[Your College Logo]

Industrial Internship Report on

” Password Manager”

Prepared by

[Ashish Arun Sonawane]

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 Project is a Python-based application designed to provide users with a secure and convenient solution for managing their passwords. The project leverages key libraries and functionalities to ensure robust password encryption and generation. 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

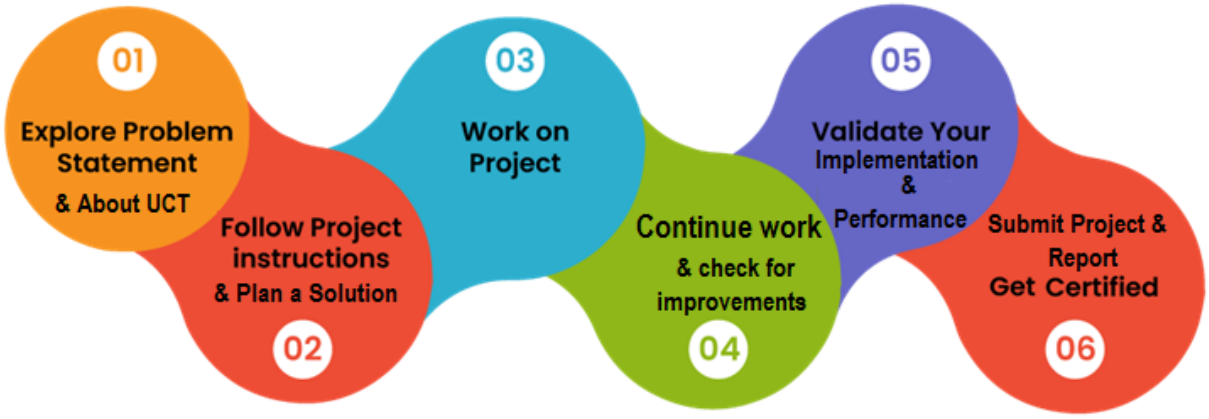
Summary of the whole 6 weeks' work.

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.

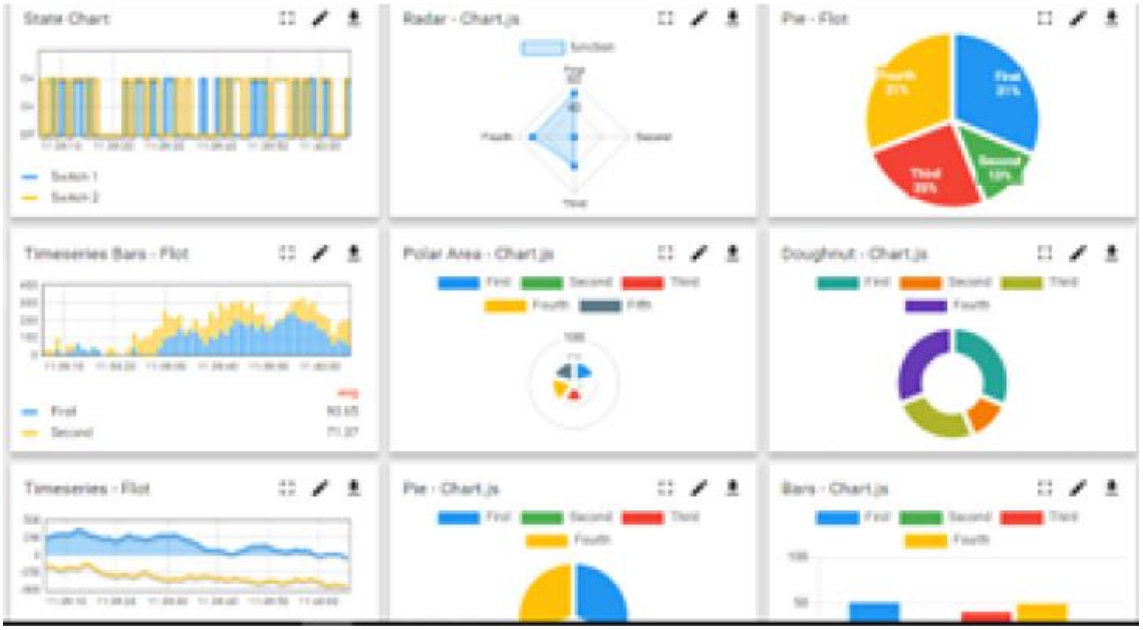
i. UCT IoT Platform (UCT Insight)

