# A HOST BASED CREDENTIALS VAULT USING MD5 AND SHA256 HASHING

**By**

**Muhia Alex Karumbi**

**122234**

A Cybersecurity project 1 Submitted to the School of Computing and Engineering Sciences in partial fulfillment of the requirements for the award of the Degree in Bachelor of Science in Computer Networks and Cybersecurity at the School of Computing and Engineering Sciences of Strathmore University

Nairobi, Kenya

June 2023

# 

# Declaration and approval

I declare that this work has not been previously submitted and approved for the award of a degree by this or any University. To the best of my knowledge and belief, the work contains no material previously published or written by another person except where due reference is made in the work itself.

Student’s Name: Muhia Alex Karumbi. - 147479

Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Proposal of **Muhia Alex Karumbi** has been reviewed and approved by **Mr.Tiberius Tabulu**

Supervisor’s Name: Tiberius Tabulu

Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# ABSTRACT

With the increasing reliance on digital services and the proliferation of online accounts, the need for secure and convenient password management has become critical. Currently in the technological space there is an existing gap where lower end devices do not have a password vault which can store passwords safely. Therefore, users are tempted to use common passwords which can be cracked or leaked to the public and lead to loss of sensitive information. This is because users have multiple accounts which all require passwords for logging in, all which cannot be remembered by memory. This project proposes to develop a python-based credentials vault which is will store user passwords for different accounts. The vault will also generate strong and unique passwords that ought to be at least 16 characters long and contain combinations of unique characters and symbols. The credentials vault which will be developed by the Python framework will be able to run on multiple Operating systems by default. The vault will use SHA256 which will be used to hash the master password which will then has the specific passwords after they have been hashed the MD5 hashing algorithm, these hashing algorithms are used by the program to provide safe password hashing and storage. In order to prevent unwanted access, a master password authentication system is also used.

# Table of Contents

[Declaration and approval ii](#_Toc137797466)

[ABSTRACT iii](#_Toc137797467)

[Table of Contents iv](#_Toc137797468)

[List of Figures viii](#_Toc137797469)

[List of Abbreviations ix](#_Toc137797470)

[Chapter 1: Introduction 1](#_Toc137797471)

[1.1 Background Information 1](#_Toc137797472)

[1.2 Problem Statement 2](#_Toc137797473)

[1.3 Objectives 3](#_Toc137797474)

[1.3.1 General Aim 3](#_Toc137797475)

[1.3.2 Specific Objectives 3](#_Toc137797476)

[1.4 Research Questions 3](#_Toc137797477)

[1.5 Justification 4](#_Toc137797478)

[1.6 Scope 4](#_Toc137797479)

[1.7 Limitations and Delimitations 4](#_Toc137797480)

[Chapter 2: Literature Review 6](#_Toc137797481)

[2.1 Introduction 6](#_Toc137797482)

[2.2 Digital User Identities 6](#_Toc137797483)

[2.2.1 Principles Guiding Digital Identities 7](#_Toc137797484)

[2.2.2 Parameters and Considerations in Password vaults 9](#_Toc137797485)

[2.3 Hashing Algorithms for Password Storage 10](#_Toc137797486)

[2.4 Password Vaults and Password Storage 11](#_Toc137797487)

[2.5 Related Works 11](#_Toc137797488)

[2.5.1 Mac OS Keychain 11](#_Toc137797489)

[2.5.2 Google Passwords 12](#_Toc137797490)

[2.5.3 Last Pass 13](#_Toc137797491)

[2.5 Conceptual Framework 13](#_Toc137797492)

[Chapter 3: Methodology 15](#_Toc137797493)

[3.1 Introduction 15](#_Toc137797494)

[3.2 Development Methodology 15](#_Toc137797495)

[3.2.1 Requirement Analysis 15](#_Toc137797496)

[3.2.2 Design 16](#_Toc137797497)

[3.2.3 Prototyping 16](#_Toc137797498)

[3.2.4 Customer Evaluation 16](#_Toc137797499)

[3.2.5 Refining Prototype 16](#_Toc137797500)

[3.3 Justification of the Methodology 17](#_Toc137797501)

[3.4 Tools and Techniques 17](#_Toc137797502)

[3.4.1 Python Programming Language 17](#_Toc137797503)

[3.4.2 Integrated Development Environment (IDE) 17](#_Toc137797504)

[3.4.3 Version Control System 17](#_Toc137797505)

[3.4.4 Cryptographic Libraries 17](#_Toc137797506)

[3.4.5 SQLite3 Database 18](#_Toc137797507)

[3.4.6 Tkinter 18](#_Toc137797508)

[Chapter 4: System Design 19](#_Toc137797509)

[4.1 Functional Requirements 19](#_Toc137797510)

[4.1.1 User Registration and Authentication 19](#_Toc137797511)

[4.1.2 Credentials Management 19](#_Toc137797512)

[4.1.4 Password Generation 19](#_Toc137797513)

[4.2 Non-Functional Requirements 20](#_Toc137797514)

[4.3 Use Case Diagram 21](#_Toc137797515)

[4.4 System Architecture Diagram 21](#_Toc137797516)

[4.5 Sequence Diagram 22](#_Toc137797517)

[4.6 Entity-Relationship Diagram 24](#_Toc137797518)

[4.7 Class Diagram 24](#_Toc137797519)

[4.8 State Machine Diagram 25](#_Toc137797520)

[Appendices 29](#_Toc137797531)

[Appendix 1: Gantt chart for this Project 29](#_Toc137797532)

[Appendix 2: Project Proposal Marking Guide 30](#_Toc137797533)

# List of Figures

[Figure 2.1 NIST's Digital Identity Model 6](#_Toc137793311)

[Figure 2.2 Conceptual Framework 14](#_Toc137793312)

[Figure 3.1 Prototype Model 15](#_Toc137793313)

[Figure 4.1 Use Case Diagram 21](#_Toc137793314)

[Figure 4.2 System Architecture Diagram 22](#_Toc137793315)

[Figure 4.3 Sequence Diagram 24](#_Toc137793316)

[Figure 4.4 Entity-Relationship Diagram 24](#_Toc137793317)

[Figure 4.5 Class Diagram 25](#_Toc137793318)

[Figure 4.6 State-Machine Diagram 26](#_Toc137793319)

# List of Abbreviations

DFD –Data Flow Diagram

ERD –Entity-Relation Diagram

FSM-Finite State Machine Diagram

MD5 -Message Digest Algorithm 5

SHA256 -Secure Hash Algorithm 256-bit

# Introduction

## 1.1 Background Information

Authentication plays a critical role in today’s society which includes access to our personal accounts. There are digital identities of different users from individuals to organizations and even companies which all have accounts which are password protected. These accounts all require credentials for the protection of sensitive information to avoid being leaked and most of the times it is use of passwords (Alhothaily et al, 2017).

