|  |  |
| --- | --- |
| 1 | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

Senior Project Report

**Password Manager**

**Project Group Members:**

Youssef Sherif Aly (201901748)

Essa Ahmed Kamel(202004969)

Khalifa Ebrahim Yousuf (202002710)

Mohamed Dhia Abdaoui (202005886)

**Supervisor**: Dr. Mohammad Saleh

**2023-2024**

This project report is submitted to the Department of Computer Science and Engineering of Qatar University in partial fulfillment of the requirements of the Senior Project course.

# Declaration

This report has not been submitted for any other degree at this or any other University. It is solely our work except where cited in the text or the Acknowledgements page. It describes work carried out by us for the senior project. We are aware of the university policy on plagiarism and the associated penalties, and we declare that this report is the product of our own work.

Student: Youssef Sherif Aly Date: 21st September 2023

Signature: youssef aly

Student: Essa Ahmed Abou Jabal Date: 21 September 2023

Signature: 

Student: Date:

Signature:

# Abstract

< **ToDo:**

* The abstract is a brief overview of your project and its objectives.
* It should present an accurate summary of the problem your project has addressed and a summary of your solution.
* Highlight key achievements and most important conclusions.
* The length of your abstract should not exceed 500 words.

/>

<**ToDo:** **Writing the final report**

* To produce the final report, you can follow the following recommended steps:
* Where appropriate, copy material from your interim report into the final report. Go through that material, and update it based on changes that have occurred in your project between last semester and now.
* Revise the Abstract and enhance it by adding the project’s key achievements and most important conclusions. The last paragraph should highlight the novelty of your design (e.g., what makes your design unique and what are the impacts of your engineered solution, etc.).
* Fill in all of the appropriate material required for the final report.
* Update the Table of Content, the List of Figures, and the List of Tables.
* Review the whole document to make sure that it is coherent and to ensure that it addresses all the requirements listed in the Project Guide and the Project Grading Rubrics. Also make sure that the tense used is the present tense and the past and not the future (e.g., avoid ‘we will’ or ‘system should’ and report what has been done) except in the Future Work section.
* Seek your supervisor’s feedback and address any issues raised.
* Note that the template is only provided as a guide. In consultation with your supervisor, you can add other sections to align it with the nature of your project.
* If you are using Word ‘Track Changes’ you must accept all the changes before submitting your report.
* Keep the ToDo instructions and only remove them from the **final revised report** that you will submit **after** the Senior Project presentation and after addressing the examiners feedback.

/>

# Acknowledgment

< **ToDo:**

* Acknowledge any assistance you received for your project.

/>

Table of Contents

[Declaration ii](#_Toc127951596)

[Abstract iii](#_Toc127951597)

[Acknowledgment iv](#_Toc127951598)

[List of Figures vii](#_Toc127951599)

[List of tables viii](#_Toc127951600)

[1. Introduction and Motivation 9](#_Toc127951601)

[1.1. Problem statement 9](#_Toc127951602)

[1.2. Project objectives 9](#_Toc127951603)

[1.3. Expected benefits and impacts on various contexts 9](#_Toc127951604)

[1.4. Market Research and Business Viability 10](#_Toc127951605)

[2. Background and related work 10](#_Toc127951606)

[2.1. Background 10](#_Toc127951607)

[2.2. Related work 11](#_Toc127951608)

[3. Requirements analysis 12](#_Toc127951609)

[3.1. Software development process 12](#_Toc127951610)

[3.1.1. Applying the software development process 12](#_Toc127951611)

[3.2. Functional requirements 12](#_Toc127951612)

[3.3. Non-functional requirements 13](#_Toc127951613)

[3.4. Assumptions 14](#_Toc127951614)

[3.5. Ethics 14](#_Toc127951615)

[4. Project Plan 15](#_Toc127951616)

[4.1. Project milestones 15](#_Toc127951617)

[4.2. Project timeline 15](#_Toc127951618)

[4.3. Anticipated risks 15](#_Toc127951619)

[5. Solution Design 16](#_Toc127951620)

[5.1. High-level design 16](#_Toc127951621)

[5.1.1. Alternative solutions and tradeoffs 16](#_Toc127951622)

[5.1.2. Selected solution overview 16](#_Toc127951623)

[5.1.3. High level architecture 17](#_Toc127951624)

[5.2. Structural model 17](#_Toc127951625)

[5.3. Behavioral model 17](#_Toc127951626)

[5.4. Database design 18](#_Toc127951627)

[5.5. User interface design 18](#_Toc127951628)

[5.6. Design patterns 18](#_Toc127951629)

[6. Implementation 19](#_Toc127951630)

[6.1. Hardware/software used 19](#_Toc127951631)

[6.2. Challenging issues and solutions 19](#_Toc127951632)

[7. Testing and evaluation 20](#_Toc127951633)

[7.1. Functional testing 20](#_Toc127951634)

[7.2. Non-functional testing 20](#_Toc127951635)

[8. Conclusion 21](#_Toc127951636)

[9. Future work 22](#_Toc127951637)

[10. Student reflections 23](#_Toc127951638)

[References 24](#_Toc127951639)

[Appendix A – Use cases specification 25](#_Toc127951640)

[Appendix B – Test cases specification 26](#_Toc127951641)

[Other Appendices 27](#_Toc127951642)

# List of Figures

Figure Y1. Password Manager architecture………………………………………………………………………………………7

[Figure 1. Use cases diagram 13](#_Toc127949606)

# List of tables

[Table 1. Expected benefits and impacts on various contexts 10](#_Toc127949795)

[Table 2. Use cases summary 13](#_Toc127949796)

[Table 3. Evaluation plan for the non-functional requirement 13](#_Toc127949797)

[Table 4. Project-related Software Engineering Code of Ethics and Professional Responsibilities 14](#_Toc127949798)

[Table 5. Milestone of the project 15](#_Toc127949799)

[Table 6. Risks 15](#_Toc127949800)

[Table 7. Hardware/software to be used 19](#_Toc127949801)

# Introduction and Motivation

The purpose of this introduction is to serve as the gateway to understanding the core problem that our project aims to tackle, showing its significance, the associated obstacles and a basic overview of our proposed solution and its anticipated impacts.

In today’s digital age, the significance of a password cannot be underestimated, as it is the key that grants users’ authority over their online accounts. Regrettably, a substantial number of individuals have fallen prey to data breaches due to the widespread reliance on weak, default, or repeatedly used passwords. This unfortunate consequence can be traced back to a simple rationale that predictable passwords and password reuse make it easier to remember and convenient for users who might be unaware of the security risks hidden behind that convenience. This vulnerability becomes especially apparent when a single breach in one service jeopardizes multiple other accounts. Astonishingly, these poor password practices have contributed to a staggering 81% of corporate data breaches, highlighting the magnitude of the problem at hand.

This is where our password manager application comes in, aimed with the sole purpose of addressing the pervasive issues associated with weak, recycled and default passwords. Our goal is to provide users with a seamless and efficient experience that encourages them to protect their privacy and secure their accounts across all the services they use. Our password manager aims to fill in the gap between security and user convenience.

By providing an intuitive interface, secure password storage, and the capacity to generate robust passwords, we aim to provide users with a tool that elevates their online security without sacrificing the ease and comfort they seek. With our application at their disposal, users can confidently navigate the landscape of the digital realm, secure in the knowledge that their accounts and sensitive data remain well-guarded.

## Problem statement

The problem at hand is to design and develop a secure and user-friendly password manager application that combines some of the most desired and unique features of other products currently in the market while addressing the pervasive issues of weak, reused, and default passwords. The goal is to empower users to enhance their online security while maintaining the convenience they desire by offering a seamless and efficient experience for managing their passwords across various online services.



