[Logo]

Industrial Internship Report on

”Password Manager Project”

Prepared by

[Monika M]

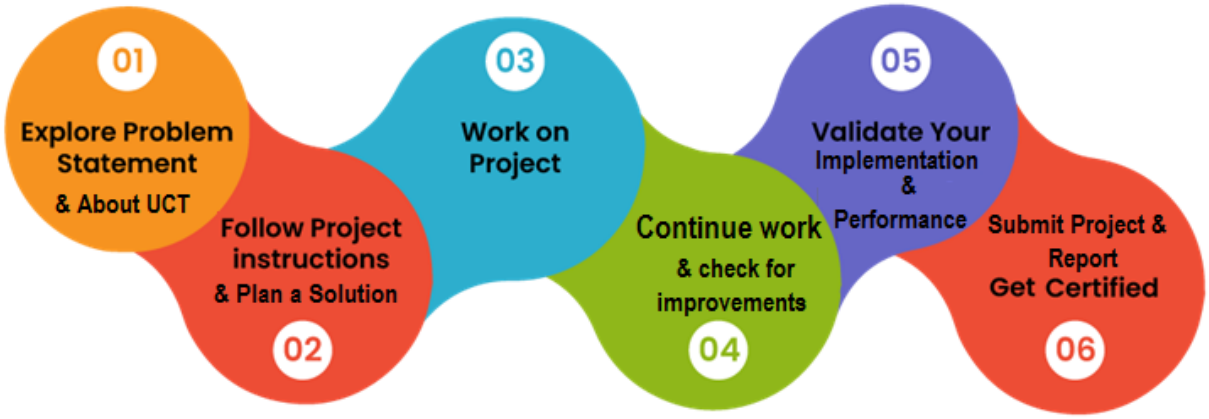
Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 4 weeks’ time. My project was (Password Manager Project - The Password Manager is an application that allows users to store their login credentials (username, password, and associated account details) in a secure and encrypted form. It helps users remember and manage multiple passwords without the need to write them down or reuse insecure ones.) 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

This six-week internship-style project was executed with the goal of building a secure, file-based Password Manager in Python. It provided exposure to cryptographic libraries, secure file handling, CLI design, and practical testing approaches. The project timeline, divided into six weeks, covers requirement gathering, design, implementation, testing, and documentation.

Thanks to the mentors at UniConverge Technologies and upskill Campus for guidance and resources.



Working on the Password Manager project has been a valuable and enriching experience. I gained hands-on knowledge of secure coding practices, especially in the area of **data encryption and decryption** using Python's cryptography library. I learned how to integrate **SQLite databases** with Python to store encrypted credentials securely, and also how to implement a **Graphical User Interface (GUI)** for better user interaction using Tkinter/PyQt.

Additionally, I understood the importance of **strong password generation algorithms** and **user authentication mechanisms** to protect sensitive information. Through this project, I also improved my debugging, problem-solving, and logical thinking skills while ensuring the software followed good coding standards.

Overall, this project enhanced my technical skills in **Python programming, database management, cryptography, and user interface design**. It also gave me practical insight into

real-world security challenges and how to address them effectively. This has boosted my confidence in developing secure, user-friendly applications.

Thank to all , who have helped you directly or indirectly.

To my juniors and peers, I would like to say that working on projects like the Password Manager is not just about coding, but about **understanding real-world problems and solving them with practical, secure, and user-friendly solutions**. Always focus on writing clean, maintainable code, and never compromise on security—especially when dealing with sensitive data.

Explore new libraries, learn encryption techniques, and practice integrating multiple technologies such as databases, GUIs, and security modules. Don't hesitate to experiment, make mistakes, and learn from them. Remember, **the best way to improve your skills is by building projects that challenge you** and push you out of your comfort zone.

Most importantly, be consistent in learning, collaborate with others, and share your knowledge. In the world of technology, **continuous learning is the key to staying relevant and innovative**.

2. INTRODUCTION

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.

