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

Senior Project Report

**PassGuard: 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**

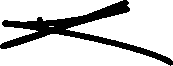
This project report is submitted to the Department of Computer Science and Engineering of Qatar University in partial fulfillment of the requirements of the Senior Project course

# Declaration

This report represents our original work, and any external sources utilized have been appropriately cited. It details the work we have undertaken for our senior project. We are aware of the university’s strict policy on plagiarism and the corresponding penalties, and we declare that this report is the product of our own work.

Student: Youssef Sherif Aly             Date: 21st September 2023

Signature:



Student: Essa Ahmed Abou Jabal             Date: 21st September 2023

Signature: A black and white image of a frog

Description automatically generated

Student: Khalifa Ebrahim Yousuf Date: 21st September 2023

Signature: Khalifa

Student: Mohamed-Dhia Abdaoui Date: 21st September 2023

Signature: Mohamed-Dhia Abdaoui

# Abstract

< **ToDo:**

* The abstract is a brief overview (500 words max) of your project's objectives, proposed solution, accomplishments:
* Clearly introduce the problem your project has addressed. Highlight the significance of addressing the problem.
* Provide a concise overview of your proposed solution.
* Highlight key achievements and most important conclusions.

/>

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. The prevalence of weak and recycled passwords poses a significant threat to online security. A study by the National Institute of Standards and Technology (NIST) found that 81% of data breaches were caused by weak or reused passwords [[1]](#R1). This vulnerability is further compounded by the practice of reusing passwords across multiple accounts, a single breach exposing a multitude of sensitive information.

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 secure 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. Additionally, our application securely stores sensitive documents, providing users with a centralized repository for their important files. This eliminates the need for scattered storage locations and enhances accessibility.

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. Our application offers a robust set of security features, including:

* **Master Password Protection:** Users only need to remember one master password to access all their stored credentials, ensuring convenience without compromising security.
* **Industry-Standard Encryption:** Passwords and documents are encrypted using industry-standard encryption techniques, safeguarding sensitive data from unauthorized access.
* **Local Storage:** All passwords and documents are stored locally on the user's device, giving them complete control over their stored credentials.
* **Password Export/Import:** Users can easily export and import their passwords, ensuring backup and portability with other devices.

By providing an intuitive interface, secure password and document storage, and the capacity to generate robust passwords, we aim to provide users with a tool that elevates their security without sacrificing the ease and comfort they seek.

<**ToDo:** **Before submitting your final report:**

* **Complete all Report Sections**: Ensure that each section of the report is thoroughly developed following the provided ToDo instructions for each section.

Remember that the template serves as a guide. In collaboration with your supervisor, you may tailor the report structure as necessary by adding sections or subsections that align with the specific nature of your project.

* **Coherence and Compliance Check**: Thoroughly review the entire document to ensure its cohesiveness and alignment with the Project Guide and Project Grading Rubrics. Confirm that all specified requirements have been addressed.
* **Supervisor Feedback**: Seek feedback from your supervisor and promptly address any concerns or issues raised in their feedback. Collaborate with your supervisor to refine the report content.
* **Update Table of Contents and Lists**: Update the Table of Contents, List of Figures, and List of Tables to reflect any changes or additions made during the report's development.
* **Track Changes**: If utilizing Word's 'Track Changes' feature, ensure that all changes are accepted before submitting your final report.
* **Retain ToDo Instructions**: Keep the ToDo instructions as they will be valuable for both semesters. Remove them only when preparing the *Final Revised Report* after the Senior Project Presentation and after addressing examiners feedback at the end of Semester 2.
* **Semester 2 Final Report Additional Tasks:**
  + **Review of Semester 1 Content**: Revisit the content produced in Semester 1 and update it to reflect any changes that have occurred in your project since last semester.
  + **Abstract Enhancement**: Revise the Abstract to include key project achievements and significant conclusions. The final paragraph should emphasize the uniqueness of your design and discuss the impacts of your engineered solution.
  + **Tense Alignment**: Review the entire document to ensure consistent usage of present and past tenses while avoiding the inclusion of future tense (e.g., 'we will' or 'system should') except in the Future Work section.

/>

# Acknowledgment

< **ToDo:**

* Acknowledge any assistance you received for your project.

/>

We extend our gratitude to Allah for granting us the strength and resilience to initiate and tackle this project. We owe the success of this project to the invaluable support we have received from our friends and family, whose encouragement fueled our determination to push forward. We also owe a special thanks to our Supervisor Dr. Mohammed Saleh whose guidance, support, and time have been invaluable to us and to the quality of this project.

# Table of Content

[Declaration ii](#_Toc151848222)

[Abstract iii](#_Toc151848223)

[Acknowledgment v](#_Toc151848224)

[List of tables ix](#_Toc151848225)

[1. Introduction and Motivation 11](#_Toc151848226)

[1.1. Problem statement 11](#_Toc151848227)

[1.2. Project objectives 14](#_Toc151848228)

[1.3. Expected benefits and impacts 15](#_Toc151848229)

[1.4. Market Research and Business Viability 16](#_Toc151848230)

[2. Background and related work 25](#_Toc151848231)

[2.1. Background 25](#_Toc151848232)

[2.2. Related work 27](#_Toc151848233)

[3. Requirements analysis 34](#_Toc151848234)

[3.1. Software development process 34](#_Toc151848235)

[3.2. Applying the software development process 36](#_Toc151848236)

[3.3. Functional requirements 36](#_Toc151848237)

[3.4. Non-functional requirements 38](#_Toc151848238)

[3.5. Assumptions 40](#_Toc151848239)

[3.6. Ethics 41](#_Toc151848240)

[4. Project Plan 43](#_Toc151848241)

[4.1. Project milestones 43](#_Toc151848242)

[4.2. Project timeline 45](#_Toc151848243)

[4.3. Anticipated risks 49](#_Toc151848244)

[5. System Architecture 51](#_Toc151848245)

[5.1. High level system architecture 51](#_Toc151848246)

[5.2. Alternative solutions and tradeoffs 52](#_Toc151848247)

[6. Detailed Design 55](#_Toc151848248)

[6.1. Structural model 55](#_Toc151848249)

[6.2. Behavioral model 57](#_Toc151848250)

[6.3. Database design 63](#_Toc151848251)

[6.4. User interface design 64](#_Toc151848252)

[6.5. Design patterns 66](#_Toc151848253)

[7. Implementation 71](#_Toc151848254)

[7.1. Hardware/software used 71](#_Toc151848255)

[7.2. Challenging issues and solutions 75](#_Toc151848256)

[8. Testing and evaluation 77](#_Toc151848257)

[8.1. Functional testing 77](#_Toc151848258)

[8.2. Non-functional testing 90](#_Toc151848259)

[9. Conclusion 90](#_Toc151848260)

[10. Future work 91](#_Toc151848261)

[11. Student reflections 92](#_Toc151848262)

[References 95](#_Toc151848263)

[Appendix A – Use cases specification. 97](#_Toc151848264)

[Appendix B – Test cases specification 107](#_Toc151848265)

[Other Appendices 111](#_Toc151848266)

# List of Figures

[Figure 1: Password Manager architecture [6] 11](#_Toc151848027)

[Figure 2: Market share of password management software worldwide in 2023 [9] 16](#_Toc151848028)

[Figure 3: Global password management market from 2015 to 2025 [10] 17](#_Toc151848029)

[Figure 4: Age Data from Arabic responses 17](#_Toc151848030)

[Figure 5: Age Data from English responses 18](#_Toc151848031)

[Figure 6: Field of Study from Arabic responses 18](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848032)

[Figure 7: Field of Study from English responses 18](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848033)

[Figure 8: Occupation from Arabic Responses 19](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848034)

[Figure 9: Occupation from English Responses 19](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848035)

[Figure 10: "Would a password manager be useful for you?" Arabic responses. 20](#_Toc151848036)

[Figure 11: "Would a password manager be useful for you?" English responses. 21](#_Toc151848037)

[Figure 12: Password Strengths figure shown to the participants [11]. 22](#_Toc151848038)

[Figure 13: Arabic Responses to Password Strength question 22](#_Toc151848039)

[Figure 14: English Reponses to Password Strength question 23](#_Toc151848040)

[Figure 15: Nord Pass UI and Desktop Application [20] 28](#_Toc151848041)

[Figure 16: Bit Warden UI and Desktop Application [21] 29](#_Toc151848042)

[Figure 17: Dash Lane UI and \*Old\* Desktop Application [22] 29](#_Toc151848043)

[Figure 18: Zoho Vault UI and Desktop Application [23] 30](#_Toc151848044)

[Figure 19: 1Password UI and Desktop Application [24] 30](#_Toc151848045)

[Figure 20: Keeper UI and Desktop Application [25] 31](#_Toc151848046)

[Figure 21: LogMeOnce UI and Desktop Application [26] 32](#_Toc151848047)

[Figure 22: Password Boss UI and Desktop Application [27] 32](#_Toc151848048)

[Figure 23: RoboForm UI and Desktop Application [28] 33](#_Toc151848049)

[Figure 24: Iterative Model 35](#_Toc151848050)

[Figure 25: Use Case Diagram 37](#_Toc151848051)

[Figure 26: SDP 1 Gantt Chart. 48](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848052)

[Figure 27: SDP 2 Gantt Chart. 49](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848053)

[Figure 28: MVC High-level architecture 51](#_Toc151848054)

[Figure 29: React Component Hierarchy 56](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848055)

[Figure 30: Repository Class Diagram 57](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848056)

[Figure 31: Sign Up Design Sequence Diagram 58](#_Toc151848057)

[Figure 32: Login Design Sequence Diagram 58](#_Toc151848058)

[Figure 33: Add Credential Design Sequence Diagram 59](#_Toc151848059)

[Figure 34: Update Design Sequence Diagram 59](#_Toc151848060)

[Figure 35: Remove Credential Design Sequence Diagram 59](#_Toc151848061)

[Figure 36: Autofill Design Sequence Diagram 60](#_Toc151848062)

[Figure 37: Search Credential Design Sequence Diagram 60](#_Toc151848063)

[Figure 38: Generate Password Design Sequence Diagram 61](#_Toc151848064)

[Figure 39: Export Design Sequence Diagram 61](#_Toc151848065)

[Figure 40: Import Design Sequence Diagram 62](#_Toc151848066)

[Figure 41: Recovery Access Design Sequence Diagram 62](#_Toc151848067)

[Figure 42: Database Design Model 64](#_Toc151848068)

[Figure 43: Color Scheme 65](#_Toc151848069)

[Figure 44: Typography 65](#_Toc151848070)

[Figure 45: Conceptual Dashboard with Figma 66](#_Toc151848071)

[Figure 46: MVVM Design Pattern 68](#_Toc151848072)

[Figure 47: Using ternary operators to change color depending on the password strength. 68](#_Toc151848073)

[Figure 48: Using the use State Hook to decide on whether to expand the navigation bar. 69](#_Toc151848074)

[Figure 49: Instantiating one global Prisma client instance. 69](#_Toc151848075)

[Figure 50: Using the export of Prisma to enable access to all methods and objects of Prisma. 70](#_Toc151848076)

[Figure 51: User Database before Adding the User. 78](#_Toc151848077)

[Figure 52: Expected and Actual Results of Creating a User. 79](#_Toc151848078)

[Figure 53: Results of creating a new user. 79](#_Toc151848079)

[Figure 54: User database after adding a User. 80](#_Toc151848080)

[Figure 55: Trying to add the same user to the User database. 80](#_Toc151848081)

[Figure 56: Expected and Actual Results of retrieving an existing user from the database. 81](#_Toc151848082)

[Figure 57: Results of retrieving an existing user. 82](#_Toc151848083)

[Figure 58: Expected and Actual results of adding a credential to a non-existing user. 82](#_Toc151848084)

[Figure 59: Receiving error for adding a credential to a non-existing user. 83](#_Toc151848085)

[Figure 60: Expected and Actual results of adding a credential to an existing user. 83](#_Toc151848086)

[Figure 61: Results of Adding a new credential. 83](#_Toc151848087)

[Figure 62: Before adding a new credential (filling the form) 84](#_Toc151848088)

[Figure 63: After adding a new credential. 84](#_Toc151848089)

[Figure 64: Credential database before updating the credential. 85](#_Toc151848090)

[Figure 65: Expected and Actual results of updating a credential for an existing user. 85](#_Toc151848091)

[Figure 66: Results of Updating an existing credential. 86](#_Toc151848092)

[Figure 67: Credential database before after updating the credential. 86](#_Toc151848093)

[Figure 68: Expected and Actual results of updating a non-existing credential. 86](#_Toc151848094)

[Figure 69: Results of updating a non-existing credential. 87](#_Toc151848095)

[Figure 70: Credential database before deleting the credential. 87](#_Toc151848096)

[Figure 71: Expected and Actual results of deleting an existing credential. 88](#_Toc151848097)

[Figure 72: Results of testing the 3 main operations (CUD) 88](#_Toc151848098)

[Figure 73: Credential database after deleting the credential. 89](#_Toc151848099)

[Figure 74: Results of deleting a non-existing credential. 89](#_Toc151848100)

# List of tables

[Table 1: Expected benefits and impacts on various contexts. 15](#_Toc151848183)

[Table 2: Password Analysis Evaluation 23](#_Toc151848184)

[Table 3: Key similarities and differences between out project and the related work 34](https://qucloud-my.sharepoint.com/personal/ya1901748_qu_edu_qa/Documents/CS_SDP_MergedReportTemplate_V2%201.docx#_Toc151848185)

[Table 4: Use Cases Summary 37](#_Toc151848186)

[Table 5: Evaluation plan for the non-functional requirement 39](#_Toc151848187)

[Table 6: Project-specific code of ethics and professional responsibilities 42](#_Toc151848188)

[Table 7: Implementation Tiers 43](#_Toc151848189)

[Table 8: Project milestones 44](#_Toc151848190)

[Table 9: Project timeline SDP 1 45](#_Toc151848191)

[Table 10: Project Risks 50](#_Toc151848192)

[Table 11: Alternative Tools & Solutions 53](#_Toc151848193)

[Table 12: Hardware/software to be used. 71](#_Toc151848194)

# Introduction and Motivation

## Problem statement

< **ToDo:**

* Give a clear (and preferably formal) definition of the problem you aim to solve. Additionally, consider including an overview figure to visually depict the problem for enhanced clarity.
* Highlight the significance of tackling the problem (i.e., why it is important). Support your argument with relevant references and a real-world example to motivate your project.
* Outline the technical and non-technical challenges associated with the project using bullet points. For example, integrating machine learning models into your solution is a technical challenge, while the need for collecting labeled data is a non-technical challenge.
* Explain aspects to classify this problem as a complex computing problem:
* Complex computing problems include one or more of the following characteristics: involving wide-ranging or conflicting technical issues and tradeoffs, having no obvious solution, involving diverse groups of stakeholders, including many component parts or sub-problems, involving multiple disciplines, or having significant consequences in a range of 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 [[2]](#R2). In addition, according to a late 2019 Harris Poll, 75% of Americans struggle to keep track of all their passwords [[3]](#R3). 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 [[4]](#R4). 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 [[5]](#R5). 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 retrieval when necessary.

A diagram of a computer system

Description automatically generated

Figure : Password Manager architecture [[6]](#R6)

Our team is committed to creating a safe and user-friendly password manager app that addresses the widespread problems of weak, reused, and default passwords. Our group's strong interest in cyber security drives us to investigate and address the complex problems related to password management. Our goal is to develop a password manager that is highly secure and has a smooth and easy-to-use interface. Striking a balance between safety and comfort is difficult since it requires thorough consideration to be given to user interface design, multi-factor authentication protocols, password generating algorithms, and encryption techniques. As a result, its complexity, and the challenge to strike a balance between security and usability, developing a password manager is a complicated task that's ideal for a senior-level design project.

We have outlined the major technical and non-technical challenges below and have prioritized them accordingly to tackle each one comprehensively to successfully reach an optimal solution that encompasses the most crucial components.

Technical Challenges:

* **Robust Encryption:** Creating robust encryption methods to transfer and store private user credentials securely.
* **User Authentication:** Making sure user authentication is safe and intuitive.
* **Password Generation:** Developing a powerful and effective password generation algorithm that produces strong unique passwords.
* **Intuitive UI**: Creating a password management interface that is easy to use and intuitive for all users.
* **Password Recovery and Backup:** Implementing a secure password backup and recovery method that is user-friendly.
* **Password Analysis:** Implementing password vulnerability scanning to improve user's password management.
* **Compliance with Privacy Regulations:** Making sure the application conforms with laws governing user privacy and data protection.
* **Credential auto-filling:** This is especially difficult because we are developing a native Desktop application with the passwords stored locally on the user's device.

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**

The application tries to overcome several complexities that we have categorized into the following main characteristics.

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

< **ToDo:**

* Make sure your objectives are specific, measurable, achievable, relevant, and time-bound (SMART). Use bullets format.
* Clearly state what you aim to achieve with your project and define your objectives in terms of measurable outcomes against which the success of the project will be judged.
* Break down your objectives into smaller, more manageable sub-objectives that can be achieved incrementally.
* Use action verbs to describe your objectives, such as design, implement, test, or evaluate.
* In the Conclusion section you must summarize the project objectives and discuss their level of attainment.

/>

We have categorized our main general goal of the project and have sequentially broken it down into the following main and sub-objectives so that we could successfully prioritize and implement them accordingly.

**Main Objective**

Develop a User-Friendly Password Manager Application that aligns with the project’s goal of enhancing user convenience and security within 8 months. To achieve this, we have segmented the main objectives into manageable sub-objectives that follow **SMART** as shown below**.**

Sub-Objectives

* Design an easy-to-use UI that is suitable for a wide range of users and perform user testing to improve the interface based on feedback by the end of Fall 2023.
* 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 throughout the project duration.
* Develop a password generation functionality that generates strong, unique passwords by utilizing established password generation algorithms during Spring 2023.
* Ensure that password data is consistently accessible on all devices the user desires to enhance user convenience and accessibility by the end of the project.
* Implement multi-factor authentication to enhance security and user confidence by the end of Spring 2023.
* 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 by the end of Fall 2023.
* 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

< **ToDo:**

* List and discuss your project contributions.
* List and explain in table format the expected benefits and the **local and global impacts** of your project on individuals, organizations, and society. For example:
* Highlight the ways in which the project's outcomes will benefit the project stakeholders, such as users and customers, both individuals and organizations.
* Discuss the potential for your project in creating social and cultural changes within communities and individuals.
* Evaluate the anticipated contributions of your solution towards addressing global issues, such as climate change, poverty alleviation, literacy promotion, economic growth and competitiveness, sustainable development, and overall human well-being.
* Revisit and enrich this section towards the end of the project.

