Finish application basics and database
Sprint 3:
Objective 3 – Encryption
• Create algorithm for hashing and salting passwords.
Sprint 4:
Objective 4 – Sync
• Create server-side code for syncing database
• Implement client side code for sync
Milestone 3: Finish hashing algorithm and password checker
Sprint 5:
Objective 4 – Password Generator and Checker
• Create password generator.
• Create rules for strong password.
• Create password checker.
Sprint 6:
Objective 5 – Two factor authentication

• Create system for two factor authentication.
Milestone 4: Finish password gen, two factor
Testing
• Perform system testing.
• Document issues and bugs.
• Correct bugs and issues found.
Analysis
• Analyse and evaluate system to see if it meets objectives.
• Analyse user feedback.
• Propose changes.
Milestone 5: Finish second half report.
Report
• Complete writing report.
• Proofread and format report.
Milestone 7: Finish project and polish

3.1.2. Schedule Gantt chart



3.2. Project evaluation

Evaluating the projects deliverables will be done in the form of testing. The first type of testing is called unit testing. Certain sections within the software can be tested independently allowing the result from a particular important function to be checked against its desired result.

The next type of testing is called acceptance testing, this will be used to check the deliverables meet and fulfil the requirements set out at the beginning of the project. User behaviour is replicated and then the outcome is recorded to ascertain if the objective has been met. This will be done throughout the project at the end of each sprint, as well as at the end so that issues can be caught early. The next type of testing that will be used is Integration testing, this is done continuously throughout the development process. For this project this will be done at the end of each sprint, the completed individual parts will be taken and tested alongside each other to ensure that they are functional, this will also help to find and catch bugs early.

Performance testing will also be undertaken to ensure the system can handle the required number of users. As the system only needs to support up to 6 users in a session this testing does not need to be performed thoroughly and instead as an exercise of proof of concept. There is not plan for other people to test the system, so all this testing will be performed by the project creator.

4. Appendix

4.1. Appendix A – References

Atlassian, n.d. Kanban - A brief introduction [WWW Document]. Atlassian. URL https://www.atlassian.com/agile/kanban (accessed 10.20.20).

BillWagner, n.d. C# docs - get started, tutorials, reference. [WWW Document]. URL https://docs.microsoft.com/en-us/dotnet/csharp/ (accessed 10.20.20).

cplusplus.com - The C++ Resources Network [WWW Document], n.d. URL https://www.cplusplus.com/ (accessed 11.1.20).

Explore Windows 10 OS, Computers, Apps, & More | Microsoft [WWW Document], n.d. URL https://www.microsoft.com/en-us/windows (accessed 11.1.20).

How to Create a Strong Password and Beat the Hackers | Avast [WWW Document], n.d. URL https://blog.avast.com/strong-password-ideas (accessed 11.1.20).

Java Programming Language [WWW Document], n.d. URL https://docs.oracle.com/javase/7/docs/technotes/guides/language/ (accessed 10.20.20).

macOS Catalina - Apple (UK) [WWW Document], n.d. URL https://www.apple.com/uk/macos/catalina/ (accessed 11.1.20).

Most hacked passwords revealed as UK cyber survey exposes gaps in online security [WWW Document], n.d. URL https://www.ncsc.gov.uk/news/most-hacked-passwords-revealed-as-uk-cyber-survey-exposes-gaps-in-online-security (accessed 11.1.20).

MySQL [WWW Document], n.d. URL https://www.mysql.com/ (accessed 11.1.20).

PASSWORD | meaning in the Cambridge English Dictionary [WWW Document], n.d. URL https://dictionary.cambridge.org/dictionary/english/password (accessed 11.1.20).

RC522 RFID Module [WWW Document], n.d. . Components101. URL https://components101.com/wireless/rc522-rfid-module (accessed 11.3.20).

RFID -Software [WWW Document], n.d. URL https://neuftech.net/rfid (accessed 11.3.20).

