**Password Manager with Python**

Created by: Mario Getaw, Jarrett Wilson, & Joshua Moore

**COMP-460**

**Senior Capstone 2025**

**Computer Science Department**

Dr. Sunday Ngwobia

**Abstract**

This project presents the design and development of a cross-platform password manager application created using Python. The goal was to implement core security features such as local encryption, user authentication, and secure password storage while maintaining simplicity and accessibility. The application includes a graphical user interface (GUI) and integrates two-factor authentication (2FA) to enhance account security. Notably, instead of relying on a dedicated backend or database, the app utilizes Google Drive for cloud-based password backup and recovery, allowing users to retrieve credentials across devices. By combining cryptographic techniques with a cloud storage API, the system demonstrates how secure password management can be achieved within a lightweight desktop application. This project serves as a proof of concept that explores user-focused security features and cross-device accessibility, while acknowledging the limitations of a prototype environment in contrast to industry-grade password managers. The final product delivers a working model that balances security, usability, and practical design within the scope of an undergraduate capstone.

**Introduction**

In the digital age, individuals rely on a growing number of online accounts, each requiring secure and unique login credentials. With the rise in data breaches and cyber threats, managing these credentials has become both a necessity and a challenge. Password managers have emerged as a practical solution, offering users a secure and centralized way to store, retrieve, and manage passwords. This capstone project explores the development of a lightweight, standalone password manager built entirely in Python.

The goal of the project was to create an application that supports essential password management functionality—such as secure storage, encryption, user authentication, and cloud-based backup—while remaining accessible to non-technical users. To enhance security and user trust, the application implements local encryption using the Fernet module from the cryptography library and integrates two-factor authentication (2FA) via time-based one-time passwords (TOTP). For remote data access and backup, the system uniquely leverages Google Drive instead of a conventional database, enabling cross-device synchronization without requiring complex backend infrastructure.

By focusing on core security principles and usability, this project offers a practical example of how password management solutions can be built from the ground up. It is intended as a learning tool and prototype, emphasizing the feasibility of developing secure and functional applications with limited resources and within an academic environment.

**Literature Review**

**System/Hardware Requirements**

The password manager has minimal requirements, as it works on any computer that can install the Python library. To use the Cloud feature, you first need to set up a Google account with Google Drive. When storing a password, it will direct you to Google, where you can select your account or create one if you don't already have one.

**Software Requirements**

The software must allow users to create an account to save passwords and to log in later. Along with this functionality, the software must be compatible with a 2 factor authentication app to allow for added protection. The software must be able to encrypt the user’s data to allow for a level of protection for the passwords. The system must also have a user-friendly GUI. Furthermore, the application should work quickly and smoothly to allow for a seamless user experience.

**System/Hardware Requirements**

The password manager requires the system to be able to install Python libraries to run locally on the device. To use the Cloud feature, you first need to set up a Google account with Google Drive. When storing a password, it will direct you to Google, where you can select your account or create one if you don't already have one. Therefore, for cloud services, an internet connection is also required.

**User Requirements**

The users of the program would require a simple GUI, so that they can feel confident navigating the app. They would also require the ability to edit and retrieve existing passwords, as well as generate new secure passwords on the app. Users would also expect some level of encryption for their stored passwords and login information, so they can feel confident their data will not be breached. Finally, users would expect there to be a cloud backup of their data in case they wanted to log in across devices, or if their local data was corrupted or deleted for whatever reason.

**Project Performance, Uniqueness, and Similarity to Others**

Our app performs as intended. A simple password manager app was created as a proof of concept to demonstrate the fundamentals of secure credential storage and user authentication. The primary goal was to explore encryption techniques and user interface design in a controlled development environment. It served as a learning tool or prototype rather than a fully deployed product, lacking the rigorous security auditing, compliance certifications, and scalability features necessary for real-world use. Because it was not intended to handle sensitive user data outside of testing scenarios, it remained a conceptual project without exposure to live users or production environments. Our application takes a unique approach by utilizing Google Drive for password storage instead of a traditional database. This decision was driven by the goal of simplifying implementation while ensuring cross-device accessibility. By leveraging Google Drive’s cloud infrastructure, we enabled users to securely access their stored passwords from any device with an internet connection, streamlining the user experience without the complexity of managing a custom backend database. This method also allowed for easier integration within the scope of our project, aligning with our focus on accessibility and ease of use.