/>

The alarmingly high frequency of password breaches, which is made worse by the general usage of weak and reused passwords, highlights the pressing need for safe and convenient password management solutions. According to Verizon's Data Breach Investigations Report 2023, studies show that a startling 81% of data breaches include compromised or stolen passwords, underscoring the urgent need for stronger security measures [[7]](#R7). Furthermore, Microsoft reported that 60% of users reuse their passwords for several accounts, which leaves big gaps that hackers take advantage of [[8]](#R8).

Our password manager app contributes to these issues by providing a safe and easy-to-use method of storing and managing passwords. Some of these contributions include:

**Determining and evaluating user requirements and needs:** To determine the requirements and expectations of prospective password manager users, we carried out a thorough investigation. Usability testing, interviews, and surveys were all part of this. Our password manager was designed with a wide range of users' needs in mind, thanks to the insights gathered from our study.

**Creating and designing the password manager with security and privacy in mind:** We created a user-friendly and safe password manager software program that gives the user full control of his credential management. In addition, many features in the application make it simple to create, store, and manage strong passwords.

Table : 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 characteristics, preferences, and pain points.
* Analyze the competing solutions and how does yours differ from those offered by competitors? Showcase the novel features of your solution and the tangible benefits it delivers.
* Develop a business plan including your business model, pricing, marketing strategy to bring your product to market and make it competitive.

/>

PassGuard is intended for a wide range of users, including students, employees, and businessmen (Mass Market). However, it is flexible and may be improved or adapted to certain business domains or objectives. For example, it may be improved for government security needs or tailored with extra features to meet the needs of a particular business. Password cracking is becoming more common, as noted earlier, which makes our application pertinent and necessary for the public. To emphasize this point, please refer to [section 1.1](#ProblemStmt) for more details. The below figures show some crucial statistics that highlight the importance of PassGuard and its market need.

A graph of data with numbers and percentages

Description automatically generated

Figure : Market share of password management software worldwide in 2023 [[9]](#MR1)

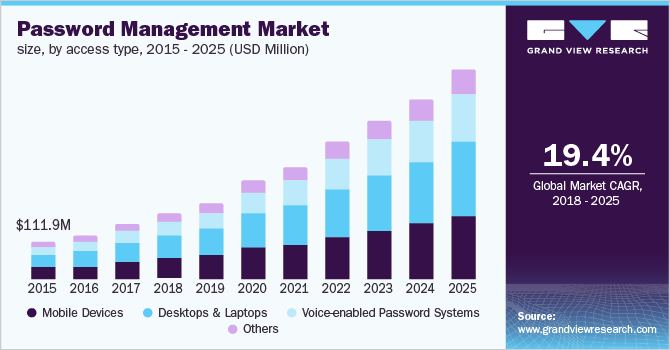


Figure : Global password management market from 2015 to 2025 [[10]](#MR2)

In addition, we conducted a survey with 227 individuals (146 men and 81 females) spanning a range of age groups, sectors, and jobs to assess PassGuard’s applicability and efficacy. The following data provides specifics on the outcomes.

Forms response chart. Question title: الفئة العمرية
. Number of responses: 175 responses.

Figure : Age Data from Arabic responses

Forms response chart. Question title: Age group
. Number of responses: 52 responses.

Figure : Age Data from English responses

Forms response chart. Question title: مجال التخصص (عمل أو دراسة)
. Number of responses: 175 responses.

Figure : Field of Study from Arabic responses

Forms response chart. Question title: Field of Study
. Number of responses: 51 responses.

Figure : Field of Study from English responses

Forms response chart. Question title: العمل
. Number of responses: 175 responses.

Figure : Occupation from Arabic Responses

Forms response chart. Question title: Occupation
. Number of responses: 52 responses.

Figure : Occupation from English Responses

The survey analysis covers three main points:

1. **Survey Sample Scope Analysis**

Based on the information provided, the survey sample was categorized into eight main groups, considering common and significant characteristics such as job and age that will help to get a better understanding and analysis of the answers.

The target groups, representing 97.3% of the total dataset, include:

* Engineering Male Students (Between 18 to 25 Years Old): 64
* Non-Engineering Male Students: 29
* Engineering Female Students: 22
* Non-Engineering Female Students: 20
* Female Employees (Over 20 Years Old): 27 (11 In Education; 22 Are Above 30 Years Old)
* Male Employees (Over 20 Years Old): 38 (17 In Education, 10 In Engineering)
* Housewives: 6
* Male & Female (Below 18 Years Old): 12

1. **Need For the Application (Based on the Sample Scope)**

The survey initiated with a direct question: “Would a password manager be useful for you?” Among respondents, 42% expressed that a password manager would be highly beneficial.

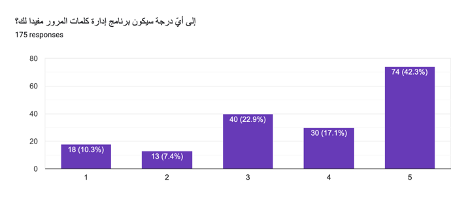


Figure 10: "Would a password manager be useful for you?" Arabic responses.

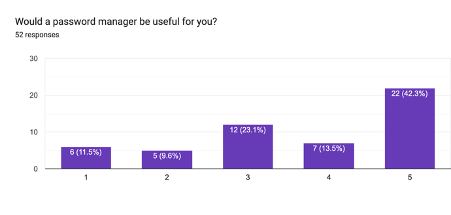


Figure 11: "Would a password manager be useful for you?" English responses.

1. Security behavior

This section encompasses four key aspects representing the primary services that a password manager aims to streamline.

* The Method of Password Storage.
* The Frequency of Forgetting Passwords.
* The Tendency to Reuse Passwords.
* History of Security Breaches.

A notable 31.8% of respondents rely on memorizing passwords, while 25.6% prefer storing them in files on their phones or computers. Merely 14.4% use a password manager. Regarding memory, 38.7% occasionally forget their passwords (rating 2 on a scale of 5). Furthermore, 36% reuse the same password for 1 to 3 accounts, and 31% for 4 to 10 accounts. Interestingly, a significant majority (87.6%) have not experienced password leaks. Only a limited 12% admitted to falling victim to cyber-attacks 1 to 3 times.

It is noteworthy that 52.4% of respondents were unfamiliar with password managers, potentially explaining the applications’ low usage rate.

1. Security knowledge

Given that the application is designed for the public, it is essential to assume that users may lack prior knowledge of security measures. To determine users’ awareness of their password strength, we posed a direct question: “How strong do you think your password are?” Subsequently, participants were presented with a figure illustrating the actual password strength. We then readdressed the question to highlight any disparities between their perceived and actual password strength.



Figure 12: Password Strengths figure shown to the participants [[11]](#R11).

The results were as follows:

Forms response chart. Question title: حسب رأيك، ما مدى قوة كلمات المرور التي تضعها؟
. Number of responses: 175 responses.

Figure 13: Arabic Responses to Password Strength question

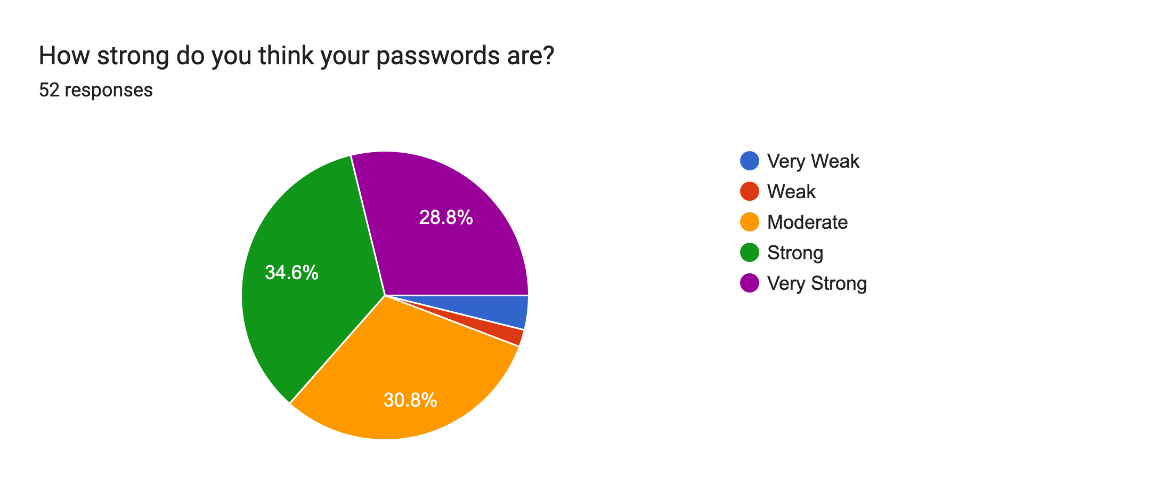


Figure 14: English Reponses to Password Strength question

The results analysis can be summarized in this table as follows:

Table : Password Analysis Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Security Level | Password strength self-evaluation 1  (fig.13,14) | Color in figure | Password strength self- evaluation 2 | Matching ratio between color & evaluation 2 | Difference between Evaluations 2 and 1 |
| Very weak  (purple) | 5 | 13 | 11 | 5 (38%) | +6 |
| Weak (red) | 5 | 82 | 44 | 40 (49%) | +39 |
| Moderate (orange) | 101 | 61 | 81 | 44 (72%) | -20 |
| Strong (yellow) | 77 | 32 | 55 | 23 (71%) | -22 |
| Very strong (green) | 39 | 39 | 36 | 29 (74%) | -3 |

As expected, after seeing the password strength graph, the answers in self-evaluation (2) for weak passwords increased, while for the strong passwords decreased.

While it is acknowledged that the matching ratio between the color of strength of password and self-evaluation (2) is low, the overall outcome still underscores a widespread misjudgment of password strength. This reveals not only a misperception of users’ own password security but also a lack of robust security knowledge among users, although there might be some confusion due to potential shortcomings in the survey explanation.

* **Observations**: It is noteworthy that the group with the largest disparity between perceived and actual password strength deserves special attention. This suggests a potential gap in understanding, which could be due to the survey interpretation or to a general lack of security awareness.
* **Recommendations**: It is advisable not to assume user understanding, particularly during the implementation of adding or updating a credential. That’s why we should take that in consideration by incorporating guidelines, having straight forward activity flows, and a user-friendly interface that can be instrumental in closing this knowledge gap.

1. **Business Considerations & Application Features**

Then, we wanted to know the essential features desired by users when using a password manager. According to the survey responses, the top type of accounts individuals wishes to secure are their Bank accounts (Over 60%), Email accounts (Around 50%), and Social Media accounts (Around 50%). On the matter of pricing, a significant 70% are unwilling to pay for such a service, with only 25% expressing a willingness to pay for usage time rather than a subscription. This poses a substantial business challenge that requires resolution through either a robust marketing strategy to persuade users to pay or by identifying alternative income sources not directly tied to customers.

A substantial 61% of respondents expressed trust in using a password manager. However, the remaining 39% cited various reasons for their lack of trust:

Primary Concerns

* Privacy concerns, particularly distrust in third-party services (especially non-open source).
* Potential hacking incidents leading to password leaks, given its attractiveness as a target for hackers.
* Lack of awareness about the application

These concerns have been already addressed in our business plan:

Trust Assurance

* Users can trust PassGuard as it operates offline, residing on their personal devices. Saving passwords online would’ve raised privacy concerns.
* Decentralization of databases (each user having their local database) minimizes the convenience for hackers to target multiple users. However, potential vulnerabilities require attention from the implementation and security team.
* We need to include and implement a comprehensive marketing plan to educate users about password managers since a lot of people don’t know about it.

Regarding features, nearly all suggested features were equally demanded by respondents, with slight preferences for the generation of strong passwords, password recovery, and auto-fill features.

Concerning portability, a majority preferred the app on smartphones, while fewer favored it on desktops. The current plan is to develop it as a desktop app initially, with future plans to expand to a mobile app.

We also received valuable suggestions from respondents that included the following:

* Implementation of MFA (Multi-Factor Authentication) mechanisms for authentication.
* Several fail-safes included beyond “Forget Password” for cases where the primary account of a user or device access is lost.
* Consideration of making the application open source.

Another critical aspect of business viability is Pricing. As previously discussed, many individuals are hesitant to pay for such services despite their importance. The existence of competitors in the market providing comparable services with different prices, as will be discussed in Section 2.2 below, further complicates issues.

Taking all these factors into consideration, we have decided to offer our application for free to casual users. Customized versions, which are a part of our plans, will be available for specific sectors or businesses seeking additional features like document and application locking. The pricing for these versions will be determined through negotiations with individual customers, considering factors such as security requirements, special requested features, and the size of the customer.

There is also the potential to generate revenue through in-app advertising. However, this option requires careful consideration to assess its impact on application security, functionality, and user experience.

Regarding marketing, our strategy primarily involves leveraging social media. Additionally, partnerships with institutions and government entities are crucial to reach large user groups, particularly employees who could benefit from our application PassGuard.

# Background and related work

## Background

< **ToDo:**

* This section aims to equip the reader with the essential background information to understand your project.
  + Provide a brief overview of the field or domain that your project is focused on including any relevant history or trends.
  + Identify and clarify the major relevant concepts, theories, or technologies that are necessary for understanding your project.

/>

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 Cyber threats' domain & The Importance of Password Management**

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. We have highlighted a few of the key issues that outline the importance of password management below.

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.

Important Terminologies

**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 (encrypted 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

<**ToDo:**

* Discuss related works and solutions that have addressed the same problem:
  + Under the guidance of your supervisor, identify and cite related works and solutions. A list of 10 to 15 references should be adequate to include in this section. This can include academic papers, previous project reports, or other relevant sources.
  + Present a summary of existing solutions and evaluate **their strengths and limitations**.
  + Highlight the uniqueness and novelty of your solution by explaining how it distinguishes itself from existing solutions or builds upon them. This may include novel design, algorithm, or other distinctive aspects of your solution. To enhance clarity, we recommend employing a table to summarize the main points of comparison between your project and the related work. This table can be organized with the solutions (including yours) as rows and the key features as columns, or vice versa.

/>

Developers aim to improve password managers with new features but face criticisms and security issues. PassGuard 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. This section will be discussing 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 [[7]](#E1), 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, PassGuard 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 15: Nord Pass UI and Desktop Application [[20]](#E14)

2 - Bit Warden

Bit warden stands out as an affordable and feature-rich open-source password manager [[8]](#E2) 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, PassGuard 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.

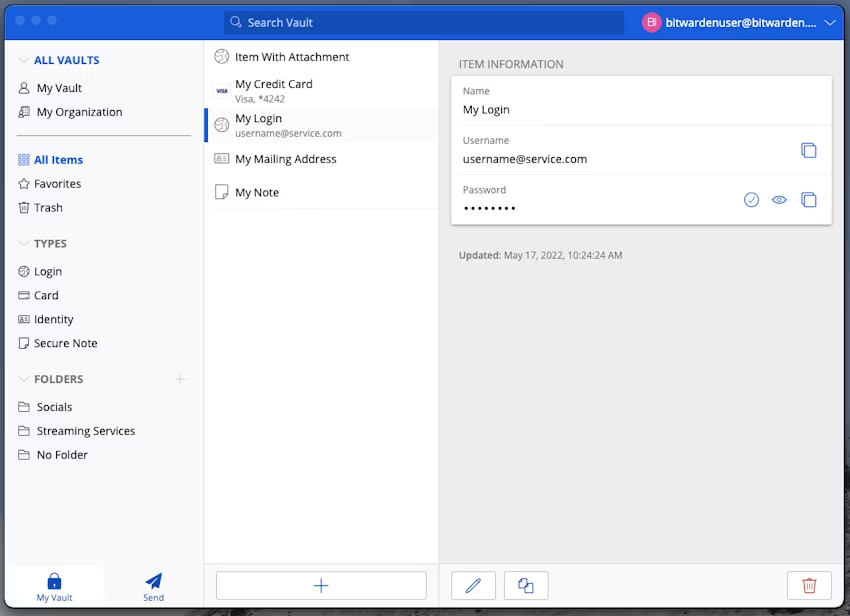


Figure 16: Bit Warden UI and Desktop Application [[21]](#E15)

3 - Dash Lane

Dash lane, a leading 2023 password manager, offers robust security with AES encryption, biometric authentication, and an integrated VPN [[9]](#E3). 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 PassGuard 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 17: Dash Lane UI and \*Old\* Desktop Application [[22]](#E16)

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 [[10]](#E4). Comparatively, it may have a steeper learning curve than some competitors. When it’s compared to PassGuard, usability 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 18: Zoho Vault UI and Desktop Application [[23]](#E17)

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 it lacks a free version and money-back guarantee, it offers a 14-day trial. Notably, Travel Mode enhances security during travel [[11]](#E5). 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 PassGuard, there is no need to worry about the subscription plans, the issues with form filling, and import options since all of these are provided for the user with ease.

