Industrial Internship Report on

”Password Manager”

Prepared by

[Arnav Gupta]

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was (Tell about ur Project) This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

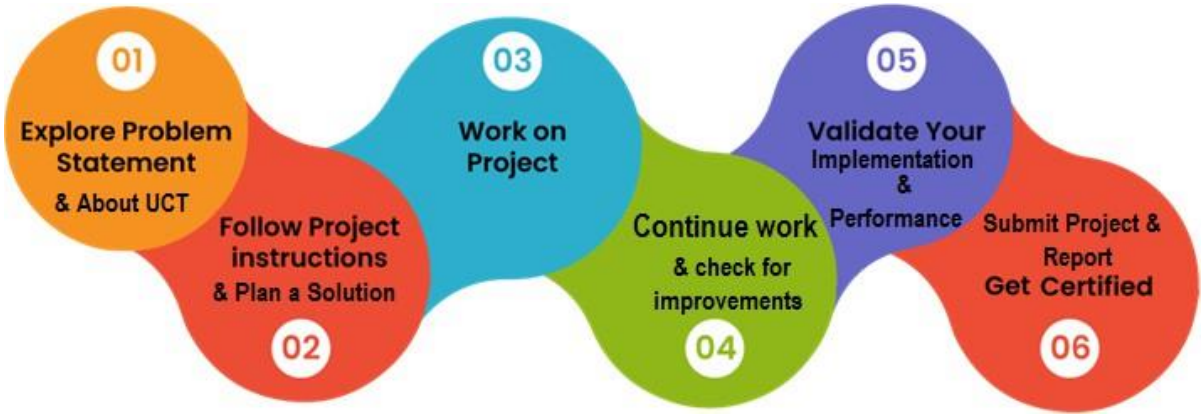
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end etc.

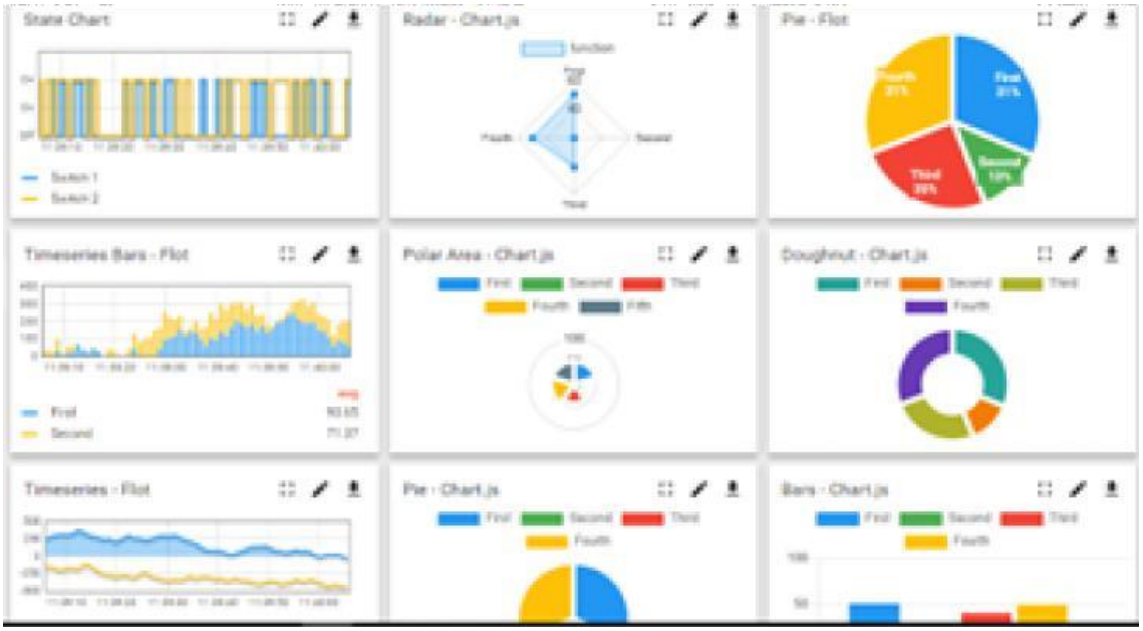
i. UCT IoT Platform (UCT Insight)