SQL Injection [WWW Document], n.d. URL https://www.w3schools.com/sql/sql_injection.asp (accessed 11.1.20).

SQL Introduction [WWW Document], n.d. URL https://www.w3schools.com/sql/sql_intro.asp (accessed 11.1.20).

SQL Server 2019 | Microsoft [WWW Document], n.d. URL https://www.microsoft.com/en-gb/sql-server/sql-server-2019 (accessed 11.1.20).

The most popular database for modern apps [WWW Document], n.d. . MongoDB. URL https://www.mongodb.com (accessed 11.1.20).

The Scrum Framework Poster [WWW Document], n.d. . Scrum.org. URL https://www.scrum.org/resources/scrum-framework-poster (accessed 3.24.21).

Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio [WWW Document], n.d. URL https://visualstudio.microsoft.com/ (accessed 11.1.20).

What Exactly Is GitHub Anyway?, 2012. . TechCrunch. URL https://social.techcrunch.com/2012/07/14/what-exactly-is-github-anyway/ (accessed 10.20.20).

What is a relational database? [WWW Document], n.d. URL https://www.oracle.com/uk/database/what-is-a-relational-database/ (accessed 11.1.20).

What is Arduino? [WWW Document], n.d. URL https://www.arduino.cc/en/Guide/Introduction (accessed 11.3.20).

What is GDPR, the EU's new data protection law? [WWW Document], 2018. . GDPR.eu. URL https://gdpr.eu/what-is-gdpr/ (accessed 10.20.20).

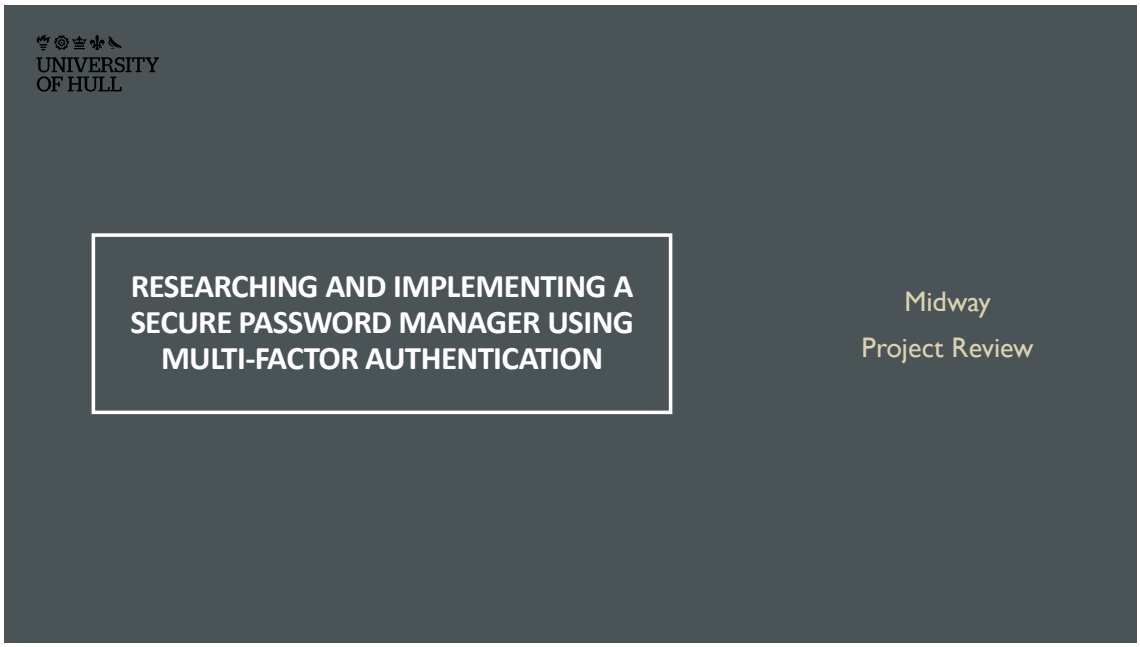
What is Linux?, n.d. . Linux.com. URL https://www.linux.com/what-is-linux/ (accessed 11.1.20).