Before beginning development, we conducted research on leading password managers such as NordPass and 1Password to gain insight into industry standards and functionality. Through this comparison, we identified key features that make these platforms robust and user-friendly, including cloud-based database systems and seamless browser integration. However, given the scope of our project and the limited time and resources available, implementing such advanced capabilities was not feasible. Instead, we focused on achievable goals, such as designing a clean, intuitive user interface and enabling cross-device accessibility. While our app does not yet include the full range of enterprise-level security features, it reflects a thoughtful balance between functionality and practicality within the constraints of our development environment.

**Class Diagram –**

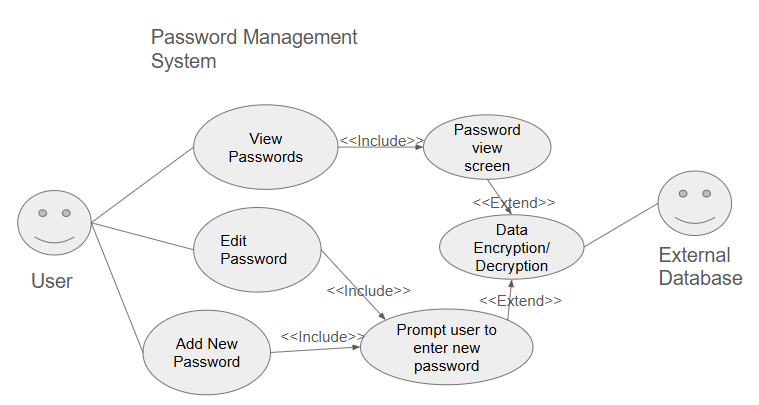
A screenshot of a computer

AI-generated content may be incorrect.

**Use Case Diagrams –**

A diagram of a sign in system

AI-generated content may be incorrect.



**Architecture –**

A diagram of a computer system

AI-generated content may be incorrect.

**Ethical/ Legal and Security Concerns**

While our app adheres to ethical and legal standards, it does present notable security limitations. It was not developed to provide enterprise-level or high-security protection. Although passwords are stored using basic encryption methods, the implementation lacks the robustness needed to prevent decryption by individuals with advanced knowledge of password-cracking techniques. As such, we do not recommend relying on this application for safeguarding sensitive credentials. Instead, it is intended as a simplified demonstration of how a password manager functions, serving educational or conceptual purposes rather than secure, real-world usage.

**Project Performance, Uniqueness, and Similarity to Others**

Our app performs as intended. A simple password manager app was created as a proof of concept to demonstrate the fundamentals of secure credential storage and user authentication. The primary goal was to explore encryption techniques and user interface design in a controlled development environment. It served as a learning tool or prototype rather than a fully deployed product, lacking the rigorous security auditing, compliance certifications, and scalability features necessary for real-world use. Because it was not intended to handle sensitive user data outside of testing scenarios, it remained a conceptual project without exposure to live users or production environments. Our application takes a unique approach by utilizing Google Drive for password storage instead of a traditional database. This decision was driven by the goal of simplifying implementation while ensuring cross-device accessibility. By leveraging Google Drive’s cloud infrastructure, we enabled users to securely access their stored passwords from any device with an internet connection, streamlining the user experience without the complexity of managing a custom backend database. This method also allowed for easier integration within the scope of our project, aligning with our focus on accessibility and ease of use.

Before beginning development, we conducted research on leading password managers such as NordPass and 1Password to gain insight into industry standards and functionality. Through this comparison, we identified key features that make these platforms robust and user-friendly, including cloud-based database systems and seamless browser integration. However, given the scope of our project and the limited time and resources available, implementing such advanced capabilities was not feasible. Instead, we focused on achievable goals, such as designing a clean, intuitive user interface and enabling cross-device accessibility. While our app does not yet include the full range of enterprise-level security features, it reflects a thoughtful balance between functionality and practicality within the constraints of our development environment.

**Conclusion**

**References**