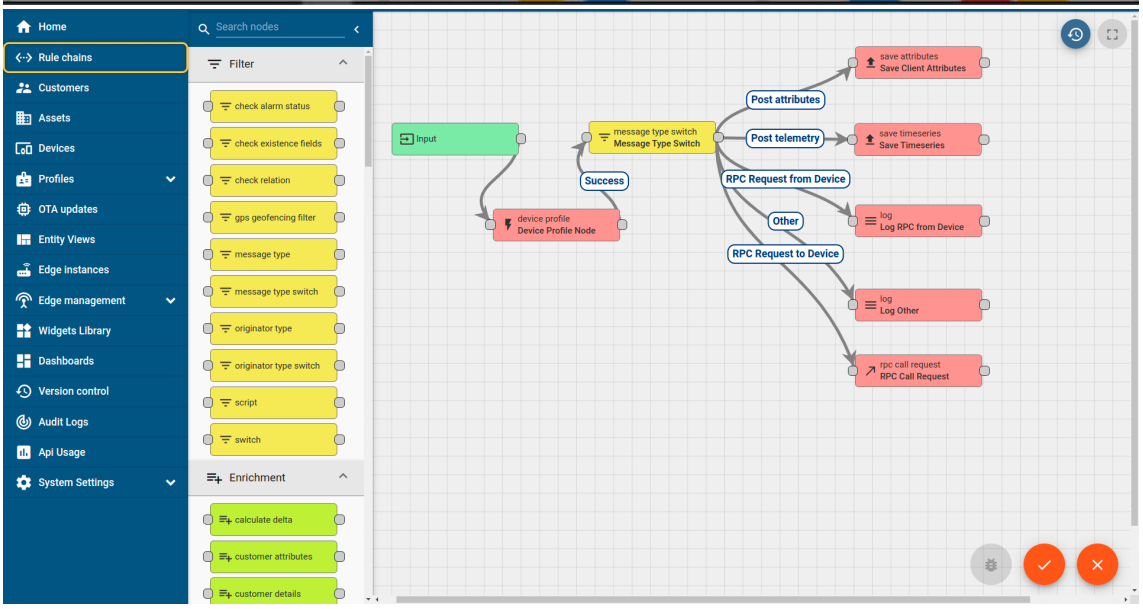
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

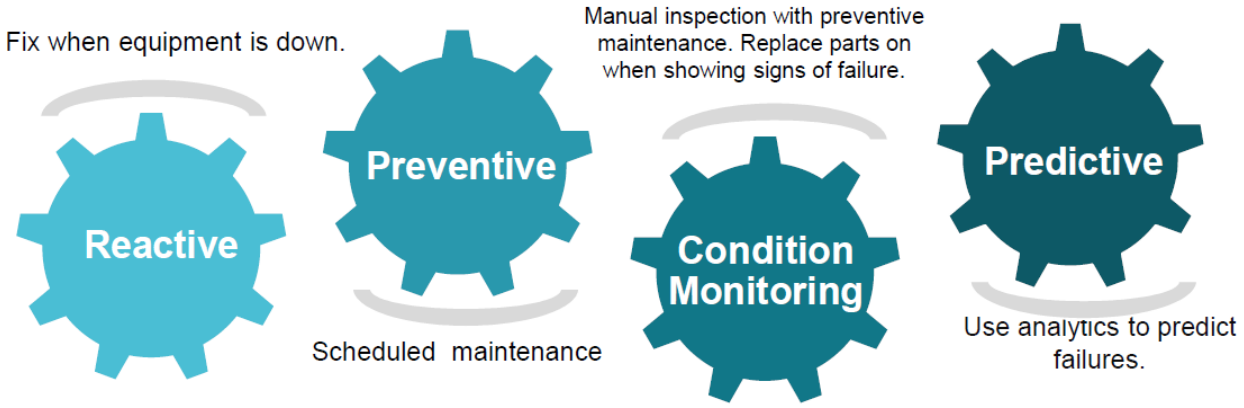
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