Users often reuse the same password across different websites for their multiple accounts or they either slightly change the password for a different site. The importance of robust password management cannot be overstated. Weak or compromised passwords can result in unauthorized access to personal, financial, or sensitive information, leading to identity theft, data breaches, and other cybersecurity incidents. Additionally, password-related vulnerabilities can have severe consequences for businesses and organizations, including financial losses and damage to their reputation (Alhothaily et al, 2017).

Considering the challenges mentioned above, the development of a password vault presents a compelling solution. A password vault is a secure digital repository that allows individuals to store and manage their passwords in an organized and encrypted manner. By employing a password vault, individuals can generate and store complex, unique passwords for each online account, significantly enhancing the security of their digital identities (Raymond and Jacques, 2018).

The use of encryption and hashing algorithms in password vaults adds an additional layer of security by protecting stored passwords and making them more resistant to unauthorized access. Encryption transforms passwords into unreadable cipher text using algorithms like AES and RSA, ensuring sensitive information remains protected. Hashing algorithms, such as MD5, SHA-1, SHA-256, and bcrypt, convert passwords into irreversible hash values that are stored instead of the actual passwords (Cambou, 2019).

Therefore based on the challenges in handling credentials and the possibility of using vaults for this, there is need for host based password vaults that help end users securely and conveniently store their credentials for easier and secure retrieval for subsequent account activities .The use of MD5 and SHA256 hashing algorithms can be used as an additional layer of security to the password storage process. These widely recognized and robust hashing algorithms provide encryption mechanisms that make it computationally costly for an attacker to reverse-engineer the original passwords from their hashed representations. The password vault's objectives extend beyond secure password storage. It seeks to offer features that streamline password management, such as generating strong and unique passwords for new accounts, promoting good password hygiene practices.

## 1.2 Problem Statement

The widespread use of online accounts and the increasing frequency of data breaches have highlighted the critical need for secure password management. Individuals often struggle to maintain good password hygiene practices such as strong and unique passwords across multiple accounts. This leads to weak passwords and passwords reuse, which leaves their accounts vulnerable and more susceptible to compromise by malicious users which exposes individuals to significant risks, including unauthorized access, identity theft, and compromised personal information.

This work thus proposes a host-based password vault that utilizes MD5 and SHA256 hashing algorithms the project will leverage on the, Tkinter framework and hashlib libraries, these robust mechanisms that make it extremely difficult for attackers to crack, brute-force or decipher stored passwords. The vault will also be initialized by system once the user enters the master password. The project also incorporates password generation as a feature that, facilitates the creation of strong, unique passwords that meet the minimum threshold for length and complexity for new accounts. This will reduce the vulnerability of user accounts and the associated losses in data and value

## 1.3 Objectives

### 1.3.1 General Aim

The aim of the project is to create a Python-Based Password Vault Application which will use hashing algorithms like MD5 and SHA256 for password access and storage of multiple user accounts through a single master password.

### 1.3.2 Specific Objectives

This are the specific objectives:

1. To review digital user credential storage principles, parameters and considerations.
2. To investigate different hashing algorithms that are used in password vaults access and password storage.
3. To review the gaps in the digital identities’ credential storage cyber security space on password vaults.
4. To develop a python-based OS initiated password vault for multi account user credential management through a single master password.
5. To test the password vault system.

## 1.4 Research Questions

1. What are the principles, parameters and considerations of user credential storage?
2. What are the different encryption and hashing algorithms used in password vaults for access control and password storage?
3. What are the existing gaps in password practices related to the storage of digital identities' credentials in password vaults?
4. How can a python-based password vault be developed that allows for the management of multiple user credentials through a single master password?
5. How is the credentials vault going to handle user data and process the different user password combinations?

## 1.5 Justification

The actualization of this project will help solve the problem of poor password hygiene by enhancing enhances password security which will prevent unauthorized access of accounts through use of robust encryption and storage mechanisms .This will help prevent associated losses in data and value to individuals and even organizations. The project helps protect personal and sensitive information, reducing the risk of password leaks and potential data breaches. Moreover, the project promotes better password hygiene for the user by implementing a random password generator .This reduces the risk of data loss which can lead to serious personal breaches of information.

## 1.6 Scope

The scope of the password vault project encompasses several key areas and considerations. Firstly, the project will revolve around the development of a Python-based password vault application. This entails designing and implementing features that enable users to securely store, manage, and retrieve their passwords. Additionally, the project will involve the implementation of the MD5 and SHA256 hashing algorithms for password encryption. Integrating these algorithms into the password vault application will ensure the secure storage of passwords in hashed form, enhancing overall data security. Another aspect of the project's scope is the implementation of a master password authentication mechanism. Users will be required to enter a master password to access the password vault and retrieve their stored passwords. The project will focus on designing and implementing a secure authentication process to protect against unauthorized access.

The password vault application which will be designed by the Python framework by default will be able to run on multiple Operating Systems. Ensuring multi-platform support requires considerations and adaptations to ensure compatibility and functionality across different operating systems.

## 1.7 Limitations and Delimitations

While the project aims to create a secure password vault application, it is important to acknowledge that no system can be completely immune to all security risks. The project will mitigate known vulnerabilities, but it may not address all potential threats or evolving security concerns.

The project focuses on developing a password vault application with multi-platform support for Windows, macOS, and Linux. However, due to technical constraints or differences in operating systems, certain functionalities or features may not be fully compatible across all platforms.

The project aims to create a user-friendly password vault, but individual users may have unique requirements or preferences that cannot be fully addressed within the project's scope. Customization options may be limited to ensure a streamlined and standardized user experience.

Finally, the project's scope is contingent on external factors such as availability of resources, third-party libraries, and APIs. Any limitations or constraints imposed by these external dependencies may affect the overall development and functionality of the password vault.

# Literature Review

## 2.1 Introduction

In this chapter, related works according to the proposal at hand are discussed as well as the gaps that exist which show the significance of the proposal project. A conceptual framework which is a roadmap of how the system is working will follow.

## 2.2 Digital User Identities

Digital user credentials refer to the information that uniquely identifies and authenticates an individual for accessing digital systems, services, or resources. They typically consist of a combination of username, password, and sometimes additional factors such as biometrics, security questions, or one-time passwords. User credentials are essential for establishing the identity of users and granting them authorized access to protect digital websites and their respective accounts (Alhothaily et al, 2017).

