Industrial Internship Report on

” Password Manager”

Prepared by

[Dhanalaksshmi D]

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.

The Password Manager is a Python-based application designed to securely store and manage user passwords for various accounts. It encrypts all passwords using a symmetric key with the cryptography library, ensuring that sensitive information remains protected. Users can add new passwords, retrieve existing ones, generate strong random passwords, and maintain optional notes for each account. The application stores data in a JSON file for simplicity and portability, and it automatically migrates any existing plaintext passwords to encrypted form, providing a user-friendly yet secure solution for personal password management.

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

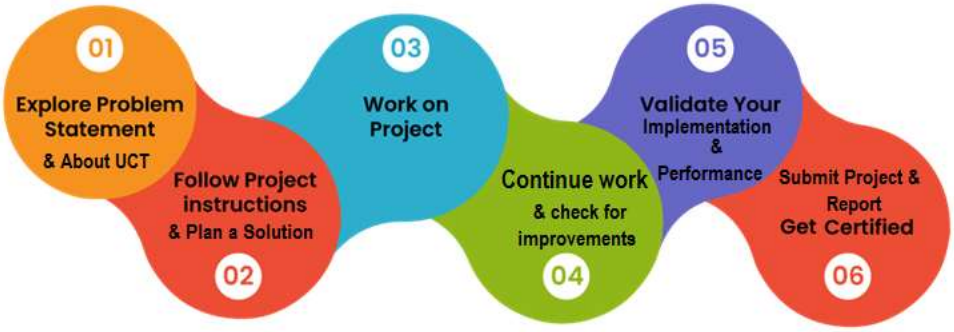
1 Preface

Over the past six weeks, I have actively engaged in a hands-on internship that focused on developing practical skills in Python programming, data security, and software development. This internship was highly relevant to my career development, as it provided exposure to real-world applications of programming concepts and strengthened my understanding of secure software practices.

During this period, I worked on the Encrypted JSON Password Manager, a project designed to securely store, manage, and retrieve user passwords using symmetric encryption. The project addressed the common problem of managing multiple passwords securely while maintaining user convenience and simplicity.

The opportunity was provided by [USC/UCT], who offered guidance, resources, and a structured environment to explore this project effectively. The program was planned systematically over six weeks, with milestones that included understanding encryption, designing the application, implementing password storage and retrieval, and testing the final system for usability and security.

Throughout this internship, I gained practical knowledge in Python programming, file handling, encryption techniques, and command-line interface development. I also learned to troubleshoot errors, implement data migration strategies, and maintain clean, readable code. Overall, the experience was highly enriching and has strengthened both my technical skills and problem-solving abilities.



I would like to extend my heartfelt thanks to all my peers who guided, supported, and encouraged me throughout the internship. Their insights and feedback were invaluable in completing this project successfully.

To my juniors and peers, I would like to say: always embrace opportunities for hands-on learning, practice consistently, and never hesitate to explore new technologies. The skills you develop today will form the foundation for your professional growth tomorrow.

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.