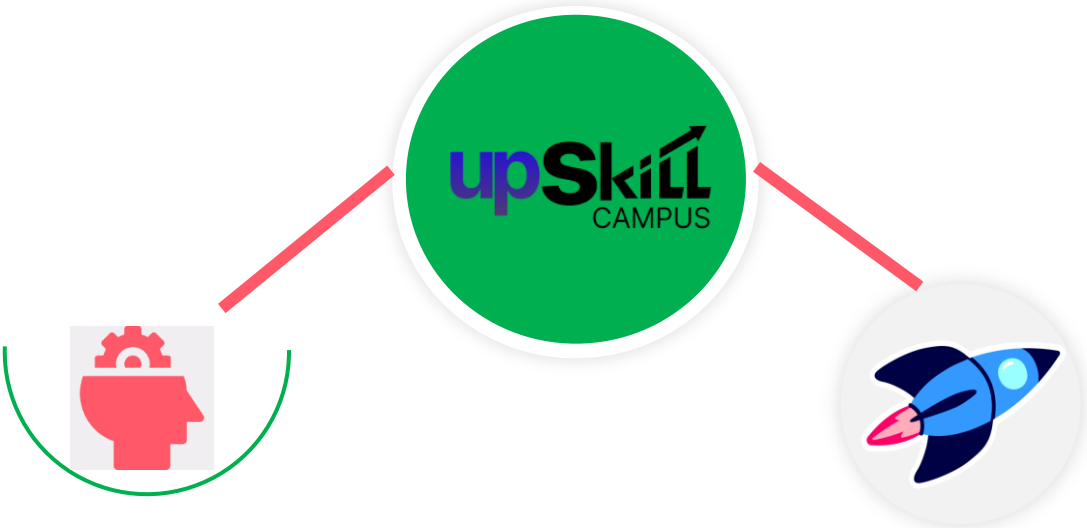
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

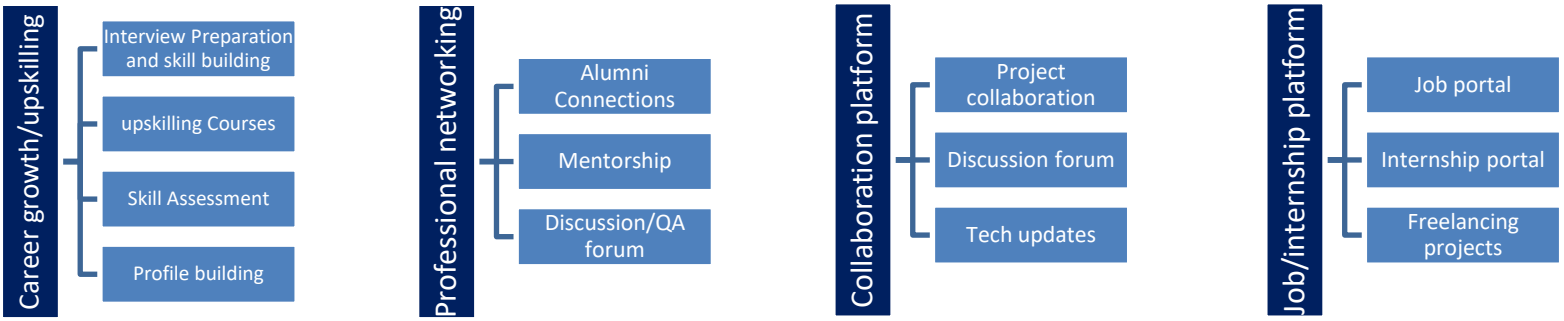
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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. sqlite3: The Python standard library module for interacting with SQLite databases. You can refer to the official Python documentation for sqlite3.

2. hashlib: Although imported, it's not used explicitly in the provided code. The Python documentation for hashlib can provide information on cryptographic hash functions.

3. cryptography.fernet: A module from the cryptography library used for AES encryption. You can refer to the official Cryptography documentation for more details, especially the section on Fernet.

4. base64: The Python standard library module for encoding and decoding base64 strings. You can find information in the official base64 documentation.

5. random and string: Python standard library modules used for generating random passwords. Documentation for random and string can be referenced.

2.6 Glossary

	Terms	Acronym
1. SQLite	Lightweight database management system.	None
2.Hashlib	Python library for secure hashing.	None
3. Fernet	Symmetric encryption tool in cryptography.	None
4.AES Encryption	Advanced Encryption Standard.	AES
5. Base64	Encoding scheme for binary data.	None
6. Random	Python library for random number generation.	None
7. String	Python library for string manipulation.	None
8.Password Encryption	Process of securing passwords.	None
9.Password Decryption	Reversing encrypted passwords.	None
10.User Interface (UI)	Interaction point between user and application.	UI
11.Cryptographic Key	Information for encryption/decryption.	None
12.Password Generation	Creating strong, random passwords.	None
13.ASCII Letters/Digits/Punctuation	Character sets for passwords.	ASCII
14. Exit Option	Menu choice for exiting.	None
15.Command-Line Interface (CLI)	Text-based user interaction.	CLI

3 Problem Statement

Design and implement a simple Password Manager application in Python to address the following key requirements:

1.Secure Password Storage:

Develop a system to securely store user passwords using encryption techniques.