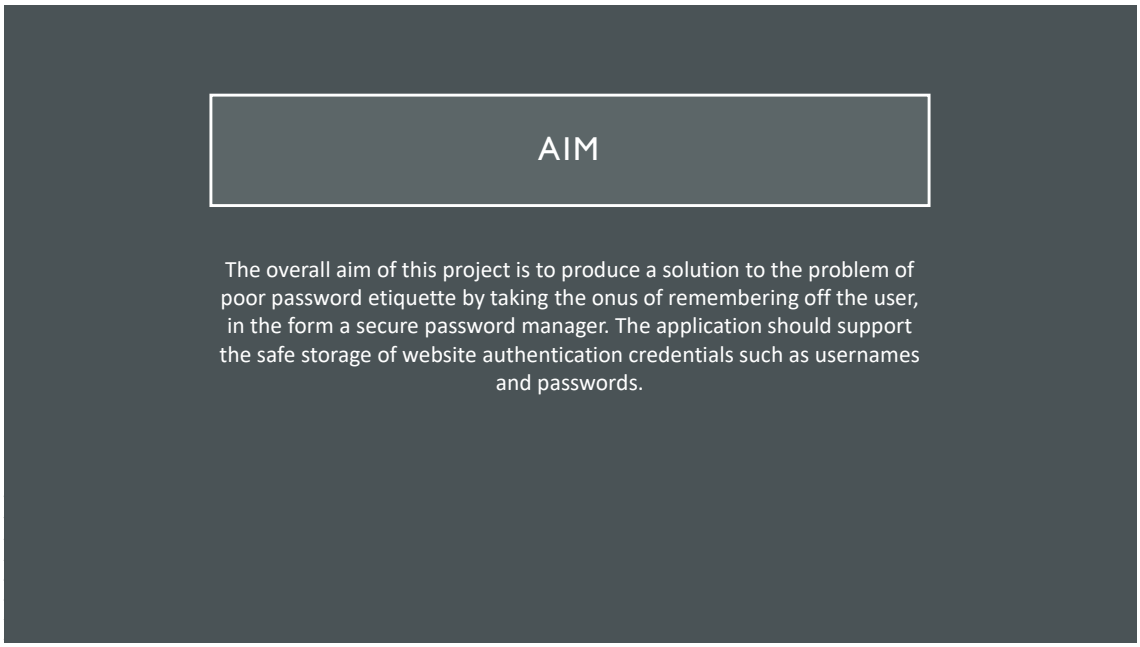
What is Scrum? [WWW Document], n.d. . Scrum.org. URL https://www.scrum.org/resources/what-is-scrum (accessed 10.15.20).

