Report C3F1 Password Manager

Software Architecture and Pattern Design Project

D. Calò **M. Capuano** **L. Crispino** **F. Ficarella**

Academic Year 2024/2025

Professor: Marina Mongiello
Supervisors: Pwnzer0tt1 Team

Contents

Abstract

This report outlines the development process and design considerations of a secure and user-friendly Password Manager called C3F1. The project was conceived to address the increasing need for robust cybersecurity solutions in an era where diglital threats continue to grow with great frequency and sophistication. The Password Manager is designed to help individuals or even organizations securely store, manage and generate strong passwords and other sensitive information, such as authentication tokens and confidential notes. The core focus of the project lies in striking a balance between security and usability. The Password Manager employs state-of-the-art encryption algorithms to ensure that user data remains secure from unauthorized access while maintaining a user-friendly interface to cater to diverse technical proficiencies. The key features of the project include encrypted password storage, automatic strong password generation, TOTP code generation, and data breach detection. Our team of four, collaborated to design a solution that adresses critical need for robust cybersecurity in today's digital landscape. This Password Manager also emphasizes best practices in cybersecurity by promoting the adoption of strong and unique passwords for all accounts. By mitigating the risks associated with weak or reused passwords, the project contributes to reducing the likelihood of data breaches and improving overall security hygiene. The report delves into the technical architecture, the challenges faced and the solutions implemented during the development process. The project demonstrates how innovative design principles and cutting-edge technology can converge to create a reliable and accessible tool. Ultimately, this Password Manager aspires to contribute to the broader goal of enhacing online security and promoting some best practices in digital identity management

1 Introduction

Numerous articles have been published discussing the security of password managers, aiming to guide their future development. Some of these articles delve into the vulnerabilities associated with password managers, offering readers a defense-in-depth approach to enhance their security. A significant distinction can be observed between articles when considering the type of password manager, such as open-source versus proprietary (licensed) solutions. Specifically, open-source password managers tend to be more susceptible to cybersecurity vulnerabilities compared to their proprietary counterparts, which often exhibit fewer security issues. It is widely recognized that password-based authentication on the web is inherently insecure. One of the primary challenges associated with this method lies in the cognitive burden placed on users to create and manage secure, random passwords for the numerous websites they access [14]. Research indicates that many users, perhaps out of rational necessity, have resorted to creating weak passwords and reusing them across multiple platforms. This widespread behavior further underscores the importance of secure password management solutions, as discussed in the aforementioned articles [7, 23, 13]. By addressing these vulnerabilities and user habits, password managers—whether open-source or proprietary—play a crucial role in mitigating the risks associated with password-based authentication.

1.1 Most commonly used password

Talking about the most commonly used password [2], is an important evaluation, for understanding why the use of a password manager is higly raccomend. Many users continue to rely weak, easily guessable passwords-such as "123456" or "password"-depsite widespread awareness of the risks associated with such practices. This behavior stems from cognitive limitations in generating and remembering unique, complex passwords for each account, as we can see in the graph 1, the most common password used are very simply, that can be easly find by a *bruteforce attack* in less than one second. Password managers are designed to mitigate these issues by generating, storing, and autofilling strong passwords, thereby reducing the reliance on user memory. However, the persistence of weak passwords even with the availability of these tools highlights several interrelated concerns:

- **Adoption Gap**: Not all users adopt password managers, either due to a lack of awareness, mistrust in their security, or resistance to change in habitual behaviors;

- **Misuse of Password Managers**: Even among users who utilize password managers, there can be a tendency to store commonly used weak passwords, which undermines the tool's security benefits;

- **Vulnerability Propagation**: Weak passwords, when reused across multiple accounts, amplify the impact of data breaches, as attackers can exploit compromised credentials to gain unauthorized access to other services;

- **Trust and Security in Password Managers**: The effectiveness of password managers depends on the trustworthiness of the tool itself. Vulnerabilities in the password manager software or cloud storage can expose even well-generated unique passwords to potential compromise.

Some of all this concept listed before will be discuss later in our report, to analyze also the actual state-of-art, of this technology.

1.2 Mitigate the Human Risk

Password managers are designed to address the challenges posed by the increasing complexity and quantity of passwords users must manage. A secure password manager can generate and autofill strong, unique passwords for various websites, alleviating users from the cognitive burden of remembering them. Moreover, by autofilling credentials based on the specific context of a webpage, password managers provide a degree of protection against *phishing attacks* by ensuring credentials are entered only on legitimate sites. A significant feature offered by modern password managers is cloud-based synchronization, enabling seamless access to stored passwords across multiple devices. This combination of enhanced security and user convenience positions password managers as a highly effective tool with minimal deployment costs. Their utility is widely endorsed in technical literature; for instance, a recent US-CERT publication highlights:

> *" A Password Manager is one of the best ways to keep track of each unique password or passphrase that you have created for*

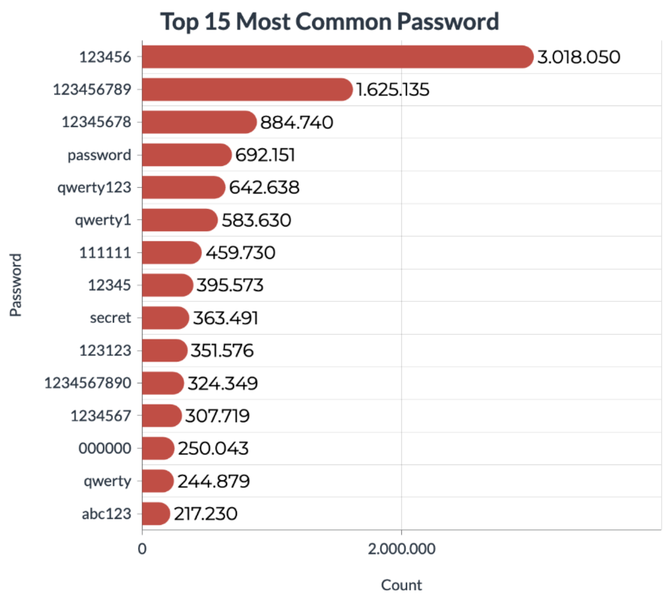


Figure 1: Most commonly used Password according to NordPass Report 2024

your various online accounts without writing them down on a piece of paper and risking that others will see them. "

As a result, it is unsurprising that an increasing number of users are turning to password managers as a solution to mitigate password fatigue and bolster account security.