The various entities and interactions that comprise the digital identity model used here are illustrated in Figure 2.1 (Grassi et al, 2017)

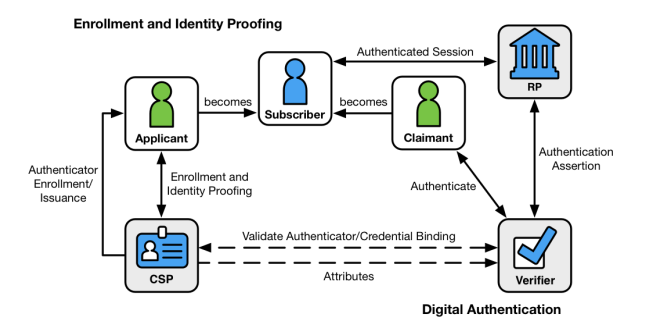


Figure . NIST's Digital Identity Model

The left side of the diagram shows the enrollment, credential issuance, lifecycle management activities, and various states of an identity proofing and authentication process. The usual sequence of interactions is as follows:

1. An applicant applies to a CSP through an enrollment process.
2. The CSP identity proofs that applicant. Upon successful proofing, the applicant becomes a subscriber.
3. Authenticator(s) and a corresponding credential are established between the CSP and the subscriber.
4. The CSP maintains the credential, its status, and the enrollment data collected for the lifetime of the credential (at a minimum). The subscriber maintains his or her authenticator(s).

The right side of Figure 2.1 shows the entities and interactions involved in using an authenticator to perform digital authentication. A subscriber is referred to as a claimant when he or she needs to authenticate to a verifier. The interactions are as follows:

1. The claimant proves possession and control of the authenticator(s) to the verifier through an authentication protocol.
2. The verifier interacts with the CSP to validate the credential that binds the subscriber’s identity to their authenticator and to optionally obtain claimant attributes.
3. The CSP or verifier provides an assertion about the subscriber to the RP, which may use the information in the assertion to make an authorization decision.
4. An authenticated session is established between the subscriber and the RP.

### 2.2.1 **Principles Guiding Digital Identities**

In password vaults, several underlying principles are implemented to ensure robust security and effective password management. Some of these principles include open designs, complete mediation, security by obscurity, fail safe defaults and authentication mechanisms.

Open designs: Password vaults often follow the principle of open designs, which means that the implementation and security mechanisms of the vault are transparent and open to scrutiny. Open designs allow for peer review, independent audits, and security assessments, which contribute to identifying potential vulnerabilities and enhancing overall security.

Complete mediation: Complete mediation is a principle that emphasizes the need for all accesses to sensitive resources, such as stored passwords in a vault, to be validated and authorized. In the context of password vaults, complete mediation ensures that every access request to retrieve or modify a stored password goes through a validation and authorization process. This principle helps prevent unauthorized access to sensitive information and ensures that only authenticated and authorized users can retrieve their stored passwords (Stallings and Brown, 2015).

Fail-safe defaults: Fail-safe defaults refer to the principle of configuring a system or application with secure settings as the default, ensuring that it operates securely even if the user does not make any additional configuration changes. In the case of password vaults, fail-safe defaults could include features like automatically enabling two-factor authentication, enforcing strong password policies, or implementing encryption by default. Fail-safe defaults help protect users who may not have extensive knowledge of security practices or who may overlook certain configuration options, ensuring that the password vault operates with a high level of security out-of-the-box (Stallings and Brown, 2015).

Security by obscurity: Security by obscurity is a principle that suggests that the security of a system should not rely solely on keeping its inner workings or design secret. While open designs promote transparency, security by obscurity acknowledges that certain aspects of the system may benefit from being less predictable or less known to potential attackers. In the context of password vaults, security by obscurity can involve obfuscating certain implementation details or using non-standard approaches to protect against common attack vectors. However, it is important to note that security by obscurity should not be relied upon as the sole line of defense, but rather complement other robust security measures (Stallings and Brown, 2015).

### 2.2.2 Parameters and Considerations in Password vaults

Authentication mechanisms: Authentication mechanisms play a crucial role in password vaults to verify the identity of users and protect access to stored passwords. Common authentication mechanisms include the use of a master password, two-factor authentication (2FA), biometric authentication, or hardware-based security tokens. These mechanisms provide an additional layer of security by requiring users to provide something they know (e.g., password) or something they have (e.g., physical token) to gain access to the password vault. Robust authentication mechanisms are essential to prevent unauthorized access to stored passwords (Stallings and Brown, 2015).

There are several important considerations and parameters to keep in mind. First, it's essential to ensure that passwords are of sufficient length which is of recommended 16 characters long. Longer passwords provide more possible combinations, making them harder for attackers to crack. Additionally, passwords should be complex, incorporating a combination of uppercase and lowercase letters, numbers, and special characters. This complexity adds an extra layer of security by salting the password ([Zhang-Kennedy](https://ieeexplore.ieee.org/author/37085790633) et al, 2016).

Another critical consideration is to use unique passwords for each account or service. Reusing passwords across multiple accounts increases the risk of a single password compromise leading to unauthorized access to multiple accounts. It's also important to avoid common patterns or easily guessable information in passwords. Sequential numbers, birthdates, and repetitive characters should be avoided as they make passwords predictable and easier to crack ([Zhang-Kennedy](https://ieeexplore.ieee.org/author/37085790633) et al, 2016).

Regularly changing passwords is a good practice, particularly for critical accounts. This reduces the risk of long-term exposure and unauthorized access. Enabling two-factor authentication (2FA) whenever possible increases the security of the passwords. With 2FA, a second form of verification, such as a unique code sent to a mobile device, is required along with the password ([Zhang-Kennedy](https://ieeexplore.ieee.org/author/37085790633) et al, 2016).

## 2.3 Hashing Algorithms for Password Storage

Different hashing algorithms are used in password management and security systems to transform passwords or other data into unique hash values.

MD5 (Message Digest Algorithm 5): MD5 is a widely known and frequently used hashing algorithm. It takes an input message and produces a 128-bit hash value. However, MD5 is considered weak for password hashing due to its vulnerability to collision attacks, where two different inputs can produce the same hash value. (Zhouyo et al, 2022).

SHA-1: SHA-1 produces a 160-bit hash value, which is represented as a sequence of 40 hexadecimal characters. It takes an input message and applies a series of logical and arithmetic operations to generate the hash value. SHA-1 was widely used in the past for various cryptographic applications, including password storage and digital signatures. However, SHA-1 has been found to be vulnerable to collision attacks, where two different inputs can produce the same hash value (Rohit et al, 2019).