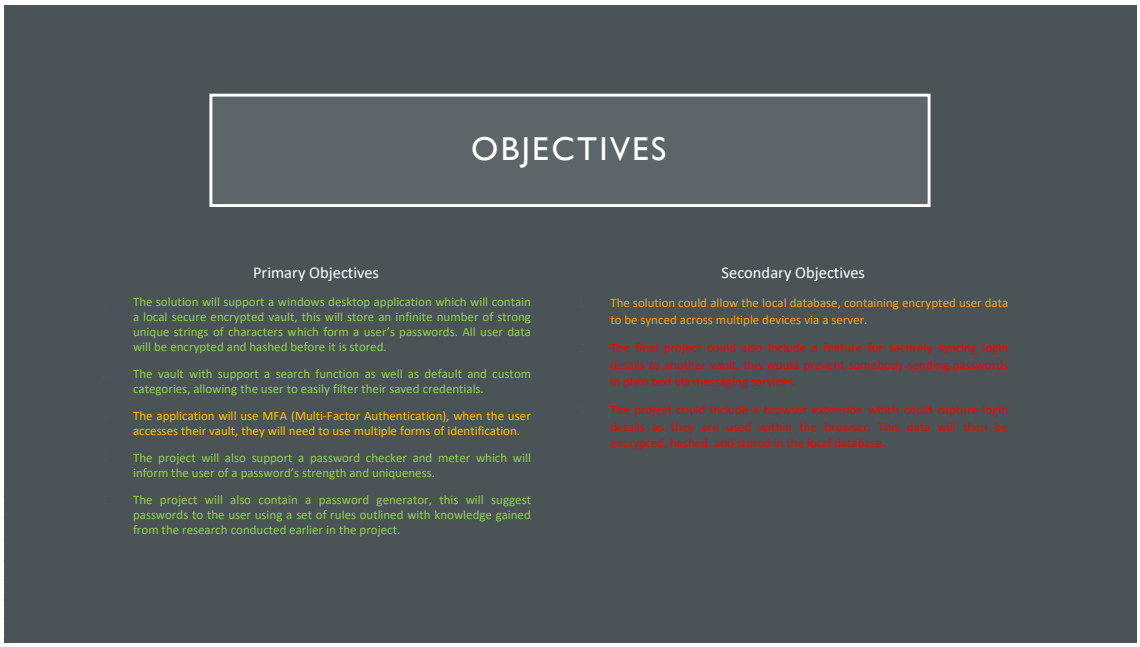
Project Initiation Document

What is Trello? - Trello Help [WWW Document], 2020. URL https://help.trello.com/article/708-what-is-trello (accessed 10.20.20).

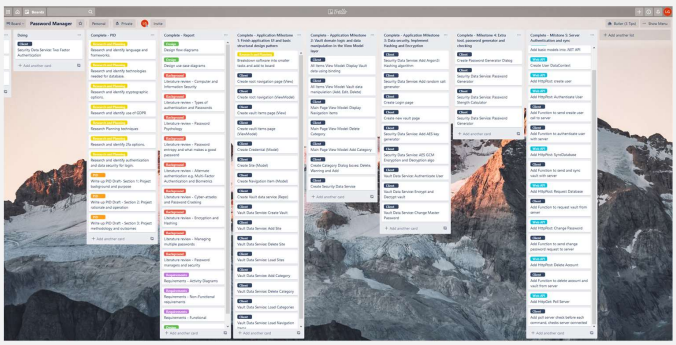
10.17 Mid Project Review




AIM
The overall aim of this project is to produce a solution to the problem of poor password etiquette by taking the onus of remembering off the user, in the form a secure password manager. The application should support the safe storage of website authentication credentials such as usernames and passwords.