2.Password Generation:

Provide functionality to generate strong and random passwords for users.

3.User Interaction:

- Create a user-friendly command-line interface (CLI) allowing users to:
- Add passwords for different accounts securely.
- Retrieve stored passwords by decrypting them.
- Generate strong passwords based on user preferences.
-

4. Data Persistence:

Optionally integrate SQLite for persistent storage of encrypted passwords.

5.Enhanced Security:

Implement password encryption using the Fernet module from the cryptography library to ensure data confidentiality.

6.User Guidance:

Include informative prompts and messages to guide users through the password management process.

7.Expected Outcomes:

- A functional password manager application with a clear and intuitive user interface.
- Secure encryption and decryption of passwords using Fernet.
- Strong and random password generation based on user specifications.
- Optional: Integration with SQLite for persistent storage.

4 Existing and Proposed solution

1. **Provide summary of existing solutions provided by others, what are their limitations?**

- **Summary :-**

1. Commercial Password Managers:

- Advantages: Feature-rich, with secure storage, password generation, and cross-device synchronization.
- Limitations: Often require a subscription, potential for centralization of sensitive data.

2. Open Source Password Managers:

- Advantages: Customizable, community-driven development.
- Limitations: May lack certain advanced features, user interface may vary.

3. Built-in Browser Password Managers:

- Advantages: Seamless integration with browsers, easy to use.
- Limitations: Limited features, reliance on browser security.

4. Command-Line Password Managers:

- Advantages: Lightweight, scriptable.
- Limitations: May lack a graphical interface, less user-friendly.

- **Limitations of Existing Solutions:**

1. Security Concerns:

- Some commercial solutions face security breaches, raising concerns about the safety of stored passwords.

2. Usability:

- Complex interfaces and features may overwhelm users, leading to poor adoption.

3. Platform Dependence:

- Browser-based solutions tie users to specific platforms or browsers.

2. **What is your proposed solution?**

Develop a lightweight and secure Password Manager using Python with the following features:

1. User-Friendly Interface:

- Implement a simple yet intuitive command-line interface (CLI) for ease of use.

2. Strong Encryption:

- Utilize the Fernet module for robust AES encryption to secure user passwords.

3. Password Generation:

- Provide a password generation function for creating strong and random passwords.

4. Optional Persistent Storage:

- Integrate SQLite for optional persistent storage of encrypted passwords.

5. Enhanced Security Measures:

- Implement best practices for secure password handling, such as key generation and protection.

3. What value addition are you planning?

1. Simplicity and Accessibility:

- Deliver a straightforward solution accessible to users of varying technical backgrounds.

2. Enhanced Security:

- Prioritize the security of stored passwords through robust encryption methods.

3. Flexibility:

- Offer optional persistent storage, allowing users to choose based on their preferences.

4. Educational Component:

- Include informative prompts to guide users and promote best password practices.

5 Proposed Design/ Model

1.User Interface:

Command-Line Interface (CLI):

- Create a straightforward CLI with interactive menus for user interaction.

- Clearly display options: Add Password, Retrieve Password, Generate Strong Password, and Exit.

2. Data Storage:

SQLite Database (Optional):

- Design a database schema with tables for account names and encrypted passwords.

- Allow users to choose between in-memory storage or SQLite for persistent storage.

3. Encryption and Decryption:

Fernet Encryption:

- Utilize the cryptography.fernet.Fernet module for AES encryption.

- Dynamically generate a Fernet key for each session and securely manage its lifecycle.

4. Password Generation:

Random Password Generation:

- Develop a function to generate strong and random passwords based on user-specified length.

- Utilize the random and string modules to ensure password strength.

5. Main Loop and User Interaction:

Continuous Interaction:

- Implement a main loop to allow users continuous interaction until they choose to exit.

- Display a user-friendly menu with options and prompt for user input.

6. Error Handling:

Robust Exception Handling:

- Implement error handling to gracefully manage potential issues during encryption, decryption, and user input.

- Provide informative error messages to guide users in case of errors.

