KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY

Industrial Internship Report on

"Password Manager in Python"

Prepared by

SATYA PRIYANKA KAMUJU

Department: Computer Science & Engineering

Institution: Kakinada Institutions of Engineering and Technology

Internship Provider: upskill Campus (USC) & UniConverge Technologies Pvt Ltd (UCT)

Project Duration: 6 weeks

Executive Summary

This report details the development of a **Password Manager** application written in **Python**. The objective of the project was to design, implement, test, and document a secure, user-friendly desktop-password manager that helps users store, generate, and manage credentials safely. The system leverages modern cryptographic practices for local data encryption, a reliable password-generator, and an intuitive CLI/GUI for day-to-day use.

Key deliverables:

- A working Python-based password manager with encrypted local storage.
- Password generation utility with customizable rules.
- Features: Add, view (decrypt), edit, delete, search and export (encrypted) credentials.
- Documentation, test plan, and recommended future work for cloud sync and multi-device support.

Table of Contents

1. Preface

Internship Preface Image - Teamwork and Coding

Internship Preface Image - Teamwork and Coding

This project was completed as part of a 6-week industrial internship facilitated by upskill Campus (USC) in collaboration with UniConverge Technologies (UCT). The primary goal was to build a useful, secure password manager that demonstrates software design, secure coding practices, and user-focused features. The report documents the motivation, design decisions, implementation, testing, and future improvements.

Python Programming Image - Code Development

Python Programming Image - Code Development

---

2. Introduction

2.1 About UniConverge Technologies Pvt Ltd

UniConverge Technologies (UCT) provides digital transformation services and industrial solutions. Their focus areas include IoT, Cloud, Security, and full-stack development—making them a suitable industrial partner for projects requiring best practices and secure implementations.

2.2 About upskill Campus (USC)

upskill Campus provides career development and industry-aligned project opportunities for students. USC supported mentorship and the project framework for this internship.

2.3 Objective

Design and implement a secure, easy-to-use password manager in Python to help users safely store and manage credentials offline, with scope for future secure synchronization.

2.4 References

1. Python official documentation — https://docs.python.org/
2. Cryptography library documentation — https://cryptography.io/
3. OWASP Password Storage Cheat Sheet

2.5 Glossary

- **AES**: Advanced Encryption Standard

- **PBKDF2**: Password-Based Key Derivation Function 2
- **CLI**: Command Line Interface
- **GUI**: Graphical User Interface

---

3. Problem Statement

Users often reuse weak passwords across multiple services or store them insecurely (notes, spreadsheets). The objective is to provide a local, encrypted vault that securely stores credentials and helps users generate strong, unique passwords.

---

4. Existing and Proposed Solution

Existing solutions: Commercial managers (1Password, LastPass, Bitwarden) provide robust cloud features but may be overkill for simple offline storage or learning projects. Many basic scripts lack proper encryption, using plaintext or weak hashing.

Proposed solution: A lightweight Python application that:

- Stores records in an encrypted file using AES-GCM.
- Uses a master password to derive encryption keys (PBKDF2 with salt).
- Provides password generation (length, charset options).
- Offers CRUD operations on entries plus secure export/import.

Value additions:

- Clear CLI (and optional GUI) flows for non-expert users.
- Secure by-default choices (strong KDF iteration counts, authenticated encryption).
- Simple JSON-based data model for easy extension.

Code Submission(GitHub Link):
https://github.com/Satyapriyanka511/upskillcampus/blob/main/Passwordmanager.py
Report submission:
https://github.com/Satyapriyanka511/upskillcampus/blob/main/placeholder.txt

5. Proposed Design / Model

6. Implementation Details

6.1 Tech Stack

- **Language:** Python 3.10+
- **Libraries:** `cryptography`, `secrets`, `argparse`, `json`, `pyperclip` (optional), `PySimpleGUI` (optional GUI)
- **Storage:** Local file `vault.pwm` (binary encrypted file)

6.2 Core Modules

1. **vault.py** — Handles encryption/decryption, key derivation, load/save.
2. **cli.py** — Command-line interface for user interactions.
3. **generator.py** — Strong password generator with options.
4. **utils.py** — Helper functions (validation, clipboard copy).

6.3 Security Considerations

- Use authenticated encryption (AES-GCM) to prevent tampering.
- Do not store master password anywhere.
- Use high iteration counts for KDF and a long random salt.
- Zero sensitive data from memory if possible (Python has limitations).
- Encourage users to backup the encrypted vault file and remember their master password.

7. Performance Test

7.1 Test Plan / Test Cases

- **TC1:** Create a new vault and ensure it can be opened with the master password.
- **TC2:** Add 1,000 entries — measure load and save times.
- **TC3:** Attempt decryption with wrong password — must fail securely.
- **TC4:** Password generation randomness checks (basic uniqueness frequency test).

7.2 Test Procedure

- Automated scripts generate entries and time save/load operations.
- Manual checks for CLI responsiveness.

7.3 Performance Outcome

- With 1,000 small entries (metadata + credential pairs), load/save operations completed under a second on a typical modern laptop (i5, 8GB RAM). Exact numbers vary by hardware and iteration count for PBKDF2.

---

8. My Learnings

- Gained practical experience with cryptographic libraries and key management.
- Improved software design skills: modularization, clean CLI flow, and error handling.
- Learned about user-experience tradeoffs for security-first applications.

---

9. Future Work Scope

- Add secure cloud sync (end-to-end encrypted) with optional multi-device support.
- Implement hardware-backed key storage (e.g., OS keychain integration or YubiKey support).
- Add GUI enhancements and browser extension for autofill.
- Implement secure sharing of credentials among trusted users.

---

10. Appendix

Code snippets and How to run

1. Install dependencies:

```
python -m pip install cryptography pyperclip
```


2. Run CLI:

```
python cli.py --vault vault.pwm
```

3. Example: generating a 16-character password using `secrets`:

```python
import secrets, string
alphabet = string.ascii_letters + string.digits + string.punctuation
pwd = ''.join(secrets.choice(alphabet) for _ in range(16))
```

Acknowledgements

Declaration

I hereby declare that the project work presented in this report is my own, and all sources used have been duly acknowledged.

Date: 02/11/2025

Signature: K. Satya Priyanka

End of Report