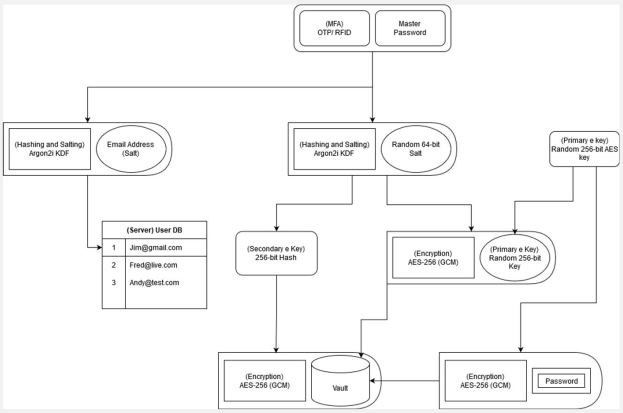

OBJECTIVES
Primary Objectives
The solution will support a windows desktop application which will contain a local secure encrypted vault, this will store an infinite number of strong unique strings of characters which form a user's passwords. All user data will be encrypted and hashed before it is stored.
The vault with support a search function as well as default and custom categories, allowing the user to easily filter their saved credentials.
The application will use MFA (Multi-Factor Authentication), when the user accesses their vault, they will need to use multiple forms of identification.
The project will also support a password checker and meter which will inform the user of a password's strength and uniqueness.
The project will also contain a password generator, this will suggest passwords to the user using a set of rules outlined with knowledge gained from the research conducted earlier in the project.
Secondary Objectives
The solution could allow the local database, containing encrypted user data to be synced across multiple devices via a server.
The final project could also include a feature for securely syncing login details to another vault. This would prevent somebody sending passwords in plain text via messaging services.
The project could include a browser extension which could capture login details as they are used within the browser. This data will then be encrypted, hashed, and stored in the local database.

PROGRESS SO FAR!

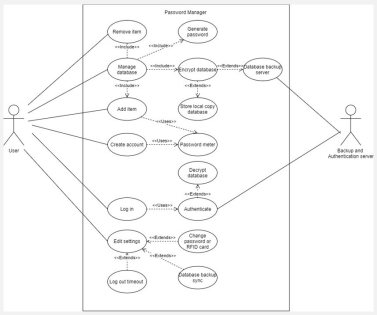


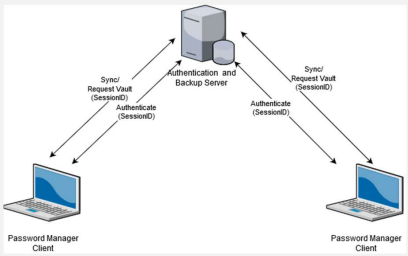

LITERATURE REVIEW
Password Managers
Computer and Information Security
Types of authentication and Passwords
Password Psychology
Password entropy and what makes a good password
Alternate authentication e.g. Multi-Factor Authentication and Biometrics
Cyber-attacks and Password Cracking
Encryption and Hashing
Managing multiple passwords
Password managers and security
Alternative Password Management Methods
Technologies
Architecture
Programming Languages and Frameworks
Database and Markup
Configuration Control
Code Linting