**Figure Y1: Password Manager architecture**

Technical Challenges:

* Implementing Robust Encryption: Developing strong encryption mechanisms to securely store and transmit sensitive user passwords.
* User Authentication: Ensuring secure and user-friendly methods for user authentication.
* Password Generation: Creating an effective password generation algorithm that produces strong, unique passwords.
* Usable User Interface: Designing an intuitive and user-friendly interface for managing passwords, suitable for all users.
* Password Backup and Recovery: Implementing a secure and user-friendly password backup and recovery methods.
* Security Auditing: using tools for regular security audits and alerts for password-related vulnerabilities.
* Compliance with Privacy Regulations: Ensuring the application complies with data protection regulations and user privacy requirements.
* Auto-filling credentials: This is particularly challenging since we are working on a native Desktop application.

Non-Technical Challenges:

* User Education: Educating users about password best practices and the importance of using a password manager.
* User Trust: Building trust in the security and privacy of the application, as users entrust it with their sensitive data.
* User Experience: Balancing security with user experience to create an application that users find easy and pleasant to use.
* Adoption and Advertising: Encouraging users to adopt our password manager and effectively highlight its benefits.

**Problem Classification based on its Computing Complexity**

Conflicting Technical Issues Tradeoffs:

* + The application will need to balance usability with security. Creating a user-friendly interface while ensuring robust encryption and protection against breaches is a technical challenge.
  + Choice of handling user data either locally or by using cloud storage requires extensive thinking and measuring of trade-offs between security and accessibility across devices.
  + Decisions such as the choice of encryption algorithms, key management, and password storage methods involve trade-offs between security and performance.

No Obvious Solution:

* + Designing a password manager that provides both convenience and strong security is not straightforward.
  + The best approach to user authentication, password storage, and secure password sharing may require innovative thinking.

Including Many Component Parts or Sub-Problems:

* + A password manager involves various components, such as user interfaces, encryption modules, database management, and some kind of synchronization across devices.
  + Addressing each component's unique challenges and ensuring they work effectively together is essential.

Involving Multiple Disciplines:

* + Developing a password manager requires expertise in software development, cryptography, user experience design, and potentially will require legal and privacy considerations.
  + Cross-disciplinary collaboration is often necessary to create a robust optimal solution.

## Project objectives

Main Objective

Develop a User-Friendly Password Manager Application that aligns with the project’s goal of enhancing user convenience and security.

Sub-Objectives

* Design an easy-to-use UX that is suitable for a wide range of users and perform user testing to improve the interface based on feedback.
* Implement secure encryption for data storage that meets or exceeds industry best practices by utilizing well known encryption algorithms and libraries since data security is a fundamental aspect of the project's objectives.
* Develop a password generation functionality that generates strong, unique passwords by utilizing established password generation algorithms.
* Ensure that password data is consistently accessible on all devices the user desires to enhance user convenience and accessibility.
* Implement multi-factor authentication to enhance security and user confidence.
* Design alerts by utilizing automated security scanning tools and implementing alert mechanisms that inform the user of their password strengths and recommend possible solutions to create a stronger password.
* Evaluate the compliance with privacy regulations related to user data protection and how it is stored which will require continuous evaluation and monitoring throughout the project's lifecycle. This is essential for user trust and legal compliance.

## Expected benefits and impacts on various contexts

The importance of our password manager app project lies in its crucial role in addressing and mitigating several significant challenges related to secure password management, and data protection. This project tackles these issues head-on, offering concrete solutions that enhance security for individuals and organizations alike. According to the 2021 Data Breach Investigations Report, in 2019, 80% of hacking-related breaches were reportedly linked to passwords and stolen credentials [[1]](#Bookmark4). In addition, according to a late 2019 Harris Poll, 75% of Americans struggle to keep track of all their passwords [[2]](#Bookmark3). In today’s world enhancing data security by any means is crucial, our password manager will protect login credentials through encryption and secure storage, contributing to overall data security by preventing unauthorized access to accounts. A study by IBM security reported that the average cost of a data breach in 2020 was $3.86 million [[3]](#Bookmark2). Today, many cybercriminals rely on bad password management to successfully hack into an enterprise’s networks and systems. In 2020 research conducted by the Ponemon Institute, it was revealed that a significant 59% of organizations continue to depend solely on their employees' memory to manage their passwords [[4]](#Bookmark1). This situation highlights the challenge employees face when choosing passwords, as they tend to opt for ones that are easy to remember. Given the growing complexity of password strength requirements, enterprises are faced with the imperative task of equipping users with suitable tools, such as password managers. These tools not only store passwords securely but also facilitate password generation and automatic filling when necessary.

Personal Interest and Career Goals

We became interested in this project due to the increasing importance of cybersecurity in our digitally connected world, even though there are great password managers out there, we have decided to face the challenge of developing one ourselves and delve into security and software development while also aligning with our career goals in addition to attempting to create an application that combines some of the most useful features into one place. As the field of cybersecurity continues to evolve, we hope that our involvement with this project enhances our skills and knowledge as well as allowing us to contribute to the broader goal of making the digital world safer for everyone.

Table 1. Expected benefits and impacts on various contexts

|  |  |
| --- | --- |
| Context | Expected benefits and impacts |
| Individuals | Users can enjoy an enhanced digital security against cyber threats, reducing the risk of personal data breaches. A simplified and easy-to-use password management application will encourage users across the board to use its tools as a way of protecting their personal information while maintaining conveniency. |
| Organizations | Employees in all organizations can utilize our password manager to spend less time on password-related issues and more on establishing efficiency and effectiveness, thus increasing the organization’s productivity. In addition, our tool is valuable in the sense that it eliminates the need for employees to write down their passwords, which is a common insecure practice in most organizations. |
| Society | Usage of our password manager will contribute to a safer online environment, reducing the risk of identity theft and fraud as well as allowing users to adopt such responsible online practices. |
| Global | Our app can contribute to a thriving local cybersecurity industry, furthering Qatar’s technical regional capabilities and boosting competitiveness. |

## Market Research and Business Viability

< **ToDo:**

* Conduct market research to address the following:
* Describe the market need and the market size.
* Identify the target customers and their demographic.
* Describe the competing products and how does yours differ from that offered by competitors? Highlight the novel features of your product and the benefits it offers.
* Develop a business plan including your business model, pricing, marketing strategy to bring your product to market and make it competitive.

/>

# Background and related work

## Background

In an era dominated by digital interactions and online services, the management of passwords has become a pivotal component of personal and organizational cybersecurity. This background report provides essential context and insights into our project's major concepts, issues, and critical problems.

Overview of the field

Cyber threats, breaches of personal data and unauthorized access to sensitive information constitute a constant struggle in today’s digital world. Passwords were always an integral element of security, but individuals and businesses today face several challenges to securely protect access to their online accounts while also maintaining the proper password practices recommended by the cyber specialists and different standards in place. That is why this wide domain is always in need of secure, yet easy-to-use password management solutions that could carefully mitigate data breaches and security deficiencies for all users.

Key problems

Our password management system project aims to deal with some of the most critical problems in the field including:

**Poor password strength:** a lot of users are having trouble creating strong, specific passwords for their accounts and this makes them susceptible to unauthorized access.

**Password Reuse:** The common practice of reusing passwords across multiple accounts increases the risk of widespread compromise in the event of a breach.

**Vulnerabilities in current systems:** vulnerabilities which could allow an attacker to gain access to the system can exist in some of the currently used password management systems.

**The need for user-friendly solutions:** users often find that existing password management systems are hard to use or difficult to exploit, which can lead to a decrease in the adoption of secure passwords.

History and the rise of key players in the field

Since the beginning of computing, passwords have been a critical element in digital security. Initially password management started with users manually writing down passwords or using basic methods like using the same password for multiple accounts. This led to major security gaps and user inconveniences. Eventually, local and cloud-based password managers like LastPass, Dashlane, and 1Password rose in the field to offer a more secure and convenient way to manage passwords. In addition to the password managers, leading cybersecurity experts and organizations in the field actively advocate for improved password practices and security measures.