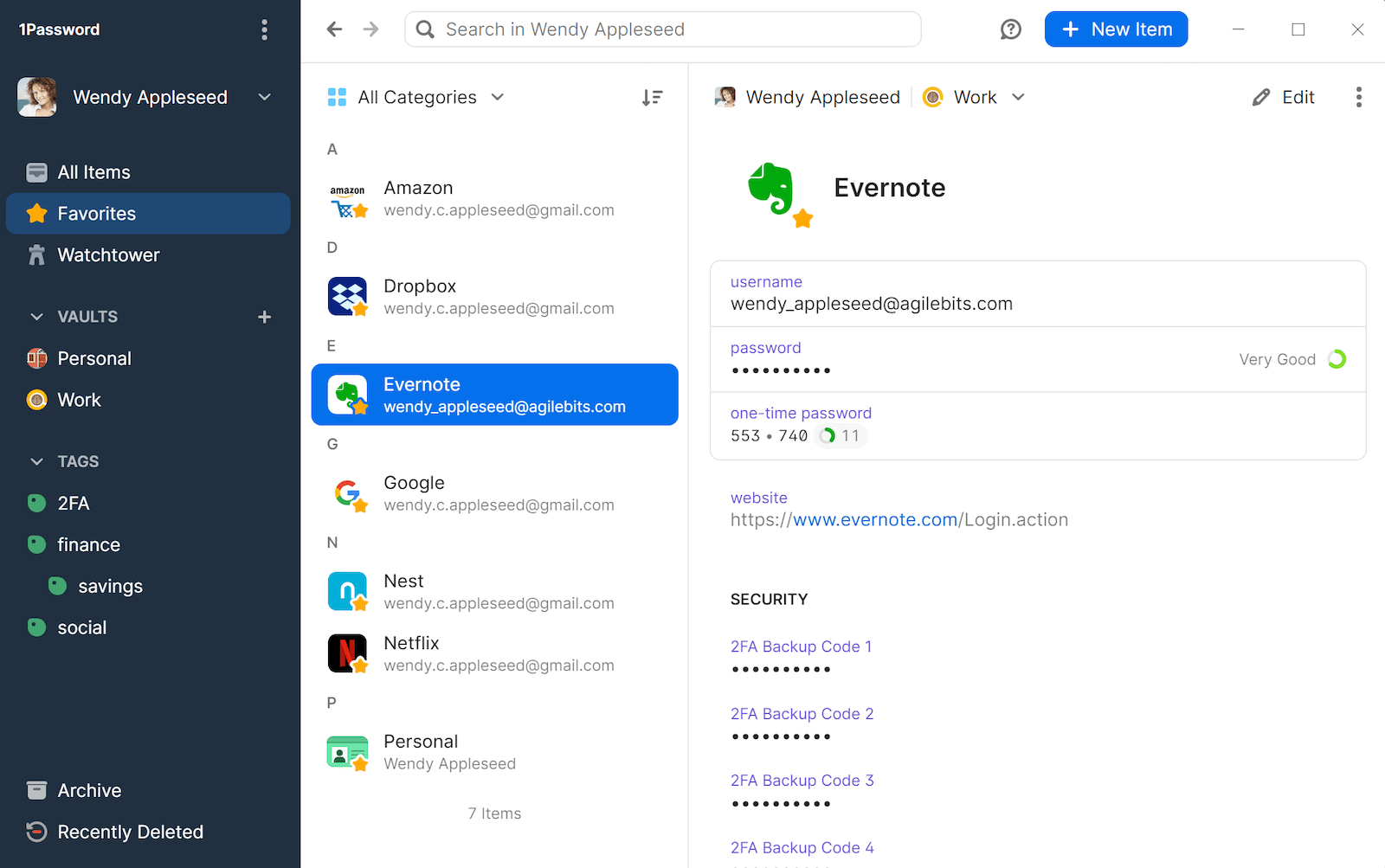


Figure 19: 1Password UI and Desktop Application [[24]](#E18)

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 [[12]](#E6). 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 to other projects, PassGuard provides a free dark mode option for users who prefer reduced screen brightness, offering all features at no charge, and simplifies the account recovery process.

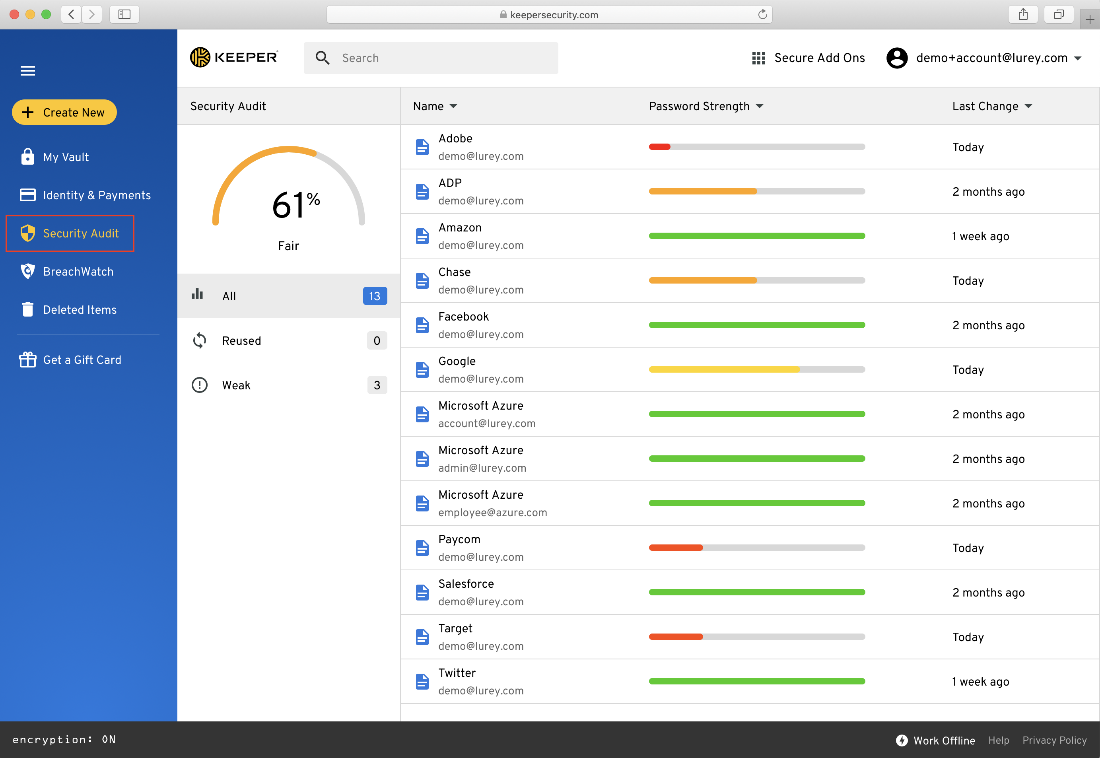


Figure 20: Keeper UI and Desktop Application [[25]](#E19)

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 [[13]](#E7). However, some features are costly, making it relatively expensive, and its wide range of features might be overwhelming. In contrast, PassGuard offers all features for free, ensuring accessibility for all users.

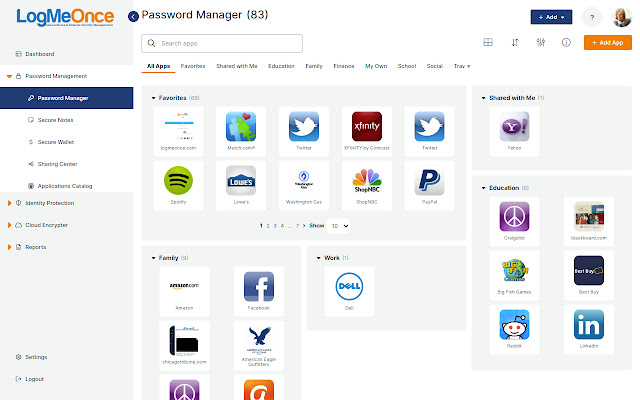


Figure 21: LogMeOnce UI and Desktop Application [[26]](#E20)

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 [[14]](#E8). 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 PassGuard through user-friendly installation and expandable password sharing across devices.

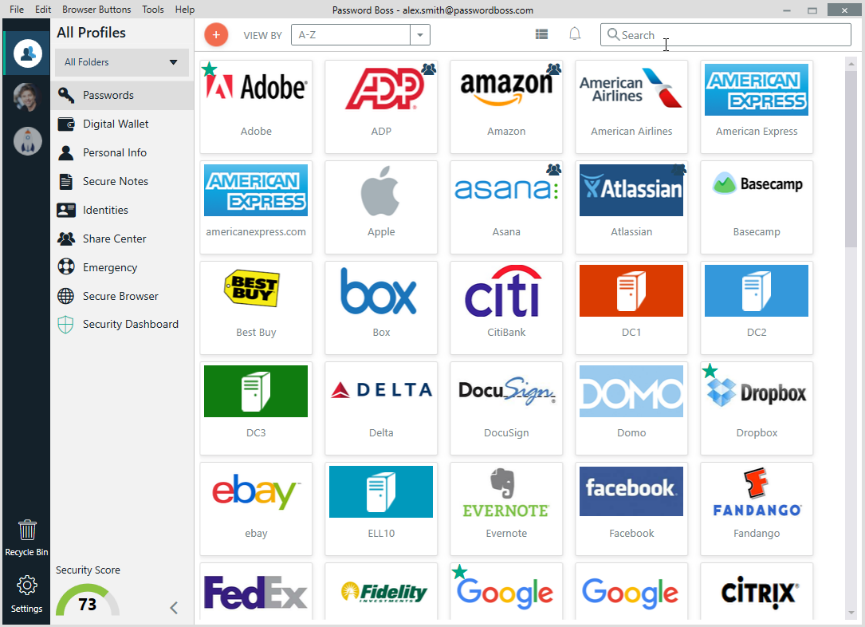


Figure 22: Password Boss UI and Desktop Application [[27]](#E21)

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 [[15]](#E9). One significant distinction between PassGuard and RoboForm is the array of features and functionalities integrated into our program, providing users with a comprehensive experience, and eliminating the need of additional password managers.

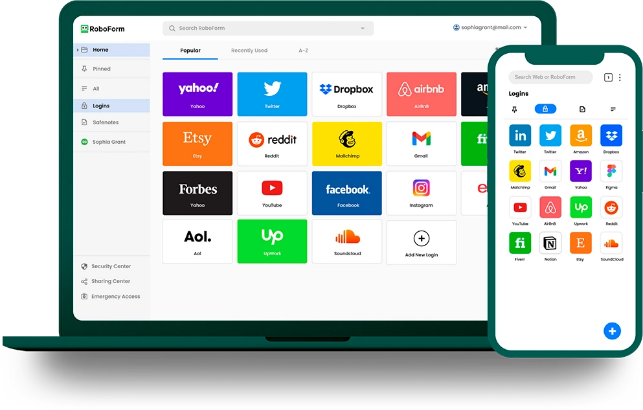


Figure 23: RoboForm UI and Desktop Application [[28]](#E22)

**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 [[16]](#E10). 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 [[17]](#E11). 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 [[18]](#E12). 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 [[19]](#E13).

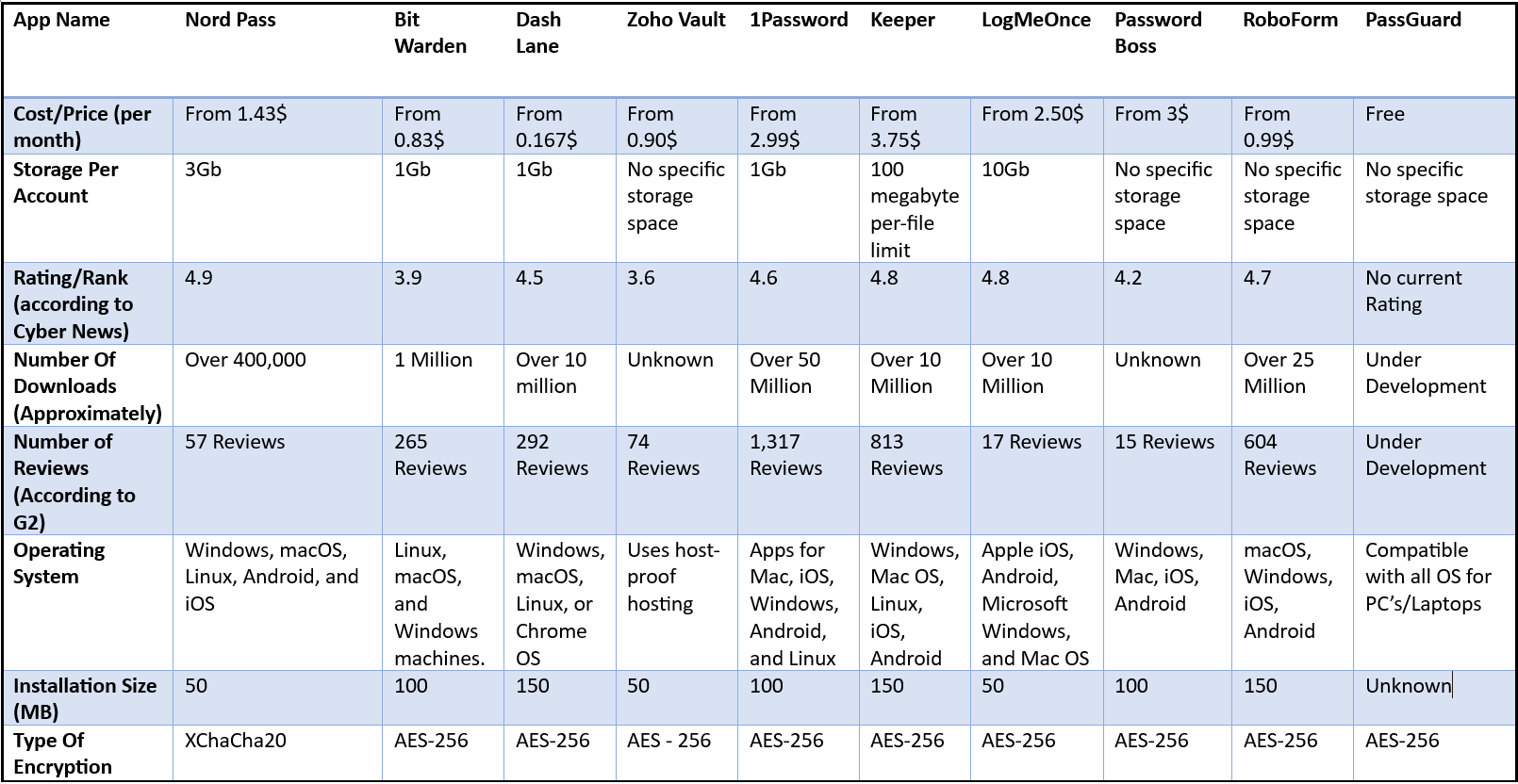


Table 3: Key similarities and differences between out project and the related work

# Requirements analysis

## Software development process

<**ToDo:**

Among the various possible software development processes introduced in your software engineering course, select, and **justify** the most appropriate process that aligns with the specific requirements of your project.

/>

Many software development processes could have been used for our project, such as Waterfall, Scrum, and the V-model. After an in-depth examination of the different possibilities, we chose to apply the iterative-Waterfall model. This software development process is the most convenient for our project, since the project requirements are clear from the beginning, and they are unlikely to change. Most importantly, the security-oriented nature of this project requires an iterative approach that allows us to review and improve the application’s architecture to iteratively treat security vulnerabilities, as security is a major requirement for the application. In addition, our project is a desktop app, so improving the Graphical-user interface (GUI) based on the feedback is a crucial step of PassGuard’s development.

Other models provide iterative characteristics such as spiral or scrum. However, they are not the most suitable because we are not expecting to face continuous changes. Waterfall is better as the project requirements are clear from the beginning. Nevertheless, we got inspired from Agile methodology in the organization of weekly meetings.

The iterative waterfall model consists of different iterations, in each one the team passes through most stages of the traditional waterfall model depending on the iteration objectives. (e.g. Requirement Analysis is just needed in the first iteration). The main steps of the iterations include:

1. **Requirement Analysis:** In this phase, both functional and non-functional requirements are defined based on the application objectives.
2. **Design:** In the design phase, the software is designed using different UML diagrams like class diagram, sequence diagram, and use case diagrams.
3. **Implementation:** In the implementation, requirements are transformed into actual coding.
4. **Testing:** After completing the coding phase, software testing starts using different test methods to make sure that the software works properly.
5. **Deployment:** After completing all the phases, the software is deployed to its work environment.
6. **Review:** After the product is launched, a review phase is performed to check the behavior and validity of the developed product.
7. **Maintenance:** The continuous improvement, resolving bugs, applying few updates in the system...etc.

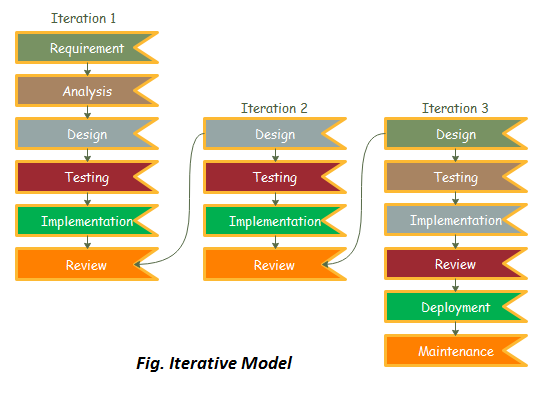


Figure 24: Iterative Model

## Applying the software development process

</**ToDo:**

* Describe and provide evidence on how you effectively **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.
* Utilize in-text citations to refer to relevant sections of the report, providing concrete evidence of the proper application of the chosen software development process. For instance, you can reference specific sections within the report that elaborate on the requirements gathering phase, design phase, testing procedures, or project management strategies.
* 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.

/>

Throughout our senior design project, we followed the enhanced waterfall model, a structured approach that helped us manage our tasks in distinct stages: [Requirement Analysis](#RA) in terms of determining the functional and non-functional requirements, [Design](#DD) where we worked on the detailed design of PassGuard and designed a numerous amount of UML diagrams, [Implementation](#Implementation), [Testing and Review](#Testing) where we initially performed unit testing as part of the functional testing.

Overall, this method provided a clear path forward and allowed us to adjust our plans smoothly whenever needed. Each stage had its own focus, making it easier to handle changes in our project requirements. By thoroughly testing each component before putting them together, we ensured a higher quality outcome for our project. However, we faced a few challenges like strictly respecting milestone deadlines because of the various academic commitments that we had; thankfully, our initial project plan took this into consideration as it was flexible from the beginning to accommodate such delay.

## Functional requirements

< **ToDo:**

* Gather, document, and analyze the project requirements.
* Present the requirement specifications using suitable methods and tools, such as a use case diagram and a use case summary table.
* Produce detailed specifications for each use case and add it to *Appendix A - Use cases specification*.
* You may use flowcharts to describe the sequence of actions and data flow of process(es) required for your project.
* After identifying and describing all the use cases, prioritize them and determine your project's scope by highlighting the use cases you intend to design and implement.