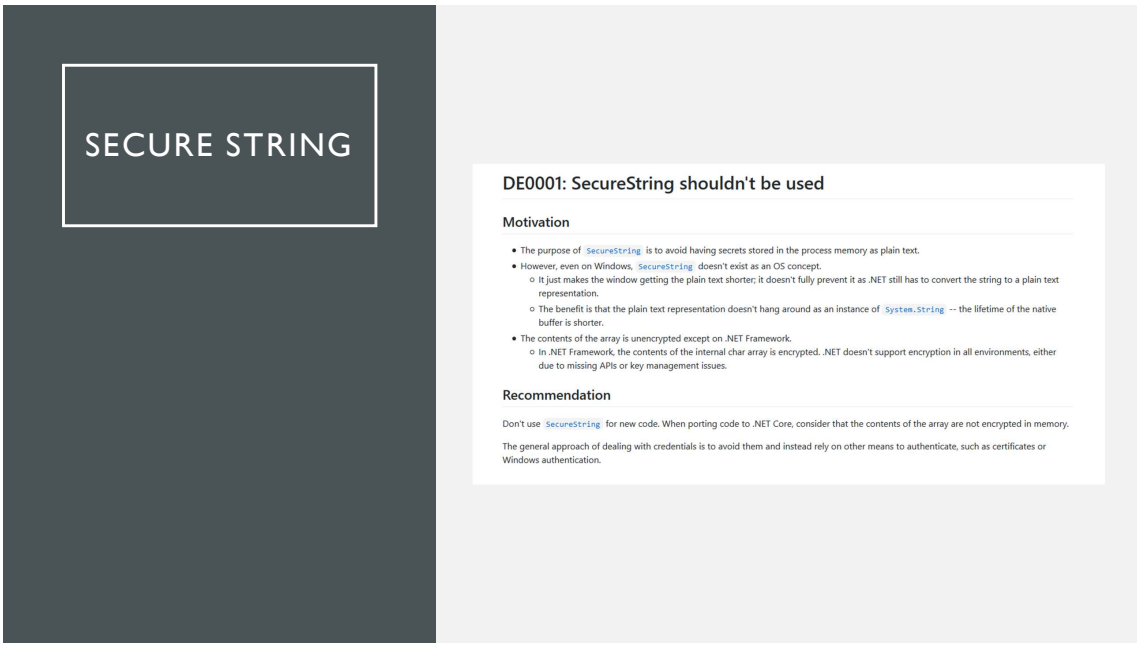
SECURITY MODEL



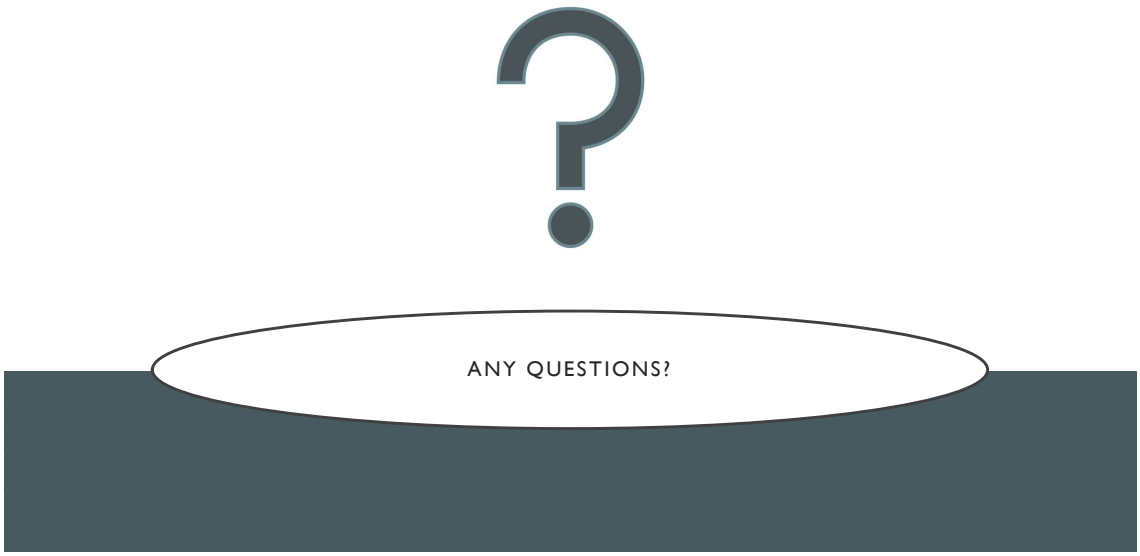
SYSTEM DESIGN





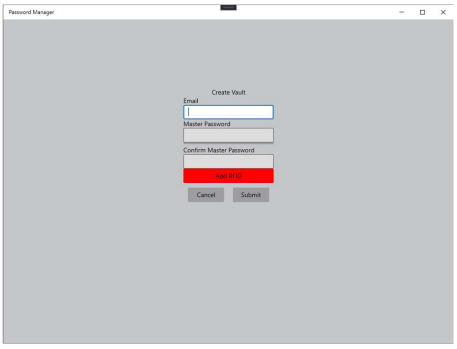

SECURE STRING
DE0001: SecureString shouldn't be used
Motivation
The purpose of SecureString is to avoid having secrets stored in the process memory as plain text.
However, even on Windows, SecureString doesn't exist as an OS concept.
It just makes the window getting the plain text shorter; it doesn't fully prevent it as .NET still has to convert the string to a plain text representation.
The benefit is that the plain text representation doesn't hang around as an instance of System.String -- the lifetime of the native buffer is shorter.
The contents of the array is unencrypted except on .NET Framework.
In .NET Framework, the contents of the internal char array is encrypted. .NET doesn't support encryption in all environments, either due to missing APIs or key management issues.
Recommendation
Don't use SecureString for new code. When porting code to .NET Core, consider that the contents of the array are not encrypted in memory.
The general approach of dealing with credentials is to avoid them and instead rely on other means to authenticate, such as certificates or Windows authentication.