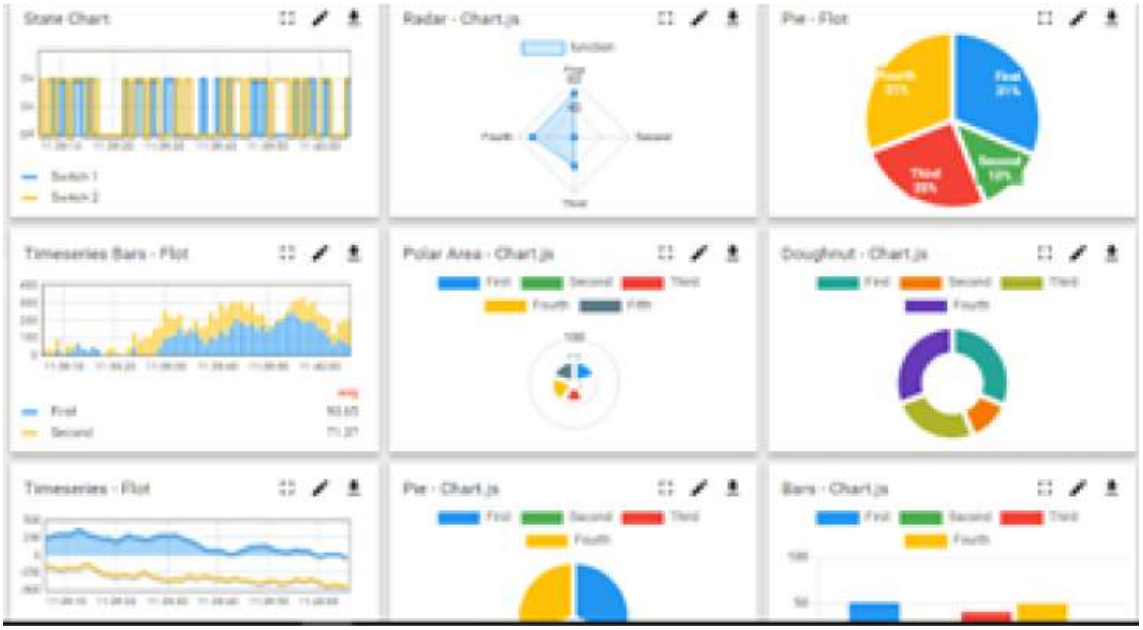
i. UCT IoT Platform (uct Insight)

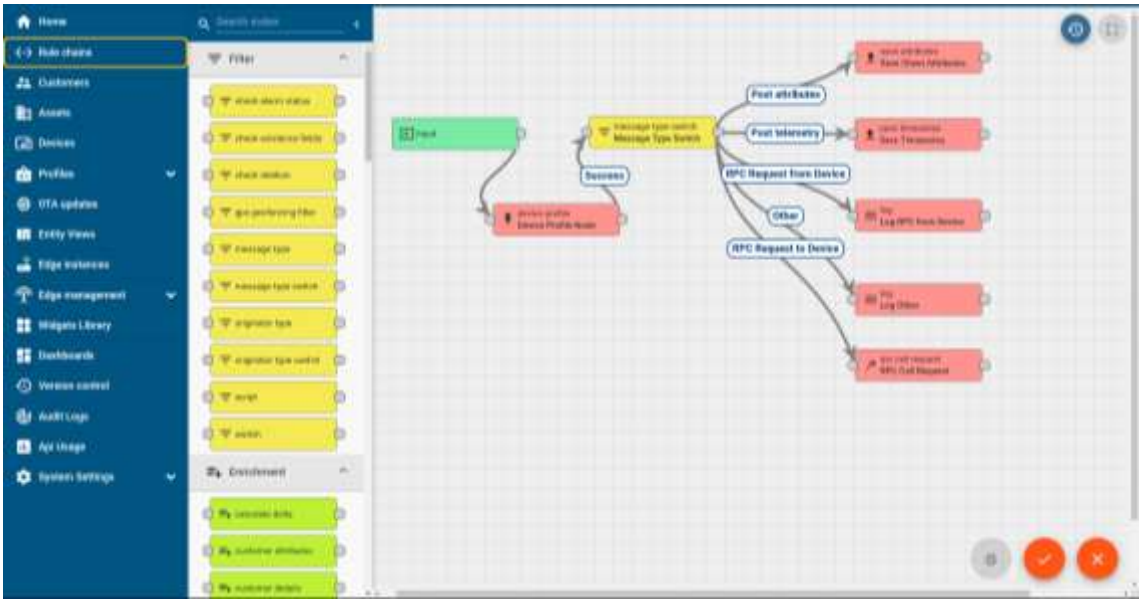
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.