/>

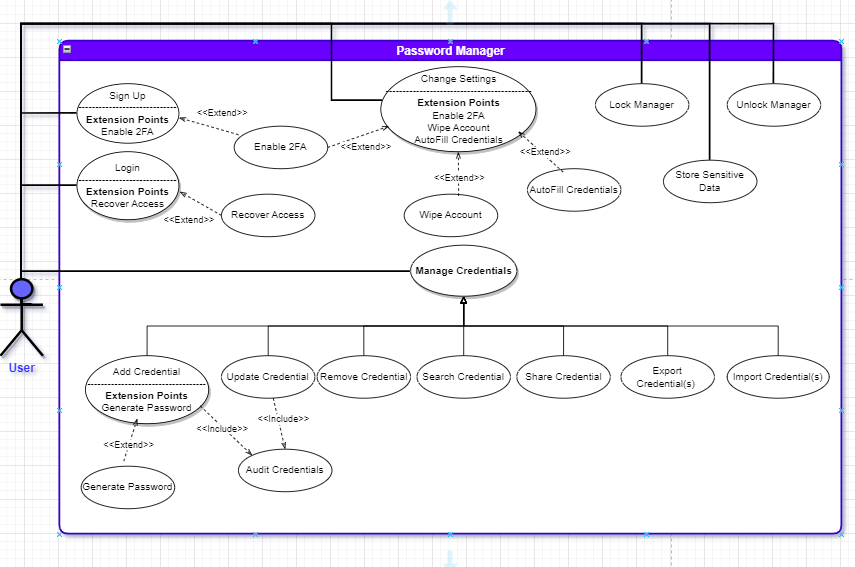


Figure 25: Use Case Diagram

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:

Table : Use Cases Summary

|  |  |  |
| --- | --- | --- |
| **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** | 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

<**ToDo:**

* Identify non-functional requirements (i.e., desired quality attributes) and clearly document them using measurable scenarios, such that you can later test whether they are met (see Section *‎8. Testing and evaluation*).
* Categorize the requirements into different categories such as performance, reliability, security, usability, portability, and maintainability.
* Define precise, measurable targets for each requirement to facilitate future evaluation of their achievement during the testing phase. For instance, when addressing a performance requirement, it is crucial to define explicit metrics, such as maximum response time or throughput. Rather than merely stating a vague aim of ‘fast system’, provide specific measurable target like ‘Registering a new user should complete in under 10 seconds’. Likewise, regarding security requirements, avoid generic statements and specify the types of attacks the system must safeguard against and the threats it must withstand.
* Keep in mind: A non-functional requirement must be specific and measurable to be considered effective and valid.
* Include an evaluation plan for each non-functional requirement to test whether the system is meeting the specified targets. Include details of specific scenarios, tools or methods for evaluating each non-functional requirement.

/>

Table : Evaluation plan for the non-functional requirement

|  |  |  |
| --- | --- | --- |
| Non-functional requirement | Measurable targets | Evaluation Plan |
| 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. |
| 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. |
| 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. |
| 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. |
| 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. |

## Assumptions

<**ToDo:**

* List any project-specific assumptions. These are statements or conditions that are believed to be true or expected to occur during the course of a project but are not guaranteed. They can become potential failure points that need to be monitored and managed.

For instance, in the case of a QU Indoor Navigation app project, one assumption could be the ability of obtaining AutoCAD floor plans for several buildings from QU Facilities Department.

* A bulleted list should be used for this section.

/>

* We are expecting that the user interface to be intuitive enough that users with varying technical proficiency can utilize it without significant issues.
* We aim for the system to maintain responsiveness regardless of the number of credentials stored.
* We expect our system to be designed to have the capability to manage an increased number of users if necessary.
* At the time of development, the cryptographic algorithms and methods employed to safeguard stored passwords are intended to adhere to the most up-to-date security standards and practices.
* We seek to provide users with the option to Export/Import their credentials for use across multiple desktop devices as needed.
* All external components, libraries, and services integrated into the system are expected to be reliable and secure.
* This system is meant to function with popular operating systems such as Windows and macOS, offering appropriate levels of compatibility and functionality.
* Even for academic projects, we strive to ensure our systems comply with conventional cybersecurity practices and guidelines.

## Ethics

<**ToDo:**

* Identify and discuss the legal and ethical considerations and professional responsibilities relevant to your project while taking into account the perspectives of stakeholders such as clients, users, team members, and the community impacted by the project. These ethical aspects can vary widely depending on the nature of the project, such as responsible management of confidential data, data privacy, security, accessibility, respecting intellectual property rights and avoiding plagiarism, and many other pertinent considerations.
* Define project-specific code of ethics by drawing upon relevant resources such as the [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 codes of ethics to effectively address the identified ethical considerations and professional responsibilities throughout the project's lifecycle. Your choices should align with the scope, objectives, and unique challenges of your project.

Do not copy and paste the entire ACM and IEEE codes of ethics into your report. Instead, review them and, in your report, concentrate on discussing those codes that are most applicable to your particular project.

* + Detail how you plan to incorporate and adhere to the selected codes of ethics at various stages of the project’s lifecycle.
  + Demonstrate your critical thinking by considering alternative courses of action and providing well-justified choices based on logical and persuasive reasoning.
  + Conclude your discussions with a table that summarizes the ethical considerations and professional responsibilities relevant to your project.

/>

Ethical issues and professional responsibilities

As we develop our password manager system, we must follow specific rules and stay true to them. We need to fully own our system and be ready for anything that comes from it. It is vital not to do anything that could hurt people or society. Keeping people's information private and being answerable for our actions are top rules we should show clearly. Once we finish building, we must check and test our system properly before letting users have it. Based on trusted groups like ACM and IEEE, we have a list of primary rules and duties for software builders. Think of table 4 as our guiding book for the job.

Table : Project-specific code of ethics and professional responsibilities

|  |  |  |
| --- | --- | --- |
| Sec. No | Code | How to address the ethical issue / professional responsibility during your project |
| 1.1. IEEE | **Responsibility:** Accept full responsibility for their own work | As a software developer we take full ownership of the programs we create, covering every aspect of their development, functionality, and impact. |
| 1.3. ACM | **Avoid harm:** Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy, or harm the environment. The ultimate effect of the work should be to the public good. | We should never release our application to the public without verifying that the program is secure, meets strict security requirements, and follows the appropriate guidelines put in place by experts. |
| 2.1. IEEE | **Being honest:** Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education | We should work within our areas of expertise and be transparent about any restrictions on our experience and training. |
| 2.5. ACM | **Confidentiality:** Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law. | We are responsible for upholding and safeguarding user privacy. It is essential to be able to create systems that safeguard the user's confidential data from unauthorized access. |
| 3.8. IEEE | **Accountability:** Ensure that specifications for software on which they work have been well documented, satisfy the users requirements and have the appropriate approvals | We are responsible for ensuring that software specifications are documented in a manner that satisfies user needs and adheres to established protocols. |
| 3.10. IEEE | **Ensure debugging and testing:** Ensure adequate testing, debugging, and review of software and related documents on which they work. | We should thoroughly test, troubleshoot, and evaluate the software and related documentation before releasing the program. |
| 3.14. IEEE | **Integrity of data:** Maintain the integrity of data, being sensitive to outdated or flawed occurrences. | We should maintain the integrity of data and pay close attention to the detection and correction of obsolete or incorrect data to ensure the integrity of software systems. |

# Project Plan

## Project milestones

< **ToDo:**

* Create a table that outlines and describes the project key milestones. 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.

/>

In this section, the project planning will be laid out in detail providing brief information about the different milestones, their deliverable in Table 5 and assignment of each task in section 4.2. It was done on tiers-based approach, where in each iteration of waterfall explained in section 3.1, we aim to accomplish one tier as shown in the figure below.

Table : Implementation Tiers

|  |  |
| --- | --- |
| Tiers | Features |
| Tier 1 (Working but not secure) | * Saving passwords in file or local DB (encrypted with key saved in a file) * Health of password (including past leaked, passwords comparison) * Generation of password * Good GUI * Searching, adding, modifying, deleting passwords |
| Tier 2 (Working with basic security) | * (Tier 1) * Hashing and Encryption * More secure app architecture |
| Tier 3 (Better User Experience with basic security) | * (Tier 2) * Export/Import of credentials * Account recovery |
| Tier 4 (Enhance security) | * (Tier 3) * MFA |
| Tier 5 (Future work) | * (Tier 4) * Lock app & document access |

Table : Project milestones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Milestone | Description | Deliverables | Duration (Days) |
| M1 | **Project foundation and Design** | * Choosing the project idea, brainstorming ideas, and doing the literature review, as well as drawing up the project plan. * Finishing Chapters 1 to 5 in the report | * Project plan * Project design | 35 days |
| M2 | **Implementation** | Implementing around 30-40% of the project, completing Tier 1. | Tier 1 implemented | 21 days |
| M3 | **Testing and feedback (1)** | Testing the functionalities implemented in Milestone 2 and record feedback (in appendix) | Testing and feedback report | 7 days |
| M4 | **Applying feedback** | Applying all comments in the feedback report | Implementation updated | 7 days |
| M5 | **Last checkup** | Finalization and update of the report | Report updated and submitted for supervisor’s review | 7 days |
| M6 | **Review and improvement (SDP1)** | Recording supervisor’s comments and update the report based on it for final submission (SDP1), and finalization of the presentation | Final version of report and presentation submitted for SDP1 | 7 days |
| Iteration (2) | | | | |
| M7 | **Design Review (1)** | Review of the design based on tier 2 | Design reviewed | 7 days |
| M8 | **Implementation improvement (1)** | Implementation of tier 2 | Tier 2 implemented | 14 days |
| M9 | **Testing and review (2)** | Testing and review, including functional and non-functional requirements | Implementation reviewed | 7 days |
| Iteration (3) | | | | |
| M10 | **Design Review (2)** | Review of the design based on tier 3 and 4 | Design reviewed | 7 days |
| M11 | **Implementation improvement (2)** | Implementation of tier 3 and 4 | Tier 3 and 4 implemented | 14 days |
| M12 | **Testing and review (3)** | Testing and review including users’ feedback | Implementation reviewed | 7 days |
| M13 | **Finalization of deliverables** | Finalization of all deliverables for SDP2 discussion.  Note that M13 will be passively incremental during the Spring 2024 semester. | All deliverable submitted | 14 days |

## Project timeline

< **ToDo:**

* The project timeline defines **who** will do **what** and **when** to ensure efficient project execution and progress tracking. You need to:
* Develop a detailed project timeline that breaks down each milestone into smaller, manageable tasks.
* Estimate the time required to complete each task.
* Assign the tasks to individual team members and define a work schedule stating the start and end date of each task. You can use Microsoft Project or Excel to create your project timeline. Insert a [Gantt chart](https://en.wikipedia.org/wiki/Gantt_chart) and a project timeline table in this section.

Additionally, keep in mind the overall project duration, which typically spans around 14 weeks for each semester. It is vital to decide the roles and responsibilities of each team member to avoid any ambiguity.

* Be sure to include time to document your work at each milestone to incrementally produce your final report. Additionally, schedule time for project evaluation, reflection on the experience gained, and the finalization and review of both the report and presentation.

/>

Note that the planning timeline is 12 weeks per semester (excluding holidays) to give the team flexibility to encounter any potential unexpected risk as mentioned in section 4.3.

Table : Project timeline SDP 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Task name | Start Date | End Date | Duration (Days) | Assigned To | Dependencies |
| 1.1 | **Brainstorming ideas** | Sept 01, 2023 | Sept 07, 2023 | 7 days | All team members |  |
| 1.2 | **Literature review** | Sept 01, 2023 | Sept 07, 2023 | 7 days | Youssef |  |
| 1.3 | **Project description** | Sept 06, 2023 | Sept 06, 2023 | 1 day | All team members |  |
| 1.4 | **Multi-criteria analysis** | Sept 06, 2023 | Sept 06, 2023 | 1 days | All team members |  |
| 1.5 | **Report: Ch.1** | Sept 08, 2023 | Sept 10, 2023 | 3 days | Youssef  Mohamed-Dhia | 1.2, 1.1, 1.3 |
| 1.6 | **Report: Ch.2** | Sept 11, 2023 | Sept 12, 2023 | 2 days | Khalifa  Essa | 1.5 |
| 2.1 | **Use cases** | Sept 12, 2023 | Sept 15, 2023 | 4 days | All team members | 1.6 |
| 2.2 | **Use case specifications** | Sept 18, 2023 | Sept 25, 2023 | 8 days | All team members | 2.1 |
| 2.3 | **Report: Ch.3** | Sept 15, 2023 | Sept 18, 2023 | 4 days | All team members |  |
| 2.4 | **Report: Ch.4** | Sept 19, 2023 | Sept 21, 2023 | 3 days | Mohamed-Dhia |  |
| 2.5 | **Report: Ch.5** | Sept 22, 2023 | Sept 25, 2023 | 4 days | Youssef  Khalifa |  |
|  |  |  |  |  |  |  |
| 2.6 | **Structural Model** | Sept 26, 2023 | Sept 27, 2023 | 2 days | Youssef  Essa |  |
| 2.7 | **Sequence diagrams** | Sept 28, 2023 | Oct 04, 2023 | 7 days | All team members |  |
| 2.8 | **Database design model** | Oct 01, 2023 | Oct 04, 2023 | 4 days | Essa  Khalifa |  |
| 3.1 | **Interface implementation** | Oct 05, 2023 | Oct 18, 2023 | 14 days | Youssef  Mohamed-Dhia |  |
| 3.2 | **Database implementation** | Oct 05, 2023 | Oct 18, 2023 | 14 days | Khalifa  Essa | 2.8 |
| 3.3 | **DB - Interface Connection** | Oct 19, 2023 | Oct 26, 2023 | 8 days | Khalifa  Essa | 3.1, 3.2 |
| 4.1 | **Testing and feedback** | Oct 27, 2023 | Nov 01, 2023 | 6 days | Essa  Khalifa | 3.3 |
| 4.2 | **Applying feedback** | Nov 02, 2023 | Nov 09, 2023 | 8 days | Khalifa | 4.1 |
| 5.1 | **Report: Ch. 8** | Nov 10, 2023 | Nov 12, 2023 | 3 days | Essa |  |
| 5.2 | **Report: Ch. 9,10, and 11** | Nov 13, 2023 | Nov 14, 2023 | 2 days | All team members |  |
| 5.3 | **Review of report for first submission** | Nov 15, 2023 | Nov 29, 2023 | 15 days | All team members | 4.1, 5.2, 5.1 |
| 6 | **Review of report for final submission** | Nov 28, 2023 | Dec 01, 2023 | 4 days | All team members | 5.3 |

The following table is the planned timeline for SDP2 that the team aims to follow.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Task name | Start Date | End Date | Duration (Days) | Assigned To | Dependencies |
| 7 | **Design Review (1): improve security of app architecture** | Jan 29, 2024 | Feb 04, 2024 | 7 days | All team members |  |
| 8.1 | **Implementation of remaining interfaces** | Feb 05, 2024 | Feb 19, 2024 | 14 days | All team members | 7 |
| 8.2 | **Applying security improvement in Implementation** | Feb 05, 2024 | Feb 19, 2024 | 14 days | All team members | 7 |
| 9 | **Testing and Review (2)** | Feb 20, 2024 | Feb 27, 2024 | 7 days | All team members | 8.2 |
| 10 | **Design Review (2)** | Feb 28, 2024 | Mar 03, 2024 | 7 days | All team members | 9 |
| 11.1 | **Add the account recovery functionality** | Mar 04, 2024 | Mar 25, 2024 | 21 days | Youssef  Mohamed-Dhia | 10 |
| 11.2 | **Add the import/export functionality \*** | Mar 04, 2024 | Mar 25, 2024 | 21 days | Essa  Khalifa | 10 |
| 12 | **Testing and Review (3)** | Mar 26, 2024 | Apr 02, 2024 | 7 days | All team members | 11.1, 11.2 |
| 13.1 | **Preparing poster** | Apr 03, 2024 | Apr 17, 2024 | 14 days | Youssef  Khalifa | 12 |
| 13.2 | **Preparing other deliverables** | Apr 03, 2024 | Apr 17, 2024 | 14 days | Essa  Mohamed-Dhia | 12 |
| 13.3 | **Review of deliverables for final submission (SDP2)** | Apr 18, 2024 | May 16, 2024 | 28 days | All team members | 13.1, 13.2 |

Below are the Gantt charts that summarizes the above tables for SDP-1 and SDP-2:

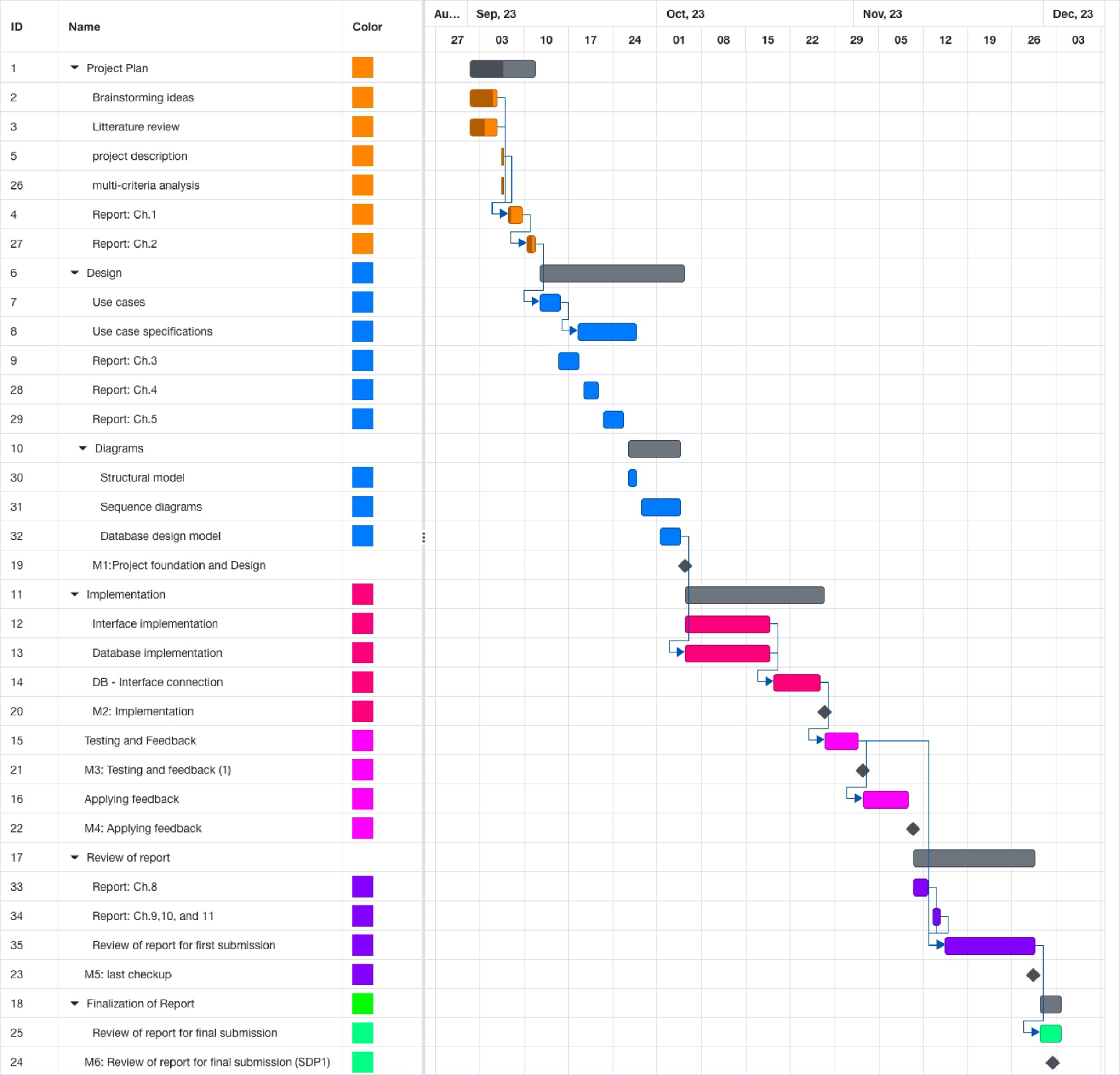


Figure 26: SDP 1 Gantt Chart.