DEMO
Password Manager
Hot Reload
File
Tools
Sync
All Items
Favourites
Social
Bank
Games
Google
google.com
Facebook
facebook.com
Github
github.com
Steam
steam.com
Epic Games
epic.com
Add
Name
Google
Username
Jim
Password
Domain
google.com
Notes
Category
Social
Save
Delete


ANY QUESTIONS?

10.18 Manual - How to

PASSWORD MANAGER MANUAL



Create Vault

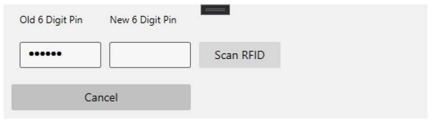
Sign up to the service with your usual email address. The important here is to use a strong password. Remember this is the only password you will need to remember from now on, so make it strong and memorable.

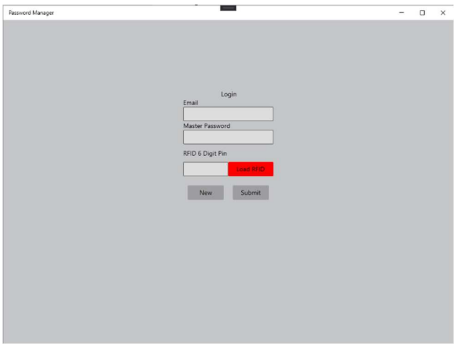
*Hint – One good technique is to choose three words which have nothing in common, stick some capitals, numbers and special characters in there randomly to finish it off.

Add RFID

On the Create Vault page press the 'Add RFID' button to open the pop up. If a brand new MIFARE Classic 1k is being used then the default pin is auto filled, otherwise replace this with the old pin and enter your new 6-digit pin.

*Hint - Make sure your selected pin is made up of a mixture of numbers with no repeating values.





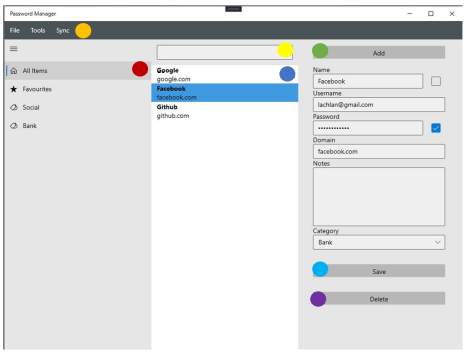
Login

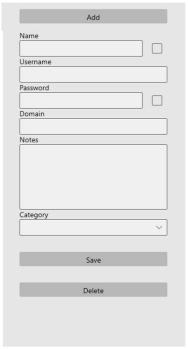
Login using the email and password of the account you just created. Place the RFID card on the reader and enter the 6 Digit RFID pin, press the Load RFID button, this will change to green if the pin was correct.

*Hint - Both boxes are case sensitive so make sure you enter the details exactly as you did when you created the vault.

All Items Page

- **Categories**
- **Credentials**
- **Add Credential Button**
- **Save Credential Button**
- **Delete Credential Button**
- **Search Bar**
- **Settings bar**





Add Credential

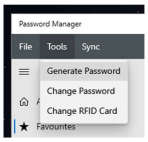
Fill in the information for the credential, it must include a valid site name, domain and password to be able to save.

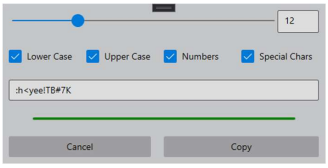
*Hint – Make sure the password is strong, you do not need to remember it so why not use the password generator.