2 Literature Review

This section provides a comprehensive review of the current state-of-the-art in password manager adoption and technology. We begin by examining statistical trends derived from various studies identified during the research phase. Subsequently, we delve into the progression of password manager technology over time, particularly its evolution in addressing persistent security challenges. Finally, we conclude with a comparative analysis of some of the most widely used password managers available today.

2.1 Statistical Trends

One of the key studies referenced in this review was conducted by Brett Cruz [8], who examined password manager usage across a representative sample of over 94 million individuals in the United States. The results of this study reveal an upward trend in the adoption of password managers over the last few years, highlighting important shifts in user behavior and security practices.

Based on the data, in 2022, only 21% of the surveyed population in the United States reported actively using a password manager. This figure increased substantially to 34% in 2023, marking a notable rise in adoption within a single year. By 2024, the percentage of users employing pass-

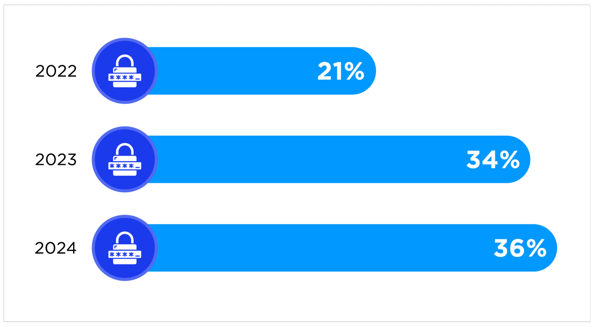


Figure 2: Adoption trends of password managers in the US over recent years.

word managers rose slightly to 36%. Although the growth rate appears to have slowed, this trend is still viewed as a positive indicator of increasing awareness and adoption of secure password management practices.

The findings also shed light on the types of devices where password managers are being utilized. Password manager installations remain consistently high on mobile devices, reflecting their widespread use as primary access points to online accounts. Moreover, the adoption of password managers on personal computers has surged significantly, suggesting that users are extending their reliance on these tools across multiple platforms. Notably, 77% of password manager users reported installing the software on more than one device, an increase from 71% in the previous year. This trend underscores the growing importance of cross-device synchronization as a critical feature for password manager users.

Despite the encouraging rise in adoption, a significant portion of the population continues to refrain from using password managers. Alarmingly, nearly one-third of adults who do not utilize password management solutions have fallen victim to identity theft or credential theft within the past year. This statistic highlights the risks associated with inadequate password practices and the potential benefits of adopting password managers.

Interestingly, public attitudes toward password managers seem to be evolving. Last year, 71% of individuals who did not use a password manager expressed openness to adopting one in the future. Although the overall increase in adoption during the past year was modest, current surveys suggest that more than three-quarters of Americans are now willing to consider using a password manager. This growing willingness indicates the potential for a more significant uptick in adoption rates in the near future, driven by increasing awareness of the security benefits offered by these tools.

2.2 Evolution of Password Managers

While password managers (PMs) have significantly improved digital security practices, their use alone is often insufficient to guarantee account safety. This shortfall becomes evident when users store weak, guessable, or pre-

On wich devices you use a password manager ?	2021	2022	2023	2024
Mobile	77%	84%	84%	83%
PC	75%	75%	77%	90%
Tablet	46%	44%	44%	36%

Table 1: Trends on Devices on which Password Managers are Installed

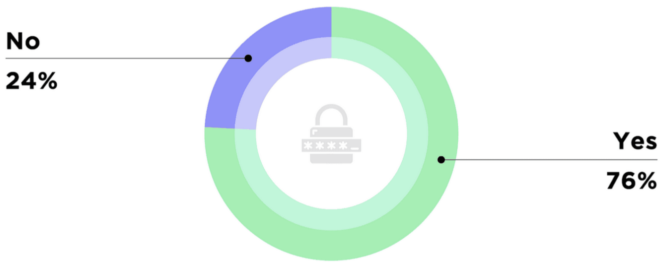


Figure 3: Projected willingness to adopt password managers among non-users.

viously breached passwords. To address this issue, modern PMs include "checkup" features designed to identify and report insecure passwords within a user's vault.

Password managers alleviate the need for users to remember complex passwords by generating, storing, and autofilling strong, random, and unique passwords for different accounts 2. PMs can be broadly categorized into three types [19]:

- **Browser-based PMs**: These are integrated into web browsers such as Google Chrome and Firefox and come pre-installed as part of the browser environment.

- **Third-party PMs**: These are standalone applications that users must install separately. Many are offered on a subscription basis, although open-source and free alternatives are available.

- **System PMs**: These are embedded into operating systems, such as Apple's iCloud Keychain, and are automatically available to users of the corresponding systems.

Despite the availability of diverse password manager options, bad password practices continue to persist. Research reveals that even among PM users—particularly those relying on browser-based or system-integrated solutions—password reuse across accounts remains prevalent[19, 22]. Moreover, many users generate their passwords manually, utilizing PMs solely for storage purposes, rather than taking full advantage of their password-generation capabilities.

To counteract these behaviors and enhance security, many PMs incorporate security audit tools known as "checkups." These tools evaluate the strength of stored passwords and alert users to weak or reused credentials [19]. Advanced checkups can identify passwords found in public breach datasets and notify users accordingly. While such breach reporting features are often restricted to subscription tiers in third-party PMs, browser-based and system-integrated PMs generally offer these services for free.

The criteria for flagging weak passwords vary across PMs. A password may be deemed weak based on factors such as:

- Results from the PM's internal password strength checker.
- Discovery of the password in a known data breach.
- Password reuse across multiple accounts.
- Expiration dates assigned to passwords in corporate or enterprise settings.