Figure 27: SDP 2 Gantt Chart.

## Anticipated risks

< **ToDo:**

* Present a table that outlines potential risks, which are events that could jeopardize the successful completion of the project.
* Outline the approach to mitigate each identified risk, ensuring the project's smooth progression and minimizing disruptions.

/>

As a team we should be expecting potential risks and setbacks that might occur during the project’s lifecycle. Even though it is difficult to accurately predict what might jeopardize our progress, we have outlined some events in the below table that we might have to deal with.

Table : Project Risks

|  |  |
| --- | --- |
| **Risk** | **Approach to mitigate the risk** |
| **Unexpected Personal events** | Keeping up with each other’s tasks and assist each other when any of us is in need. |
| **Academic Overload** | Stick with the project milestones and work on time management. |
| **Time Contraints** | Proper planning and task distribution  Weekly meetings with supervisor. |
| **Application Deployment** | Early testing and building a prototype for debugging. |
| **Unexpected Code Errors** | Proper research and locating the source of the error effectively |

# System Architecture

## High level system architecture

< **ToDo:**

* Present a brief overview of the solution and the system's architecture, allowing stakeholders to understand its key components and interactions.
* Include a high-level system architecture diagram of the proposed solution that serves as a visual guide for the detailed design. The diagram should show how your solution is decomposed and organized into components. Use labeled boxes to represent components and arrows to show how they interact. Indicate external systems or services that the system may interface with.
* Describe the key responsibilities and functions of each component of your high-level architecture. Explain the interfaces they expose or utilize.
* Discuss the key interactions between the identified components, encompassing aspects like data flow, communication protocols, and dependencies between components. Use diagrams, flowcharts, or textual descriptions to clarify these interactions.
* Discuss the architectural style (e.g., Layered, MVC, SPA) used by your solution. Justify the choice of this architectural style, clarifying how it aligns with the project's requirements.

/>

We have chosen the **Model-View-Controller** (**MVC**) architectural pattern for our password manager project due to its ability to provide a structured and organized approach to software development. **MVC** promotes a clear separation of concerns, allowing us to modularly deal with the core components of our application. The **Model** is responsible for managing data and business logic ensuring that our users' sensitive information remains secure. **View** handles therendering of the user interface, offers flexibility and ease of modification in response to changing design requirements. Finally, the **Controller** acts as a bridge between the **Model** and the **View**, facilitating smooth interactions and user experience. By adopting **MVC**, we not only enhance the maintainability and scalability of our application but also ensure that we can implement features like real-time password strength updates and robust security measures efficiently and coherently throughout the project.

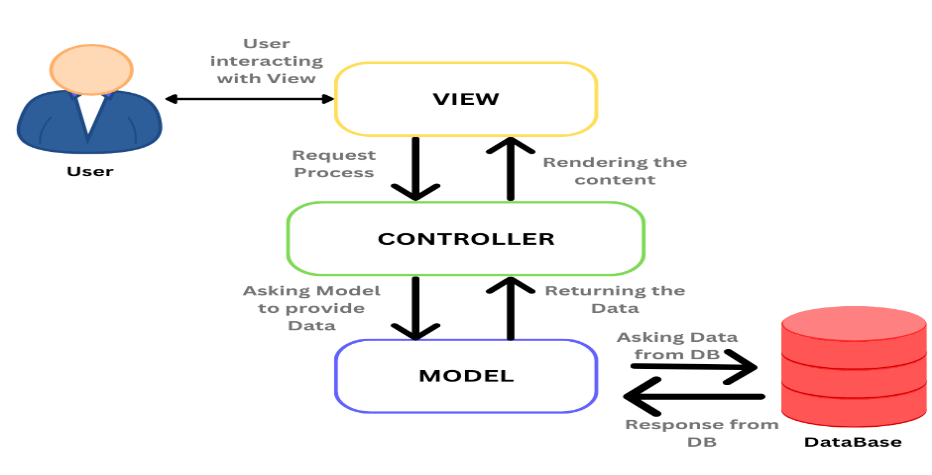


Figure 28: MVC High-level architecture

When a user interacts with the application, the **Controller** captures requests through the **View**. The **Controller** thencommunicates with the **Model** to perform operations like encryption/decryption, password management, strength calculation and DB querying. The **Controller** updates the **View** to reflect those changes.

## Alternative solutions and tradeoffs

< **ToDo:**

* Discuss alternative solutions and architectural styles that you have considered to deliver the project requirements. Analyze their respective tradeoffs and merits.
* Provide sufficient justifications for selecting the preferred solution architecture and design. Clearly articulate why the chosen approach is superior to the alternative solutions, highlighting its advantages and benefits.

After extensive discussion within our team regarding the most suitable architectural style that aligns with our expectations for PassGuard, our research has pointed out that MVC (Model-View-Controller) and layered architectural styles are the two most suitable ones. Even though layered architecture ensures maintainability and ease of updates as each layer can scale independently, managing interactions among the different layers is complex and affects overall app performance due to the overhead of data transportation. In addition, its rigid structure may lead to cascading changes as alterations in one layer often necessitate adjustments in interconnected layers, thus limiting flexibility.

With these tradeoffs in mind, our team opted for the MVC (Model-View-Controller) architectural style due to its streamline modular structure and its best merit of allowing us as a team to concurrently work on different parts of the application with barely any cascading. Moreover, the testability MVC provides with the architecture’s inherent design allows for easier and more effective unit testing of individual components which aligns perfectly with our commitment to delivering a robust and dependable management system. With all that in mind, the flexibility provided by MVC is crucial for our application’s future enhancements and modifications enabling smooth adjustments and addition of features when needed.

In addition to the architectural styles, we have also highlighted some of the alternative software tools and frameworks that were considered for PassGuard’s development alongside a brief description of what they offer and their tradeoffs. ([Refer to section 7.1 for the chosen software and frameworks and their advantages](#SW)).

Table : Alternative Tools & Solutions

|  |  |
| --- | --- |
| Alternative Tools  & Solutions | Advantages & Tradeoffs |
| Nw.js | * NW.js facilitates the creation of native desktop applications utilizing web technologies, like Electron, but with a greater degree of direct control over Chromium browser. * It offers a higher degree of compatibility with Node.js, as well as greater control over the app window. * Despite its advantages, the development community of NW.js is smaller than that of Electron, and the documentation is less comprehensive, making it difficult to find solutions and community support. |
| AngularJS | * Angular is a powerful framework for web application development that is supported by Google and is renowned for its robust features and comprehensive solutions. * It offers a wide range of features from the outset, such as dependency injection, template support, and reactive programming via RxJS. * However, it also has a steeper learning curve and may be overly complex for smaller projects, while its opinionated nature may limit its flexibility in certain areas. |
| Webpack | * Webpack is a powerful JavaScript module bundler that can be configured in a variety of ways, making it one of the most widely used web development tools. * It is well-suited to complex applications due to its extensive plugin ecosystem and its configurable nature. * However, it can be challenging to configure, particularly for those who are new to web development. Additionally, the development server is typically slower than Vite. |
| Sequelize | * Sequelize is an advanced Object-Oriented Programming Language (ORM) for the Node.js programming language. * It supports a variety of SQL dialects, provides robust transaction support, supports associations, supports both fast and slow loading, and offers a range of other features. Due to its maturity and its widespread use, Sequelize has access to many resources and a supportive community. * However, it may be more verbose and difficult to set up and use than Prisma. |
| PostgreSQL | * PostgreSQL is an open-source, high-performance relational database with a wide range of features. * It is suitable for a variety of applications, from single-user workloads to large-scale data warehouses. PostgreSQL offers several advantages over SQLite, such as enhanced indexing capabilities, comprehensive search capabilities, and the ability to support complex queries and relations. * However, it also requires more configuration and maintenance than SQLite, making it an ideal choice for larger projects. |
| Bootstrap | * Bootstrap is a popular CSS framework that provides a selection of pre-designed elements and responsive design capabilities. * It is suitable for rapid application prototyping and requires minimal CSS knowledge to begin with. * However, it may result in "bootstrap-like" designs without the ability to customize them, and offers limited flexibility compared to utility-oriented frameworks such as Tailwind CSS. |
| JavaScript | * JavaScript is one of the most widely used languages in the web development world. It is a dynamic language that is interpreted and used in both the front-end and back-end of web applications. * It offers several advantages, such as ease of use and a large ecosystem. Additionally, there is no compilation step required, resulting in a simplified setup. * However, it does not have the same static typing capabilities as TypeScript, which may result in increased runtime errors and less code self-documentation in larger applications. |

# Detailed Design

<**ToDo:**

* During semester 1, you will design, implement, and test **25% to 30% of the essential use cases** to deliver the core and most importantfeatures of your solution. In semester 2, you will complete the remaining use cases.
* 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. The detailed design serves as a reference and actionable guidance for implementing the solution.
  + Use suitable methods and tools such as class diagrams, sequence diagrams, activity diagrams and state diagrams.
* Note that for every design aspect you will present in the sections below, you must:
* **Identify and evaluate possible alternative solutions** (i.e., design choices) and analyze their respective tradeoffs.
* Provide **detailed justifications** on why you have preferred and selected a particular solution or design choice over others. Highlighthow your design choicesenabled you to achieve the desired system quality attributes documented in *Section ‎3.2 Non-Functional Requirements*. Additionally, you should highlight the novel aspects of your design.

Further justify your design choices by discussing relevant **design principles** that influenced your system design. For example, highlight how your design complies with key software design principles such as separation of concerns, abstraction, modularity, high cohesion, and low coupling.

/>

## Structural model

<**ToDo:**

* **You may break this section into subsections** to clearly document the detailed structural model of each major component or use case.
* The structure of your software components could be documented using:
* Class diagrams to model the structure and relationships among classes, aiding in both conceptual understanding and practical implementation. If the model is too big, partition the diagram. For instance, you may present entity classes, repository classes, and service classes as separate diagrams.
  + - * For each class, provide specifications for attributes (names and data types) and methods (parameter names, types, and return types).
      * Clearly depict inheritance, composition, or aggregation relationships between classes. Define the association name and specify multiplicities at both ends.
      * Accompany each diagram with a succinct explanation. Include a few paragraphs discussing significant design decisions, such as how inheritance and polymorphism were employed to enhance system design. Additionally, briefly discuss how design principles such as the ‘Information Expert’ and ‘Whole controls the Parts’ principles were applied to your design.
* Design the interfaces exposed by external services that the system interacts with (e.g., BankingService, MailService). Specify input parameters, return types, and exceptions for each of these service interfaces.

/>

PassGuard is a React based app, and its structure could be visualized with a component Hierarchy or what is also called a UI structural tree which is very crucial in understanding how data flows through our React app, where components play a role like classes in OOP. Below is the visual representation of what has been implemented thus far in PassGuard.

React Component Hierarchy

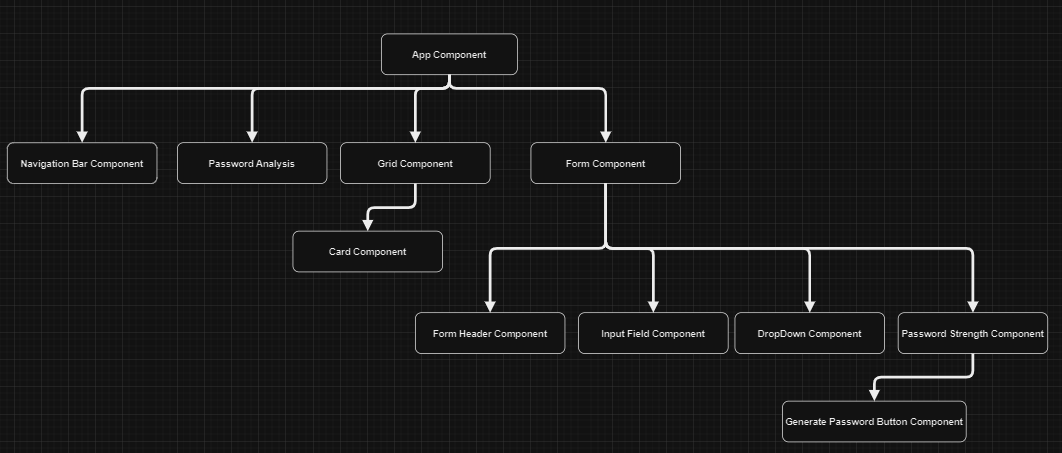
The below tree is made up of nodes, where each node represents a component. The root node (App component) is the first node that React renders and it is the parent node. The tree then goes down rendering the child components where each arrow in the tree represents a parent-child relationship and further renders takes place till we reach the leaf nodes.

Figure 29: React Component Hierarchy

Repository Class Diagram

The below repository class diagram shows the classes / services that interact with the database, we decided to divide the operations to (Create, Update, Delete) in the UserManagementService, and the (Read / Retrieve) operations in the UserQueryService. And by using these services in components or utility classes, it will ease the process of communication with the database.

A screenshot of a computer program

Description automatically generated

Figure 30: Repository Class Diagram

## Behavioral model

<**ToDo:**

* **You may break this section into subsections** to clearly document the detailed behavioral model of each major component or use case. The behavioral model specifies the sequence of interactions between objects or components to depict how specific use cases are executed within the system.
* The behavioral aspects of your software components could be documented the following techniques:
* **Activity diagrams** to model the flow of activities and control logic within the system or a complex use case. They are particularly well-suited for modeling complex processes, workflows, or business logic. They capture conditional branching, parallel processing, and decision points.
* **Design Sequence Diagrams (DSD)** to document interactions and communication between objects or components over time to realize a specific use case. They emphasize the timing and dependencies of interactions. Ensure that method calls in the DSD use methods from the class diagram and must show the parameters and the return type of each method.
* **State diagrams** to represent the different states and transitions of important objects or components within the system. Describe the conditions triggering state transitions
* Present detailed algorithms for important functions within the software. Include pseudocode, or flowcharts to explain the logic.

/>

A close-up of a blueprint

Description automatically generated

Figure 31: Sign Up Design Sequence Diagram

A diagram of a recovery access

Description automatically generated

Figure 32: Login Design Sequence Diagram

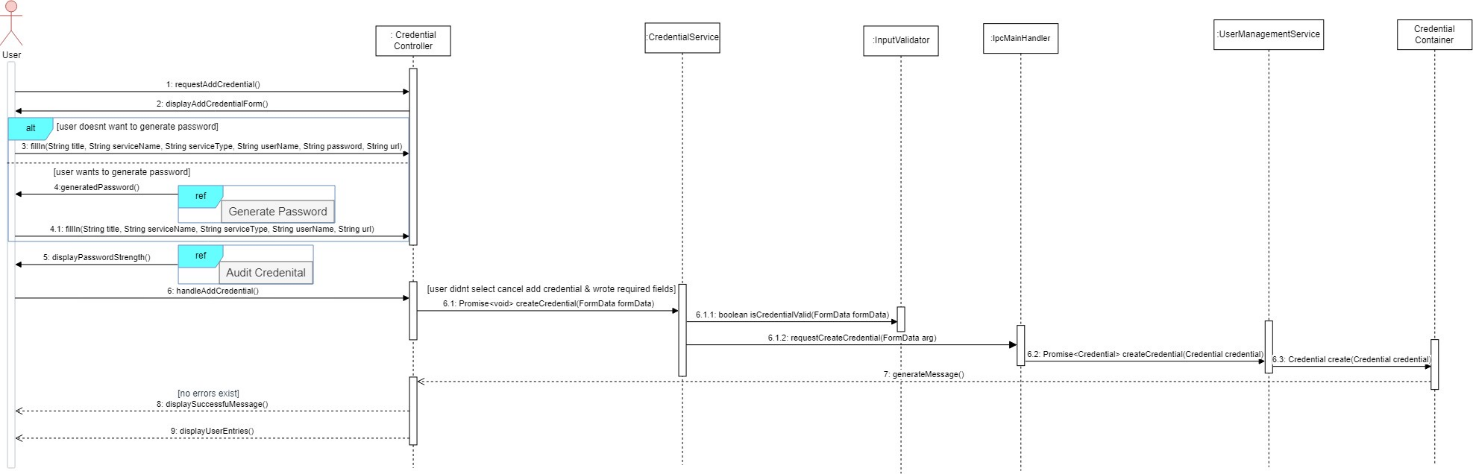


Figure 33: Add Credential Design Sequence Diagram

A close-up of a diagram

Description automatically generated

Figure 34: Update Design Sequence Diagram

A diagram of a diagram

Description automatically generated with medium confidence

Figure 35: Remove Credential Design Sequence Diagram

A diagram of a diagram

Description automatically generated with medium confidence

Figure 36: Autofill Design Sequence Diagram

A diagram with text on it

Description automatically generated

Figure 37: Search Credential Design Sequence Diagram

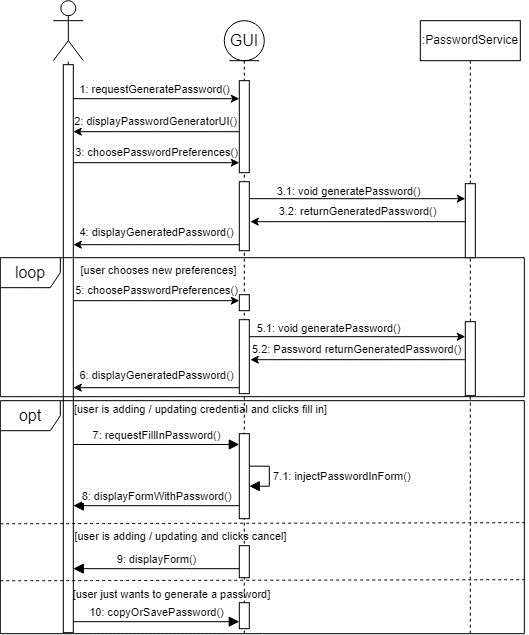


Figure 38: Generate Password Design Sequence Diagram

A screenshot of a computer

Description automatically generated

Figure 39: Export Design Sequence Diagram

A diagram of a company

Description automatically generated

Figure 40: Import Design Sequence Diagram

A diagram of a project

Description automatically generated with medium confidence

Figure 41: Recovery Access Design Sequence Diagram

## Database design

<**ToDo:**

* Include a logical data model describing the major entities, their relationships, associated attributes, and respective data types. Ensure 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 generating separate diagrams for each component or subsystem.