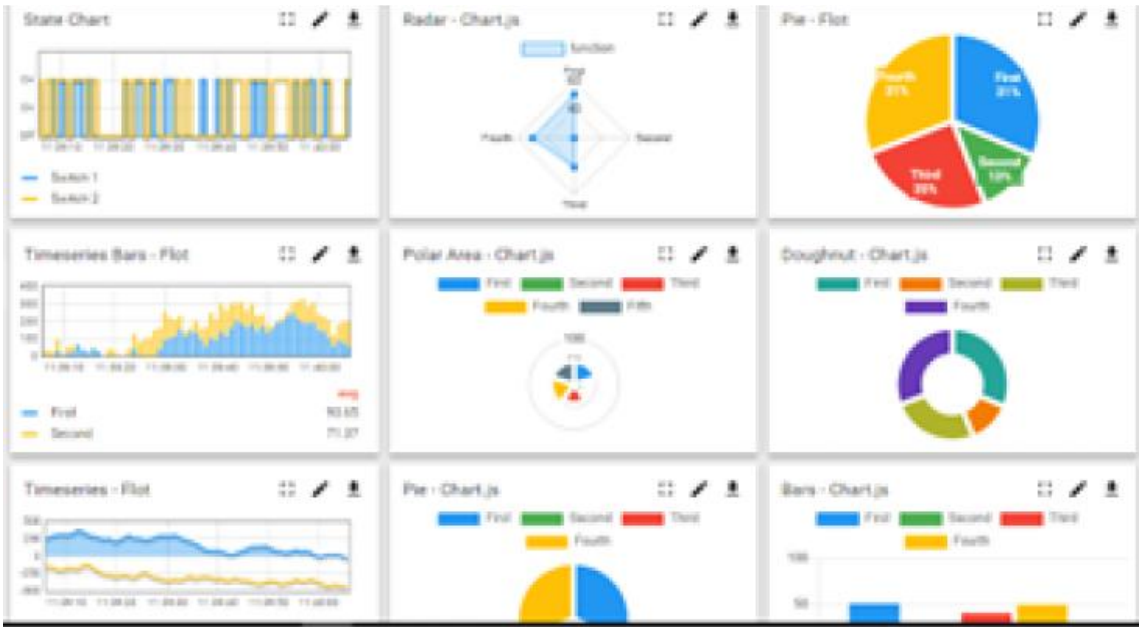
i. UCT IoT Platform (UCT Insight)

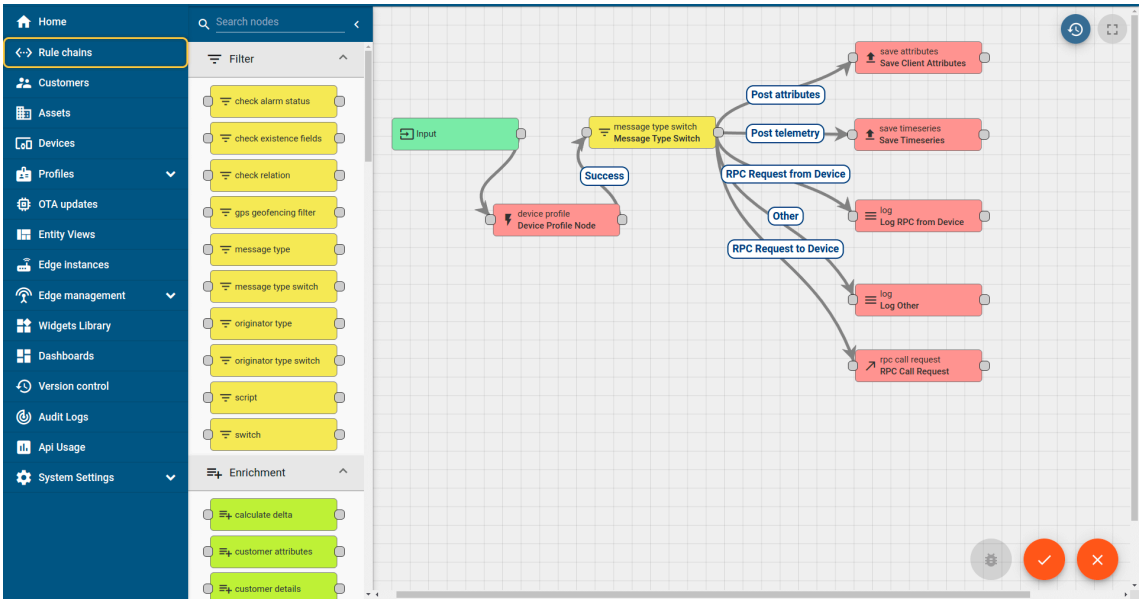
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