Feature	**Percentage**
Synchronization to access logins across multiple devices	91%
Sharing of logins with family members and friends	34%
Password generation	67%
Two-factor Authentication	85%

Table 2: Usage of various features in password managers.

2.3 Comparison of Password Managers

To establish a robust foundation for developing our password manager, we conducted an in-depth analysis of several existing solutions that are widely utilized by the community [20, 15]. Recognizing that most password manager users access these services via mobile devices, our research focused on seven of the most downloaded password managers available on the Google Play Store and Apple App Store: *pCloud Pass*, *Kaspersky Password Manager*, *Dashlane*, *NordPass*, *Keeper*, and *RoboForm*. In this section, we examine these tools in detail, highlighting the features they offer, including password generation, storage, autofill capabilities, and security audits. Additionally, we compare their functionalities to identify their respective strengths and limitations. This comparative analysis provides valuable insights into current best practices and gaps in the market [15], which inform the design and development of our proposed solution.

2.3.1 pCloud Pass

pCloud Pass [21], integrated with the pCloud cloud service, is one of the most secure and reliable password managers available for both Android and iPhone. It offers a high level of security through advanced encryption protocols and robust safety measures implemented within the cloud service. Additionally, its password generator is customizable, and access to the app can be secured with biometric authentication, such as fingerprint recognition or facial unlocking.

The app provides a free plan (pCloud Pass Free), which can be used indefinitely but is limited to a single device. Users who wish to synchronize their passwords across multiple devices can opt for a paid subscription, starting at €29 per year. Notably, pCloud Pass is one of the few services offering a lifetime subscription option, allowing users to pay once and enjoy the service without recurring fees.

Key features include:

- Integrated password generator.
- Biometric access options.
- Lifetime subscription available.
- Lacks tag-based password management.

2.3.2 NordPass

NordPass [26], developed by the renowned NordVPN service, is another excellent password manager available for smartphones. This tool not only stores passwords securely but also saves credit card information and banking credentials.

NordPass helps users enhance their password security by identifying weak or reused credentials. The app notifies users of overly simple or frequently reused passwords, recommending replacements generated randomly by the app itself. It employs strong encryption, supports two-factor authentication (2FA), and stores all credentials in a zero-knowledge vault, ensuring that only the legitimate owners can access their data.

The free version of NordPass is limited to a single device. To unlock full functionality across multiple devices, users can opt for a paid subscription starting at €1.29 per month.

Key features include:

- Monitoring for weak or breached passwords.
- Two-factor authentication and biometric access.
- Free version available.
- Limited password organization options.

2.3.3 Dashlane

Dashlane [24] stands out as one of the top password managers for those seeking a secure solution to store both personal and corporate credentials. It automatically saves and autofills passwords and payment details, while also monitoring the dark web for potential breaches or compromises.

The app supports two-factor authentication and biometric access to enhance security and prevent unauthorized use. Dashlane offers a free plan with no time limits, but like other services, it is restricted to a single device. To synchronize passwords across multiple devices, users must subscribe to a paid plan, starting at €3.62 per month.

Key features include:

- Password breach monitoring.
- Biometric access security.
- Free version available.
- Usability could be improved.

2.3.4 1Password

1Password [3] is a widely recognized password manager designed for securely storing and using credentials in an encrypted manner. In addition to passwords, it stores credit card details, email addresses, and critical documents using 1 GB of secure storage space.

The app also provides features like password recovery for accidentally modified credentials, generating strong passwords, and configuring passwordless access systems using biometric unlocking and notification-based authentication.

Key features include:

- Passwordless access systems.
- Document storage with 1 GB capacity.
- Credential recovery options.
- Autofill functionality could be improved.

2.3.5 Kaspersky PM

Kaspersky Password Manager[1] can be used as part of Kaspersky's antivirus suite or as a standalone app. It includes a random password generator, synchronization across browsers and devices, and storage for credit card details and addresses. The app also provides cloud storage for important documents, secured with advanced encryption and two-factor authentication.

Kaspersky Password Manager does not offer a free version but provides a 14-day free trial. After the trial, users can subscribe to the service for €16.99 per year, with a 30-day money-back guarantee.

Key features include:

- Secure password generator.
- Encrypted storage for documents.
- Limited offline accessibility.

2.3.6 Keeper

Keeper [25] focuses heavily on security, offering tools to protect encrypted vaults from hackers and malicious actors. It monitors passwords, checks for breaches in the dark web, and notifies users about weak or repeated passwords.

In addition to password management, Keeper offers secure password sharing options and encrypted messaging for private communication. A free 30-day trial is available, after which users can subscribe to a plan starting at €2.92 per month, billed annually.

Key features include:

- Dark web monitoring.
- Encrypted messaging.
- Secure password sharing.
- Customer support could be improved.

2.3.7 RoboForm

RoboForm [27] is a robust password manager offering advanced security features to protect stored credentials. It supports all major operating systems and integrates seamlessly with browsers via dedicated extensions.

The app employs AES-256 encryption and two-factor authentication, configurable through TOTP apps like

Google Authenticator. It also supports offline access, organizes credentials into folders, and includes an autofill feature. RoboForm offers a free version with unlimited password storage on a single device, with paid plans starting at €1.66 per month for multi-device synchronization.

Key features include:

- Built-in secure password generator.
- Offline functionality.
- Free version available for single-device use.
- Limited additional features, suitable for basic use.

3 Comparison of analyzed Password Managers

To provide a comprehensive overview of the password managers analyzed, we now compare their features, strengths, and limitations. This comparison aims to highlight key differences and commonalities, assisting users in selecting the most suitable solution for their needs. Table 3 summarizes the main characteristics of each password manager.