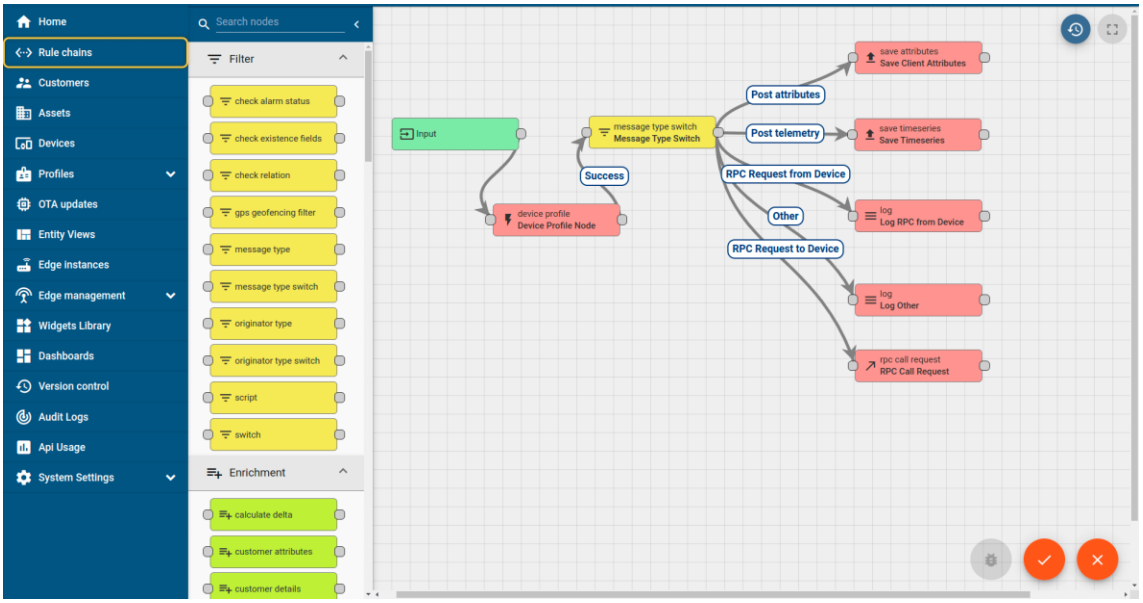
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. **Smart Factory Platform** (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

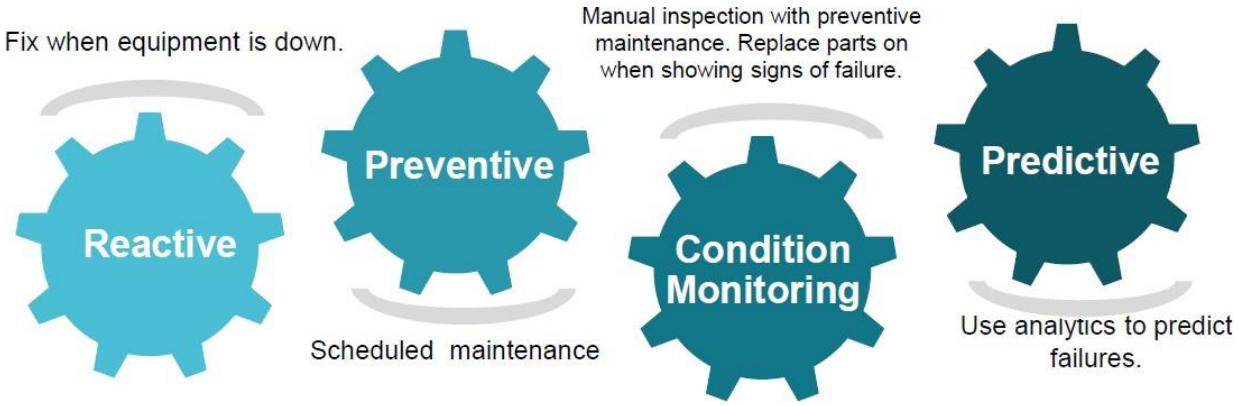
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