SHA-256: SHA-256, on the other hand, produces a 256-bit hash value, represented as a sequence of 64 hexadecimal characters. It is a more secure and robust algorithm compared to SHA-1. SHA-256 applies a more complex set of operations and transformations to the input message, resulting in a larger hash size and increased resistance to attacks. It is designed to be computationally secure against collision attacks, pre-image attacks, and second pre-image attacks. SHA-256 is widely used in various cryptographic applications, including password storage, digital certificates, integrity checking, and data integrity verification (Rohit et al, 2019).

bcrypt: bcrypt is a specialized hashing algorithm designed specifically for password hashing. It utilizes a variation of the Blowfish encryption algorithm, incorporating salt and a configurable number of iterations. The use of salt adds a random component to the hashing process, making it more challenging for attackers to precompute hash tables (rainbow tables) for password cracking. bcrypt is known for its slow computational speed, which makes it resistant to brute-force and dictionary attacks (Stallings and Brown, 2015).

## 2.4 Password Vaults and Password Storage

A password vault, also known as a password manager, is a software based application designed to securely store and manage passwords for various online accounts and services. In today's digital age, where individuals often have numerous online accounts, it can be challenging to remember complex and unique passwords for each account. Password vaults offer a convenient and secure solution to this problem by providing a centralized and encrypted storage system for passwords (Raymond and Jacques, 2018).

The hashed versions of passwords play a critical role in password vaults. Hashing algorithms are used to transform passwords into unique hash values, which are stored in the password vault instead of the actual passwords. Hash functions are one-way mathematical algorithms that convert input data, such as passwords, into fixed-length strings of characters. These hash values are not reversible, meaning they cannot be easily converted back to the original passwords. This ensures that even if the password vault is compromised, the actual passwords remain protected (Raymond and Jacques, 2018).

## 2.5 Related Works

### 2.5.1 Mac OS Keychain

The Mac OS Keychain is a built-in password management system for Apple devices, including macOS for the MacBook and iOS devices such as iPhone. It securely stores passwords, credit card information, and other credentials on the device and they are accessed via either biometric authentication which is face ID or through the device’s password.

Pros:

1. Seamless Integration: The Keychain is seamlessly integrated into the Apple ecosystem, providing easy access to stored passwords across Apple devices.
2. Biometric Authentication: Keychain supports biometric authentication methods such as Touch ID and Face ID, adding an extra layer of convenience and security.

Cons:

1. Limited Cross-Platform Compatibility: The Keychain is primarily designed for Apple devices, limiting its usefulness for users who use non-Apple platforms or devices.
2. Dependency on Apple Ecosystem: Users heavily invested in the Apple ecosystem will benefit most from the Keychain, while users of other platforms may find limited compatibility (Apple Developer, 2023).

### 2.5.2 Google Passwords

Google Passwords is a password management feature offered by Google, providing password storage and autofill capabilities across Google services and platforms. It is a web based password vault which prompts you on entry of user credentials to save the user’s credentials. The user credentials can then be accessed later via the user’s Google account password.

Pros:

1. Seamless Integration with Google Services: Google Passwords integrates smoothly with various Google services, making it convenient to use and access passwords across android devices.
2. Advanced Security Measures: Google implements strong security measures such as two-factor authentication prevent unwanted access.
3. Password Health Monitoring: Google Passwords alerts users of weak or compromised passwords, promoting better password hygiene.

Cons:

1. Limited to Google Ecosystem: Google Passwords is primarily optimized for Google services and may not offer the same level of integration or compatibility with other platforms or applications.
2. Privacy Concerns: There may be concerns regarding data privacy and the collection of user information which is shared to Google and used even for ad purposes.
3. Limited Cross-Browser Support: While Google Passwords works well with the Google Chrome browser, compatibility with other web browsers may be limited (Google, 2023).

### 2.5.3 Last Pass

LastPass is a popular third-party password manager that provides a secure platform to store and manage passwords across multiple devices and platforms. It safely stores passwords and grants access to the technology and services people rely on every day using sign-on and password management.

Pros:

1. Cross-Platform Support: LastPass is compatible with various operating systems and browsers, allowing users to access their passwords from different devices.
2. Secure Password Generation: LastPass offers a password generator that creates strong and unique passwords which promotes password hygiene.
3. Advanced Features: LastPass provides additional features such as secure password sharing, emergency access, and multi-factor authentication, enhancing password management capabilities.

Cons:

1. Dependency on Master Password: The security of LastPass relies heavily on the strength and protection of the user's master password. Compromising the master password could lead to unauthorized access to all stored passwords.
2. Limited Free Version Features: While LastPass offers a free version, some advanced features and functionalities are available only in the premium version (LastPass, 2023).

## 2.5 Conceptual Framework

The figure 2.2 is a simplified illustration that shows how the application works. The user logs in via a master password, the master password is then hashed using SHA256.The user is then able to add his user credentials which include his account and password. The user is also able to generate random passwords which are then hashed by MD5 and further hashed using the encrypted master password. The hashed passwords are then stored in a MySQL database.

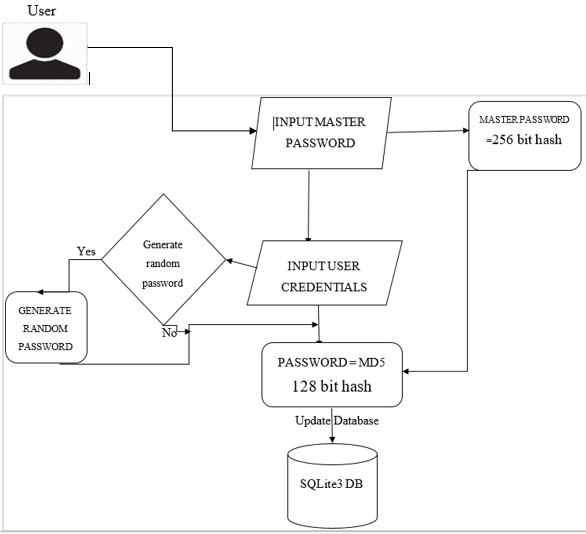


Figure . Conceptual Framework

# Methodology

## 3.1 Introduction

In this chapter, the methodology used in the developed system is discussed together with the requirements. It also highlights the system development tools and techniques that have been used in the development of the system.