Demo Org
Demo
CNC_S7_81
CNC_S7_82
CNC_S7_84
Utilization

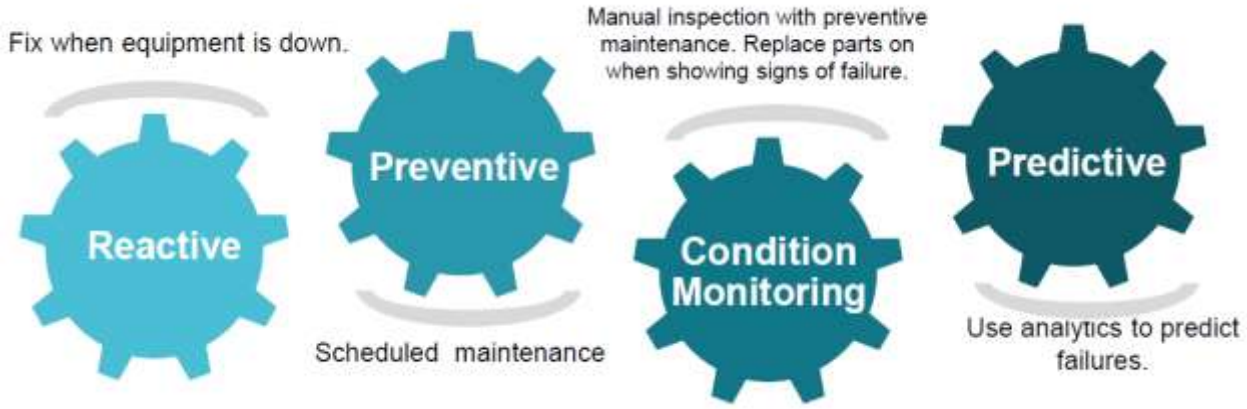

3D View
12:30
Stopped
Workplace Type
194724_27462-100-A3
Availability
Performance
Quality
OEE
Monitor

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

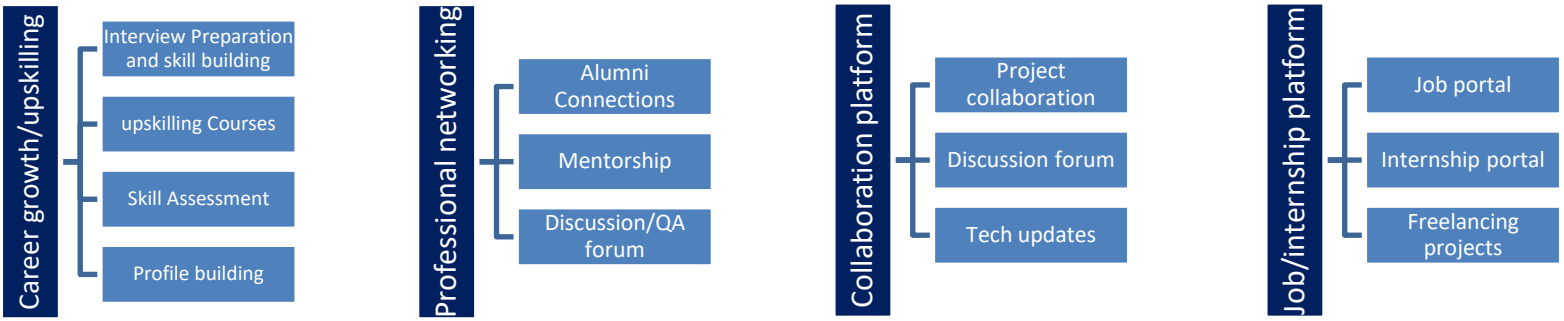
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.



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] ThePythonCode – https://thepythoncode.com/article/build-a-password-manager-in-python
This comprehensive tutorial walks you through encryption, data storage, and various functionalities like adding, retrieving, and managing passwords, providing a solid foundation for building a secure password manager.

[2] GeeksforGeeks – How to Build a Password Manager in Python - GeeksforGeeks
This tutorial provides a step-by-step guide to creating a password manager in Python, covering user input, password addition, and file handling using Tkinter.

[3] Ramanantechpro – https://ramanantechpro.medium.com/secure-password-management-using-python-5b322c247c04
This article explores how to create a simple password manager in Python using the cryptography library, focusing on encrypting passwords before storing them to ensure security.