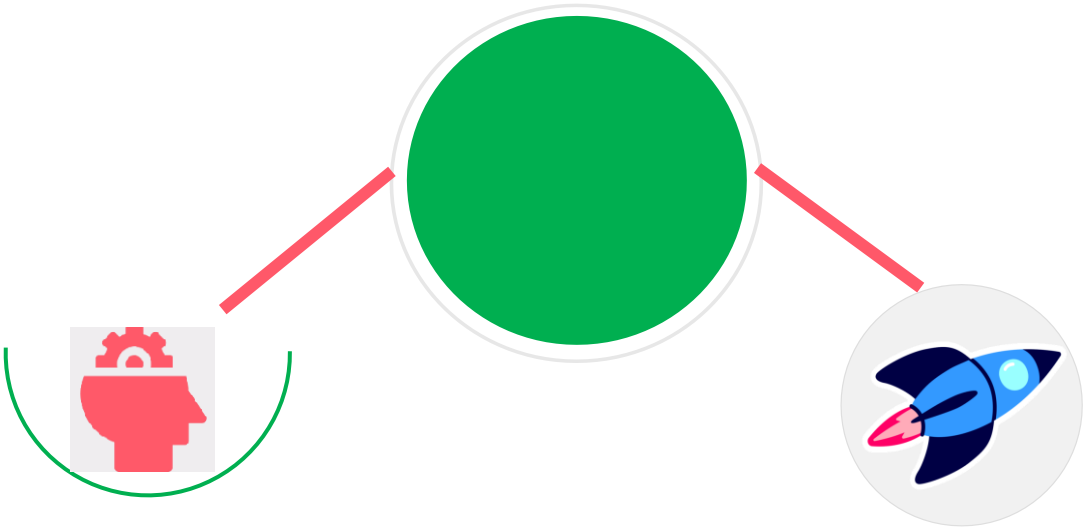
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

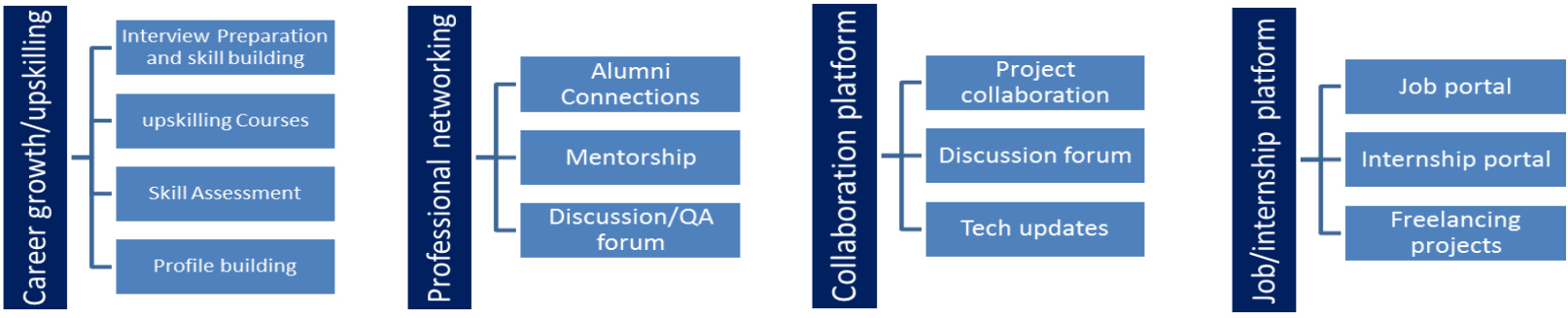
USC is a career development platform that delivers personalized executive coaching in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] Python Software Foundation. Python 3 Documentation. Retrieved from https://docs.python.org/3

[2] Cryptography Developers. Cryptography 41.0.3 Documentation. Retrieved from https://cryptography.io/en/latest/

[3] Schneier, B. (2015). Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons.

2.6 Glossary

Terms	Acronym
Advanced Encryption Standard	AES
Application Programming Interface	API
Data Encryption Standard	DES

Graphical User Interface		GUI
JavaScript Object Notation	JSON	
Authentication	AUTH	
Command Line Interface	CLI	
Two-Factor-Authorization	2FA	
Public Key Infrastructure	PKI	
Transport Layer Security	TLS	
Access Control List	ACL	

3 Problem Statement

Problem Statement

In the assigned problem statement, the major challenge is the **secure management of digital passwords** in an environment where users often struggle to remember multiple complex credentials. Relying on weak or reused passwords exposes individuals to risks such as data breaches, identity theft, and unauthorized access. Existing solutions, while effective, are often either paid, overly complex, or dependent on third-party cloud services that may raise privacy concerns.

Therefore, the problem is to design and implement a **lightweight, local, and user-friendly password manager** that:

- Protects user data using strong encryption techniques.
- Secures access through a master password.
- Provides a simple interface to store, retrieve, and manage credentials.
- Ensures passwords remain confidential by avoiding plaintext storage.