Table 3: Comparison of Password Managers

PM	**Free Version**	**Biometric Access**	**Key Features**
pCloud Pass	Yes	Yes	Lifetime subscription, customizable password generator, lacks tag-based organization
NordPass	Yes	Yes	Zero-knowledge encryption, breach monitoring, 2FA support
Dashlane	Yes	Yes	Dark web monitoring, autofill, 2FA support, corporate credential management
1Password	No	Yes	Passwordless access, encrypted document storage (1 GB), credential recovery
Kaspersky PM	14-day trial	Yes	Secure password generator, encrypted cloud storage, synchronization across devices
Keeper	30-day trial	Yes	Dark web monitoring, encrypted messaging, secure password sharing
RoboForm	Yes	Yes	Offline access, AES-256 encryption, TOTP-based 2FA, folder-based organization

3.1 Key Observations

The comparative analysis reveals the following insights:

- **Free Versions and Trials**: Most password managers offer a free version or trial; however, they are often limited to a single device. Dashlane, pCloud Pass, and NordPass stand out by providing free access without a time limit, whereas Keeper and Kaspersky Password Manager only offer time-limited trials.
- **Biometric Access**: All analyzed password managers support biometric authentication (e.g., fingerprint or facial recognition), enhancing security and ease of use.
- **Unique Features**:
  - *pCloud Pass* is notable for its lifetime subscription model.
  - *NordPass* excels with its zero-knowledge encryption.
  - *Dashlane* offers robust corporate credential management and dark web monitoring.
  - *1Password* introduces passwordless access and encrypted document storage.
  - *Kaspersky Password Manager* integrates seamlessly with its antivirus suite.
  - *Keeper* adds secure encrypted messaging as a differentiating feature.
  - *RoboForm* supports offline access and extensive device compatibility.
- **Limitations**: Many tools impose restrictions in their free versions, particularly concerning device synchronization. Additionally, features like organization through tags or improved autofill precision are inconsistently available across tools.

3.2 Conclusion

While all the analyzed password managers offer robust security measures, their suitability depends on user preferences and specific use cases. For those seeking a one-time payment model, *pCloud Pass* is an excellent choice. *NordPass* and *Dashlane* are ideal for users requiring comprehensive breach monitoring and corporate solutions. Meanwhile, *Keeper* and *1Password* stand out for their innovative features, such as encrypted messaging and passwordless access. Finally, *RoboForm* and *Kaspersky Password Manager* provide reliable, straightforward solutions with extensive compatibility and accessibility options.

4 Methodology

We decided to follow the Agile methodology in our project, employing DevOps and Scrum methods. Before discussing how we applied these methods, it's useful to define them.

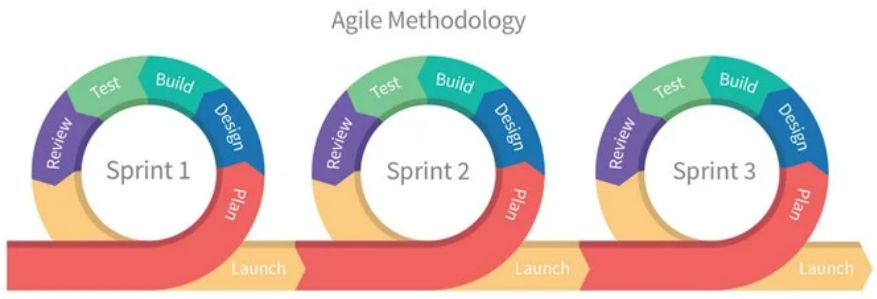


Figure 4: Visualization of how Agile Methodology works

4.1 Agile Methodology

Agile software development emphasizes collaboration and self-organizing teams, but it does not negate the presence of managers[4, 5]. These teams are cross-functional, and managers ensure that team members possess the necessary skills and the environment for success. Agile focuses on practices that foster collaboration, but also includes technical practices to address uncertainty in software development [11].

4.1.1 Advantages of Agile

Agile's most compelling feature is its flexibility, speed, and commitment to continuous improvement. Lightweight pre-project planning allows teams to adjust as needed[9], enabling creativity and freedom throughout the process. Key advantages include:

- Teams can refine and re-prioritize the backlog, allowing quick adaptation to changes.
- Development can begin without a predefined end goal, offering freedom to redirect based on progress.
- Iterative cycles help identify and resolve issues early, ensuring improved quality and collaboration.
- Continuous client feedback ensures that development aligns with user needs and leads to a sophisticated final product.

4.2 Agile Methodologies

Agile development is an umbrella term for iterative methodologies focused on adaptive planning, faster delivery, and continuous improvement. It encompasses several methods that share core values, with feedback from each iteration refining the product [29]. Agile fosters collaboration and empowerment, making it highly popular in business development. According to VersionOne's 2017 State of Agile Report, 94% of organizations practice some form of Agile, though it is often not fully implemented.

4.3 Scrum

Scrum is a widely used Agile method that focuses on iterative development with fixed roles, responsibilities, and meetings. It uses time-bound sprints (typically one or two weeks) for regular software delivery [28]. Scrum is characterized by four key ceremonies that guide the team through each sprint:

1. **Sprint Planning**: A meeting where the team decides what to include in the sprint. Once decided, no additional tasks are added.
2. **Sprint Demo**: A meeting to present the work completed during the sprint.
3. **Daily Stand-up**: A short meeting to discuss progress and roadblocks.
4. **Retrospective**: A review of the sprint to identify areas for process improvement.

Teams use a *Scrum board* to track the workflow, moving tasks through stages such as *To Do*, *In Progress*, *Code Review*, and *Done*.

4.4 Conclusion

Agile methodologies and Scrum provide structured yet flexible approaches to software development. By focusing on iterative progress, continuous feedback, team collaboration, and automation, these methods foster an environment of adaptability and continuous improvement, ensuring successful project outcomes.