Basic concepts used in our project

**Cryptographic Techniques:** A wide range of terminology and concepts to secure information and communications including **encryption**, **decryption**, **ciphers**, **keys** and other terminologies.

**Encryption:** The process of converting plaintext (unencrypted data) into ciphertext (encrypted data) using **cryptographic techniques** for the objective of making sensitive data unreadable to unauthorized users.

**Decryption:** The reverse process of encryption, where ciphertext (encypted data) is transformed back into plaintext.

**Ciphers:** Algorithms used for encryption and decryption of data.

**Keys:** Values used in cryptographic algorithms to control encryption and decryption processes.

**Authentication:** The identity of the individuals who are attempting to access a password protected account is verified through user authentication.

**Salting and hashing:** The key technology used in harmony with ciphers to secure passwords is the use of salting and hashing, which ensures that passwords are kept protected in case of a data breach.

**User Interface UI Design:** In password management systems, user friendly interfaces are necessary to make it easier for users to enter and manage passwords.

**Security Frameworks and Standards:** Best practice for password management is provided by industry standard security frameworks and guidelines, such as those of the National Institute of Standards and Technology.

## Related work

Developers aim to improve password managers with new features but face criticisms and security issues. Our solution combines the best concepts from popular password managers for enhanced security and functionality. We prioritize secure data storage and strive to introduce value to existing options. We discuss the related work in two sections, each relating to different types of relevant sources. First, we discuss the most used password manager applications. Second, we discuss password organizer books, research, study, and an analysis about password managers. We conclude with a table that summarizes the similarities and differences in various aspects, providing a comprehensive overview.

**Password Management Applications**

**1- Nord Pass**

Nord Pass is one of the top password managers in 2023 [[1]](#NordPass), its known for its secure and user-friendly interface, robust security features like zero-knowledge policy and multi-factor authentication, and ease of use in tasks like password saving, generation, auto-filling, and sharing across devices. While it lacks some advanced features and leans on subscriptions, it offers excellent customer support and various plan options. In contrast, our free application stands out for its accessibility on multiple devices and no password vault limitations, making it a valuable alternative for users seeking a cost-effective solution.

A screenshot of a phone login screen

Description automatically generated

**Figure 1: Nord Pass UI and Desktop Application**

**2- Bit Warden**

Bit warden stands out as an affordable and feature-rich open-source password manager [[2]](#BitWarden) with robust security features like encryption and two-factor authentication, though it's not as user-friendly as some competitors. Its free version is generous, but concerns exist about data storage and past breaches. However, Bit warden’s commitment to open-source transparency strengthens its security. In contrast with Bit Warden, our application is user friendly with an attractive UI, and when users want to access passwords from different devices, we assure that it’s secure and encrypted before sending."

A screenshot of a computer

Description automatically generated

**Figure 2: Bit Warden UI and Desktop Application**

**3- Dash Lane**

Dash lane, a leading 2023 password manager, offers robust security with AES encryption, biometric authentication, and an integrated VPN [[3]](#DashLane). However, it comes at a higher cost than some competitors like Nord Pass, and the absence of desktop apps may be a drawback for many users who prioritize this feature. Nonetheless, Dash Lane is recognized for its top-tier security and user-friendly features. This is where our application comes in handy. Many people use their desktops as their main device so having the desktop application is considered a priority for many users, which Dash Lane doesn’t provide.



**Figure 3: Dash Lane UI and \*Old\* Desktop Application**

**4- Zoho Vault**

Zoho Vault is a business-focused password manager with strong security, a cost-effective free version, various browser extensions, and phone support. It offers robust user management and Two-Factor Authentication options but may have a somewhat confusing interface. While not ideal for simplicity, it excels in business settings with granular password sharing control and integration with popular apps [[4]](#ZohoVault). Comparatively, it may have a steeper learning curve than some competitors. When it’s compared to our application, usability for us seems much easier and simpler, and any individual can use the app it is not specifically designed for businesspeople.

A screenshot of a computer

Description automatically generated

**Figure 4: Zoho Vault UI and Desktop Application**

**5- 1Password**

1Password is a top-ranked password manager in 2023 known for its cross-platform compatibility, attractive interfaces, competitive pricing, and strong security. It employs encryption, biometric authentication, zero-knowledge architecture, and multi-factor authentication for data protection. Although lacking a free version and money-back guarantee, it offers a 14-day trial. Notably, Travel Mode enhances security during travel [[5]](#OnePassword). Drawbacks include a clumsy autofill and lack of password inheritance. Overall, it's customizable, user-friendly, and highly secure, making it a standout choice in the password manager category. Compared to our implementation, there is no need to worry about the versions, the issues with form filling, and import options since all of these are provided for the user with ease.

**Figure 5: 1Password UI and Desktop Application**

**6 - Keeper**

Keeper Password Manager & Digital Vault is a highly secure and user-friendly password management solution known for versatile multi-factor authentication, secure password storage, customizable vault entries, and encrypted messaging with Keeper Chat [[6]](#Keeper). While it no longer offers a free tier and charges for some advanced features, it remains a solid choice, but there's potential for improvement by including standard dark web monitoring and additional features like a VPN or Travel Mode. In comparison with our project, ours provides free dark mode for users who like less light being emitted from the device and other features that are close to what keeper has.

**Figure 6: Keeper UI and Desktop Application**

**7 – LogMeOnce**

LogMeOnce Password Management Suite is a feature-rich password manager, rivaling Dash Lane with its extensive feature set, including smartphone-based authentication for password less login [[7]](#LogMeOnce). However, some features are costly, making it relatively expensive, and its wide range of features might be overwhelming. In contrast, our project offers all features for free, ensuring accessibility for all users.

A screenshot of a computer

Description automatically generated

**Figure 7: LogMeOnce UI and Desktop Application**

**8 – Password Boss**

Password Boss is a versatile cloud-based password management software suitable for personal and business use, offering secure storage, synchronization, password sharing, and encryption for robust security [[8]](#PasswordBoss). While it provides a free plan for one-device local storage and premium plans for unlimited device synchronization, two-factor authentication, and remote data deletion, it does have some drawbacks, including installation issues and slow email support. Users can try it with a 30-day free trial or a risk-free 30-day money-back guarantee, and it distinguishes itself from our application through user-friendly installation and expandable password sharing across devices.



**Figure 8: Password Boss UI and Desktop Application**

**9 – RoboForm**

RoboForm is a top-rated Password Manager and Web Form Filler known for automating password entry and form filling. It offers convenient one-click Logins, standard features like autofill, and strong encryption, all at an affordable price. However, it has limitations like slower customer support and may lack some advanced features [[9]](#RoboForm). One significant distinction between our project and RoboForm is the array of features and functionalities integrated into our program, providing users with a comprehensive experience, and rendering additional password managers unnecessary.



**Figure 9: RoboForm UI and Desktop Application**

**Password Organizer Books, Research, Study, and an Analysis**

An analysis written by Timothy Oesch's assesses the security of contemporary password managers in desktop and mobile contexts. The study identifies weak passwords and autofill vulnerabilities based on 147 million generated passwords, comparing iOS and Android autofill frameworks to uncover security issues [[10]](#Analysis). Qualitative insights from user interviews are provided, emphasizing the use of multiple managers and password reuse trends. While the analysis offers valuable insights with a large dataset, it lacks exploration of usability and specific recommendations. A more comprehensive evaluation including usability, strengths, weaknesses, and user behavior would enhance its utility.

Password organizer books are a physically secure but limited alternative for storing login credentials. They offer simplicity, reduced software vulnerabilities, and data privacy as strengths but lack digital features like autofill, synchronization, and have challenging recovery options if lost [[11]](#BookOrganizer). They may also pose security risks if physically accessed and have limited integration capabilities. The choice between them and password manager software depends on specific security needs and preferences.