/>

Our database design model provides a thorough illustration of PassGuard system architecture, and an explanation of each relation is given below along with the meaning of the relationships:

* **User:** This entity stores all the information related to the user especially for getting authentication to PassGuard, where each user will have a unique id and his PII will be stored in the attribute data which is a JSON string. And the relationships this entity has are: (**One User** has **Many Credentials**), and (**One User** has **Many Security Questions**).
* **Credential:** This entity holds the information about the credential that the user has, where each credential will have a unique id, and each credential belongs to a specific user. It is considered one of the most crucial entities due to it storing a lot of important information. And the relationships this entity has are: (**One Credential** belongs to **One Trash**), and (**Many Credentials** belong to **One User**).
* **Security Question:** The main purpose of this entity is to store the security questions of the users, where each security question will have an answer to it, and by this entity we can ensure that the user is answering the questions (when needed) correctly. And the relationship this entity has is: (**Many Security Questions** belong to **One User**) because one security question doesn’t ensure fully the identity of the user.
* **Trash:** Used to store the credentials that get deleted, the purpose of this entity is to recover accidentally deleted credentials and retain them for a certain period for compliance or audit purposes. And the relationship this entity has is: (**One Trash** belongs to **One Credential**).

A screenshot of a computer

Description automatically generated

Figure 42: Database Design Model

## User interface design

<**ToDo:**

* Discuss the target users and highlight relevant characteristics that influenced the design decisions.
* Describe the overall layout and visual design of the user interface, including aspects such as 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 navigation aids or controls for moving between screens to achieve a particular use case.
* 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. />

PassGuard is designed for a wide range of users, from tech-savvy individuals to others who just want to make managing their credentials easier. The goal of the design choices was to create an interface that is user-friendly and appealing to all users. While the sleek, modern style appeals to younger audiences and those who value simplicity of use, the user-friendly functionality serves those who value ease of use above all else. In addition to trying to appeal to a wide range of user tastes, the combination of a visually appealing, modern style and intuitive usage enables accessibility and usefulness for a wide range of people negotiating the difficulties of digital security.

**Color Scheme**

The color scheme of PassGuard is modern and appealing. The main background color is neutral white, which is easy on the eyes. The text is black, making it easily readable. The buttons include a mixture of our secondary light blue color for (confirming) and black for (cancelling). While user interaction effects use our primary color, yellow. Our Accent color is used briefly throughout our app.



Figure 43: Color Scheme

**Typography**

We have carefully selected font pairs that ensure compatibility, readability, and visual harmony within our design. Roboto are Nunito are well known fonts that go well together across different screens in addition to creating a balanced and inviting visual contrast.

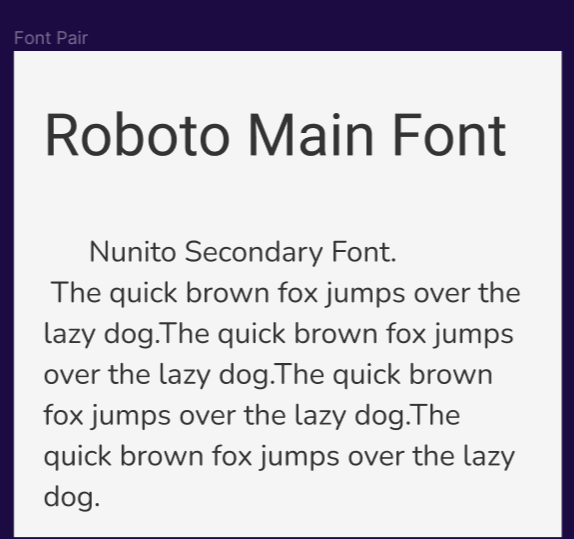


Figure 44: Typography

**Conceptual Dashboard Design Done with Figma**

Careful thought was taken when creating the main app dashboard to ensure that it encompassed a wide range of crucial use cases for most users. Prioritizing user-centricity, the design placed frequently used features in easy-to-reach locations to maximize productivity. The dashboard was streamlined to reduce pointless navigation and provide a more user-friendly and smoother interface. This method allows users to interact with the app's main features with ease and maximizes user satisfaction by adhering to the principle of simplicity.

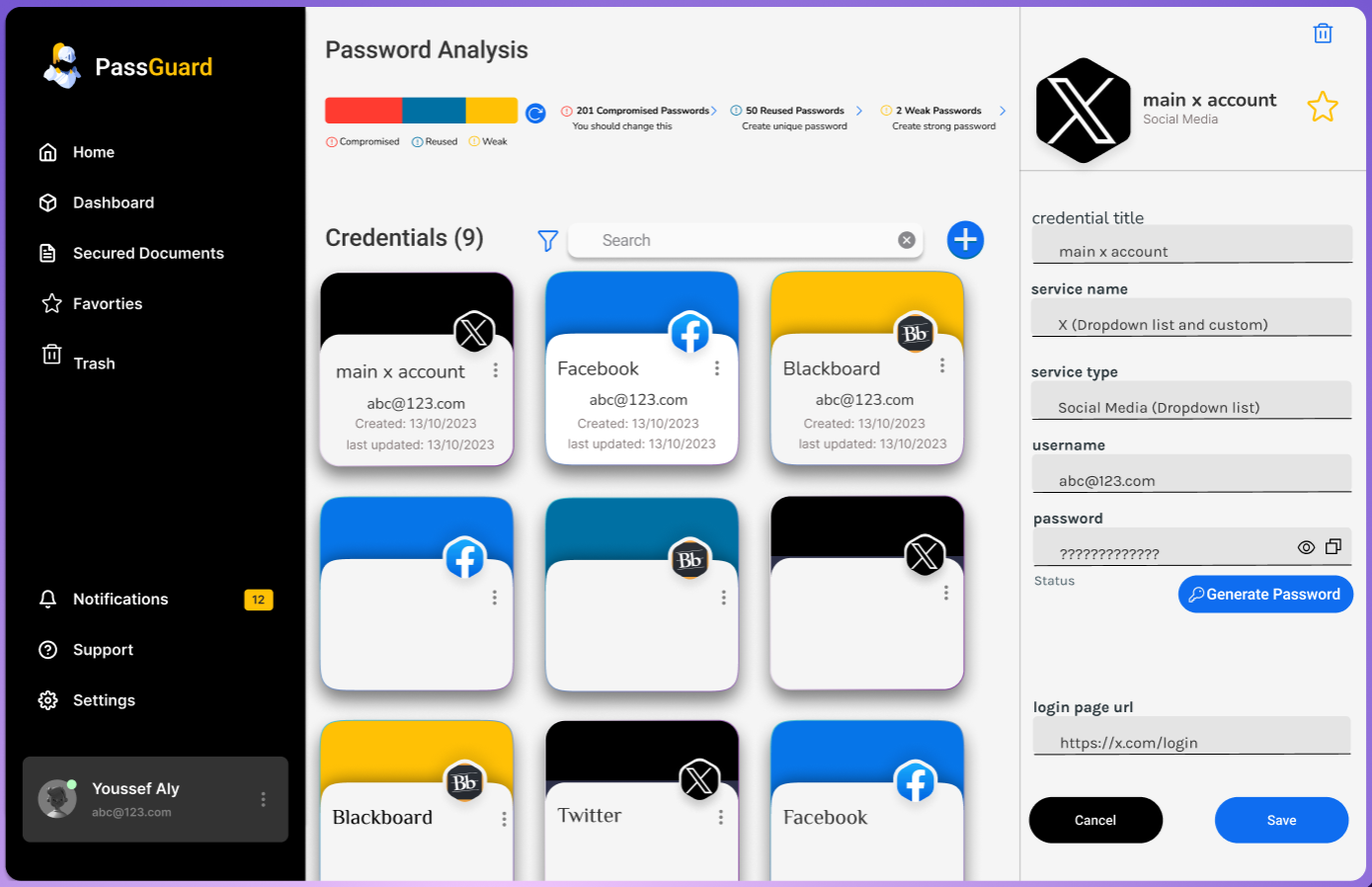


Figure 45: Conceptual Dashboard with Figma

## 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 has been applied to your design.
* Evaluate the impact of selected pattern on your system quality attributes. Highlight the benefits and potential problems or limitations introduced by the selected pattern.
* This subsection is extremely important and required. />

**Model-View-View Model (MVVM):**

The MVVM is a design pattern that provides a clear separation of concerns, making it easier to develop, test, and maintain software applications. What MVVM does is to separate an application into Views, View Models and Models.

**-Views** are responsible for presenting data to the user and capturing user input, which includes the UI and the Components, text, and animations.

**-View Models** act as a mediator between the Views and the Models, it is where the controls for interacting with View are held, and the elements of the View are bound with the controls in View Model.

**-Models** manage the application data and have the required methods to interact with the database.

MVVM was chosen for several reasons (Advantages).

* This design pattern is mainly used for **Desktop applications** with data binding capabilities which was beneficial for us, and it is used a lot with projects using React.
* Easy to **reuse** components, due to them having the same functionalities but maybe for example different labels.
* Business logic is **decoupled** (modular) from UI, where UI is for rendering the Components and Logic is in another file for handling user inputs, this enhances the maintainability of the app.
* It provides an essential structure for **collaborative development** among multiple developers.
* MVVM also simplifies the development and **maintenance** in large Desktop Applications.
* **Performance** wise it minimizes the expensive calculations by preventing unnecessary renders, which improves the overall app performance and user experience.

MVVM also has potential problems / limitations (Disadvantages).

* Sometimes the different components and codes might reach a point where they are not maintained or get a lot and confusing (**having multiple components results in confusion**).
* It is also thought that MVVM is an **overkill** for simple UI’s.
* The View Model can sometimes be **hard to design** where managing it might get frustrating.
* Data binding could affect the **debugging** which would be a bit difficult for detecting errors.

We had a main file called App which is the starting point of the application, where it calls all the other needed components to render them on the **View** side. The **View Model** is an essential part where the data is received from the View side and handled. For example, deconstructing the form data to make out of it a Credential object to be sent to the Model. Our **Model** has the schema of the database and the repository where the CRUD operations are done on the data and stored in the DB.

A diagram of a diagram of a schema

Description automatically generated

Figure 46: MVVM Design Pattern

**Conditional Rendering Design Pattern in React:**

With React, conditional rendering is a popular design technique that lets you render various elements or components according to specific criteria. It's an effective method that lets you design responsive and dynamic user interfaces, therefore it responds to user interactions and data changes. We decided to use it due to the complexity of the logic, reusability, and granularity of control.

In terms of Design Implementation, it has a lot of approaches, but the approach we used was using **Ternary Operators**, **where** it provides a compact way to render one or two elements based on a condition.

**A screen shot of a computer program

Description automatically generated**

Figure 47: Using ternary operators to change color depending on the password strength.

This approach has logical **benefits / advantages** like having **Dynamic User Interfaces** that adapt to user actions, **improves organization** of the code, and **promotes reusable components** that can be conditionally rendered in different contexts.

But it also had **limitations / problems** like **affecting the performance** of the application, **making the code harder to read** which leads to convoluted code and difficulty in tracing data flow, and could introduce **unnecessary component duplication**.

**Hook Design Pattern in React:**

This design pattern is a very powerful tool for managing state and side effects in React applications, its **advantages** include.

* **Improved Reusability** where they can be used across different components which therefore reduces redundancy.
* It **simplifies the component structure** by eliminating the need for methods and class inheritance, making the components more concise and easier to understand.
* **Enhances the composition** by allowing a more flexible component, which combines stateful logic in a more intuitive way.
* **Promotes cleaner code** and **improves maintainability** by separating stateful logic from rendering logic.

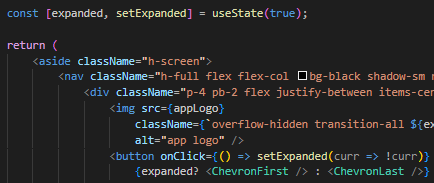


Figure 48: Using the use State Hook to decide on whether to expand the navigation bar.

Although it is very useful, it has also its **limitations** such as: **Requires initial learning effort** for developers who aren’t used to this pattern, considered **complex when debugging** especially for beginners, has the **potential to be overused**, which can lead to difficulty in understanding the components.

**Singleton Design Pattern:**

This design pattern was used specifically for Electron running the app, Service classes, and Testing where it ensured that there is one instance used of Prisma Client across all PassGuard applications, making it easier to control and manage the behavior of the whole application. Also, it allowed us to access the Prisma client from anywhere in the application.

A computer screen shot of text

Description automatically generated

Figure 49: Instantiating one global Prisma client instance.

A computer screen with colorful text

Description automatically generated

Figure 50: Using the export of Prisma to enable access to all methods and objects of Prisma.

This pattern served our application with **Benefits** like:

* **Ease of understanding** the pattern, which made it accessible to all our team, of all experience levels.
* **Simple to implement**, doesn’t need a large amount of code.
* **Provides centralized control** over the Prisma Client instance, which simplifies the execution and test setup, and the interaction with the database.
* **Easily accessible** from any part of the application.

However, there are also some potential **limitations** associated with this pattern which includes:

* The creation of a global instance **can lead to memory** leaks if not managed properly, like for example the Prisma client requires disconnection after being used.
* Causes **difficulty in isolating tests** because the same instance is being used. For example, we have 2 test files and whenever we want to run the tests, both must run (can’t run only a single file).
* Made it **challenging to use dependency injection**, which is making a class independent of its dependencies, as the Prisma client instance isn’t injected into the components.

# Implementation

## Hardware/software used

< **ToDo:**

* List in a table and justify the hardware/software platforms and tools used for the design and implementation.
* Explain why each technology was chosen and its role within the architecture.
* Provide links to relevant documentation for each hardware or software component.

/>

Table : Hardware/software to be used.

|  |  |
| --- | --- |
| Software & Tools used | Description & Justification |
| Electron    Documentation: <https://www.electronjs.org/docs/latest/> | * Using web technologies like HTML, CSS, JavaScript, and TypeScript, desktop apps may be developed on several platforms with the help of the free framework Electron. * It offers a single code base for desktop programs running on Linux, Mac OS, and Windows and incorporates Node.js to offer strong back-end capabilities. * For production applications, Electron is the best option because of its large community, library, and support. * This community is crucial for resolving shared problems and integrating various capabilities. Electron is also a better secure option because it is more generally accepted in the industry and more mature. |
| React    Documentation:  <https://react.dev/learn> | * React is an open-source JavaScript library designed for the development of user interfaces, particularly web applications, with a component-oriented architecture. * It offers a range of advantages, such as the ability to create reusable UI components, a virtual Document Object Model (DOM) implementation that optimizes performance for dynamic user interfaces, and a large library and tool ecosystem that allows for easier access to resources and skilled developers. * These factors contributed to the decision to choose React over Angular. |
| Vite    Documentation:  <https://vitejs.dev/> | * Vite is a cutting-edge front-end development tool that offers a comprehensive suite of features to streamline the development process. * It utilizes esbuild to expedite the bundling process and provide quick hot module replacement, as well as instant server start and reloading. Additionally, Vite is optimized for speed and efficiency, with native TypeScript, JavaScript, CSS, and more support available. * Vite was chosen as the preferred choice over Webpack due to its speed and efficiency, as well as its simplified configuration setup that reduces configuration overhead. |
| Prisma    Documentation:  <https://www.prisma.io/docs> | * With Prisma, you may access and query databases safely and openly using a type-safe database toolkit. * It supports a large variety of databases and is made to be simple to use. Because of its emphasis on Type Safety, Clean Syntax, and robust TypeScript integration, it is a favored option among developers. * Moreover, it provides a simple method for Database Schema Definition, Migration, and Queries, which makes it perfect for teams who value quick development. |
| SQLite    Documentation:  <https://www.sqlite.org/docs.html> | * Lightweight Database: SQLite is a self-contained, serverless, and lightweight database engine that requires minimal setup and administration. * Portability: It's highly portable, allowing easy integration into various applications and platforms without extensive configuration. * Reliability: Despite its lightweight nature, SQLite offers ACID (Atomicity, Consistency, Isolation, Durability) compliance, ensuring data integrity and reliability even in high-demand environments. * Additionally, SQLite is a resource-efficient database, providing sufficient functionality without server-based database overhead, which is especially advantageous for desktop applications. |
| Tailwind CSS    Documentation:  <https://v2.tailwindcss.com/docs> | * Tailwind CSS is a CSS framework that provides developers with the ability to quickly create custom designs without having to leave HTML. * It offers a range of advantages, such as rapid prototyping and great flexibility. Additionally, the utility-first approach allows for a more maintainable CSS than traditional stylesheets, which is why we chose Tailwind CSS over Bootstrap. * This utility-first approach provides designers with more design freedom and finer control over styling, resulting in more unique and tailored designs. * Additionally, Tailwind CSS helps to maintain cleaner and more legible HTML and CSS than traditional frameworks, particularly in large projects. |
| TypeScript    Documentation:  <https://www.typescriptlang.org/fr/docs/> | * Type Safety: TypeScript adds static typing to JavaScript, catching errors during development and enhancing code reliability. * Modern Features: It supports modern JavaScript features while offering additional functionalities like interfaces, enums, and more, aiding in cleaner and more maintainable code. * Tooling Support: TypeScript integrates well with various development tools and IDEs, providing excellent editor support, refactoring capabilities, and easy adoption within existing JavaScript projects. |
| Visual Studio Code    Documentation:  <https://code.visualstudio.com/docs> | * Code Editing: VS Code offers a robust and customizable platform for writing and editing code across multiple programming languages. * Extensions & Integrations: It supports a wide range of extensions and integrations, enhancing functionality and adapting to various development needs. * Productivity & Collaboration: Its features like debugging tools, Git integration, and collaborative coding capabilities boost productivity and teamwork among developers. |
| Draw.io    Documentation:  <https://www.drawio.com/doc/> | * Diagram Creation: Draw.io provides a user-friendly platform to create various diagrams like flowcharts, org charts, and more. * Collaboration: It allows real-time collaboration, letting multiple users work on diagrams simultaneously. * Integration & Flexibility: Integrates with various platforms and offers flexibility in exporting diagrams to different file formats, making it versatile for different needs. |
| Microsoft Project    Documentation:  <https://learn.microsoft.com/en-us/project/> | * Project Management: Microsoft Project streamlines project planning, scheduling, and resource management. * Task Tracking: It helps monitor tasks, timelines, and dependencies in a visual format. * Team Collaboration: Allows teams to coordinate, share progress, and work together efficiently on complex projects. |
| GitHub    Documentation:  <https://docs.github.com/en> | * Collaboration: Working together on projects is made easy using GitHub. * Version control: Monitors modifications to help with code organization. * Open-Source Hub: Promotes an international community for code improvement. |
| Jest  Jest Logo PNG Transparent & SVG Vector - Freebie Supply  Documentation:  <https://jestjs.io/docs/getting-started> | * Testing Simplicity: Jest simplifies the testing process by providing an intuitive and easy-to-use testing framework for TS. * Feature-Rich: It offers a comprehensive suite of features, including built-in assertions, mocking capabilities, parallel test execution, and enhancing testing efficiency. * Configuration Ease: Jest comes with sensible defaults and minimal configuration requirements, allowing developers to get started with testing quickly and smoothly. |

## Challenging issues and solutions

<**ToDo:**

* Discuss the challenging issues encountered during the implementation and how they were addressed.
* Present lessons learnt from this experience, offering valuable insights and guidance to benefit others in similar situations.