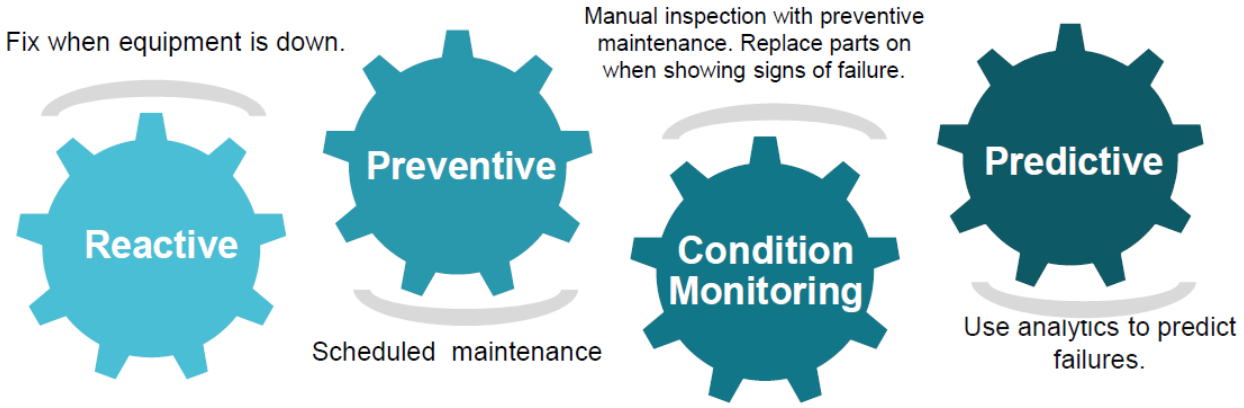
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

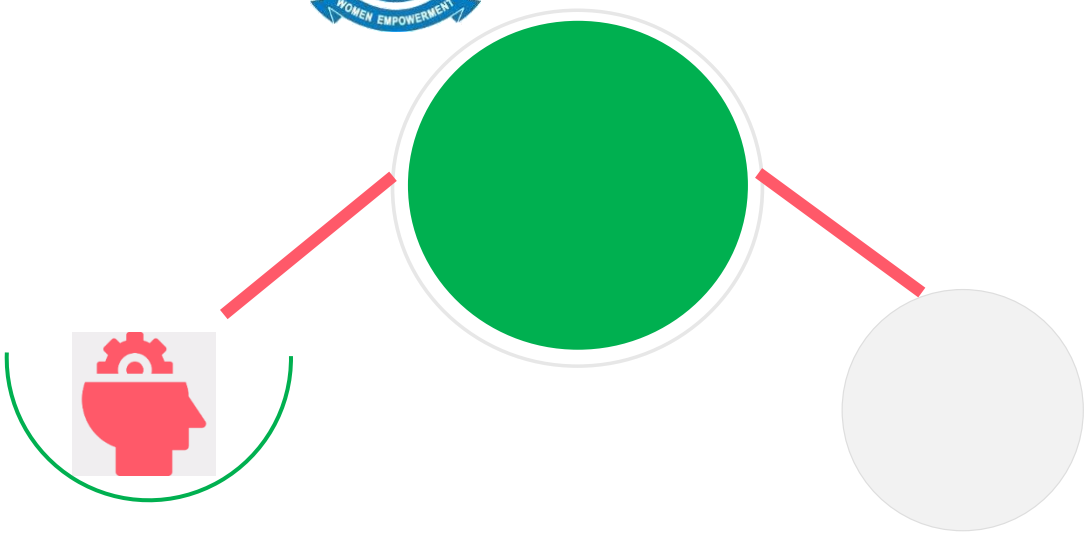
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

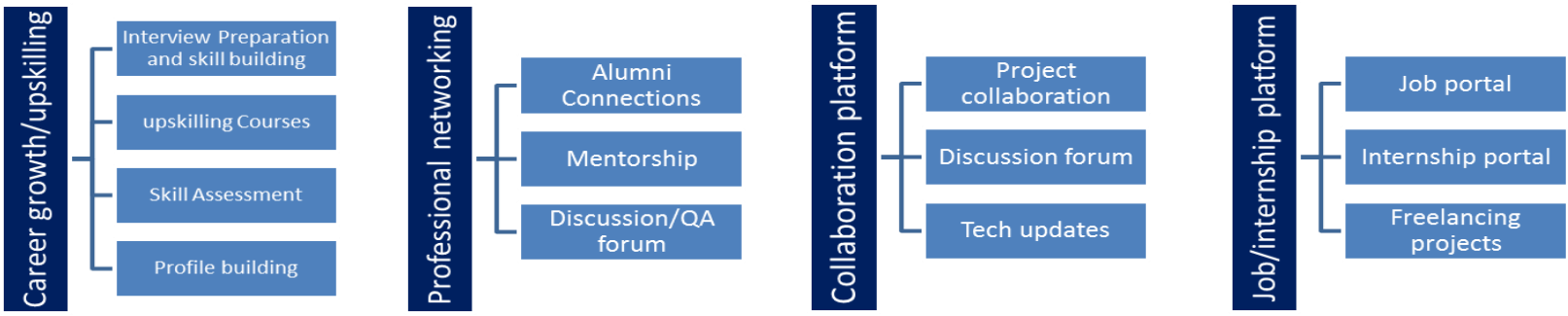
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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/



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.