An additional reference underscores the growing demand for robust password management systems due to the rise of online services. It evaluates password manager effectiveness, usability, and security, proposing parameters for a 2022 system [[12]](#Study). Key findings stress the significance of longer passwords for security, endorse AES-256 encryption, and acknowledge user cognitive load but overlook real-world testing, security-usability trade-offs, and user experience aspects.

Lastly, research done by Ramakrishna Ayyagari investigates factors influencing password manager adoption and emphasizing security considerations. It offers valuable insights, supported by survey data, but faces limitations like a small sample size, contradictory findings on trust and ease of use, and the need for behavior-based research, signaling the necessity for further study with larger samples and behavioral metrics for a more comprehensive understanding [[13]](#Research).

A screenshot of a computer

Description automatically generated**Table 1: Key similarities and differences between out project and the related work**

# Requirements analysis

< **ToDo:**

* In this section, you are describing and justifying your selection of the most appropriate software development process for your specific project and presenting the project requirements.

/>

## Software development process

<**ToDo:**

* Discuss possible alternative software development processes that can be applied for your project.
* Select an appropriate software development process for your project.
* Provide sufficient justifications why the selected software development process is the most suitable for your specific project.
* Describe how you are intending to apply the selected software development process to your project.

/>

## Applying the software development process

</**ToDo:**

* Document and provide evidence on how you **applied** the adopted Software Development Process throughout the project to produce the solution. This should also be reflected in the project plan presented in section ‎4.
* Use in-text referencing to relevant sections of the report to provide evidence of the proper usage of the adopted software development process. For example, you can reference specific sections of the report that describe the requirements gathering process, design phase, testing process, or project management techniques.
* Discuss any related issues, challenges, difficulties, advantages or drawbacks, and lessons learnt during the software development process. For example, you can describe how the chosen methodology helped to address specific project challenges or how it fell short in some areas.
* This subsection is extremely important and required.

/>

## Functional requirements



Figure 1. Use cases diagram

Table 2. Use cases summary

We have categorized the different use cases into distinct groups based on their priority and importance (**Essential (E)**, **Recommended (R)**, and **Optional (O))** as shown in the table below:

|  |  |  |
| --- | --- | --- |
| ID | Use case | Brief description |
| E1 | **Sign up** | New users who want to use the password manager application create an account. |
| E2 | **Login** | Registered users access their stored credentials using their master password. |
| E3 | **Add Credential** | Users can add new login credentials (username and password) for websites or applications to their password manager. |
| E4 | **Update Credential** | Users can edit and update existing login credentials stored in their password manager. |
| E5 | **Remove Credential** | Users can delete login credentials they no longer need from their password manager. |
| E6 | **AutoFill Credentials** | The password manager can automatically fill in login credentials when users visit websites or apps, streamlining the login process. |
| E7 | **Search Credential** | Access and retrieve specific stored credentials quickly. |
| E8 | **Generate Password** | Users can use the password manager to generate strong, unique passwords for new accounts or to replace weak passwords. |
| E9 | **Export Credentials** | The password manager can export login credentials for usage across multiple devices, ensuring users have access to their passwords everywhere. |
| E10 | **Import Credentials** | The password manager can import login credentials for availability purposes. |
| E11 | **Lock Manager** | Users can lock their password manager, preventing access to stored credentials for security purposes or even set a timeout period for the application to lockout. |
| E12 | **Unlock Manager** | After locking, users can unlock the password manager by only providing their master password. |
| R1 | **Audit Credentials** | Users can review and audit the security of their stored credentials, checking for weak or duplicated passwords. |
| R2 | **Change Settings** | Users can customize settings within the password manager, such as changing master passwords, enabling or disabling features, and adjusting security preferences. |
| R3 | **Recover Access** | In case of a forgotten master password, users can initiate a recovery process to regain access to their password manager account. |
| R4 | **Wipe Account** | Users can permanently delete all data and credentials stored in their password manager account, ensuring data privacy in the event of an account closure. |
| R5 | **Enable 2FA** | Users can enhance the security of their accounts by enabling 2FA. They have the option to link their accounts with a secondary authentication method, adding an extra layer of security to prevent unauthorized access. |
| O1 | **Share Credential** | Users can securely share login credentials with trusted individuals. |
| O2 | **Store Other Sensitive Data** | Users can store various sensitive documents and information within the platform. This includes passports, social security numbers, financial records, or any other confidential data. Robust encryption is implemented to protect this information from unauthorized access. |

## Non-functional requirements

Table 3. Evaluation plan for the non-functional requirement

|  |  |  |
| --- | --- | --- |
| Non-functional requirement | Measurable targets | Evaluation Plan |
| 1 - Performance | - The application must ensure a response time of under 10 seconds for 90% of requests, including text rendering.  - User registration should be completed within 20 seconds, logging in should take no more than 5 seconds. | - Simulate various scenarios, including situations like a slow hardware device. Any instances that exceed predefined maximum thresholds will be highlighted.  - Performance metrics such as Throughput, Average Response Time, and Deadline will be monitored. If an action's performance surpasses the desired target, we will optimize it using the appropriate algorithm.  - Ensure this through load testing / stress testing. |
| 2 - Reliability | - No data loss is permissible; all data delivery outcomes must be verifiable within a 5-second timeframe.  - The system will incorporate appropriate failover mechanisms and implement backup procedures to mitigate the impact of disasters.  - User information and login records are consistently and continuously stored in the database to prevent data loss.  - The system must operate without failure in at least 85% of use cases within a given month. | - Regularly pinging the backend and acting on any errors that arise.  - Verifying the presence and retrieval of user data in the database.  - Monitoring and recording the number of critical production bugs over a specific period or calculating the mean time to failure. |
| 3 - Security | - All communication between the front end and back end must be authenticated and encrypted.  - The system must implement robust security measures to protect sensitive data and uphold the CIA triad, safeguarding users' personal and credential assets.  - Personal user information and login credentials should only be accessible to authorized users.  - The databases should be protected by proper form of validation to prevent access from unauthorized people to users’ private information and mitigate attacks and threats like Phishing Attacks. | - Implement best practices to enhance security, including end-to-end encryption.  - Provide authentication mechanisms, such as multi-factor authentication, and conduct tests to ensure that no unauthorized users can access each other's data.  - Encrypt and hash all data stored in the database to establish robust security measures for data protection.  - Employ countermeasures against brute force attacks, which may involve utilizing tools like John the Ripper. |
| 4 - Usability | - The system's interface should be user-friendly and easily understandable for all users. Most features should be designed to be intuitive, taking the user no more than 2 minutes to use, without necessitating extensive training. | - Make a Hallway Testing where we involve a minimum of 50 users in a testing session where they perform key actions within the system. Collect their feedback to make necessary improvements.  - Make a survey to assess the application's ease of use by distributing the survey to more than 50 students and carefully consider the feedback received. |
| 5 - Portability | - The application is compatible with Windows 10 and must maintain its behavior and performance when running on Windows 11 without any modifications | - We will conduct testing on both iOS and Windows systems and address any errors that may arise |
| 6 – Availability | - The system should remain operational 24 hours a day, 365 days a year, with the assistance of backup modules. | - Reliable software implies its availability.  - Testing during development and operation includes measuring failure detection, correction time, and app restart duration |
| 7 – Scalability | - The application must be capable of managing more than 300 logins/accounts per user.  - The volume of data processed per user can potentially reach up to 281 terabytes. | - Our app's storage allocation depends on the user's device capacity; if a user's device has ample storage, then we already gave him a maximum amount of storage.  - Our app is designed to handle user databases of any size or number of logins.  - We will conduct tests with users who have over 100 logins to identify and address any potential issues with the app. |

## Assumptions

<**ToDo:**

* List any specific project assumptions. Assumptions are potential failure points in a project. They need to be monitored and managed. A bulleted list could be used for this section.