/>

**Challenging Issues Encountered**

Our team faced several challenges during the implementation of the password manager, including:  
  
**Creating a Unique UI:** A major challenge was in designing a UI that was both unique and easy to use. We did not want PassGuard to appear generic as the other password managers, yet at the same time the app should be intuitive and user friendly.

**Selecting the Right Tools and Technologies:** It was the first desktop app the team had ever developed, thus, the appropriate tools and software had to be figured out. We also had to learn to blend these technologies together which required a lot of research and debugging.  
  
**Learning New Tools:** For instance, in the front end, we adopted React and Electron which were new tools to us. In addition, we had to learn the ins and outs of Inter-process communication to successfully communicate with the database. This made the project even longer and more complicated.  
  
**Linking Front-end and Back-end:** Another challenge was integrating the front-end system with the database that processes user credentials. Therefore, our team had the obligation to make sure that the app had a mechanism for storing and retrieving user data before the end of SDP 1.

**Solutions Implemented**

To address these challenges, our team took the following steps:

**Conducting Design Research:** With this in mind, we carried out an extensive study on the design principles and practices for UI to be distinctive and friendly to the users. We also built a UI prototype in Figma to help in the actual implementation.  
  
**Extensive Learning:** We utilized documents, tutorials, and online resources to master the new tools and technologies we needed for implementation. We also looked for solutions when we encountered errors.  
  
**Trial and Error:** Using documentation alongside trial-and-error we learnt React and Tailwind. In addition, we tested various approaches and code fragments until we arrived at the needed solution.  
  
**Team Collaboration:** Our team met frequently and discussed ideas of how to move forward. Moreover, we carried out vast exploration in the most appropriate approaches to match up the front-end and back-end.

**Lessons Learned**

Our team learned several valuable lessons from this experience, including:  
  
**The Importance of Design Research:** It is critical that designers conduct an extensive design study, so as to generate a distinctive UI that is also easy to use.  
  
**The Value of Documentation and Tutorials:** Tutorials and documentation are essential tools when learning new tools and technologies.  
  
**The Importance of Trial and Error:** The new programming languages and tools are best mastered through experimentation and trying out.  
  
**The Power of Team Collaboration:** A properly functioning team collaboration can speed up the process of development and help develop better solutions.

**Guidance for Others**

Based on our experience, we offer the following guidance to others who might face similar challenges:

**Start with a Clear Vision:** Set out objectives and goals of the project before the process of development.  
  
**Conduct Thorough Research:** Research widely on design parameters, best practices, and applicable technologies.  
  
**Choose the Right Tools:** Selecting the best suited tools and techniques to suit the needs of the project.  
  
**Embrace Trial and Error:** Remember, you do not know it all so do not hesitate to experiment and try out new things.  
  
**Seek Help When Needed:** Do not shy away from asking for help from your colleagues, mentors and online resources in case of problems.

**Collaborate Effectively:** Establish a working area that will foster free expression of ideas and handling of challenges among the team members.

# Testing and evaluation

<**ToDo:**

* During semester 1, you will test the use cases you have implemented. Then in semester 2 you will complete the testing of the whole solution.

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 project, 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 ‎3.1.
* 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 B - Test cases specification*. Functional testing will allow you to find errors and defects, then fix them and identify possible improvements. You need to have a comprehensive set of tests that verify 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.

/>

**E1: Sign Up Use Case**

Unit Testing:

For testing the Sign-Up use case, we verified that our solution and code satisfied the requirement by making sure that the post condition is valid, and that was done by creating a new user who isn’t yet registered in PassGuard.

A screenshot of a computer

Description automatically generated

Figure 51: User Database before Adding the User.

A computer screen shot of a program

Description automatically generated

Figure 52: Expected and Actual Results of Creating a User.

A screenshot of a computer program

Description automatically generated

Figure 53: Results of creating a new user.

A screenshot of a computer

Description automatically generated

Figure 54: User database after adding a User.

A screen shot of a computer program

Description automatically generated

Figure 55: Trying to add the same user to the User database.

From the above figures and from what we expected, our test results were successful because the user wasn’t in the database, and after executing the use case the user was added to the database. This also shows that the test case has satisfied the requirements due to us expecting that an account will be created for the user. And now the user is authorized to use PassGuard.

**E2: Login Use Case**

Unit Testing:

The goal of this use case is to give access to an existing user to the PassGuard application, by providing his email address and password that were used when the user created his account. So, to test this, we tried to use the query method to retrieve a user who isn’t in the user database and what we expected was an error.

**Expected:** To get a user who already exists in the user database

**Actual Results:**

A screen shot of a computer program

Description automatically generated

Figure 56: Expected and Actual Results of retrieving an existing user from the database.

**A screenshot of a computer screen

Description automatically generated**

Figure 57: Results of retrieving an existing user.

From the above figures and from what we expected, our test results were successful because the user was in the database, and when we tested our method of retrieving that user it returned to the user. This also shows that the test case has satisfied the requirements due to us expecting that an account will be returned from the database, and now the user is authenticated and is able to login into the PassGuard application.

**E3: Add Credential Use Case**

Unit Testing:

Testing adding a credential to an existing user required us to verify and make sure that the that user is already registered so what we tried first is to add a credential to a non-existing user. These were the results.

A screen shot of a computer

Description automatically generated

Figure 58: Expected and Actual results of adding a credential to a non-existing user.

A screen shot of a computer program

Description automatically generated

Figure 59: Receiving error for adding a credential to a non-existing user.

This error was expected due to us trying to add a credential for a user who isn’t in the system, so then we tried adding a credential for an existing user and these were the results.

A screenshot of a computer program

Description automatically generated

Figure 60: Expected and Actual results of adding a credential to an existing user.

A screenshot of a computer screen

Description automatically generated

Figure 61: Results of Adding a new credential.

A screenshot of a computer

Description automatically generated

Figure 62: Before adding a new credential (filling the form)

A screenshot of a computer

Description automatically generated

Figure 63: After adding a new credential.

From the above figures and from what we expected, our test results were successful because the user was in the database, and when we tested our method of adding a new credential to him it created it successfully. This also shows that the test case has satisfied the requirements due to us expecting that a new login credential has been added and stored in the database.

**E4: Update Credential Use Case**

Unit Testing:

An existing user might sometimes want to edit / update an existing credential in the database, so what we did was from the previous section (Add Credential) we used the same credential and updated it to see whether the service method will update it, and here are the results.



Figure 64: Credential database before updating the credential.

A screen shot of a computer program

Description automatically generated

Figure 65: Expected and Actual results of updating a credential for an existing user.

A screenshot of a computer program

Description automatically generated

Figure 66: Results of Updating an existing credential.

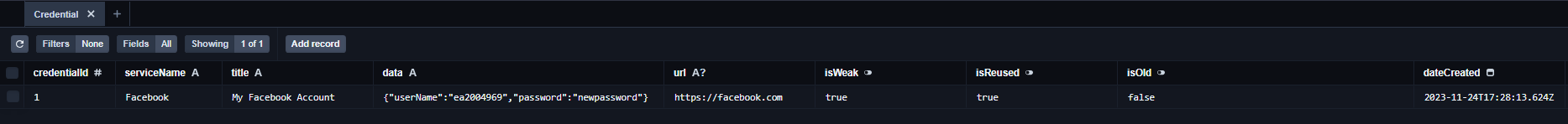


Figure 67: Credential database before after updating the credential.

We also tried to update a non-existing credential for an existing user, and these were the results.

A screen shot of a computer program

Description automatically generated

Figure 68: Expected and Actual results of updating a non-existing credential.

A screen shot of a computer program

Description automatically generated

Figure 69: Results of updating a non-existing credential.

From the above figures and from what we expected, our test results were successful because the credential was in the database, and when we tested our method of updating the credential it was updated successfully. This also shows that the test case has satisfied the requirements due to us expecting an updated login credential, and it has been updated and stored in the database.

**E4: Delete Credential Use Case**

Unit Testing:

An existing user might want to delete an existing credential in the database, so what we did was from the previous section (Update Credential) we used the same credential and deleted it to see whether the service method will delete it, and here are the results.

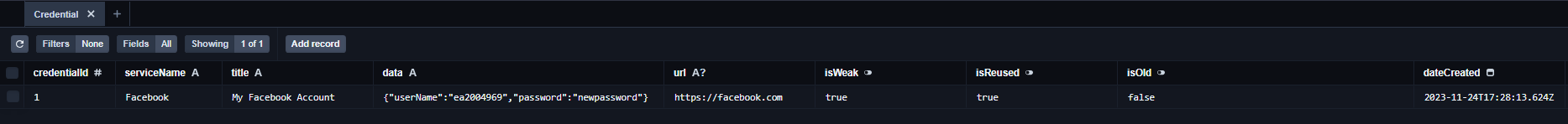


Figure 70: Credential database before deleting the credential.

A computer screen shot of a program

Description automatically generated

Figure 71: Expected and Actual results of deleting an existing credential.

A screen shot of a computer

Description automatically generated

Figure 72: Results of testing the 3 main operations (CUD)

A screenshot of a computer

Description automatically generated

Figure 73: Credential database after deleting the credential.

We also tried to delete a non-existing credential for an existing user, and these were the results.

A screenshot of a computer program

Description automatically generated

Figure 74: Results of deleting a non-existing credential.

From the above figures and from what we expected, our test results were successful because the credential was in the database, and when we tested our method of deleting the credential it was deleted successfully. This also shows that the test case has satisfied the requirements due to us expecting a deleted login credential, and it has been deleted and removed from the database.

## Non-functional testing

<**ToDo:**

* Include a sub-section for each non-functional requirement listed in Section ‎3.2.
* 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 ‎3.2) 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 attainment).
* Highlight the strengths of the solution and list its shortcomings (what worked? what didn’t work?).
* Highlight the key contributions and the novel aspects of your work.

/>

# Future work

< **ToDo:**

* You may complete this section in Semester 2.
* Suggested improvements and further work: identify areas of improvement in the project and features of interest that can be added later. 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 a sub-section for each student individual reflections:
* Describe the professional development accomplished during the project, including any (1) newly acquired technical skills, (2) interpersonal skills in areas like team leadership and effective communication (3) personal growth such as adapting to change and professional and ethical conduct. Explain the value of these newly acquired skills for your future career advancement.
* Identify key shortcomings encountered during the project that you should avoid in future projects.
* Summarize significant lessons learned from the project you intend to carry forwards to your professional life from the personal experiences and the teamwork experienced during the project.

/>

**Essa Ahmed Abou Jabal**

This project provided me with a wealth of technical skills that I hadn't anticipated during my university studies. I learned to use various project management tools such as Asana and MS Project, enabling me to organize my time effectively and adhere to a disciplined schedule. Additionally, I gained proficiency in application testing, a skill I hadn't fully appreciated before. Testing with technologies like Jest was entirely new to me, expanding my technical toolkit significantly. And not to forget, I also discovered the importance of developing locally based applications, catering to users who prioritize privacy.

On the interpersonal front, I became an expert team leader by assigning and carrying out responsibilities with ease, delivering constructive feedback, and motivating team members to accomplish their weekly tasks. Improved communication skills were another outcome, as I actively listened to team members' ideas and concerns, providing clear and concise feedback on our progress. Collaborating with diverse team members further enhanced my ability to work effectively with people from different backgrounds and preferences.

In my case, I became adaptive in the face of unexpected difficulties and project needs that at first seemed overwhelming. I also made sure to treat every member of my team with utmost respect. And in terms of future career advancement, I am confident that these acquired skills will prove invaluable either at jobs or in my own business.

Of course, the project was not without its obstacles. I underestimated the importance of requirements gathering and planning in the initial stages, resulting in delays and rework later. However, these challenges taught me valuable lessons that will undoubtedly benefit my professional life. Proceeding forward, I realize how crucial it is to gather strict specifications to comprehend the full scope of the project and exceed expectations.

**Youssef Aly**

Senior Design Project 1 has been a journey of learning and a new experience for me. I have learned and experienced new skills and software that I always hoped I would grasp during my academic period. The enthusiasm of researching UI design principles and designing my first desktop UI which my team and I would follow during development was unequalled.  
  
 The discovery that we could develop desktop apps using web tools like React and Tailwind CSS is revolutionary to me, inspiring deeper investigation into it. I got fully invested in daily learning, soaking in each concept and technique daily. Weekly meetings and project planning sessions were of great importance in developing my team leadership skills, as well as effective problem-solving strategies of overcoming the problems we came across.  
  
 The development stage also had its surprises. We found some software incompatibilities that almost put us off track. However, I viewed these challenges as opportunities for more learning and resourcefulness. I approached this problem with unbending determination where I have researched and tested other alternatives until I found alternatives.  
  
 These experiences have taught me how critical it is to engage in lifelong learning and change. I am sure the perseverance and the enthusiasm that I must learn new skills will make them crucial in my future career. Finally, I cannot wait to face the different type of challenges that will come with SDP 2 alongside my teammates and to deliver the application that we hoped we would complete when we initially discussed the idea of delving into the journey of developing our very first desktop application.

**Khalifa Yousuf**

During our SDP1, I faced several challenges that became essential learning opportunities which will benefit me throughout my career. Starting with full-stack development was overwhelming, especially when working with the backend using Prisma and SQLite and then moving to the front end with React. This experience forced me to learn new technical skills, greatly contributing to my development as a programmer. Working with these different technologies improved my understanding of each part of software development and showed me how to combine them to make a complete, working application. This was a key moment in my growth as a developer, giving me practical experience in using and combining complex tools and dealing with the fast-changing world of technology. Throughout the duration of our project, my interpersonal skills significantly developed. I gained invaluable experience in effectively collaborating within a team, particularly under the guidance of a group leader. Our weekly meetings, where we thoroughly discussed our progress, proved highly beneficial. These interactions not only enhanced my performance but also greatly contributed to our team's overall productivity and synergy. I also learned the value of adapting to change and acting professionally and ethically during challenging situations.

One key lesson I learned during this project was the importance of accurate time estimation for tasks. Previously, I tended to underestimate the time required, not fully accounting for potential obstacles or the learning curve involved with new technologies. This often led to rushed work or missed deadlines. Moving forward, I recognize the value of making more realistic time estimations. I plan to allocate sufficient time for unforeseen challenges and learning new skills, ensuring a more balanced and less stressful workflow in future projects. This approach will improve the quality of work and contribute to a more efficient and reliable project timeline.

In summary, this project taught me a lot and was beneficial. I learned how to work with new technologies and realized how important it is to appreciate the time needed for tasks. The experience improved my technical skills and gave me practical knowledge in managing projects and time better. All of this will be very useful in my future work.

**Mohammed-Dhia Abdaoui**

During the first phase of the Senior Design Project, I had the chance to work on a real complex problem. The fact that we built the project from scratch, starting with the idea and planning to the implementation and testing is a journey that helped me to approach projects in a more professional way, learning along the way planning, communication, and documentation skills. In addition to that, I was forced to push the limit of my knowledge discovering new technologies, frameworks, and tools which is an important skill that is required in the professional sector. Overall, it was a fruitful experience, and I hope to learn even more in the second phase.

# References

[1] T. Burbidge, “Cybercrime thrives during pandemic: Verizon 2021 Data Breach Investigations Report,” [*www.verizon.com*](http://www.verizon.com), May 13, 2021. <https://www.verizon.com/about/news/verizon-2021-data-breach-investigations-report>

[2] 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>

[3] 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%202019%20Google%20study%20in,different%20password%20for%20every%20account.%E2%80%AF)

[4] IBM, “Cost of a data breach 2023,” *IBM*, 2023. <https://www.ibm.com/reports/data-breach>

[5] “The Human Factor in Data Protection,” *Ponemon Institute*. <https://www.ponemon.org/news-updates/blog/security/the-human-factor-in-data-protection.html>

[6] IJRASET, “A Study for an Ideal Password Management System,” [*www.ijraset.com*](http://www.ijraset.com/). <https://www.ijraset.com/research-paper/an-ideal-password-management-system>

[7] Verizon Data Breach Investigation Report 2023, <https://www.verizon.com/business/resources/reports/2023/2023-data-breach-investigations-report-dbir.pdf>

[8] “Password health indicator - Microsoft Support,” *support.microsoft.com*. <https://support.microsoft.com/en-us/topic/password-health-indicator-5df7b4bc-cdb2-430a-9951-034accc57ff3>

[9] L. S. Vailshery, “Password management software market share 2023,” Statista, <https://www.statista.com/statistics/1331322/password-management-market-share/>.

[10] “Password management market size & growth report, 2025,” Password Management Market Size & Growth Report, 2025, [https://www.grandviewresearch.com/industry-analysis/password-management-market](https://www.grandviewresearch.com/industry-analysis/password-management-market.)