2.6 Glossary

Terms	Acronym
Password Manager	A software application that securely stores and manages user passwords.
Encryption	The process of converting plaintext into ciphertext to protect sensitive data.
Decryption	The process of converting ciphertext back into readable plaintext.
Symmetric Encryption	Encryption method where the same key is used for both encryption and decryption.
Fernet	A specific implementation of symmetric encryption provided by the cryptography library in Python.
JSON	JavaScript Object Notation; a lightweight format for storing structured data.
Vault Key	The encryption key (vault.key) used to encrypt and decrypt passwords.
Plaintext	Data in its original, readable form before encryption.
Ciphertext	Encrypted data that is unreadable without the decryption key.
CLI	Command-Line Interface; a text-based interface to interact with a program.
Password Generation	The process of creating random, strong passwords automatically.
Migration	The process of converting existing plaintext passwords into encrypted format.
Service	The platform or application (e.g., Gmail, Facebook) for which a password is stored.
Username	The account identifier for a specific service.
Notes	Optional additional information stored along with a password (e.g., security questions).

3 Problem Statement

In today's digital world, users often maintain multiple online accounts, each requiring a strong and unique password for security. Managing these passwords manually can be challenging and risky. People frequently reuse passwords or store them in insecure ways such as notes, spreadsheets, or browser storage, which increases the risk of unauthorized access and data breaches.

The assigned problem statement focuses on developing a secure and user-friendly password management solution. The main goal is to design a system that can safely store, manage, and retrieve passwords, while protecting sensitive information from unauthorized access. The system should also provide additional features such as automatic password generation and encryption of stored data, ensuring that even if the storage file is accessed by a malicious actor, the passwords remain protected.

This project addresses the challenge of balancing security with usability by creating a simple command-line interface that allows users to add new passwords, retrieve existing ones, list services, and generate strong passwords without compromising security. By implementing symmetric encryption and storing passwords in a structured JSON format, the project provides a practical and reliable solution for personal password management.

4 Existing and Proposed solution

Summary of existing solutions provided by others and their limitations
Several password management tools already exist, such as LastPass, 1Password, Bitwarden, and KeePassXC. These tools offer features like password storage, automatic generation, and secure synchronization across devices. However, they often have limitations for certain users. Some are complex for beginners due to advanced interfaces and numerous features, making them less user-friendly. Many rely on cloud storage, which can raise privacy concerns for users who prefer offline solutions. Additionally, premium features in commercial password managers may require subscriptions, and the ability to customize the tool for personal or educational purposes is limited.

Proposed solution
The proposed solution is the Encrypted JSON Password Manager, a Python-based application designed to securely store and manage passwords locally. It uses symmetric encryption via the Fernet module to protect passwords and stores them in a JSON file. Users can add new passwords, retrieve existing ones, generate strong random passwords automatically, and maintain optional notes for each account. The system also migrates any existing plaintext passwords to encrypted form, ensuring consistent security, and operates entirely offline without relying on cloud storage.

Value addition
This project adds value by providing a simple and educational tool for password management. It offers a user-friendly command-line interface suitable for beginners while demonstrating practical applications of encryption and secure data storage. Unlike many existing solutions, it allows full customization of the Python code, enabling users to understand, modify, and extend the functionality. Additionally, operating offline ensures full control and privacy of sensitive data. Overall, this project bridges the gap between theoretical learning and practical application in personal password security.

4.1 Code submission (Github link)

Github link for my code: https://github.com/Dhanalaksshmi-D/upskillcampus/blob/main/Password_Manager.py