4.5 Our Methodology

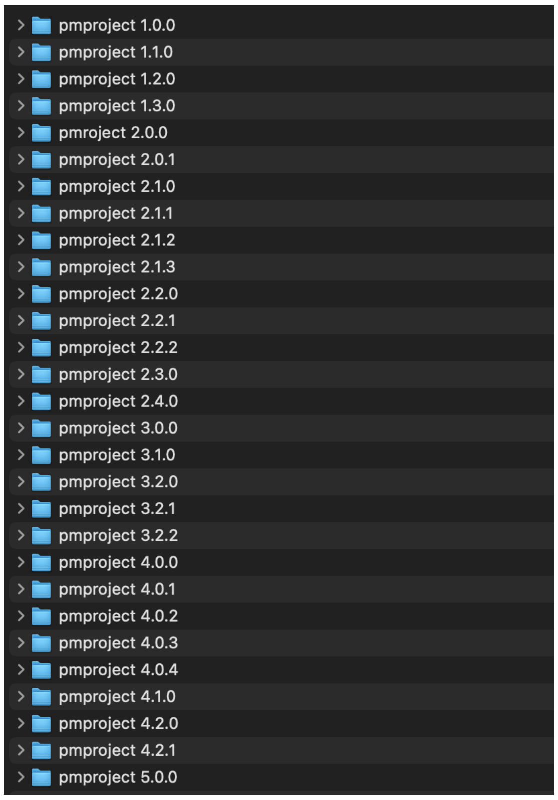
For this project, we chose to adopt a hybrid methodology that combines elements of Agile and Scrum strategies in order to maximize efficiency and collaboration. The decision to use these methodologies was driven by the need to maintain flexibility and adaptability throughout the project's lifecycle, ensuring that we could respond to challenges as they arose while keeping the momentum of the project going. One of the primary tools we employed to manage and track the progress of the project was a GitHub repository, which we utilized not only to store the code but also to manage our project backlog. This repository allowed us to effectively organize and define the tasks to be accomplished in each of the three sprints into which we divided the project. Rather than defining overly specific tasks from the outset, we intentionally kept the backlog at a high level to allow for flexibility and adjustment as the project evolved. This approach was designed to give us the freedom to adapt our plans based on new insights, feedback, or challenges encountered throughout the project. The backlog provided a general overview of the work to be done, but the actual distribution of tasks was determined more collaboratively and dynamically within the team. As the project progressed, we communicated regularly, both in person and through our group chat, to decide who would take on specific tasks. This allowed us to adjust quickly to changes in workload, skill requirements, or team availability, ensuring that our communication and teamwork remained a central focus of the project. The entire project was structured into three distinct sprints:

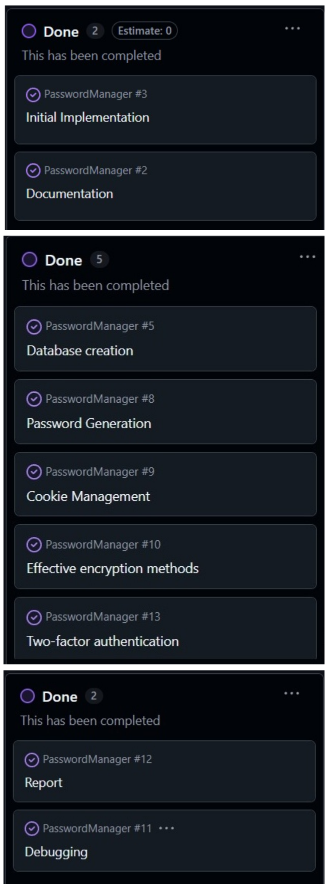
- **Sprint 1: Documentation and Initial Exploration**
  The first sprint focused primarily on gathering the necessary background information and documentation to ensure that our project was well-founded and could be implemented effectively. During this phase, we explored a variety of online libraries, resources, and tools that would be crucial to the success of our project. These resources were archived and saved for future reference, to be included in the final report's bibliography. This phase was especially important as some of our team members were engaging in a practical project for the first time, so it was essential to build a solid understanding of the relevant technologies. Alongside this documentation, we began to familiarize ourselves with the core components of the project, learning about the best practices and methodologies to apply in the development of our solution. This phase laid the groundwork for the technical work that would follow, giving us a clearer understanding of the scope of the project and how to approach its implementation.

- **Sprint 2: Implementation and Version Control**
  The second sprint was the longest and most intensive phase of the project. During this sprint, we focused on implementing the core features and functionalities that were central to the project's success. Some of the key features implemented during this phase included: *Database Creation*, *Password Generation*, *Cookie Management*, *Effective Encryption Methods*, *Two-Factor Authentication (2FA)*, *Login*, *Signup*, and *Verification of Breached Passwords*, among others. This sprint involved a significant amount of coding and problem-solving, as we worked to bring the project's ideas to life. In addition to the core development work, we also implemented a versioning strategy to help manage the updates and changes made to the project throughout the sprint. This versioning strategy was designed to ensure that we could track the progression of the project in a clear and organized manner. The versioning system we adopted followed a standard three-tier structure:

  - **Major Version**: This was used for significant changes or the addition of new features to the project. Whenever a substantial change was made, the major version number would be incremented.
  - **Minor Version**: This version was used to track smaller, incremental improvements or the addition of minor features to the project. These updates were less impactful than major changes but still represented progress.
  - **Patch Version**: This version was used for bug fixes or minor corrections. Patch updates typically followed code reviews, where issues identified by a team member were addressed and resolved.

  This versioning approach proved extremely useful, as it provided a clear record of the project's evolution and allowed us to easily track changes over time. Additionally, given the diversity of operating systems used within our team, it was important to ensure that the project remained portable and could run seamlessly across different platforms. The versioning strategy helped us maintain compatibility and ensured that each team member's work could be integrated smoothly regardless of the operating system they were using.


pmproject 1.0.0
pmproject 1.1.0
pmproject 1.2.0
pmproject 1.3.0
pmroject 2.0.0
pmproject 2.0.1
pmproject 2.1.0
pmproject 2.1.1
pmproject 2.1.2
pmproject 2.1.3
pmproject 2.2.0
pmproject 2.2.1
pmproject 2.2.2
pmproject 2.3.0
pmproject 2.4.0
pmproject 3.0.0
pmproject 3.1.0
pmproject 3.2.0
pmproject 3.2.1
pmproject 3.2.2
pmproject 4.0.0
pmproject 4.0.1
pmproject 4.0.2
pmproject 4.0.3
pmproject 4.0.4
pmproject 4.1.0
pmproject 4.2.0
pmproject 4.2.1
pmproject 5.0.0