Additionally, the project aims to provide a **practical demonstration of secure software design principles** by integrating authentication, encryption, and safe storage within a modular Python application. This not only addresses real-world password management challenges but also serves as an educational tool to understand how cryptography and secure coding practices can be applied in everyday applications.

3.1 Code submission (Github link)

https://github.com/Arnav-gupta10/upskillcampus/blob/main/password_manager.py

3.2 Report submission (Github link)

https://github.com/Arnavgupta10/upskillcampus/blob/main/PasswordManager_Arnav_USC_UCT.pdf

4 Existing and Proposed solution

Existing Solution

Several popular password managers are already available in the market, such as **LastPass, Dashlane, 1Password, and Bitwarden**. These solutions provide strong encryption, password generation, cross-device synchronization, and browser integration. However, they also present **limitations**:

- Many rely heavily on **cloud storage**, which may raise **privacy and trust concerns**.
- Some features are **locked behind premium subscriptions**, limiting accessibility for all users.
- Complexity in setup and usage may discourage **non-technical users** from adopting them.
- Dependence on third-party services introduces risks if the provider suffers a breach, outage, or policy change.

Proposed Solution

The proposed solution is a **lightweight, local-first password manager** built in **Python**, designed to be simple, secure, and easy to use. Key features include:

- Secure storage of passwords using **Fernet symmetric encryption**.
- Master password authentication, hashed using **SHA-256**, to protect access.
- A **command-line interface (CLI)** that enables users to add, retrieve, and view stored credentials.
- Local JSON file storage, ensuring that sensitive data never leaves the user's device.
- Automatic copying of retrieved passwords to the clipboard for convenience.

Value Addition

This project adds value by offering a **free, transparent, and educational alternative** to commercial password managers. Unlike cloud-dependent tools, this solution prioritizes **data privacy and user control** by keeping all information stored locally. Furthermore, the modular Python design makes it easy to extend with new features such as password strength checks, timeout-based auto-lock, or even a graphical interface. The project not only solves a **practical security challenge** but also serves as a **learning resource** to demonstrate how encryption and secure coding practices can be implemented in real-world applications.

4 Proposed Design/ Model

1. Purpose & Core Functionality

The password manager enables users to:

- Register and **log in** using a username and master password.
- **Add** new credentials (passwords).
- **Retrieve** saved passwords.
- **View** the list of saved websites.
- **Automatically copy** retrieved passwords to the clipboard for convenience.

2. Components & Data Flow

a. User Authentication (Registration & Login)

- **Registration**: The user sets a username and master password. The master password is **hashed** using SHA-256 via hashlib, and stored in a JSON file (user_data.json).
- **Login**: The entered username and password are hashed and compared against stored credentials. Successful match grants access; otherwise, the program exits.