/>

## Ethics

<**ToDo:**

* Ethical issues and professional responsibilities:
  + Identify the legal and ethical issues and professional responsibilities relevant to your project (e.g. proper handling of private information collected during the project, not to exaggerate the nature of work in the report rather the report should truly reflect the actual work completed, not to use someone’s work without citing proper reference, etc.). Consider the perspectives of diverse relevant stakeholders and articulate these with clarity and accuracy.
  + Discuss how you will address these identified ethical issues/concerns and responsibilities during the analysis, design, implementation, testing, evaluation, and delivery of your project.
* From [ACM and the IEEE Software Engineering Code of Ethics and Professional Practice](https://ethics.acm.org/code-of-ethics/software-engineering-code/) select the most relevant items to address the identified ethical issues/concerns and responsibilities in your specific project.
  + The selection should be based on appropriate project functionalities, the development tasks, the conduct of the team and interactions with other project stakeholders. Discuss and justify using/practicing them during the project work.
  + Consider alternative courses of action and justify your choices in logical manner with persuasive reasoning.
  + **Avoid** outlining general code of ethics and responsibilities not related to your specific project. **Do not cut and paste the entire ACM and IEEE codes of ethics into your report**. Instead, read through them, and in your report specifically address the ones most relevant to your specific project and discuss how you are considering them in your project.
  + At the end of your discussions, use a table to summarize the ethical issues/concerns and professional responsibilities relevant to your project.

/>

Table 4. Project-related Software Engineering Code of Ethics and Professional Responsibilities

|  |  |  |
| --- | --- | --- |
| Sec. No | Code | How to address the ethical issue / professional responsibility during your project |
|  |  |  |
|  |  |  |

# Project Plan

## Project milestones

< **ToDo:**

* In a table list and describe the major milestones of the proposed project plan. Milestones are checkpoints that have specific deliverables to produce.

For example, a milestone could involve choosing a use case (or a component from your high-level design) then designing, implementing, and testing its realization.

/>

Table 5. Milestone of the project

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Deliverables |
| M1 Design  Week 1-5 |  |  |
| M2 Implementation  Week 6-7 |  |  |
| M3 Verification and feedback  Week 8 |  |  |
| M4 applying feedback  Week 9 |  |  |
| M5 last checkup  Week 10 |  |  |
| M6 review and improvement  Week 11-12 |  |  |

## Project timeline

< **ToDo:**

* Project timeline defines who will do what and when. You need to:
* Breakdown each project milestone into manageable tasks. Using a table, provide a brief description of each task.
* Estimate the time required to complete each task.
* Allocate the tasks to individual team members and define a work schedule stating the estimated beginning and completion dates of each task. You can use Microsoft Project or Excel to create your project timeline. Insert a Gantt chart and/or a project timeline table in this section.
* Keep in mind the total number of days budgeted for the project. Also the role and the responsibilities of each team member should be clearly decided.
* Be sure to include time to evaluate your work, to reflect on the experience and to document your work at each milestone to incrementally produce your final report. You should include time to finalize and review the final report and presentation.

/>

{monitoring of work}

1. Meetings

Bi-weekly meetings: one with the supervisor during which we get overall feedback and checking on our progress, and in case we have questions. Another meeting only with the team members, to check on the progress and help each solving challenges, these meetings are relatively short (30-45mins max) for more effectiveness. They are conducted similarly as Agile meetings: “what was completed? What is to be done next? What are the obstacles”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| milestones | tasks | Deadline | description | Assigned member |
| M1 Design  Week 1-5 | Use case specifications | 25-9-23 | Details of all use case specifications | All team |
| Class diagram | 30-9-23 |  | Youssef & Mohamed-Dhia |
| Report Chpt 5 | 30-9-23 | 5.1: High lv design | Youssef & Khalifa |
| 5.2: Structural model (class diagram) | Mohamed-Dhia |
| 5.3: Behavioral model (Sequence diag. & activity diag. & state diag.) | Essa |
| 5.4: DB design | Khalifa |
| 5.5: UI design | Mohamed-Dhia & Youssef |
| 5.6: Design patters: Data flow diagram & sequence diagram | Essa |
| Planning security | 30-9-23 | How to apply security? (techniques: enc, salt & pepper…etc) | Youssef & Mohamed-Dhia |
| M2 Implementation  Week 6-7 | electron | 2-10-23 | Try electron & check if works as we want: can we just code as if it was a website? | All team |
| Front-end | 7-10-23 | Html, css | Youssef & Mohamed-Dhia |
| Back-end | 7-10-23 | mysqlite | Essa & Khalifa |
| Report Chpt 6 | 7-10-23 |  | All team |
| Testing (which type?) | 8-10-23 |  |  |
| M3 Verification and feedback  Week 8 | Security of class diagram | 12-10-23 |  |  |
| Testing (which type?) | 12-10-23 |  |  |
| General Review of Design | 12-10-23 |  |  |
| Report Chpt 7 |  |  |  |
| Design Review (security perspective) | 12-10-23 | Including zero trust architecture |  |
| M4 applying feedback  Week 9 | Update design | 19-10-23 |  |  |
| Update implementation | 19-10-23 | Based on the feedback. No new features added at this stage. |  |
| Report improvement | 21-10-23 | Overall review of the report | All team |
| M5 last checkup  Week 10 | Report Chpt 8,9,10 |  |  |  |
|  |  |  |  |
|  |  |  |  |
| M6 review and improvement  Week 11-12 |  |  |  |  |
|  |  |  |  |
|  |  |  |  |

{graph of dependency & “chain of time”}

## Anticipated risks

< **ToDo:**

* Present a table of risks highlighting the potential events that might result in failure to successfully complete the project.
* What is your approach to minimizing each risk?

/>

Table 6. Risks

|  |  |
| --- | --- |
| Risk event | Approach to minimizing the effect on project success |
| Unexpected event | Plan to completely finish the project at least 2 weeks before the deadline |
| Non-compatible implementation  (coding language,..etc) | Change it at the appropriate time |

# Solution Design

<**ToDo:**

* In this section you need to document the detailed design specifications of the software components to meet the functional and the non-functional requirements of your project.
  + You should document your detailed design using suitable methods and tools such as class diagrams, sequence diagrams, activity diagrams and state diagrams. Also, you need to provide detailed justification of design choices. Additionally, you should highlight the novel aspects of your design.
* During semester 1, you will design, implement, and test a Proof of Concept (PoC) to prove the feasibility of your design idea and deliver the core features of your solution by choosing the most important/critical use cases. It is expected that the PoC delivers 25% to 30% of the important use cases.
* Note that for every design aspects you will present in the sections below, you must:
* **Identify and evaluate possible alternative solutions** (i.e., design choices) and analyze their tradeoffs. You must provide sufficient justifications on why you have preferred and selected a particular solution or design choice over others.
* Justify your optimal design choice by **discussing how your design choices** enables you to achieve the desired system quality attributes documented in Section ‎3.3 Non-Functional Requirements.
* Further justify your design choice by discussing relevant **design principles** that influenced your system design. For example, highlight how your design comply with key software design principles such as:
* Separation of concerns: e.g., separation of the UI from the business logic.
* Abstraction: hide the component’s complexity behind simple interface.
* Modularity: divide the system into components.
* High cohesion: component’s functions should be functionally related.
* Low coupling: reduce dependencies between components.
* You could consider breaking the sub-sections listed below to clearly document and communicate your design.

/>

## High-level design

## Alternative solutions and tradeoffs

< **ToDo:**

* Identify possible alternative solutions to meet the project requirements and analyze their tradeoffs.

/>

## Selected solution overview

< **ToDo:**

* Present an overview of the selected solution. This can be done in the form of few paragraphs giving a birds-eye view of the solution.
* You must provide sufficient justifications on why your selected solution or design is better than alternative solutions.