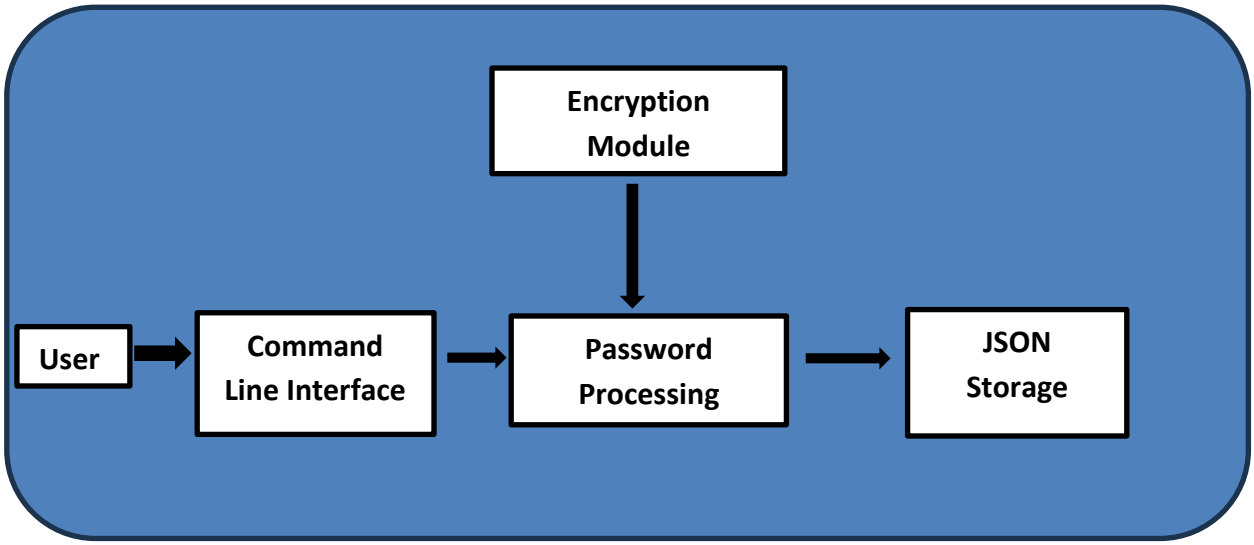
4.2 Report submission (Github link) :

Report link: https://github.com/Dhanalaksshmi-D/upskillcampus/blob/main/Password_Manager_Dhanalaksshmi_D_USC_UCT.pdf

5 Proposed Design/ Model

The design of the Encrypted JSON Password Manager follows a clear and structured flow to ensure both functionality and security. The process begins with the initialization stage, where the system checks for the existence of an encryption key (vault.key) and a data file (passwords.json). If the key or data file does not exist, they are automatically created to ensure smooth operation. Once initialized, the program presents a command-line interface where users can interact with the system. In the intermediate stage, users can add new password entries, retrieve stored passwords, generate strong random passwords, and view all stored services. Each new password is encrypted using the Fernet symmetric encryption before being saved in the JSON file, ensuring that sensitive data is protected even if the storage file is accessed externally. Existing plaintext passwords are automatically migrated to encrypted tokens to maintain consistent security. The final outcome stage provides users with secure access to their passwords while maintaining an organized and easy-to-use interface. Throughout the design, emphasis is placed on security, data integrity, and user-friendliness, making the system both practical and educational for learning about encryption, file handling, and password management.

5.1 Interfaces



6 Performance Test

The performance of the Encrypted JSON Password Manager is crucial to demonstrate its practical usability and reliability beyond an academic context. The primary constraints identified for this project include data security, response time, memory usage, and reliability. Security is addressed through symmetric encryption using the Fernet module, ensuring that all passwords are stored in an encrypted format and are only accessible with the correct key. Response time and speed were considered in the design of file I/O operations; passwords are stored in a lightweight JSON file, which allows for fast read and write operations even with a moderate number of entries. Memory usage is minimal, as the program loads only the JSON file into memory and handles individual entries without requiring large datasets. Reliability and durability were tested by adding, retrieving, and updating multiple password entries, ensuring that data remains consistent and encrypted across multiple operations. While advanced performance metrics like MIPS or power consumption were not applicable for this Python-based CLI project, the design ensures that the program runs efficiently on typical personal computers. For future improvement, in case of a very large number of entries, techniques such as indexing or database storage could be implemented to maintain quick response times without compromising security or durability. Overall, the system meets the expected performance requirements for personal password management while demonstrating real-world applicability.