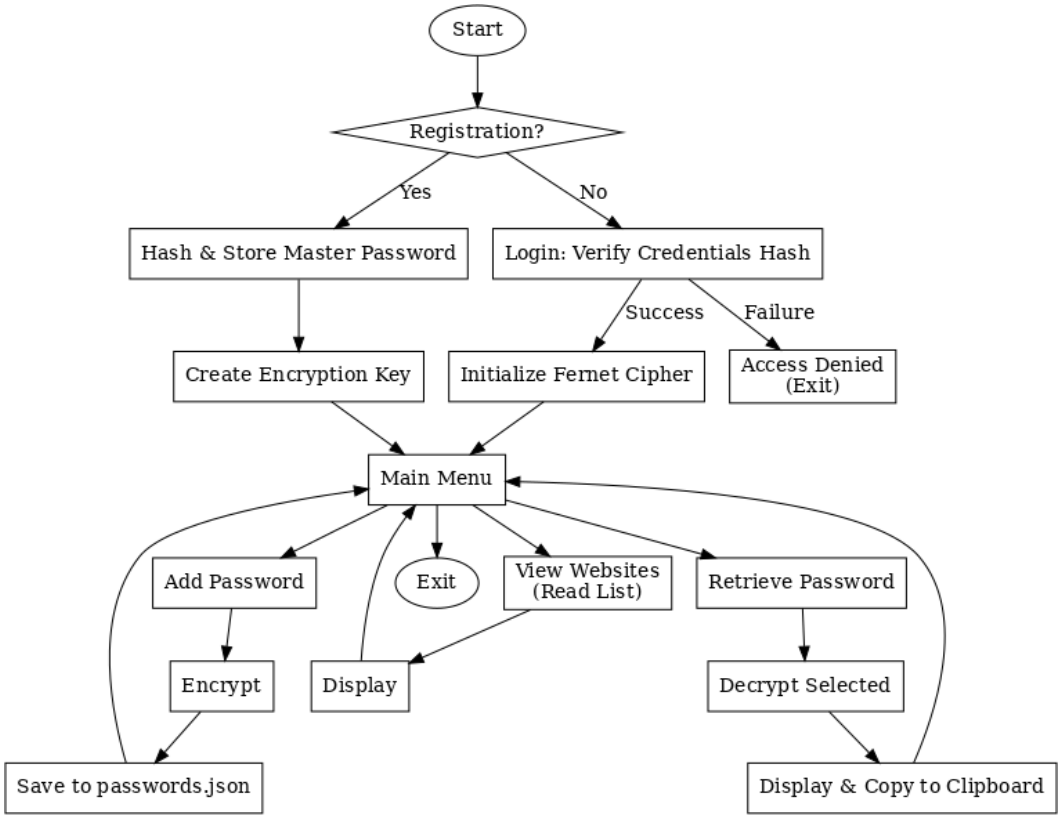
b. Key Management & Encryption Layer

- The app generates a symmetric key using Fernet.generate_key() on the first run, storing it in a file (encryption_key.key).
- A Fernet cipher is then initialized with this key to encrypt and decrypt passwords.

c. Credential Storage & Retrieval

- **Adding Passwords**: The user provides a website and password. The password is encrypted via the Fernet cipher and stored in passwords.json.
- **Viewing Websites**: The app reads the JSON file to list saved websites.
- **Retrieving Passwords**: When a website is selected, its encrypted password is decrypted and displayed, then copied to the clipboard using pyperclip.

3. Control Flow Diagram



4. Security Design Considerations

- **Password hashing**: The master password is stored securely as a SHA-256 hash—making the plaintext unrecoverable.
- **Encryption**: Uses cryptography.Fernet, which provides AES-based symmetric encryption with integrity checks.
- **Clipboard integration**: Secure user experience by copying passwords directly to clipboard via pyperclip.

5. Why This Design Works

This Project presents a **modular yet concise architecture** perfect for a beginner-friendly yet functional password manager:

- **Modular Components**: Clear separation between authentication, encryption, storage, and UI.
- **Secure Defaults**: Uses hashing and Fernet encryption—both robust and suitable for personal use.
- **Usable Interface**: CLI-driven operations with convenient clipboard integration.
- **Data Persistence**: JSON-based storage allows easy inspection and debugging.