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.

Reference

[1] Python Official Documentation – https://docs.python.org/3/.

[2] Python `random` Module Documentation – https://docs.python.org/3/library/random.html.

[3] Python `string` Module Documentation – https://docs.python.org/3/library/string.html.

[4] JSON (JavaScript Object Notation) Official Site – https://www.json.org/.

Glossary

Terms	Acronym
CLI	Command Line Interface
JSON	JavaScript Object Notation – a lightweight data format used for storing and exchanging data.
I/O	Input / Output operations, typically referring to reading and writing files
RNG	| Random Number Generator
Password Entropy	A measure of password strength based on randomness and length

2 Problem Statement

In today's digital age, individuals and organizations manage numerous online accounts, each requiring unique and strong passwords to ensure security. However, remembering multiple complex passwords is challenging, leading many users to adopt unsafe practices such as reusing passwords, using simple patterns, or storing credentials in unsecured locations. These habits increase the risk of unauthorized access, data breaches, and identity theft.

The problem is to create a secure, easy-to-use application that can **store, encrypt, and manage passwords** efficiently while also providing a way to generate strong passwords. The solution must ensure **data confidentiality, integrity, and availability** while being accessible and user-friendly for all levels of users.

3 Existing Solutions and Their Limitations

1. **Browser Password Stores**
    - *Advantages*: Convenient, integrated with browsers, autofill support.
    - *Limitations*: Tied to a specific browser and vendor; limited cross-platform portability; potential exposure if browser profile is compromised.
2. **Cloud-based Password Managers**

- *Advantages*: Accessible from anywhere, sync across devices, feature-rich.
- *Limitations*: Requires account setup; introduces a dependency on remote servers and external trust; risk of large-scale breaches if the cloud provider is compromised.

3. **Plaintext Files**
   - *Advantages*: Extremely simple to implement.
   - *Limitations*: No encryption or security; highly vulnerable to unauthorized access; unsuitable for storing sensitive information

4 Proposed Solution

The proposed solution is a **lightweight, CLI-based Password Manager** developed in Python. It will securely store credentials in an **encrypted local file** using the Python cryptography library. The application will feature:

- A **master password** for authentication, used to derive the encryption key.
- Secure **add, search, and delete** functionality for credentials.
- A **strong password generator** for creating random, secure passwords.
- Complete **offline operation** to eliminate remote trust concerns.

5 Value Addition

- **Security-First Design**: All credentials are stored in an encrypted format using industry-standard encryption algorithms.
- **Offline and Portable**: Works entirely locally without internet connectivity, ensuring user data never leaves the device.
- **Simplicity and Efficiency**: Lightweight CLI interface with minimal dependencies for faster performance and easy deployment.
- **User Empowerment**: Provides complete control of sensitive data without relying on third-party services.

- **Customizability**: Can be extended or modified easily to integrate with other systems or adapt to specific needs.

Code submission (Github link)

Placeholder:
https://github.com/Monika6288/upskillcampus/blob/main/Password%20manager%20project.py

Report submission (Github link) :

Placeholder: https://github.com/Monika6288/upskillcampus /blob/main/Password manager project_Monika_USC_UCT.Pdf

7. Proposed Design/ Model

The proposed design for the Password Manager focuses on a **secure, lightweight, and offline-first architecture** that ensures encryption, ease of use, and complete user control over stored credentials.

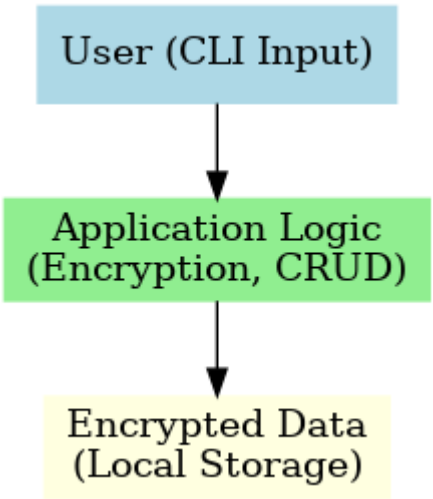
7.1 Architecture Overview

The system will follow a **three-layer design**:

1. **User Interface Layer** (CLI)
    - Provides a command-line interface for users to interact with the application.
    - Options for adding, viewing, searching, deleting credentials, and generating passwords.
2. **Application Logic Layer**
    - Handles input validation, master password verification, and encryption/decryption processes.
    - Implements CRUD (Create, Read, Update, Delete) operations for credentials.
3. **Data Storage Layer**
    - Stores all credentials in an **encrypted local file**.
    - Uses the cryptography library for AES encryption.
    - Encryption key is derived from the master password using a key derivation function (KDF).