## 3.2 Development Methodology

The methodology will involve the use of a prototyping approach. Prototyping is an iterative process that involves creating a preliminary version of the password vault system to gather feedback, identify potential issues, and refine the design. This approach allows for early user involvement and ensures that the final system meets user requirements effectively. Figure 3.1 shows the graphical illustration of the model.

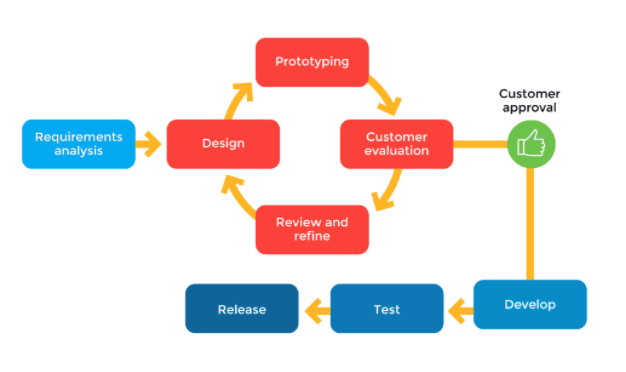


Figure . Prototype Model

### 3.2.1 Requirement Analysis

In this initial stage, the requirements for the password vault system are gathered and analyzed. The information will be gathered by research which will include literature review of existing works on password vaults. It will then be analyzed by use of qualitative analysis such as content analysis. This includes identifying the key features, functionality, and security requirements of the system. Requirements are documented and used as a basis for the subsequent design and development phases. For instance the user requirements for this project will be the various python frameworks and libraries that will be used to run the project.

### 3.2.2 Design

The design phase focuses on creating a conceptual and technical design for the credentials vault system. This includes defining the system architecture, user interface design, database structure, encryption and hashing algorithms, and error handling mechanisms. The design phase establishes the blueprint for the system's implementation.

### 3.2.3 Prototyping

The prototype development stage involves creating an initial working version of the credentials vault system. This is a basic implementation that demonstrates key functionalities and interactions. The prototype is typically developed iteratively, allowing for feedback and refinements based on user testing and evaluation. This will use tools such as Python programming language, various libraries and frameworks such as hashlib and Tkinter and also Git.

### 3.2.4 Customer Evaluation

The prototype will be self-tested as each prototype is developed which will provide feedback on its usability, functionality, and security. User testing helps identify any usability issues, potential vulnerabilities, or areas for improvement. Feedback is gathered and used to refine the prototype in subsequent iterations.

### 3.2.5 Refining Prototype

The prototype and the various features will be tested as the development of the application is ongoing which will be used to refine and enhance any identified issues or shortcomings. This iterative refinement process allows for continuous improvement of the system, ensuring that it meets user requirements and aligns with the research objectives.

## 3.3 Justification of the Methodology

By using prototypes, one can progressively develop and refine the credentials vault system, incorporating user feedback and ensuring it meets their needs. Prototyping allows for early validation of the system's feasibility and reducing the system’s vulnerabilities. Also, the product is tested very frequently hence minimizing the risk of any major failures in the future once the final implementation is deployed.

## 3.4 Tools and Techniques

### **3.4.1** Python **Programming Language**

The project will use Python as the primary programming language for implementing the password vault system. Python is widely known for its simplicity, readability, and extensive libraries, making it suitable for rapid development and integration with cryptographic algorithms.

### **3.4.2 Integrated Development Environment (IDE)**

An IDE such as PyCharm will be utilized for coding, debugging, and managing the development process. These IDEs provide features like code completion, syntax highlighting, and debugging tools to enhance the efficiency and effectiveness of the development workflow.

### **3.4.3 Version Control System**

Git will be employed to track changes, manage different versions of the source code, and facilitate collaboration among team members. Git allows for easy branching, merging, and rollback of code changes, ensuring a smooth and organized development process.

### 3.4.4 Cryptographic Libraries

The project will leverage existing cryptographic libraries, such as hashlib and cryptography, to implement hashing algorithms. These libraries provide functions and methods for secure password hashing (e.g., SHA-256) and encryption (e.g., AES) to ensure the confidentiality and integrity of user credentials.

### 3.4.5 SQLite3 Database

In the credentials vault project, SQLite3 will be used to create and manage the database where user credentials, such as usernames, passwords, and associated metadata, will be stored. The system will interact with the SQLite3 database using SQL queries, allowing for efficient data storage, retrieval, and manipulation.

### 3.4.6 Tkinter

Tkinter is a Python library used for creating graphical user interfaces (GUIs). In the credentials vault project, Tkinter will be utilized to design and develop the user interface of the password vault application. It provides a set of widgets and tools for creating windows, buttons, text boxes, and other GUI components, allowing users to interact with the password vault system in an intuitive and user-friendly manner.

# System Design

All the system requirements have been gathered and analyzed to come up with an overview of the system architecture and functionality. The analysis diagrams include the Use Case diagram, Sequence diagram, Data Flow Diagram, ERD Diagrams, Class Diagrams and the FSM Diagram.

## 4.1 Functional Requirements

### 4.1.1 User Registration and Authentication

The system should allow users to register and authenticate themselves securely. This involves creating an account and logging in using unique credentials. The system must implement robust authentication mechanisms to verify users' identities and grant access only to authorized individuals which will be the use of a master password. This master password will act as the key to access the password vault and should adhere to strong password policies to ensure its effectiveness. The master password once entered will be passed to a hashing function which will hash it into 256 bits and then stored in a DB.

### 4.1.2 Credentials Management

Another one of the important functional requirement of the application is its password management and security feature which serves as the main marketing purpose. The password management functionality includes multiple use cases such as adding a password, removing a password, updating and viewing password list. This is a high priority functional requirement for the application which gives the customer the ability to save user credentials under secure encryption. The credentials vault should provide options for securely storing, updating, and deleting passwords. Users should have the ability to categorize passwords based on different accounts or services.

### 4.1.3 Password Generation

Another major functional requirement adhering to this application is the password generator feature of the application as it is for the user’s added comfort. The feature is an added to make the user’s life easier since it contains multiple password complication options for the customer to choose from to generate a strong complex and random password for their use, tailored to their specific requirements. The uniqueness and complexity of these passwords can be tested with the strength analyzer to verify that the password is strong and complex in terms of good password hygiene.

## 4.2 Non-Functional Requirements

This section of the report measure the important non-functional requirements mentioned in the report against the current prototype and plans regarding the future development.