Done 2 Estimate: 0
This has been completed
PasswordManager #3
Initial Implementation
PasswordManager #2
Documentation

Done 5
This has been completed
PasswordManager #5
Database creation
PasswordManager #8
Password Generation
PasswordManager #9
Cookie Management
PasswordManager #10
Effective encryption methods
PasswordManager #13
Two-factor authentication

Done 2
This has been completed
PasswordManager #12
Report
PasswordManager #11
Debugging

Figure 5: On the left, the versioning of our project, each folder is a version update, on the right the visualization of the three sprints starting from the first on the top going to the bottom at the third sprint

- **Sprint 3: Final Review and Report Writing**
  The third and final sprint was focused on two main objectives: finalizing the project code and preparing the final report. Throughout the development process, we had conducted regular code reviews after each update, which made the debugging phase relatively straightforward and efficient. By reviewing the code continuously, we were able to identify and resolve issues early on, preventing any major problems from arising later in the project. As a result, the debugging process did not require as much time in the final sprint, allowing us to focus more on preparing the final documentation. The final sprint was dedicated to drafting the report, which would provide a comprehensive overview of the entire project. In this report, we described the methodology we followed, the challenges we faced, the solutions we implemented, and the lessons we learned along the way. We also included a detailed literature review, which highlighted the research and resources that informed our project decisions. Writing this report was an essential part of the project, as it allowed us to reflect on our work and

present our findings in a clear and organized manner.

In conclusion, the hybrid Agile and Scrum methodology we adopted allowed us to maintain flexibility while ensuring that we met our project goals in a structured and efficient manner. By breaking the project into manageable sprints and continuously reviewing our progress, we were able to keep the project on track and deliver a high-quality final product. The collaborative approach fostered by these methodologies enabled us to work effectively as a team and adapt to challenges as they arose, resulting in a successful project outcome.

5 Development and Features

After all the high-level considerations of our project, we start to analyse all the main problems that we have, firstly discussing about the workflow of our application, and then describing all the documentation of the project. The application developed by our team combines a frontend for user interaction and a backend for data processing, storage, and security operations. The main functionalities include user authentication (with 2FA), password encryption, secure storage, and retrieval, also all this is easly suitable on any kind of device.

5.1 General Architecture

The application follows a Client-Server architecture:

- **Frontend**: Built using React (JSX), the frontend components manage:
  - User interactions through forms (Login, Register, Setup 2FA);
  - Navigation and UI rendering (e.g., Sidebar, HomePage);
  - API calls to the backend for authentication, storing/retrieving passwords, and enabling 2FA.
- **Backend**: Developed in Python (Flask), the backend handles:
  - User Registration & Login (with password hashing);
  - Two-Factor Authentication (QR code generation and OTP validation);
  - Password encryption and storage using a master encryption key;
  - Note storage;
  - Password management operations like retrieval, modification, and deletion.
  - Note management operations like retrievial, modification, and deletion.
- **Databese**: Stores user information and ecnrypted passwords through models User and SiteData.

5.2 Workflow

Now let's move on seeing the workflow of our application, that goes as it follows, giving the possibility to the user to get specialized as we can see in the 6

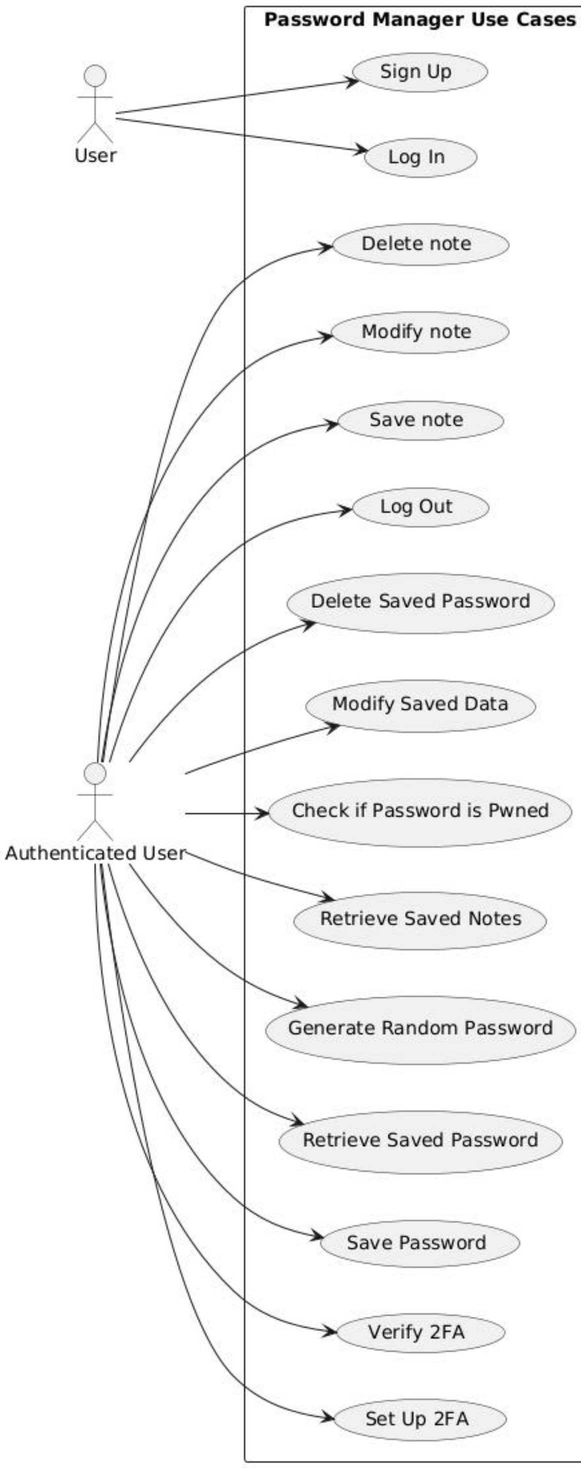


Figure 6: The use case diagram of our password manager system, showing the interactions between the system and its users.