/>

## High level architecture

< **ToDo:**

* Add a high-level architecture diagram of the proposed solution. The diagram should show how your solution is decomposed and organized into components. This should guide your detailed design.
* Describe the role and the interfaces of key components of your high-level architecture.
* Discuss and justify the architectural style (e.g., MVC, SPA) used by your solution.
* Discuss the key interactions between the identified components.

/>

## Structural model

<**ToDo:**

* **You may break this section into subsections** to document the detailed structural model of each major components or use case.
* The structure of your software components could be documented using:
* Class diagram for the whole system. If the model is too big partition the diagram using some reasonable criteria. For example, you may provide the entity classes, repository classes and the service classes as separate diagrams.
  + - * For every class, specify the declaration of all attributes (names, data types).
      * For every class, specify the signatures for all methods (i.e., parameter names and their types as well as the return type).
      * Show associations and aggregations between classes. You should define the association name and the multiplicities on both ends.
      * Show inheritance relationships between classes.
      * A brief explanation should accompany each diagram. Add few paragraphs discussing major design decisions such as how inheritance and polymorphism were used to improve the system design. Also briefly discuss how design principles such as the ‘Information Expert’ and ‘Whole controls the Parts’ principles were applied to your design.
* For the external services (i.e., boundary classes) the system interacts with (e.g., BankingService, MailService) you should design the interface exposed by each of these services.

/>

## Behavioral model

<**ToDo:**

* You may **break this section into subsections** to document the detailed behavioral model of each **major** component or use case.
* The behavioral aspects of your software components could be documented using:
* Activity diagrams to describe the overall flow of control of your application (step-by-step workflows of components in your system).
* Design Sequence Diagrams (DSD) to document how to realize key use cases. Method calls in the DSD should be numbered, and you show the parameters and the return type of each method.
* State diagrams for important parts of your system. You need to identify states and events that trigger state transitions.

/>

## Database design

<**ToDo:**

* Include a logical data model describing the major entities and the relationships between them. Also include the attributes for each entity. Make sure that the data model is consistent with (or derived from) the entities of the class diagram. If the model is too big partition the diagram into logical parts such as a diagram per component or subsystem.

/>

## User interface design

<**ToDo:**

* Discuss the target users for the system and any relevant characteristics that influenced the design decisions.
* Describe the overall layout and visual design of the user interface, including color schemes, typography, and graphics.
* Include screen mockups or wireframes for major use cases and explain the rationale behind key design decisions.
* Discuss the navigation design of the UI, including the structure of the user interface, movement between screens to achieve a particular use case, and any navigation aids or controls.
* Explain any novel aspects of your UI design and how they enhance the usability of your system. This could include innovative interactions, user feedback mechanisms, or features that support accessibility or personalization. />

## Design patterns

<**ToDo:**

* Document and evaluate the design patterns applied to your design such as the Model View Controller (MVC) pattern, the Factory pattern, the Proxy pattern, the Adapter etc.
* For each selected architectural pattern:
* Discuss the design rationale justifying the choice of the selected design pattern.
* Draw a UML diagram(s), e.g., class diagram, sequence diagram, to illustrate how the selected design pattern have been applied to your design.
* Evaluate of the used pattern the effect of selected pattern on your system quality attributes. Highlight the benefits introduced by the selected pattern potential problems or limitations introduced by the selected pattern.
* This subsection is extremely important and required. />

# Implementation

## Hardware/software used

< **ToDo:**

* List and discuss the hardware/software platforms and tools used for the design and implementation.
* Describe the role and purpose of each hardware and software component in the project. Explain why each component was chosen and how it fits into the overall project architecture.
* Provide references or links to relevant documentation, or resources for each hardware or software component.
* Use table format to present the above.

/>

Table 7. Hardware/software to be used

|  |  |
| --- | --- |
| HW/SW details | Justifications |
|  |  |
|  |  |

## Challenging issues and solutions

<**ToDo:**

* Discuss the challenging issues encountered during the implementation and how they were addressed. Lessons learnt from the experience that can be beneficial for others.

/>

# Testing and evaluation

<**ToDo:**

In this section, you should describe in detail the tests you have conducted to verify that your design and implementation satisfy the desired functional and non-functional requirements. The testing should verify and provide evidence that your solution solved the stated problem and satisfied the requirement specifications (if not, explain what is lacking).

Note that this section is a substantial portion of the grade for your final report, and it requires a significant effort.

/>

## Functional testing

<**ToDo:**

* Break this section into multiple subsections to test every use case.
* Testing should be done for different levels including at least **Unit Testing (a.k.a., component testing),** **Integration Testing, System Testing, and Acceptance Testing** to verify the correct delivery of the desired use cases documented in Section ‎3.2.
* Describe in detail the tests you have run to verify that your solution satisfies the functional requirements of your project. For example, for each use case, you should write test case(s) including the expected and actual results, run them and report the testing results. The test cases should be added to Appendix C. Functional testing will allow you to find errors/defects/faults/failures, then fix them and identify possible improvements. You need to have a comprehensive set of tests that verifies the correct functionality of every use case of your system.
* You should **present the test results**, with the appropriate level of detail in addition to accuracy and completeness**,** using tables, graphs, diagrams, screenshots, etc.
* **Discuss test results** and explain whether the implemented solution has satisfied the requirements. If not, state what is lacking or still needs improvement, then explain the reason for that.

/>

## Non-functional testing

<**ToDo:**

* Include a sub-section for each non-functional requirement listed in Section ‎3.3.
* Describe in detail the tests to **evaluate the quality of the system** (e.g., performance testing, usability testing, security testing, scalability testing, availability testing, etc. as documented in Section ‎3.3) to verify that your solution satisfies the non-functional requirements of your project. Include a summary table that shows each non-functional requirement and the measurements that prove your system met/did-not-meet the requirement. In case a requirement is not met, then explain the reason for that.
* You should **present the test results**, with an appropriate level of detail in addition to accuracy and completeness**,** using tables, graphs, diagrams, screenshots etc.
* **Discuss test results** and explain whether the implemented solution has achieved the requirements. If not, state what is lacking or still needs improvement, then explain the reason for that.

/>

# Conclusion

< **ToDo:**

* Discuss the main conclusions (e.g., match the project objectives with the achievements in your work and state the degree of achievement).
* Highlight the strengths of the solution and list down its shortcomings (what worked? what didn’t work?).
* Highlight the key contributions and the novel aspects of your work.

/>

# Future work

< **ToDo:**

* Suggested improvements and further work: identify areas of improvement in the project and features of interest that can be added later on. How the solution shortcomings could be addressed? What things could be done better? What additional resources are required to implement the extended / not-implemented design or features?

/>

# Student reflections

< **ToDo:**

* Add individual student reflections (add a sub-section for each student):
* Lessons learned from the project.
* Professional development you have achieved during your project experience (i.e., new skills gained) and explain its value for your future career.
* You can discuss (1) new technical skills acquired such as solving problems, designing and realizing solutions (2) interpersonal skills such as team leading and effective communication (3) personal growth such as adapting to change and acting professionally and ethically.
* Key shortcomings that you should avoid in future projects.
* Key lessons and new attitudes to carry forwards to your professional life from the personal experiences and the teamwork experienced during the project.

/>

# References

[1] C. T. Security, “Enterprise password security guidelines in a Nutshell: TechTarget,” Security, <https://www.techtarget.com/searchsecurity/post/Enterprise-password-security-guidelines-in-a-nutshell> (accessed Sep. 9, 2023).

[2] A. O’Driscoll, “25+ password statistics that may change your password habits,” Comparitech, [https://www.comparitech.com/blog/information-security/password-statistics/#:~:text=Another%202019%20Google%20study%20in,different%20password%20for%20every%20account](https://www.comparitech.com/blog/information-security/password-statistics/#:~:text=Another 2019 Google study in,different password for every account). (accessed Sep. 9, 2023).