1. Security – One of the most important non-functional requirements which this application should be in accordance with is security which includes the security regarding user login and registration credentials, security regarding passwords the application saves for the user and security around all the additional features the application provides to avoid leaks and exploitation. For the future development security requirements are to be satisfied to avoid any exploitation through input fields such has SQL injections through input sanitation, secure design principle development and design pattern implementation. The current application prototype also includes security measures to satisfy the security requirements at a basic stage of not being accessible until successful login.
2. Performance efficiency – The performance efficiency of the current prototype is measured to observe and conclude that the prototype is performing its functions with minimum possible response times and required resources and without putting too much load on the processing thus giving the desired output to the customer within a good time frame and performing all its required functions.
3. Data integrity: Data integrity is also a major non-functional requirement that ties in with the applications features and processes since it is utilizing a lot of customer oriented data including their credential, for recognition and authentication, passwords saved, passwords used with application feature suite functionalities etc. regarding future development I plan to encrypt these the credentials for security and data integrity maintenance purposes.
4. Reliability: For the current prototype the reliability as a non-functional requirement can also be measured since it is visible through the request and communication between the front-end and the respective back-end servers should be completed and all the functions mentioned are successfully performed with respective incoming response etc. thus receiving the desired outputs and performing the required function at the same time.

## 4.3 Use Case Diagram

This diagram is an illustration of the functionalities or use cases of the credentials vault system from the perspective of its users (actors). It shows the interactions between actors and system use cases. It shows the various functionalities that the user can perform such as entering the master password, adding their credentials and then storage and retrieval of the credentials.

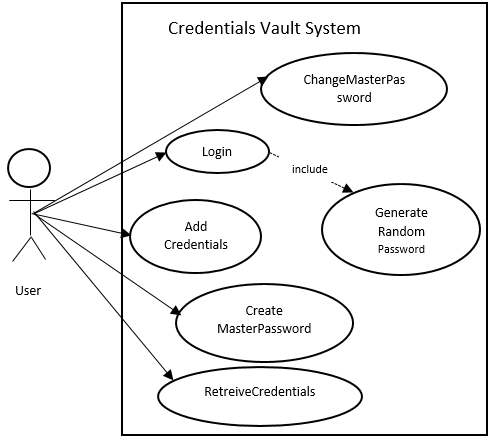


Figure . Use Case Diagram

## 4.4 System Architecture Diagram

The system architecture diagram provides a comprehensive overview of the structure and organization of the password vault system. It depicts the various components and their relationships, showcasing how they interact to fulfill the system's functionalities. The diagram includes components such as the User, User Interface (UI), Password Vault, and Database. The User Interface serves as the front-end through which users interact with the system. The Login Module ensures secure user authentication, allowing only authorized users to access the system. The Password Vault component securely stores and manages user passwords, interacting with other modules for password operations. The Database serves as the storage mechanism for persistently storing user data.

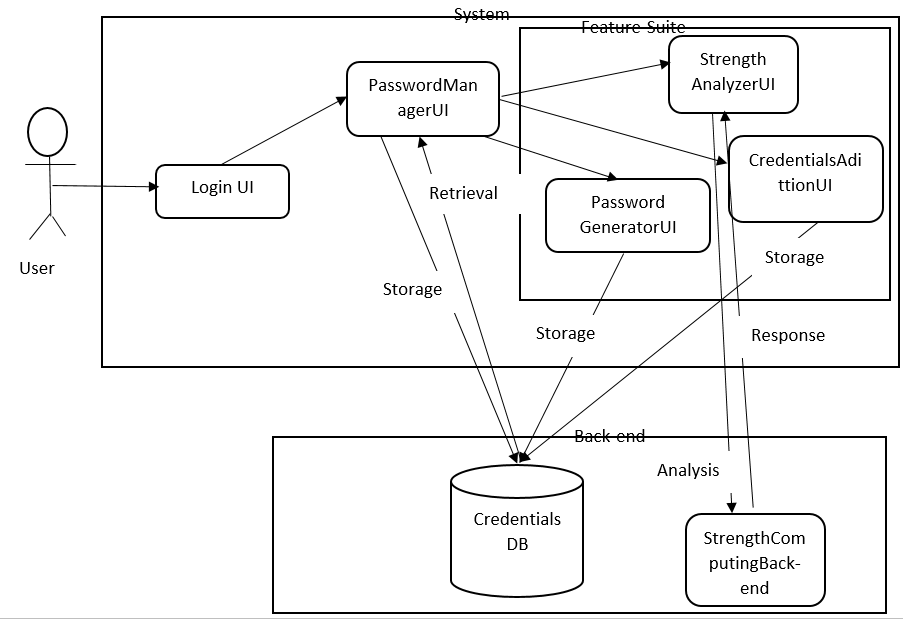


Figure . System Architecture Diagram

## 4.5 Sequence Diagram

This diagram shows the interactions and message flow between different modules in the system. It captures the dynamic behavior of the system and can be used to represent scenarios like user login, credential storage, or retrieval. It shows the sequence of method calls and the order of interactions between the different modules and the specific time intervals.