5.2.1 User Registration

1. A new User submits their *username*, *email*, and *password* through the `Register.jsx` component.
2. The frontend sends a `POST /signup` request to the backend.
3. The backend:
    - Validates user input (password strength, uniqueness of email/username).
    - Hashes the password using the `hash_password` function inside `utils.py`.
    - Stores the user information into the database through the `User` model.
4. Backend responds with:
    - **201 Created**: If user registration is successful.
    - **400 Bad Request**: If required fields are missing or the password is weak or if username or email is already taken.

5.2.2 User Login

1. The user enters their credentials using the `Login.jsx` component.
2. The frontend sends a `POST /login` request.
3. The backend verifies the credentials and also if 2FA is enabled:
    - Verifies credentials using `check_password` in `utils.py`
    - A prompt is sent for OTP verification via `POST /verify-2fa`.
    - If verified, the user is authenticated and the frontend redirect to `HomePage.jsx` on success.

5.2.3 Enabling Two-Factor Authetnication (2FA)

1. User click **Setup 2FA** on `SetupPage.jsx`;
2. Inside the Backend are manage these following things:
    - Generates a QR Code using `utils.py:get_b64encoded_qr_image`;
    - Sends the QR code image and secret key to the forntend;
3. User scans the QR code using an authenticator app (e.g., Google Authenticator);
4. The OTP is verified via `POST /verify-2fa`

5.2.4 Managing Passwords

1. Saving Passwords:
   - User inputs site details via the form in `HomePage.jsx`;
   - Frontend sends a `POST /save-password` request;
   - The backend encrypts the password using `encrypt_password` and stores it in the database unde `SiteData`;
   - The backend responds with success or error messages;
2. Retrieving Passwords:
   - User navigates to the list of saved passwords;
   - Frontend calls `GET /get-saved-passwords`;
   - Backend decrypts passwords using `decrypt_password`;
   - The backend also returns the list of passwords.
3. Modifying/Deleting Passwords:
   - Similar `POST` requests are sent for modifying (`/modify-password-data` or deleting entries (`/delete-password`

5.2.5 Security Enhancements

1. Password validation is performed using:
   - Password strength rules (`control_password`);
   - Weak password detection (`search_weak_password` with `rockyou.txt`).
2. Password exposure is checked using the **Have I Been Pwned API** `check_pwned`.

5.2.6 Password Management

- **Save Passwords**: Encrypted site passwords are stored via `POST /save-password`.
- **Retrieve Passwords**: Frontend requests `GET /get-saved-passwords`, and the backend decrypts and sends the data.
- **Modify/Delete Passwords**: Requests are handled through `POST` routes for modifications.

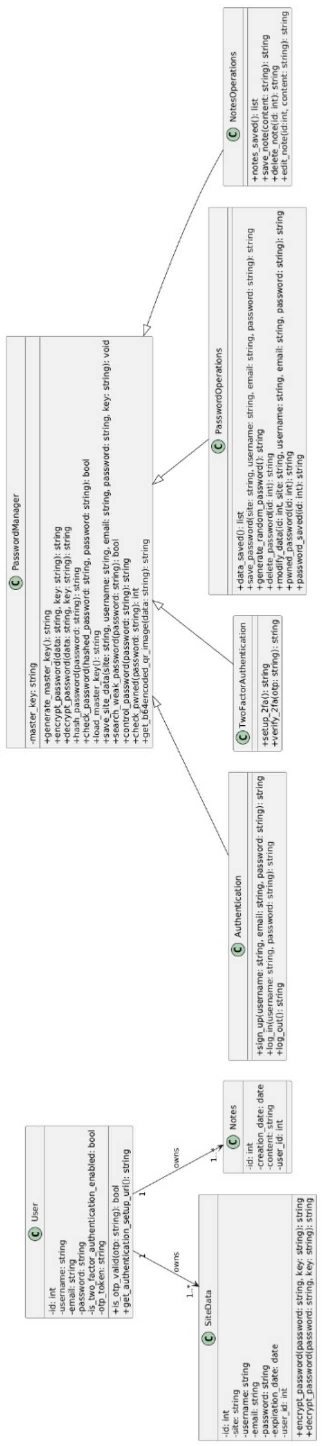

User
-id: int
-username: string
-email: string
-password: string
-is_two_factor_authentication_enabled: bool
-otp_token: string
+is_otp_valid(otp: string): bool
+get_authentication_setup_uri(): string
1
owns
1..*
1
owns
1..*
SiteData
-id: int
-site: string
-username: string
-email: string
-password: string
-expiration_date: date
-user_id: int
+encrypt_password(password: string, key: string): string
+decrypt_password(password: string, key: string): string
Notes
-id: int
-creation_date: date
-content: string
-user_id: int
PasswordManager
-master_key: string
+generate_master_key(): string
+encrypt_password(data: string, key: string): string
+decrypt_password(data: string, key: string): string
+hash_password(password: string): string
+check_password(hashed_password: string, password: string): bool
+load_master_key(): string
+save_site_data(site: string, username: string, email: string, password: string, key: string): void
+search_weak_password(password: string): bool
+control_password(password: string): string
+check_pwned(password: string): int
+get_b64encoded_qr_image(data: string): string
Authentication
+sign_up(username: string, email: string, password: string): string
+log_in(username: string, password: string): string
+log_out(): string
TwoFactorAuthentication
+setup_2fa(): string
+verify_2fa(otp: string): string
PasswordOperations
+data_saved(): list
+save_password(site: string, username: string, email: string, password: string): string
+generate_random_password(): string
+delete_password(id: int): string
+modify_data(id: int, site: string, username: string, email: string, password: string): string
+pwned_password(id: int): string
+password_saved(id: int): string
NotesOperations
+notes_saved(): list
+save_note(content: string): string
+delete_note(id: int): string
+edit_note(id:int, content: string): string

Figure 7: This UML class diagram represent the password manager system. It shows relationships between classes, attributes, and methods for components like User, PasswordManager, SiteData, Notes, Authentication, TwoFactorAuthentication, PasswordOperations, and NotesOperations.