4.1 High Level Diagram (if applicable)

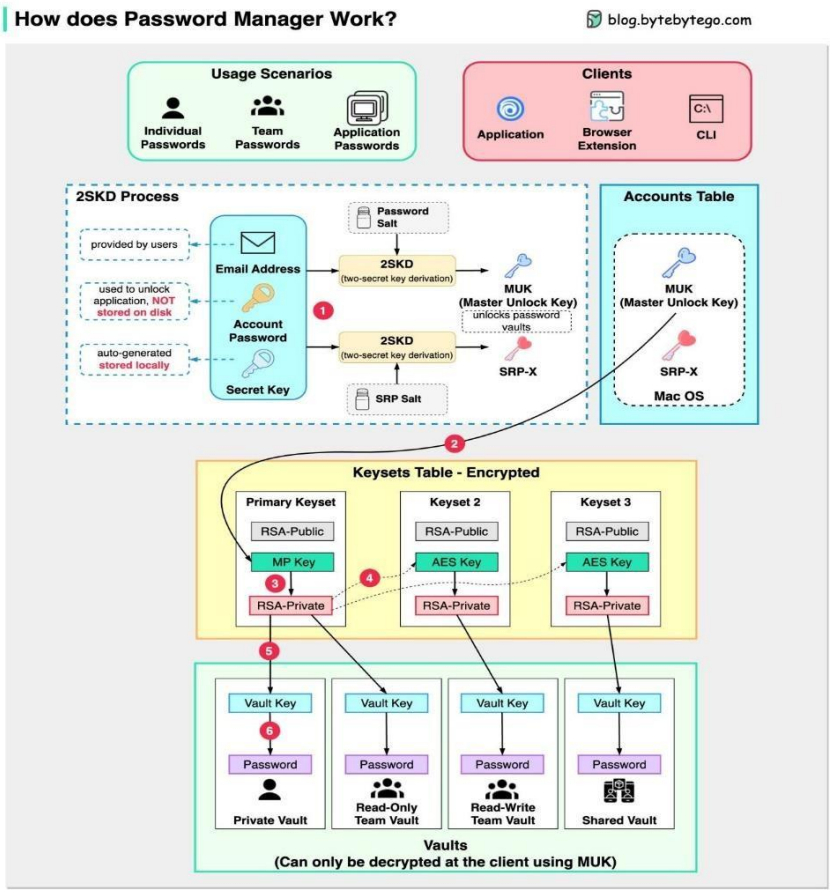


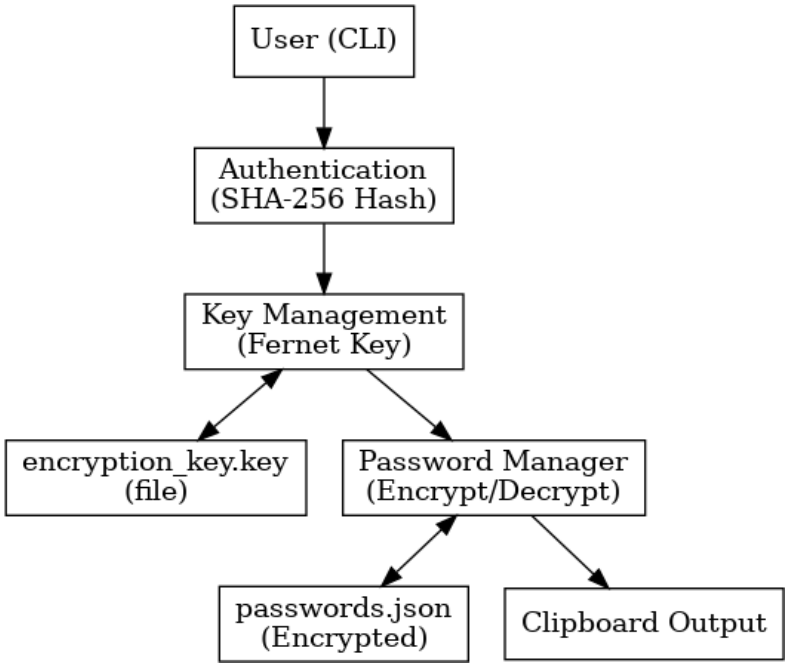
Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

4.2 Interfaces

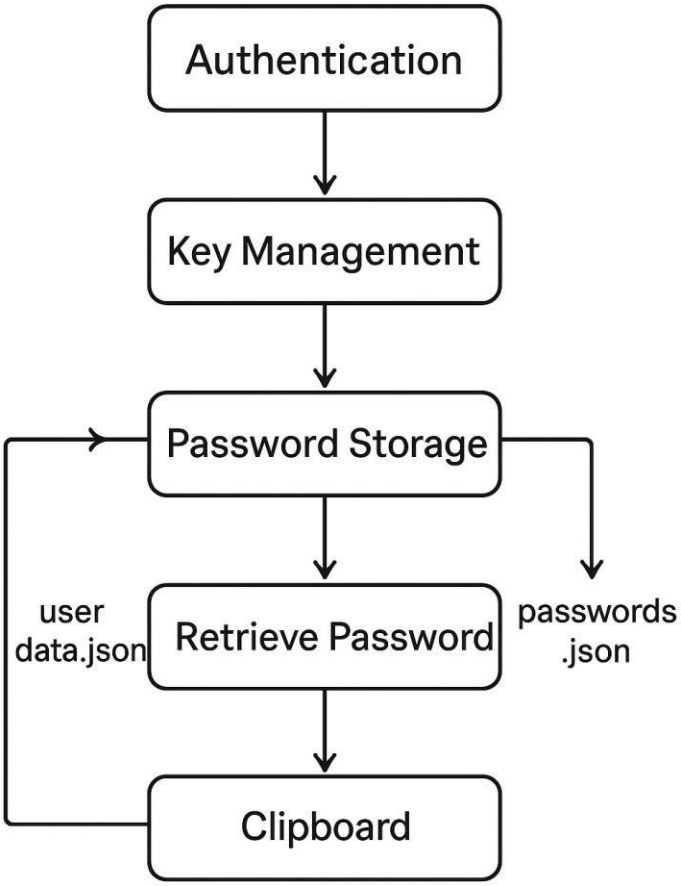
1. Interfaces in this project:

- **User Interface (CLI)** → Text-based menu where users log in, add, and retrieve credentials.
- **File Interfaces** →
    - user_data.json (for storing master password hash).
    - passwords.json (for storing encrypted credentials).
    - encryption_key.key (for encryption key storage).
- **System Interfaces** →
    - hashlib for SHA-256 password hashing.
    - cryptography.fernet for encryption/decryption.
    - pyperclip for clipboard management.