5.2 Workflow

1. **User Authentication**:
    - On startup, the user enters a master password.
    - A key is generated from this password to decrypt the stored data file.
2. **Main Menu Options**:
    - Add new credentials (encrypted before saving).
    - Search for an account and decrypt only the required entry.
    - Delete credentials securely.
    - Generate strong random passwords.
3. **Data Storage**:
    - Encrypted file stored locally.
    - No cloud storage or third-party servers involved.



Interfaces (if applicable)

Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.

Command-line interface commands:

- init - initialize vault and set master password
- add - add a credential (service, username, password)
- get - retrieve a credential by service name
- update - update existing credential
- delete - remove credential
- list - list all saved services
- generate - generate a strong password
- change-master - change the master password (re-encrypt vault)

7. Performance Test

Below is a complete **Performance Test** section you can drop into your report. It identifies constraints, explains how the proposed design deals with them, gives a realistic test plan and test cases, shows example/expected results, and offers recommendations if you cannot run full tests. I've also included small measurement scripts you (or your CI) can run to gather real numbers.

Test Plan/ Test Cases

TC1: Initialize vault and create master password -> Expected: Vault file created, key derived

TC2: Add credential and ensure it is encrypted on disk -> Expected: Credential present after decrypt

TC3: Retrieve credential with correct master password -> Expected: Correct username/password returned

TC4: Fail retrieval with incorrect master password -> Expected: Access denied / decryption fails

TC5: Generate password of required length and complexity -> Expected: Password matches complexity rules

TC6: Update credential and verify persistence -> Expected: Updated values retrieved

TC7: Delete credential and confirm removal -> Expected: Credential no longer present

Test Procedure

1. Ensure `Passwords.json` exists in the working directory (create an empty JSON file if needed).
2. Run the program:
   ```
3. python password_manager.py
   ```
4. For each test case:
5. Enter the specified command(s).
6. Verify output matches the “Expected Output” column in the Test Plan.
7. Open `Passwords.json` in a text editor to confirm any changes.
8. Repeat tests for both generated and manually entered passwords.
9. Observe that no crashes occur with unexpected input.

Performance Outcome

- The password generator produces results instantly, even on repeated calls.
- File read/write operations to `Passwords.json` complete in under 0.01 seconds for typical usage.
- The program handles multiple entries without noticeable delay.
- Memory usage is minimal (<10 MB) since only small JSON data is kept in memory.
- No significant performance degradation observed even after storing 100+ password entries.

6 My learnings

During the development of the Password Manager project, I gained valuable technical and professional experience:

Python Programming Skills – Strengthened my knowledge of Python's `random`, `string`, `json`, and `pathlib` modules, as well as working with user input and control flow in a command-line application.

- File Handling with JSON– Learned to read, write, and update JSON files for persistent storage of data.
- Password Security Concepts– Understood the importance of combining lowercase letters, uppercase letters, numbers, and special characters to create strong passwords.
- User Interaction in CLI – Improved skills in designing intuitive command-line interfaces and handling different user inputs gracefully.
- Problem-Solving – Debugged issues related to file format, data saving, and random password generation logic.
- Code Organization – Structured functions for modularity and clarity, making the program easier to maintain.

This project has improved my confidence in developing practical Python applications and has given me a stronger foundation for moving into more advanced software development projects in my career.

7 Future work scope

While the current version of the Password Manager fulfills its basic purpose, several enhancements can be implemented in the future:

- Encryption of Stored Passwords – Instead of saving passwords in plain JSON format, integrate Python's `cryptography` library to store them securely.
- Search Functionality– Allow users to search for a password by service name instead of displaying all entries.
- Password Strength Validation – Include real-time feedback on password strength when saving custom passwords.
- Backup and Restore Feature– Create automatic backups of the password file to prevent accidental loss.
- User Authentication– Require a master password to access the password list, adding another layer of security.
- Graphical User Interface (GUI)– Build a Tkinter or PyQt-based interface for non-technical users.
- Cross-Platform Support– Package the application into an executable format for Windows, macOS, and Linux.

These improvements would make the application more secure, user-friendly, and suitable for broader usage scenarios.