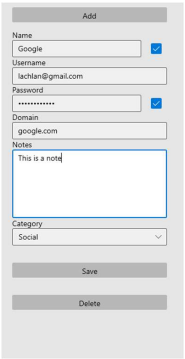
Generate Password

Use the password generator to create a random string of characters. This can then be added to one of your credentials.

*Hint – Make sure you include lower case, upper case, numbers and special characters for the strongest password.





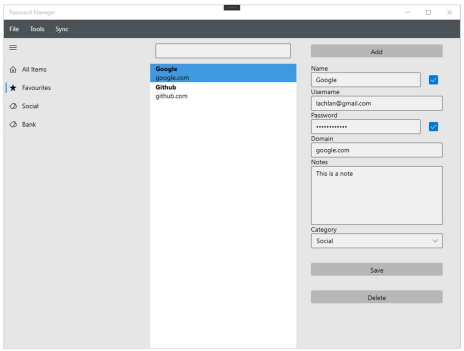


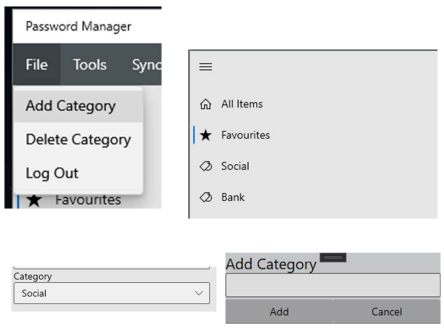
Credentials

Select a credential from the list and change any of the values associated with it. Once done press the save button to keep the changes. Credentials can also be deleted by selecting the credential you wish to delete and pressing the delete button.

Favourites

The check box in the top right of the credential breakdown section is the favourite button. When this is selected the currently displayed credential is added to the favourites. The favourited credentials can be filtered by selecting them on the left-hand navigation panel.



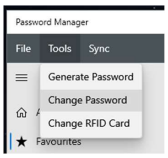


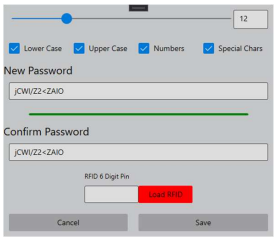
Categories

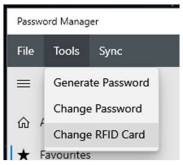
Pressing the File settings menu gives the options to create or delete a category. To delete select the category you wish to remove, and press remove in the settings menu. These categories can be used to filter the credentials from left navigation pane. Categories can be assigned to each credential from the right-hand side when they are selected.

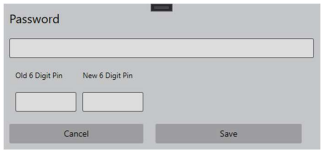
Change Password

Select the Change Password option from the settings menu. The system will suggest an auto generated password for you or allow you to enter your own password. You will also have to renter your 6 Digit RFID pin.







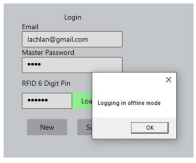


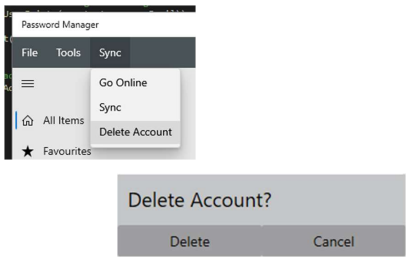
Change RFID Card

Select the Change RFID Card option from the settings menu. You need to enter your password and then the current RFID pin of the card. You can then enter the new 6-digit pin and press save.

Login Offline Mode

If there is currently a local vault file on the computer, you are attempting to login to, but you have no connection to the internet it is possible to login in offline mode. To log back in select the Go Online option from the settings and enter your login information.





Delete Account

It is possible to delete your account, this will delete the local copy as well as any information stored on the server. Simply press the delete account button from the setting menu and press delete to confirm.