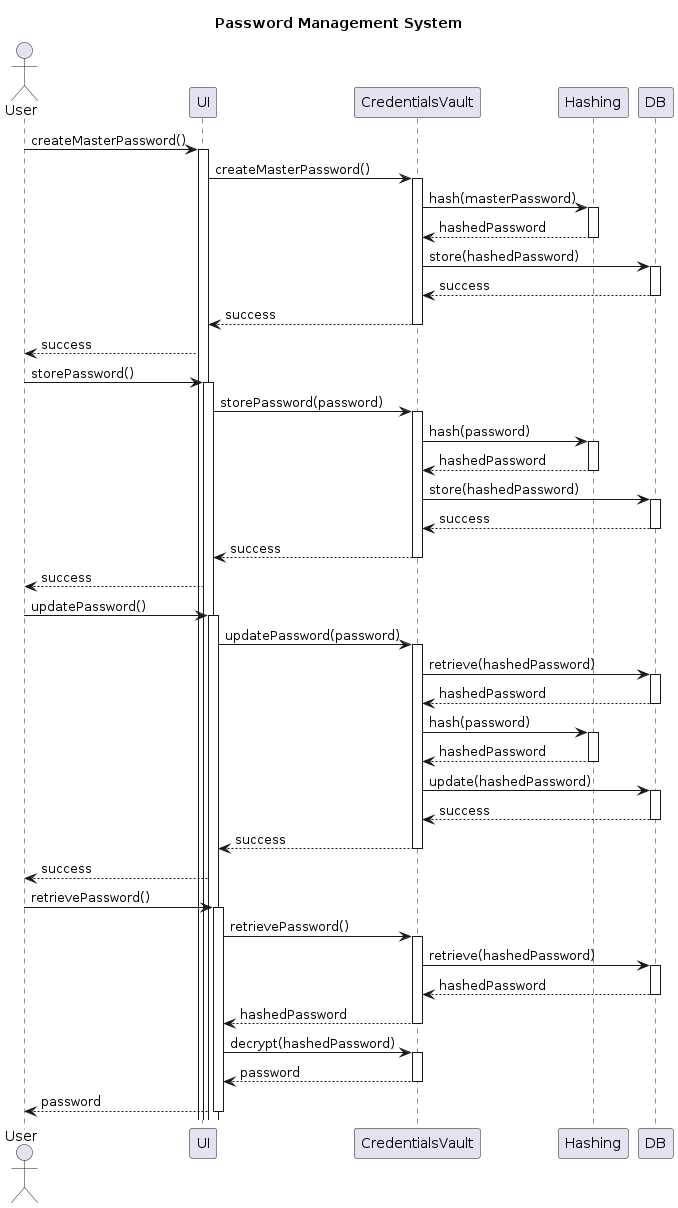


Figure . Sequence Diagram

## 4.6 Entity-Relationship Diagram

The ERD diagram represents the entities, attributes, and relationships in the database schema. It visualizes the data model and shows the relationships between entities like User, Credential, and Vault. It also indicates the cardinality and constraints of the relationships.

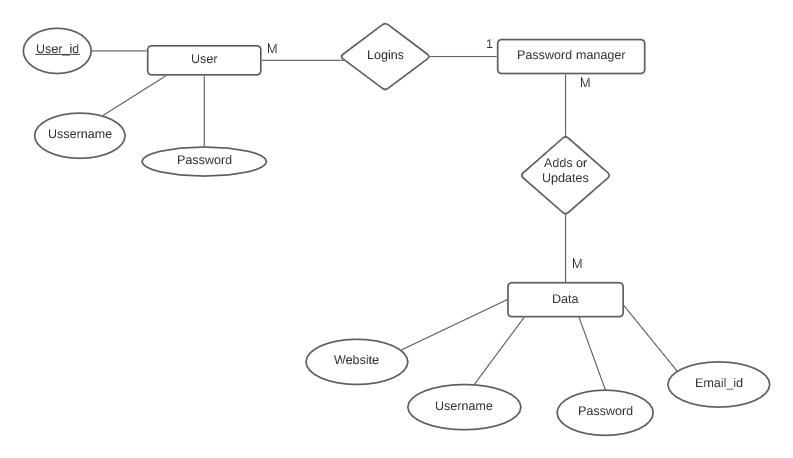


Figure . Entity-Relationship Diagram

## 4.7 Class Diagram

The Class Diagram provides a visual representation of the classes and their relationships. The main classes include User, Credentials, Database (Vault). The User class represents system users and contains attributes like username and password. The Database class handles data storage and access. The Credentials class stores account-related information. The relationships between these classes are depicted through associations, inheritance, and composition.

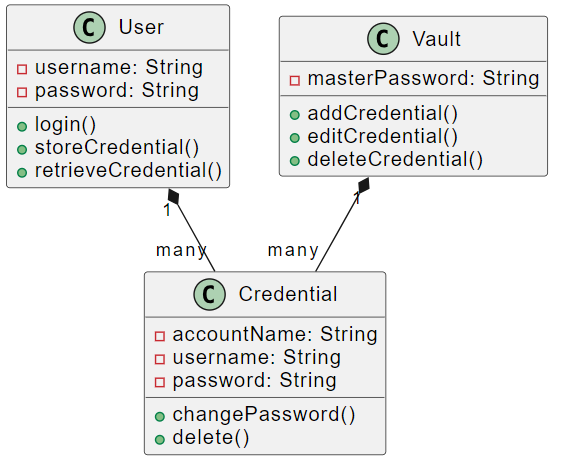


Figure . Class Diagram

## 4.8 State Machine Diagram

This diagram represents the different states and state transitions of the Credential vault system. It can depict the states like "Locked", "Unlocked," "Interrupt," "Error," and the transitions between these states based on user actions or system events.

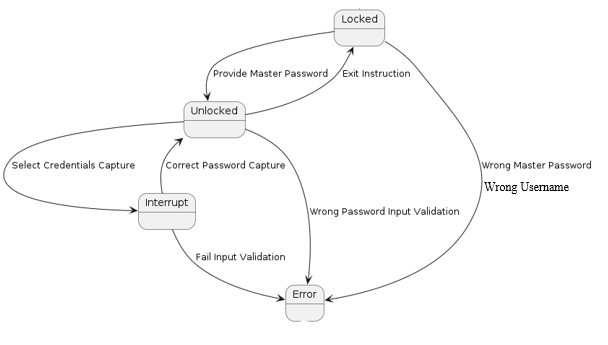


Figure . State-Machine Diagram

# References

Abdulrahman Alhothaily1,2, Chunqiang Hu1,3, (Member, Ieee), Arwa Alrawais1,4, (Member, Ieee), Tianyi Song1, Xiuzhen Cheng1 , (Fellow, Ieee), And Dechang Chen5 (2017 ,July ) A Secure and Practical Authentication Scheme Using Personal Devices.

Apple Inc. (2023) Keychain Access User Guide. <https://support.apple.com/en-ke/guide/keychain-access/kyca1083/mac>

Bertrand Cambou (2019, September) Password Manager Combining Hashing Functions and Ternary PUFs

Google (2023) Save, Manage and Protect your Passwords. <https://support.google.com/accounts/answer/6208650?hl=en&sjid=15797297605237264499-EU>

H. Zhuoyu and L. Yongzhen, "Design and implementation of efficient hash functions," 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 2022, pp. 1240-1243, doi: 10.1109/ICPECA53709.2022.9719176.

L. Zhang-Kennedy, S. Chiasson and P. van Oorschot, "Revisiting password rules: facilitating human management of passwords," 2016 APWG Symposium on Electronic Crime Research (eCrime), Toronto, ON, Canada, 2016, pp. 1-10, doi: 10.1109/ECRIME.2016.7487945.

LastPass (2023) Last Pass Technical Whitepaper.

Paul A. Grassi, Michael E. Garcia, James L. Fenton (2017) *Digital Identity Guidelines.* National Institute of Standards and Technology.