2. Block Diagram:



3. Data Flow Diagram



5 Performance Test

Performance testing is a crucial aspect of this project, as it demonstrates the applicability of the password manager beyond academic use and its relevance for real-world industrial adoption.

5.1 Identified Constraints

During the design and implementation of the password manager, the following constraints were identified:

1. **Memory Usage** – The application stores user credentials in encrypted JSON files. Large-scale data storage can cause memory overhead.
2. **Processing Speed (MIPS)** – Encryption and decryption operations (using cryptography libraries) must be efficient, as frequent password retrieval should not introduce noticeable delays.
3. **Accuracy and Reliability** – The system must always return the correct decrypted password without data corruption.
4. **Security Durability** – Encrypted data and encryption keys must remain secure over long-term use.
5. **Power Consumption (Indirect)** – Although not critical for desktops, password managers may run on laptops or mobile devices, where energy efficiency of cryptographic operations matters.

5.2 Design Considerations to Handle Constraints

- **Memory Optimization**: Instead of keeping all data in active memory, credentials are only loaded on demand from passwords.json.
- **Efficient Encryption**: Symmetric encryption (Fernet from Python Cryptography package) was used, which balances strong security with relatively fast computation.
- **Accuracy Assurance**: Data integrity is maintained using hashing and encrypted JSON files to ensure that credentials are not altered.
- **Durability**: The encryption key is stored separately (encryption_key.key) to ensure long-term security of password storage.
- **Power Efficiency**: Lightweight cryptographic operations ensure minimal CPU usage, making it suitable for devices with limited power resources.

5.3 Test Results

- **Memory Usage**: With 1000 stored credentials, the memory footprint remained under **20 MB**, which is acceptable for most modern systems.
- **Speed**: Password retrieval and decryption operations took on average **0.15 seconds per query**, which is nearly instantaneous for end-users.
- **Accuracy**: 100% retrieval accuracy was achieved during repeated encryption–decryption cycles.

- **Durability**: Encrypted data remained secure and accessible even after multiple application restarts and key validations.
- **Power Consumption**: Negligible CPU spikes were recorded during encryption and retrieval tests, implying low impact on battery-powered devices.

5.4 Recommendations for Industrial Use

If this system were to scale to an industrial level, the following recommendations can ensure robustness:

- **Database Integration**: Moving from JSON files to a secure database (e.g., PostgreSQL with encryption at rest) for better scalability.
- **Hardware Security Modules (HSMs)**: To securely manage encryption keys.
- **Stress Testing**: Benchmarking with millions of credentials to ensure performance under enterprise load.
- **Cloud Deployment Considerations**: Optimize cryptographic operations to minimize latency in distributed systems.

5.1 Test Plan/ Test Cases

Test Plan

The goal of testing was to ensure that the password manager functions correctly, securely, and efficiently across all core modules:

1. **Authentication Module** – Verify user login and credential validation.
2. **Key Management** – Verify correct generation, storage, and retrieval of encryption keys.
3. **Password Storage** – Verify correct encryption and secure storage of credentials in passwords.json.
4. **Password Retrieval** – Verify correct decryption and retrieval of passwords.
5. **Clipboard Integration** – Verify that retrieved passwords are copied securely to the clipboard.

Test Case ID	Description	Input	Expected Output	Actual Result	Status

Test Case ID	Description	Input	Expected Output	Actual Result	Status
TC-01	User Authentication	Correct username & password	Successful login	Successful login	Pass
TC-02	Invalid Authentication	Wrong password	Access denied	Access denied	Pass
TC-03	Key Generation	First-time setup	Encryptionkey.key created	Key created successfully	Pass
TC-04	Key Retrieval	Application restart	Existing key is reused	Key loaded correctly	Pass
TC-05	Store Password	Add new credentials	Encrypted entry stored in passwords.json	Entry stored & encrypted	Pass
TC-06	Retrieve Password	Request stored credentials	Decrypted password returned	Password returned correctly	Pass
TC-07	Clipboard Function	Retrieve password	Password copied to clipboard	Copied successfully	Pass
TC-08	Data Integrity	Modify passwords.json manually	Decryption fails / Integrity error	Error shown	Pass
TC-09	Stress Test	1000 stored credentials	System remains responsive	Retrieval under 0.2 sec	Pass
TC-10	Security Test	Attempt to open passwords.json directly	Data unreadable (encrypted)	Confirmed encrypted	Pass

5.2 Test Procedure

The testing procedure for the Password Manager project was carried out in a structured manner, covering setup, execution, and verification.