[11] C. Fripp, “Use this chart to see how long it’ll take hackers to crack your passwords,” *Komando.com*, Mar. 20, 2021. <https://www.komando.com/security-privacy/check-your-password-strength/783192/>

[12] S. Boyd, “NordPass review 2023 - is it any good?,” SafetyDetectives, [https://www.safetydetectives.com/best-password-managers/nordpass/.](https://www.safetydetectives.com/best-password-managers/nordpass/. )

[13] I. Valiaugaitė, “Bitwarden Password Manager Review - Cybernews,” cybernews, [https://cybernews.com/best-password-managers/bitwarden-review/.](https://cybernews.com/best-password-managers/bitwarden-review/. )

[14] A. a Pcm. security analyst and K. Key, “Dashlane,” PCMag Middle East, [https://me.pcmag.com/en/password-managers/18266/dashlane.](https://me.pcmag.com/en/password-managers/18266/dashlane. )

[15] M. Jennings, “Zoho Vault Review: Pros & Cons, features, ratings, pricing and more,” TechRadar, [https://www.techradar.com/reviews/zoho-vault.](https://www.techradar.com/reviews/zoho-vault.  )

[16] E. Goldstein, “1Password Review 2023: Is it safe + worth the price?,” SafetyDetectives, [https://www.safetydetectives.com/best-password-managers/1password/.](https://www.safetydetectives.com/best-password-managers/1password/. )

[17] K. Glamoslija, “Keeper password manager review 2023: Is it any good?,” SafetyDetectives, [https://www.safetydetectives.com/best-password-managers/keeper/.](https://www.safetydetectives.com/best-password-managers/keeper/. )

[18] N. J. Rubenking, “LogMeOnce Password Management Suite Ultimate Review,” PCMAG, [https://www.pcmag.com/reviews/logmeonce-password-management-suite-ultimate.](https://www.pcmag.com/reviews/logmeonce-password-management-suite-ultimate)

[19] G. Kallstrom, “Password boss review 2023: Expert rated 4.6/5,” Password Manager, [https://www.passwordmanager.com/password-boss-review/.](https://www.passwordmanager.com/password-boss-review/. )

[20] S. Trejos, Ed., “RoboForm review 2023: Affordable but basic,” All About Cookies, [https://allaboutcookies.org/roboform-password-manager-review.](https://allaboutcookies.org/roboform-password-manager-review.  )

[21] Oesch, Timothy, "An Analysis of Modern Password Manager Security and Usage on Desktop and Mobile Devices. " PhD diss., University of Tennessee, 2021. <https://trace.tennessee.edu/utk_graddiss/6670>

[22] Z. G., “The best password organizer books - best reviews,” Best Password Managers Reviews, [https://password-managers.bestreviews.net/the-best-password-organizer-books/.](https://password-managers.bestreviews.net/the-best-password-organizer-books/.  )

[23] S. K. Shinde and M. V. Deshpande, “IJRASET Journal for Research in Applied Science and Engineering Technology,” A Study for an Ideal Password Management System, [https://www.ijraset.com/research-paper/an-ideal-password-management-system.](https://www.ijraset.com/research-paper/an-ideal-password-management-system. )

[24] R. Ayyagari, J. Lim, and O. Hoxha, “Why Do Not We Use Password Managers? A Study on the Intention to Use Password Managers”, CMR, vol. 15, no. 4, pp. 227–245, Dec. 2019.

[25] “Securely store, Manage & Autofill Passwords,” NordPass, <https://nordpass.com/>.

[26] “The password manager trusted by Millions,” Bitwarden, <https://bitwarden.com/>.

[27] Dashlane, “Dashlane password manager,” Dashlane, <https://www.dashlane.com/>.

[28] “Zoho Vault: Password manager with single sign-on & MFA for businesses, teams & families,” Zoho, <https://www.zoho.com/vault/>.

[29] “Password manager for Families, Enterprise & Business,” 1Password, <https://1password.com/>.

[30] “Password & Secrets Management: Keeper security,” Keeper® Password Manager & Digital Vault, <https://www.keepersecurity.com/>.

[31] “Best password manager | home | logmeonce,” LogMeOnce, [https://logmeonce.com/.](https://logmeonce.com/)

[32] “Password management,” Password Boss, <https://www.passwordboss.com/>.

[33] “Next generation password manager,” Best Password Manager: No more writing down passwords, <https://www.roboform.com/>.

# Appendix A – Use cases specification.

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **R3** | **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 clicks on the “Forgot Password” option on the login screen. | | 2. Prompt the user to enter their registered email address. |
| 3. The user provides their registered email address. | |  |
|  | | 4. Verify email address syntax validity. **(See 4.a. for alternative flow).** |
|  | | 5. Find the registered email address. |
|  | | 6. Send a verification code to the email address registered. |
|  | | 7. Prompt user to enter verification code sent. |
| 8. The user enters the verification code received in their email. | | 9. Verify verification code entered by user. **(see 8.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:**  **4.a.** If the email address is invalid, display an error message and user is prompted to re-enter a valid email address.  **8.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** | **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” under the account settings tab. | | 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: E1** | **E1: Sign Up** | |
| **Brief Description** | The user creates / registers a new account to be able to use the password manager application. | |
| **Primary actors** | User | |
| **Preconditions:**   * The user must not have an existing account in the system. | | |
| **Post-conditions:**   * An account was created for the user. * 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 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)** |
|  | | 9. Assign a unique number to the account. |
|  | | 10. Inform the user with the account number |
|  | | 11. **<extend: Enable 2FA>** If the user wants to enable 2FA on his account / link their accounts with a secondary authentication method. |
|  | | 12. 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:**  **a) Performance      b) Reliability      c) Security      d) Usability      e) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: E2** | **E2: 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:**   * The user must be already registered. | | |
| **Post-conditions:**   * The user has access / is logged into the system. | | |
| **Main Success Scenario:** Access has been given to the user to 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. Validate the user details in the database. |
|  | | 6. Gain access / Log the user into the system if details are valid. **(See 6.a. for alternative flow)** |
|  | | 7. **<extend: Recover Access>** If the user forgot his master password & clicks on “Forgot Password” |
|  | | 8. Display the stored credentials / user profile page and main interface of the application. |
| **Alternative flows:**  6.a. If the user provided invalid details, generate an error message “Invalid Login”. | | |
| **Special Requirements:**  **a) Performance      b) Security      c) Usability      d) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: E3** | **E3: 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:**   * The user must open the application. * The user must be already registered. * The user must be logged into their password manager account. | | |
| **Post-conditions:**   * The user added a new login credential. * The login credential was stored in the credential’s database. | | |
| **Main Success Scenario:** Login credential has been added successfully. | | |
| **Actor Action** | | **System Response** |
| 1. The user selects “Add Credential”. | | **2**. Prompt the user to enter the new credential details. |
| 3. The user provides the required information (username, title, ...) | | 4. **<include: Audit Credential Use Case>** |
|  | | 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 “Save” button. | |  |
|  | | 7. Verify the user input. **(See 7.a. for alternative flow).** |
|  | | 8. Add the new credential to the user’s database. **(See 8.a. for alternative flow).** |
|  | | 9. Display confirmation with the newly added credential. |
| **Alternative flows:**  7.a. If the user selects cancel adding credential, open the main window / his database.  8.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:**  **a) Performance      b) Reliability      c) Security      d) Usability      e) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: E4** | **E4: Update Credential** | |
| **Brief Description** | The user edits / updates an existing login credential / entry for a specific website (domain), application, in their database. | |
| **Primary actors** | User | |
| **Preconditions:**   * The user must open the application. * The user must be already registered. * The user must have an existing credential selected to be updated. | | |
| **Post-conditions:**   * The user updated an existing login credential. * 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 selects “Update Credential” | | 2. Find the credential to be updated in the database. |
|  | | 3. Fill in the form fields with the credentials details and open the form. |
|  | | 4. Prompt the user to enter the updated credential details. |
| 5.  The user provides the updated information | | 6. **<include: Audit Credential Use Case>** |
| 7. The user clicks on the “Update” button. | | 8. Verify the user input. **(See 8.a. & 8.b. for alternative flow).** |
|  | | 9. Update the credential to the user’s database |
|  | | 10.  Display confirmation with the updated credential |
| **Alternative flows:**  8.a. If the user selects cancel updating credential, open the main window / his database.  8.b. 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:**  **a) Performance      b) Reliability      c) Security      d) Usability      e) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id: E5** | **E5: Remove Credential** | |
| **Brief Description** | The user removes an existing login credential / entry for a specific website (domain), application, in their password manager. | |
| **Primary actors** | User | |
| **Preconditions:**   * The user must open the application. * The user must be already registered. * The user must have an existing credential selected to be deleted / removed. | | |
| **Post-conditions:**   * The user deleted an existing login credential. * 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 selects “Remove Credential”. | | 2. Display a pop-up message “Are you sure you want to delete the selected entry?” (may ask for master key) |
| 3. The user responds by clicking the “Yes” button. | | 4. Verify the user input. |
|  | | 5. Remove credentials from the credential database. **(See 5.a. for alternative flow).** |
|  | | 6. Display confirmation that credential has been removed. |
| **Alternative flows:**  5.a. If the user selects cancel removing credential, open the main window / his database / credential page. | | |
| **Special Requirements:**  **a) Performance      b) Reliability      c) Security      d) Usability      e) Availability** | | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** **E6** | **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.  (technical: assuming the GIU saves details of actual open credential) | |
| **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** | |  | | --- | | **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. Find all the credentials in the credential container with similar title. |
|  | | 4. Dynamically displays all credentials **(See 4.a for alternative flow)** |
| 5. User selects searched credential | | 6. Find the selected credential in the credential database. |
|  | | 7. Open form with the credential information injected inside it. |
| Alternative flows:  4.a No credential was found: display an option “no credential was found”. | | |

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** **E8** | **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.  Account is created or being created.  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. Display the Generate Password UI with all the possible preferences (length, symbols, uppercase, lowercase, numbers) |
| 3. User chooses his preferences. | | 4. Generate strong password |
|  | | 5. Display generated password (See 5.a for alternative flow) |
| 6. User confirms and clicks on “Fill in Password” | |  |
|  | | 7. Fills in the generated password inside the password input field |
|  | | 8. Return to the normal Form page |
| Alternative flows:  5.a If user wants to re-generate a new password, then he will choose new preferences which restarts step 3. | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **E9** | **E9: Export Credentials** | |
| **Brief Description** | The user can use our password manager through multi devices and export login Credentials through multi devices | |
| **Primary actors** | User | |
| **Trigger(s)** | The user requests to transfer his login Credentials to a new device | |
| **Preconditions:**  - user account exist  - user authorization to transfer credentials | | |
| **Post-conditions:**   * All login credentials transfer to the new device * Credentials file generated successfully. | | |
| **Normal Scenario** | | |
| Actor Action | | System Response |
| 1. User click on “Export credentials” under the settings tab. | |  |
|  | | 2. Prompt the user with export options. (location, file type, file password expiry) |
| 3. User confirms selected options. | | 4. Retrieve the data. |
|  | | 5. Encrypt the data. |
|  | | 6. Save in a file. |
|  | | 7. Notify user with success. |
| **Alternative flows:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **R2** | **R2: Change Settings** | |
| **Brief Description** | User will be able to customize settings in the password manager | |
| **Primary actors** | User | |
| **Trigger(s)** | Users go to settings to change the settings | |
| **Preconditions:**   * The user is logged into the password manager. | | |
| **Post-conditions:**   * A specific Settings will be changed after the user change it through **settings.** | | |
| **Normal Scenario** | | |
| Actor Action | | System Response |
| 1. User clicks on “Settings” button. | | 2. Display settings menu. |
| 3. User selects a specific setting to modify. | | 4. **<extend: Wipe Account use case>** if the user clicks on “Wipe Account” user account will be wiped. |
|  | | 5.<**extend: AutoFill Credentials use case >** if the user enables “AutoFill Credentials” the credentials will be auto filled when the site is accessed. |
| **Alternative flows:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:**  **R1** | **R1: Audit Credentials** | |
| **Brief Description** | Users can review and audit the security of their stored credentials, checking for weak or duplicated passwords. | |
| **Primary actors** | User | |
| **Trigger(s)** | The user must enter his password in the specific area | |
| **Preconditions:**   * The user is logged into the password manager. | | |
| **Post-conditions:**   * The User will have full information about his password strength and takes the necessary action. | | |
| **Normal Scenario** | | |
| Actor Action | | System Response |
| 1. User enters or generates a password | | 2. Validate password strength. |
|  | | 3. Check for password duplication. |
|  | | 4. Display password health |
| **Alternative flows:** | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** **E10** | **E10: Import Credentials** | |
| **Brief Description** | The user will import the exported credentials | |
| **Primary actors** | User | |
| **Preconditions:**   * The user is logged in to the password manager. * The user had exported his login credentials | | |
| **Post-conditions:**   * The credentials will be imported successfully and stored in the system. | | |
| **Normal Scenario** | | |
| Actor Action | | System Response |
| 1. the user will login to the system | |  |
| 2. the user will access the import credential option | | 3.the system will show the import interface to the user |
| 4. the user will select the credentials that he wants to import | | 5. the system will ask for authentication from the user |
| 6. the user will authenticate him self | | 7. the system will ask for confirmation from the user |
| 8. the user will confirm | | 9. the system will show to the user that his request was done successfully |
| **Alternative flows:**  a. if the user failed to authenticate himself, the system will ask the user to try again. | | |

# Appendix B – Test cases specification

< **ToDo:**

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

/>

**Use Case: Add Credentials**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 1 | **Associated user case ID** | E3 |
| Test designed by | Essa Ahmed Abou Jabal | **Test design date** | 16/11/2023 |
| **Executed by** | Essa Ahmed Abou Jabal | **Execution date** | 21/11/2023 |
| **Test case name** | Add credentials | | |
| **Short description** | The user adds a new login credential for a specific website (domain), application, to their password manager. | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user must open the application.   + The user must be already registered.   + The user must be logged into their password manager account | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Click the '+' button | The system displays a form to add the credentials | As expected | Pass |  |
| 2 | add credentials title |  |  |  |  |
| 3 | add service name |  |  |  |  |
| 4 | Add service type |  |  |  |  |
| 5 | Add username/email |  |  |  |  |
| 6 | Add password | The system will display the password strength (very weak, weak, moderate, strong, very strong) | As expected | pass |  |
| 7 | Add login Page URL |  |  |  |  |
| 8 | Click ‘Save‘ button | The system will save the credentials and will display a card for saved credentials in the main page. | As expected | Pass |  |
| 9 | Repeat steps 1 to 7 using another new credential and click ‘cancel’ button | The system will close the credential form | As expected | Pass |  |
| **Post-conditions:**   1. The user added a new login credential. 2. The login credential was stored in the user’s database. | | | | | |

**Use Case: Sign Up**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 2 | **Associated user case ID** | E1 |
| Test designed by | Khalifa Yousuf | **Test design date** | 2/11/2023 |
| **Executed by** | Khalifa Yousuf | **Execution date** | 23/11/2023 |
| **Test case name** | Sign Up | | |
| **Short description** | The user creates / registers a new account to be able to use the password manager application | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user must be already registered. | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Enter first name |  |  |  |  |
| 2 | Enter last name |  |  |  |  |
| 3 | enter email address |  |  |  |  |
| 4 | enter password for this email |  |  |  |  |
| 5 | Re-enter password |  |  |  |  |
| 6 | Click ‘Sign up’ button | The system will return a message ‘account created successfully’ | As expected | pass |  |
| 7 | Check post-condition 1 |  |  |  |  |
| 8 | Repeat steps 1 and 2 |  |  |  |  |
| 9 | Enter registered email address |  |  |  |  |
| 10 | Repeat steps 4 and 5 |  |  |  |  |
| 11 | Click ‘Sign up’ button | The system will return a message ‘account already exist’ | As expected | pass |  |
| 12 | Repeat step 1 to 4 |  |  |  |  |
| 13 | Enter password that does not match entered password |  |  |  |  |
| 14 | Click ‘Sign up button | The system will return a message ‘passwords don’t match’ | As expected | pass |  |
| **Post-conditions:**   * + An account was created for the user.   + The user details are stored in the database. | | | | | |

**Use Case: Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 3 | **Associated user case ID** | E2 |
| Test designed by | Youssef Aly | **Test design date** | 27/10/2023 |
| **Executed by** | Youssef Aly | **Execution date** | 15/11/2023 |
| **Test case name** | Login | | |
| **Short 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. | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user must not have an existing account in the system. | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | enter valid email address |  |  |  |  |
| 2 | enter valid password for this email |  |  |  |  |
| 3 | Click ‘login’ button | The system will return a message ‘login successfully’ | As expected | pass |  |
| 4 | Repeat step 1 |  |  |  |  |
| 5 | Enter invalid password |  |  |  |  |
| 6 | Click ‘login’ button | The system will return a message ‘email or password is not correct’ | As expected | pass |  |
| 7 | Check post-condition 1 |  |  |  |  |
| **Post-conditions:**   1. The user has access / is logged into the system. | | | | | |

**Use Case: Update Credentials**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 4 | **Associated user case ID** | E4 |
| Test designed by | Essa Ahmed Abou Jabal | **Test design date** | 15/10/2023 |
| **Executed by** | Essa Ahmed Abou Jabal | **Execution date** | 24/11/2023 |
| **Test case name** | **Update Credential** | | |
| **Short description** | The user edits update an existing credentials | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user must open the application.   + The user must be already registered.   + The user must have an existing credential selected to be updated. | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Click on the card | The system displays options (delete, update) | As expected | Pass |  |
| 2 | Click ‘update’ | The system displays will show the form selected to update | As expected | Pass |  |
| 3 | Update required credentials |  |  |  |  |
| 4 | Click ‘Save’ button | The system displays a message ‘credentials have been updated’ | As expected | Pass |  |
| 5 | Repeat steps from 1 to 3 |  |  |  |  |
| 6 | Click ‘cancel button’ | The system displays a message ‘Are you sure you want to cancel?’ Options(yes/no) | As expected | Pass |  |
| 7 | Click ‘Yes’ button | The system will close the form and credentials will not be updated | As expected | Pass |  |
| 8 | Repeat step 6 |  |  |  |  |
| 9 | Click ‘no’ button | The system will lead you back to credentials form | As expected | Pass |  |
| **Post-conditions:**   * + The user updated an existing login credential.   + The login credential was updated in the user’s database. | | | | | |

**Use Case: Remove Credential**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case #** | 5 | **Associated user case ID** | E4 |
| Test designed by | Mohamed Dhia Abdaoui | **Test design date** | 15/10/2023 |
| **Executed by** | Mohamed Dhia Abdaoui | **Execution date** | 23/11/2023 |
| **Test case name** | **Remove Credential** | | |
| **Short description** | The user removes an existing login credential | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**   * + The user must open the application.   + The user must be already registered.   + The user must have an existing credential selected to be deleted / removed. | | | | | |
| **Step** | **Test Step** | **Expected System Response** | **Actual Result** | **Pass/ Fail** | **Comment** |
| 1 | Click on the card | The system displays options (delete, update) | As expected | Pass |  |
| 2 | Click ‘Delete’ | The system displays a message “Are you sure you want to delete?” Options(yes/no) | As expected | Pass |  |
| 3 | Click ‘Yes’ button | The system will delete selected credential | As expected | Pass |  |
| 4 | Repeat step 1 and 2 |  |  |  |  |
| 5 | Click ‘no’ button | The system will close the dialog box | As expected | Pass |  |
| **Post-conditions:**   * + The user deleted an existing login credential.   + The login credential was deleted in the user’s 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.

/>