[3] “Cost of a Data Breach Report”, IBM, <https://www.ibm.com/security/digital-assets/cost-data-breach-report/1Cost%20of%20a%20Data%20Breach%20Report%202020.pdf> ,2020 (accessed Sep. 9, 2023).

[4] “The human factor in Data Protection - Ponemon Institute,” Ponemon, <https://www.ponemon.org/local/upload/file/The_Human_Factor_in_data_Protection_WP_FINAL.pdf> (accessed Sep. 9, 2023).

# Appendix A – Use cases specification

< **ToDo:**

* Create a table for each use case. Use cases must be numbers e.g., UC01, UC02… See the ‘Use cases modeling’ slides for further details.

/>

|  |  |  |
| --- | --- | --- |
| **Use case Id:** |  | |
| **Brief Description** |  | |
| **Primary actors** |  | |
| **Preconditions:** | | |
| **Post-conditions:** | | |
| **Main Success Scenario:** | | |
| **Actor Action** | | **System Response** |
| 1. | | 2. |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
| **Alternative flows:** | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **R3** | **Recover Access** | |
| **Brief Description** | In case of a forgotten master password, users can initiate a recovery process to regain access to their password manager account. | |
| **Primary actors** | User | |
| **Preconditions:**   * The user must be already registered. * The user must have been provided with a recovery key during the initial account setup. * The user must have lost access to the application. | | |
| **Post-conditions:**   * The user has successfully reset their master password and can access their password manager account | | |
| **Main Success Scenario:** | | |
| **Actor Action** | | **System Response** |
| 1. The user launches the password manager application. | | 2. Display login screen. |
| 3. The user clicks on the “Forgot Password” option on the login screen. | | 4. Prompt the user to enter their registered email address. |
| 5. The user provides their registered email address. | | 6. Verify email address validity. **(See 6.a. for alternative flow).** |
|  | | 7. Send a verification code to the email address entered. |
| 8. The user enters the verification code received in their email into the application. | | 9. Verify verification code entered by user. **(see 9.a. for alternative flow).** |
|  | | 10. Prompt the user to enter their new master password. |
| 11. User sets a new master password. | | 12. Audit the new password **(See 11.a. for alternative flow).** |
|  | | 13. Update password and notify user with success. |
| **Alternative flows:**  **6.a.** If the email address is invalid, display error message and user is prompted to re-enter a valid email address.  **9.a.** If the verification code is invalid, display an error message and ask the user if they want to resend the code.  **11.a**. if the entered password doesn’t meet the proper security criteria, display error message and user is prompted to enter a valid password. | | |
| **Special Requirements:**  **a) Performance b) Reliability c) Security d) Usability e) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **R4** | **Wipe Account** | |
| **Brief Description** | Users can permanently delete all data and credentials stored in their password manager account, ensuring data privacy in the event of an account closure. | |
| **Primary actors** | User | |
| **Preconditions:**   * **The user must be logged into their password manager account.** | | |
| **Post-conditions:**   * **The user’s account data is completely wiped from the system, and they are logged out.** | | |
| **Main Success Scenario:** | | |
| **Actor Action** | | **System Response** |
| 1. The user selected the option “Wipe account”. | | 2. Present the user with a confirmation message to ensure the user’s intention. **(See 2.a. for alternative flow).** |
| 3. The user confirms their intention to wipe the account | | 4. Initiate the process of wiping the user’s account data. |
|  | | 5. Delete all user-related data, including passwords, and other account information. **(See 5.a. for alternative flow).** |
|  | | 6. Display confirmation message indicating that the account has been successfully wiped. |
|  | | 7. Redirect user to the login screen. |
| **Alternative flows:**  2.a. If the user decided not to proceed with wiping the account, return to the account settings.  5.a. If a technical error occurs that prevents the account from being wiped, display an error message and inform the user. | | |
| **Special Requirements:**  **a) Reliability b) Security c) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: UC01** | **Sign Up** | |
| **Brief Description** | The user creates / registers a new account to be able to use the password manager application. | |
| **Primary actors** | User | |
| **Preconditions:**   1. The user must not have an existing account in the system. | | |
| **Post-conditions:**   1. An account was created for the user. 2. The user details are stored in the database. | | |
| **Main Success Scenario:** An account has been created for the user. | | |
| **Actor Action** | | **System Response** |
| 1. The user launches the password manager application. | | 2. Display login screen. |
| 3. The user clicks on the “Sign Up” option on the login screen. | | 4. Prompt the user to enter registration information |
| 5. The user fills in / provides his email, first name, last name, and his master password. | |  |
| 6. The user clicks on the “Create Account” button. | | 7. Check if the user exists and the details are valid |
|  | | 8. Create an account with the user details if the account does not exist and the details are valid. **(See 8.a. & 8.b. for alternative flow)** |
|  | | 5. Assign a unique number to the account. |
|  | | 6. <extend: Enable 2FA> If the user wants to enable 2FA on his account / link their accounts with a secondary authentication method. |
|  | | 6. Inform the user with the account number. |
|  | | 7. Display the login page. |
| **Alternative flows:**  8.a. If the user already exists, inform the user that a new account cannot be created.  8.b. If the user provided invalid details, generate an error message “Invalid Sign Up”, and ask the user to re-enter / provide valid information regarding sign up. | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: UC02** | **Login** | |
| **Brief Description** | The user tries to log into the system / their personal account by using the email and master password to gain access to their stored credentials. | |
| **Primary actors** | User | |
| **Preconditions:**   1. The user must be already registered. | | |
| **Post-conditions:**   1. The user has access / is logged into the system. | | |
| **Main Success Scenario:** Access has been given to the user into the system. | | |
| **Actor Action** | | **System Response** |
| 1. The user launches the password manager application. | | 2. Display login screen. |
| 3. The user fills in / provides his email and master password. | |  |
| 4. The user clicks on the “Sign In” button. | | 5. Check / Validate the user details in the database. |
|  | | 6. <extend: Recover Access> If the user forgot his master password & clicks on “Forgot Password” |
|  | | 7. Gain access / Log the user into the system if details are valid. **(See 7.a. for alternative flow)** |
|  | | 8. Display the stored credentials / user profile page and main interface of the application. |
| **Alternative flows:**  7.a. If the user provided invalid details, generate an error message “Invalid Login”, and ask the user to re-enter / provide valid information regarding logging in, or choose to cancel the operation, at which point the use case ends.. | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: UC03** | **Add Credential** | |
| **Brief Description** | The user adds a new login credential for a specific website (domain), application, to their database / password manager. | |
| **Primary actors** | User | |
| **Preconditions:**   1. The user must open the application. 2. The user must be already registered. 3. The user must be logged into their password manager account. | | |
| **Post-conditions:**   1. The user added a new login credential. 2. The login credential was stored in the user’s database. | | |
| **Main Success Scenario:** Login credential has been added successfully. | | |
| **Actor Action** | | **System Response** |
| 1. The user selected the option “Add Credential”. | | **2**. Display the fill in form related to adding a new login credential |
| 3. The user fills in / provides the required information (username / email, password, website URL / name, notes). | | 4. <include: Audit Credential Use Case> **(See 4.a. for alternative flow)** |
|  | | 5.<extend: Generate Password Use Case> If user wants to generate a password for new accounts or to replace weak passwords |
| 6. The user clicks on the “Add” button. | |  |
|  | | 7. Verify the user credential and login state. **(See 7.a. for alternative flow).** |
|  | | 8. Add entry / credential in the user’s database. |
|  | | 9. Generate a message “Credential was added successfully!” |
|  | | 10. Display the main password manager UI / database of login credentials. |
| **Alternative flows:**  4.a. If the user selects cancel adding credential, open the main window / his database.  7.a. If the user leaves any mandatory fields, generate an error message “Please fill in the required fields”, and ask the client to fill them. | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: UC04** | **Update Credential** | |
| **Brief Description** | The user edits / updates an existing login credential / entry for a specific website (domain), application, in their database / password manager. | |
| **Primary actors** | User | |
| **Preconditions:**   1. The user must open the application. 2. The user must be already registered. 3. The user must have an existing credential selected to be updated. | | |
| **Post-conditions:**   1. The user updated an existing login credential. 2. The login credential was updated in the user’s database. | | |
| **Main Success Scenario:** Login credential has been updated successfully. | | |
| **Actor Action** | | **System Response** |
| 1. The user selected the option “Update Credential”. | | **2**. Display the fill in form related to updating an existing login credential |
| 3. The user updates / edits the information he wants (username / email, password, website URL / name, notes). | | 4. <include: Audit Credential Use Case> **(See 4.a. for alternative flow)** |
| 5. The user clicks on the “Update” button. | | 6. Verify the user credential and login state. **(See 6.a. for alternative flow).** |
|  | | 7. Update entry / credential in the user’s database. |
|  | | 8. Generate a message “Credential was updated successfully!” |
|  | | 9. Display the main password manager UI / database of login credentials. |
| **Alternative flows:**  4.a. If the user selects cancel updating credential, open the main window / his database.  6.a. If the user leaves any mandatory fields, generate an error message “Please fill in the required fields”, and ask the client to fill them. | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: UC05** | **Remove Credential** | |
| **Brief Description** | The user removes an existing login credential / entry for a specific website (domain), application, in their database / password manager. | |
| **Primary actors** | User | |
| **Preconditions:**   1. The user must open the application. 2. The user must be already registered. 3. The user must have an existing credential selected to be deleted / removed. | | |
| **Post-conditions:**   1. The user deleted an existing login credential. 2. The login credential was deleted in the user’s database. | | |
| **Main Success Scenario:** Login credential has been deleted successfully. | | |
| **Actor Action** | | **System Response** |
| 1. The user selected the option “Remove Credential”. | | **2**. Display a pop up message “Are you sure you want to delete the selected entries?” |
| 3. The user responds by clicking the “Yes” button. | | 4. Verify / Validate the user choice and the credentials. **(See 4.a. for alternative flow).** |
|  | | 5. Remove entry / credential in the user’s database. |
|  | | 6. Generate a message “Credential was removed successfully!” |
|  | | 7. Display the main password manager UI / database of login credentials. |
| **Alternative flows:**  4.a. If the user selects cancel removing credential, open the main window / his database. | | |
| **Special Requirements:** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **E5: Remove Credential** | |
| **Brief Description** | Users can delete login credentials they no longer need from their password manager. | |
| **Primary actors** | User | |
| **Trigger(s)** | User clicks on “remove credential” button | |
| **Preconditions:** | User logged in  the “page” of the credential to be removed is open (credential selected) | |
| **Postconditions:** | Credential deleted | |
| **Normal Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. clicks on “remove credential” button | | 2. Find the credential to be removed |
|  | | 3. Ask user to confirm in a dialbox (may ask for masterkey -with alternative flow if wrong) (See 3.a for alternative flow) |
| 1. Confirm removal | |  |
|  | | 1. Delete the credential |
| Alternative flows:  3.a The user does not confirm: get back to the previous page (credential page) | | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **E6: AutoFill Credentials** | |
| **Brief Description** | The password manager can automatically fill in login credentials when users visit websites or apps, streamlining the login process. | |
| **Primary actors** | User | |
| **Trigger(s)** | User clicks on “open in web browser” | |
| **Preconditions:** | User logged in  The “page” of the credential to log in is open (credential selected)  The website is “known” to the software | |
| **Postconditions:** | Selected website open and user credentials entered | |
| **Normal Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. User clicks on “open in web browser” | | 2. Find the credentials |
|  | | 3. Open the web page with credentials |
| Alternative flows: -None- | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | |  | | --- | | **E7: Search Credential** | |  | | |
| **Brief Description** | Access and retrieve specific stored credentials quickly. | |
| **Primary actors** | User | |
| **Trigger(s)** | User clicks on the search box | |
| **Preconditions:** | User logged in | |
| **Postconditions:** | Credential found & opened | |
| **Normal Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. User clicks on the search box | |  |
| 2. User enters credential’s name | |  |
|  | | 3. Dynamically displays all credentials with entered characters (See 3.a for alternative flow) |
| 1. Select searched credential | |  |
| Alternative flows:  3.a No credential was found: display an option “no credential was found” and suggest to create a new one | | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **E8: Generate Password** | |
| **Brief Description** | Users can use the password manager to generate strong, unique passwords for new accounts or to replace weak passwords. | |
| **Primary actors** | User | |
| **Trigger(s)** | User clicks on “generate password” button | |
| **Preconditions:** | User logged in  The “page” of **adding or updating** the credential is open | |
| **Postconditions:** | New password is generated and saved for the credential | |
| **Normal Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. User clicks on “generate password” button | | 2. Shows optional features (for eg. Compatible with mobile/ps4…etc/pseudorandom or complex phrase CHECK) with default values (user without security knowledge) |
|  | | 3. generate strong password |
|  | | 4. Ask user for confirmation (See 4.a for alternative flow) |
| 1. User confirms | |  |
|  | | 1. Credential is saved and a yellow box appears in the bottom of the credential page saying “update your password in your account and come to confirm” with confirm button |
|  | | 1. When confirmed, the page gets back to normal |
| Alternative flows:  4.a User wants to re-generate another password: clicks on “re-generate another password” button which restart step 3 | | |