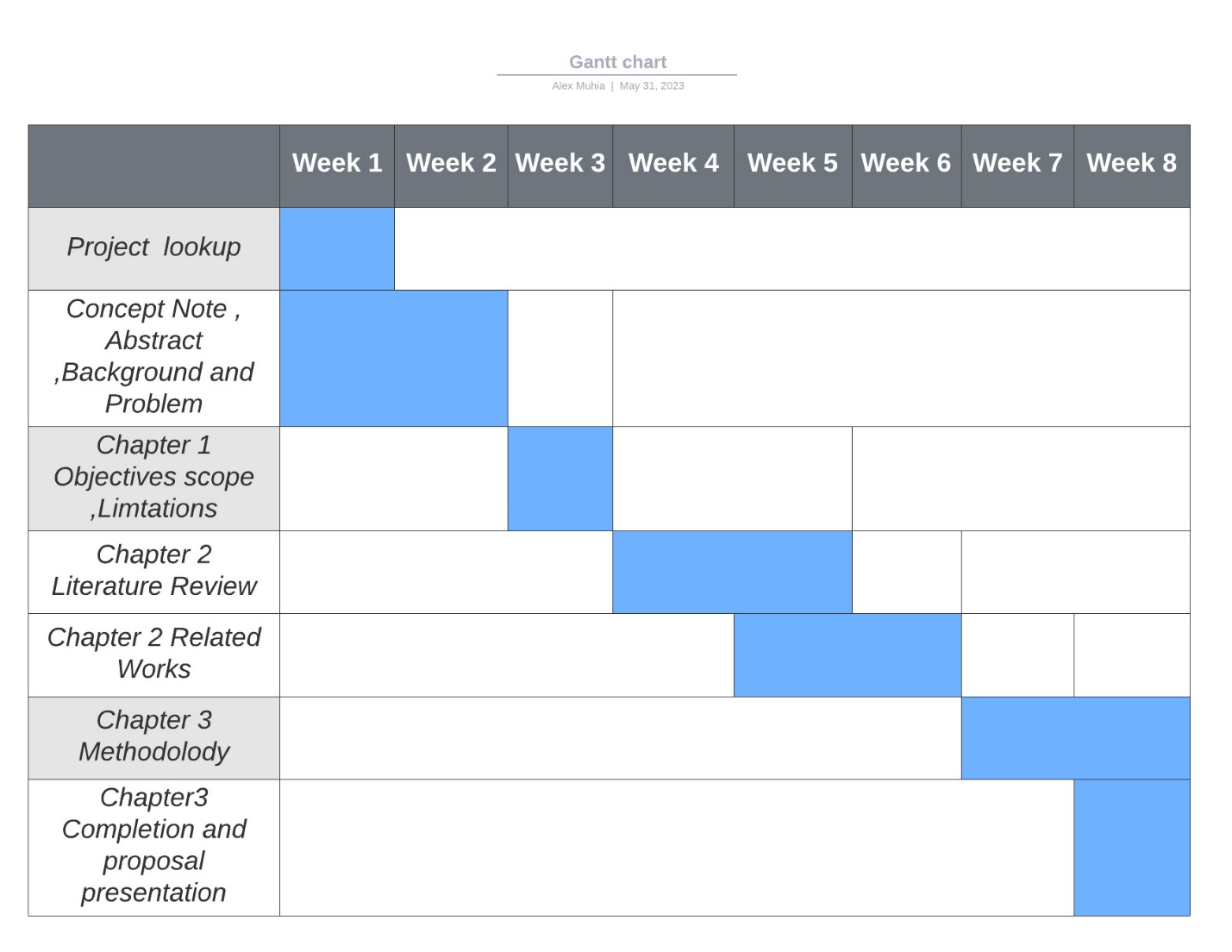
Raymond Maclean, Jacques Ophoff (2018, May) Determining key factors that lead to the adoption of password managers.

Rohit, S. Kamra, M. Sharma and A. Leekha, "Secure Hashing Algorithms and Their Comparison," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 788-792.

William Stallings, & Lawrie Brown (2015). *Computer Security Principles and Practice* (3rd ed.).Pearson Education.

# **Appendices**

## **Appendix 1: Gantt chart for this Project**



## **Appendix 2: Project Proposal Marking Guide**

**Strathmore University**

**School of Computing and Engineering Sciences**

**Project Proposal Assessment Guide**

|  |  |
| --- | --- |
| **Student Number** |  |
| **Working Title:** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation Areas** | **Weight** | **Score** | **Notes** |
| **Title page:**  Informative, concise, and appropriate | **2 pts** |  |  |
| **Abstract**  To have background, problem, solution, methodology (approach data and tools) outcomes and expectations | **2 pts** |  |  |
| **Introduction**  Background **(2)**  *A clear illustration of issue, context and audience*  Problem Statement **(2)**  *Pain points, audience, who is affected and how solution comes in to fix the pain.*  Objectives (S.M.A.R.T and Linked to Problem Statement) **(2)**  Research questions **(1)**  *Alignment of questions with objectives*  Justification **(2)**  *Should be research supported.*  Scope of Project **(2)**  *Specify boundaries of people process, HW/SW, data etc.*  Limitations **(1)**  *Challenges Expected*  Delimitation **(1)**  *To do to counter anticipated challenges* | **(13 pts)** |  |  |
| Literature Review/Related Work Objectives mapping to Literature Review **(2)**  Critique of Theoretical framework and content adequacy (**2**)  *Principles, parameters of consideration*  Discussion of technologies contextualization for the proposed work **(2)**  Citations of content and alignment to work **(2)**  Review of at least 3 systems comprehensively the working behind it **(2)**  Gaps identification, analysis relative to the proposed solution **(1)**  Conceptual Framework clear to communicate how it works, data flows, processing, actors **(3)**  *Diagram that’s clear; discussion of diagram.*  *Describe input process output storage boundaries.* | **(14 pts)** |  |  |
| Methodology Methodology and justification (**2**)  Correct Methodology Application (**1**),  Design and Development tools (**2**)  Deliverables and milestones **(2)**  *Examinable bits from ideation*  *Proposal, design, test cases documentation doc*  *Proof of concept- modules*  Gantt Chart that makes sense relative to the project **(1)** | **(8 pts)** |  |  |
| Proposal Presentation Table of Contents and List of Figures **(2)**  Are relevant references provided and formatted correctly? **(2)**  Is there a clear and proper use of language? **(1)**  Effective report structure (chapters and sections) and layout **(2)** | **(6 pts)** |  |  |
| Total Marks | **45** |  |  |

|  |  |  |
| --- | --- | --- |
| Verdict (Please tick) | Accept | Reject |

Comments (**Reasons for Reject/Accept**)