7. Security Measures:

Password Security:

- Encourage users to adopt strong and unique passwords through educational prompts.

- Implement secure practices for handling cryptographic keys and sensitive data.

8. Modularity and Code Structure:

Modular Code Design:

- Organize the code into modular functions for better readability and maintainability.

- Separate concerns such as password generation, encryption, and user interface functionalities.

9. Future Enhancements:

Scalability:

- Design the application with extensibility in mind for potential future features.

- Consider adding multi-factor authentication or biometric authentication.

10. Documentation:

User and Developer Documentation:

- Provide clear and comprehensive documentation for end-users on how to use the application.

- Include developer documentation detailing the code structure, dependencies, and any customization options.

11. Educational Features:

User Guidance:

-Include informative messages or prompts to guide users on password best practices.

- Educate users on the importance of strong and unique passwords.

13. Accessibility:

Platform Independence:

-Ensure platform independence to allow users to run the application on various operating systems.

-Make the application accessible to users with different technical backgrounds.

6 Performance Test

1. Encryption and Decryption Speed:

Evaluate the time taken for password encryption and decryption.

2. Password Generation Speed:

Assess the speed of generating strong passwords.

3. Database Interaction:

Test the efficiency of storing and retrieving encrypted passwords.

4. User Interface Responsiveness:

Check the responsiveness of the command-line interface.

5. Concurrent User Handling:

Simulate multiple users to identify potential bottlenecks.

6. Error Handling Under Load:

Evaluate error handling and recovery during stress.

7. Memory Usage:

Monitor and optimize application memory usage.

8. Scalability Testing:

Assess system scalability with increasing users or passwords.

9. Concurrency and Thread Safety:

Verify safe concurrent access and implement synchronization.

10. Security Impact:

Check for performance impact of security measures.

7 My learnings

1. Encryption Techniques:

Gained understanding and practical experience in implementing encryption methods, specifically AES encryption using the Fernet module.

2. Database Integration:

Learned to integrate and interact with databases, exploring the use of SQLite for optional persistent storage.

3. Secure Password Handling:

Acquired knowledge on best practices for securely handling passwords, including encryption, decryption, and key management.

4. Command-Line Interface (CLI) Design:

Developed skills in designing a user-friendly command-line interface, focusing on simplicity and intuitiveness.

5. Random Password Generation:

Implemented a function for generating strong and random passwords, utilizing Python's random and string modules.

6. Exception Handling:

Explored effective error-handling techniques, providing informative messages for users and maintaining application robustness.

7. Modular Code Design:

Structured code in a modular manner, improving readability, maintainability, and allowing for potential future enhancements.

8. Educational Features:

Explored ways to educate users through informative prompts, encouraging the adoption of strong and secure password practices.

9. Future Development Considerations:

Developed an awareness of potential future enhancements, such as cloud integration, browser extensions, and multi-factor authentication.

10. Documentation Skills:

Enhanced documentation skills, including providing clear user instructions and developer documentation.

11. Security Awareness:

Increased awareness of security considerations in password management applications, including regulatory compliance and data protection.

12. Cross-Platform Development:

Considered cross-platform development, exploring the potential for extending the application to different operating systems.

13 Project Management:

Gained experience in project planning, prioritization, and the iterative development process.

8 Future work scope

1. **Advanced Authentication:** Integrate multi-factor and biometric authentication.
2. **Enhanced Encryption:** Explore advanced encryption methods for increased security.
3. **Cloud Integration:**Enable cloud synchronization for multi-device access.
4. **Browser Extension:**Develop browser extension for seamless integration.
5. **Cross-Platform Support:**Extend support to desktop, web, and mobile platforms.
6. **Secure Backup:**Implement secure backup and recovery mechanisms.
7. **Password Strength Analysis:**Provide feedback on password strength and recommendations.
8. **Usage Analytics:**Implement analytics for user behavior insights.
9. **Security Auditing:**Introduce periodic security audits for stored passwords.
10 **Customization Options**:Allow users to customize themes and settings.
10. **Integration with Identity Providers:**Explore integration with popular identity providers.
11. **Open Source Contribution:**Consider open-sourcing for community contributions.
12. **User Feedback Mechanism:**Implement a user feedback mechanism for improvements.

8.1 Code submission (https://github.com/ashish750706/upskillcampus.git)

https://github.com/ashish750706/upskillcampus.git

8.2 Report submission (Github link) :

https://github.com/ashish750706/passwordmanager_Ashish_USC_UCT.pdf.git