# Appendix B – Test cases specification

< **ToDo:**

* Create a table for each test case used to test the correct delivery of your functional requirements.

/>

**Test Case Template (Example)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 3.1 | **Associated user case ID** | U3 |
| Test designed by | Abbas Ibn Firnas | **Test design date** | 15/01/2023 |
| **Executed by** | Abbas Ibn Firnas | **Execution date** | 20/02/2023 |
| **Test case name** | ATM change PIN | | |
| **Short description** | Test the ATM change PIN use case | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user has a valid ATM card   + The user has accessed the ATM by placing his ATM card in the machine   + The current PIN is 1234   + The system displays the main menu | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Click the 'Change PIN' button | The system displays a message asking the user to enter the new PIN | As expected | Pass |  |
| 2 | Enter '5555' | The system displays a message asking the user to confirm (re-enter) the new PIN | As expected | Pass |  |
| 3 | Re-enter '5555' | The system displays a message of successful operation  The system asks the user if he wants to perform other operations | As expected | Pass |  |
| 4 | Click 'YES' button | The system displays the main menu | As expected | Pass |  |
| 5 | Check post-condition 1 |  |  |  |  |
| 6 | Repeat steps 1,2,3 using another PIN say '6666' and click 'NO' button | The system is exited and displays a greeting message asking the user to place his ATM card in the machine |  |  |  |
| 7 | Check post-condition 2 |  |  |  |  |
| 8 | Repeat steps 1,2, using another PIN say '7777' | The system displays a message asking the user to confirm (re-enter) the new PIN |  |  |  |
| 9 | Enter a wrong confirmation (say ‘9876’) | The system displays a message of unsuccessful operation and asks the user to confirm the correct PIN |  |  |  |
| 10 | Re-enter ‘7777’ | The system displays a message of successful operation  The system asks the user if he wants to perform other operations |  |  |  |
| 11 | Click 'NO' button | The system is exited and displays a greeting message |  |  |  |
| 12 | Check post-condition 3 |  |  |  |  |
| **Post-conditions:**  1. The new PIN '5555' is saved in the database  2. The new PIN '6666' is saved in the database  3. The new PIN '7777' is saved in the database | | | | | |

# Other Appendices

< **ToDo:**

* The following are possible, additional appendices you may add to the project report:
* Any questionnaires, interview questions, etc. used in your project.
* Installation manual and deployment diagram
* User manual
* Any other appendices may be included to provide supporting details that could aid in understanding this report.

/>