6 Conclusion and Future Improvements

In this project, we have successfully developed a robust and secure web application designed to manage user authentication and password storage efficiently. By implementing a clear separation between the frontend and backend components, the application ensures maintainability and scalability, utilizing modern frameworks such as React for the frontend and Flask for the backend. Key functionalities include user registration, login with Two-Factor Authentication (2FA), and encrypted password management, ensuring that user credentials are securely stored and accessed. The application's architecture lays the groundwork for an extensible platform that can evolve alongside emerging security standards and user demands. Looking ahead, our team has identified a series of future enhancements aimed at improving both the security and usability of the system. Firstly, we plan to integrate SSL/TLS protocols to ensure secure communication between the client and server, safeguarding data transmissions against interception and unauthorized access. This addition will align the application with industry-standard encryption protocols, further strengthening user trust. Secondly, we aim to deploy the web application to a live environment, which will facilitate the development of a browser extension. The extension will enable autofill features, allowing users to seamlessly retrieve and input their stored credentials on websites. This enhancement will improve user experience by reducing the friction associated with manual password management. Moreover, we intend to expand the existing 2FA implementation by introducing additional verification methods, such as using a telephone number or email. These methods will provide users with greater flexibility in securing their accounts, catering to varying levels of security preferences and accessibility. Finally, a critical feature in our roadmap is the possibility of sharing passwords securely among trusted individuals or team members. This functionality will allow collaborative password management while maintaining robust encryption protocols to prevent unauthorized access during the sharing process. In conclusion, our team remains committed to enhancing this application by adopting advanced security measures, improving usability, and addressing evolving user needs. By implementing these features, we aim to deliver a comprehensive, user-friendly, and secure platform that sets a new standard for password management and authentication systems. This continued development will not only increase the application's versatility but also ensure it remains a reliable solution for individual users and teams alike.

References

[1] About kaspersky password manager. https://support.kaspersky.com/kpm-for-windows/24.2.

[2] Top 200 most common passwords. https://nordpass.com/most-common-passwords-list/.

[3] 1password Team. 1password events reporting | elastic integrations | elastic. https://www.elastic.co/guide/en/integrations/current/integration-1password.html.

[4] Agile Alliance. Agile 101, 2018. https://www.agilealliance.org/agile101/.

[5] Agile Manifesto. Principles behind the agile manifesto, 2001. https://agilemanifesto.org/principles.html.

[6] Sabrina Amft, Sandra Höltervennhof, Nicolas Huaman, Yasemin Acar, and Sascha Fahl. "would you give the same priority to the bank and a game? i do not!": Exploring credential management strategies and obstacles during password manager setup. In *Proceedings of the 2023 Conference*, pages 171–190, 2023.

[7] O. Connelly. *WordPress 3 Ultimate Security*. Packt Publishing Ltd, 2011.

[8] Brett Cruz. 2024 password manager industry report and statistics, November 14, 2024. https://www.security.org/digital-safety/password-manager-annual-report/.

[9] DevBatch. What is it they call agile software development?, 2018. https://devbatch.com/what-they-call-agile-software-development/.

[10] Kate Eby. Comprehensive guide to the agile manifesto, 2016. https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto.

[11] Kate Eby. What is the difference? agile vs. scrum vs. waterfall vs. kanban, 2017. https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban.

[12] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, page 657. ACM Press, 2007.

[13] E. Grosse and M. Upadhyay. Authentication at scale. *IEEE Security & Privacy*, 11(1):15–22, jan 2013.

[14] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. Technical report, In Proc. of NSPW, 2009.

[15] Adryana Hutchinson, Collins W. Munyendo, Adam J Aviv, and Peter Mayer. An analysis of password managers' password checkup tools. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–7. ACM.

[16] A. Huth, M. Orlando, and L. Pesante. Password security, protection, and management. Technical report, United States Computer Emergency Readiness Team, 2012.

[17] Zhiwei Li, Warren He, Devdatta Akhawa, and Dawn Song. *The Emperor's New Password Manager: Security Analysis of Web-based Password Managers:*.

[18] Jory MacKay. The ultimate guide to implementing agile project management and scrum, 2018. https://plan.io/blog/what-is-agile-project-management/.

[19] Peter Mayer, Collins W. Munyendo, Michelle L. Mazurek, and Adam J. Aviv. Why users (don't) use password managers at a large educational institution. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1849–1866, Boston, MA, August 2022. USENIX Association.

[20] Hrithik Padalia, Hitesh Patel, Amarjit Deshmukh, Mahadev Patil, Ajay Kumar, and Nripesh Kumar Nrip. A study on password manager: Users' perspective. In *2023 International Conference on Computational Intelligence for Information, Security and Communication Applications (CIISCA)*, pages 72–75. IEEE.

[21] pCloud Team. pCloud - f.a.q generali. `https://www.pcloud.com/it/help/general-help-center/`.

[22] Sarah Pearman, Shikun Aerin Zhang, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Why people (don't) use password managers effectively. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 319–338, Santa Clara, CA, August 2019. USENIX Association.

[23] M. Rochkind. Security, forms, and error handling. In *Expert PHP and MySQL*, pages 191–247. Springer, 2013.

[24] DashLane Team. Developer resources. `https://ripleyprd.wpengine.com/developers/`.

[25] Keeper Team. Keeper docs portal | keeper documentation. `https://docs.keeper.io/en`.

[26] NordPass Team. Activity logs API. `https://support.nordpass.com/hc/en-us/articles/23485641117329-Activity-Logs-API`.

[27] RoboForm Team. RoboForm password manager - user manual for windows software. `https://www.roboform.com/manual`.

[28] Madhuri Thakur. Scrum process. `https://www.educba.com/scrum-process/`.

[29] Devharsh Trivedi. Agile methodologies. *ResearchGate*, 12:91–100, 2021.

[30] Samantha Wolhuter. Waterfall vs. agile vs. scrum: What is the difference?, 2020. `https://www.wearebrain.com/blog/software-development/waterfall-vs-agile-vs-scrum/`.