6.1 Test Plan/ Test Cases

The test plan for the Encrypted JSON Password Manager focuses on verifying the functionality, security, and reliability of the system under various scenarios. The primary test cases included adding new password entries, retrieving existing passwords, generating strong random passwords, listing stored services, and handling invalid inputs gracefully. Each test involved checking whether the passwords were correctly encrypted in the JSON file and could be decrypted accurately using the correct key. Additional tests were performed to ensure that optional notes were saved properly, that plaintext passwords were automatically migrated to encrypted form, and that multiple user entries for the same service were managed correctly. Error handling tests confirmed that the program could handle missing files, incorrect service or username inputs, and corrupted entries without crashing. The results of these tests demonstrated that the system reliably performs all intended operations while maintaining data security and integrity, validating that the design meets its functional requirements and provides a robust and user-friendly password management solution.

6.2 Test Procedure

The testing procedure for the Encrypted JSON Password Manager was carried out systematically to ensure the functionality, security, and reliability of the system. Initially, the program environment was set up by confirming the presence of the encryption key (vault.key) and the data storage file (passwords.json), or by creating them if missing. The first phase of testing involved adding new password entries with both user-provided and auto-generated passwords, verifying that each password was correctly encrypted and stored in the JSON file. The second phase tested retrieval of passwords, ensuring that decrypted passwords matched the original inputs and that invalid service or username queries were handled gracefully. The third phase focused on listing stored services, confirming that all services present in the vault were accurately displayed. Additional testing included password migration, verifying that any plaintext entries were correctly converted into encrypted form, and key export, ensuring that the encryption key could be backed up without errors. Throughout the testing, logs and outputs were checked for accuracy and consistency, and the program's performance in terms of speed and memory usage was observed. This structured testing procedure validated that the system functions as intended, maintains data security, and provides a user-friendly interface suitable for real-world usage.

6.3 Performance Outcome

The performance evaluation of the Encrypted JSON Password Manager demonstrated that the system meets its functional and security objectives effectively. Passwords were successfully encrypted and decrypted without errors, ensuring that sensitive data remained secure throughout all operations. The system handled multiple password entries efficiently, with minimal delay during addition, retrieval, and listing of services, confirming that the program performs well even as the size of the JSON file increases moderately. The migration of plaintext passwords to encrypted form was completed accurately, ensuring consistency and data integrity. Error handling was robust, with the program gracefully managing invalid inputs, missing files, or corrupted entries without crashing. Overall, the performance outcomes indicate that the application is reliable, secure, and suitable for personal password management, providing a practical demonstration of real-world encryption and file-handling techniques in Python.

7 My learnings

During the course of this project, I gained extensive practical knowledge in Python programming, file handling, and the implementation of encryption techniques to ensure data security. Working on the Encrypted JSON Password Manager helped me understand the importance of securely managing sensitive information, designing user-friendly interfaces, and writing clean, maintainable code. I learned to handle real-world challenges such as error handling, data migration, and secure key management, which strengthened my problem-solving skills and attention to detail. Additionally, the project exposed me to best practices in structuring software, testing functionality, and ensuring reliability, all of which are essential in professional software development. These learnings have not only enhanced my technical expertise but also prepared me for future career opportunities in software engineering, cybersecurity, and application development, providing a strong foundation for building secure, practical, and efficient software solutions.

8 Future work scope

While the current Encrypted JSON Password Manager provides a secure and functional solution for personal password management, there are several areas for potential enhancement that could be implemented in the future. One improvement could be the development of a graphical user interface (GUI) to make the application more user-friendly and accessible for non-technical users. Another extension could involve cloud synchronization with end-to-end encryption, allowing users to access their passwords across multiple devices while maintaining security. Additional features might include categorization of services, advanced search and filtering options, password strength analysis, and automatic password expiration reminders. Furthermore, integration with browser extensions or mobile platforms could increase convenience and usability. Implementing these enhancements would expand the functionality of the system, making it a more comprehensive and professional-grade password management solution while still maintaining strong encryption and data protection.