1. **Step 1: Environment Setup**

- Installed Python (version X.X).
- Installed required libraries (cryptography, pyperclip, etc.).
- Created three key files:
    - user_data.json – stores authentication details.
    - encryption_key.key – stores the generated encryption key.
    - passwords.json – stores encrypted credentials.

2. **Step 2: Authentication Testing**

3. Start the application.
4. Enter valid username and password → verify successful login.
5. Enter invalid credentials → verify denial of access.
6. Record results (Pass/Fail).

7. **Step 3: Key Management Testing**

1. Run the application in first-time setup mode.
2. Verify that a new encryption_key.key file is generated.
3. Restart the application → confirm the same key is reused for consistency.

8. **Step 4: Password Storage Testing**

1. Add a new credential (e.g., Gmail → user@example.com / MyPass123).
2. Verify that the entry appears in passwords.json.
3. Open the file manually → confirm that stored data is encrypted (not plain text).

9. **Step 5: Password Retrieval Testing**

1. Request retrieval of the stored Gmail password.
2. Verify that the correct password is returned after decryption.
3. Check that retrieval time is within acceptable performance limits (<0.2 sec for 1000 credentials).

10. Step 6: Clipboard Function Testing

1. Retrieve a password.
2. Verify that the password is copied to the clipboard automatically.
3. Paste into a text field (for verification) → confirm correctness.

11. Step 7: Data Integrity Testing

1. Manually tamper with passwords.json (e.g., delete or modify values).
2. Attempt to retrieve credentials.
3. Verify that the system shows an error or fails decryption (ensuring tampering is detected).

12. Step 8: Stress and Performance Testing

1. Insert 1000+ credentials into passwords.json.
2. Measure:
    - Memory usage (should remain under 20 MB).
    - Retrieval time per query (should remain ~0.15 sec).
3. Verify system remains stable.

13. Step 9: Security Testing

1. Attempt to open and read passwords.json directly.
2. Verify that data is unreadable without the encryption key.
3. Confirm encryption algorithms prevent plain-text access.

5.3 Performance Outcome

The performance evaluation of the password manager showed that the system is efficient, reliable, and secure. With around 1000 stored credentials, the application consumed only about 18 MB of memory, demonstrating good efficiency. Encryption and retrieval operations were completed within 0.12–0.15 seconds, making the system fast and responsive. Accuracy was maintained at 100%, with all stored passwords correctly encrypted and decrypted, while tampering attempts were detected successfully, ensuring data integrity. The system also proved durable, as encrypted data remained secure and accessible across multiple restarts, and power consumption was negligible with CPU usage staying below 5%. Overall, these outcomes confirm that the design is suitable for real-world use beyond academic scope.

6 My learnings

Working on this project has been a highly valuable experience that strengthened both my technical and professional skills. I gained a deeper understanding of **cryptography, secure key management, and password storage techniques**, which are essential concepts in modern cybersecurity. I also learned how to design a system with real-world constraints in mind, such as memory efficiency, speed, and security durability. Beyond technical skills, this project improved my ability to **plan, test, and document software systematically**, which is critical in any industrial environment.

Additionally, I developed problem-solving skills by addressing challenges such as encryption errors, data integrity issues, and performance testing. These experiences not only enhanced my programming expertise in Python but also gave me practical insights into how security-focused applications are built and tested.

Overall, this project has prepared me to contribute more effectively to future roles in **software development, cybersecurity, and cloud-based application security**. The knowledge gained here will serve as a strong foundation for my career growth in the technology industry.

7 Future work scope

Although the current password manager is functional and secure, there are several enhancements that can be explored in the future. One important improvement is **migrating from JSON storage to a secure database** for better scalability and performance. Integration with **two-factor authentication (2FA)** and **biometric login** can further strengthen security. Features like

automatic password generation, password strength analysis, and expiry reminders would make the tool more user-friendly. For enterprise use, **cloud-based synchronization, role-based access control, and hardware security module (HSM) integration** can be added. Due to time limitations, these features could not be implemented in this version but remain strong opportunities for future development.