Industrial Internship Report on

Password Manager

Prepared by

Dyaga Sravya.

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.

1 Preface.....3
2 Introduction.....4
2.1 About UniConverge Technologies Pvt Ltd.....4
2.2 About upskill Campus.....9
2.3 Objective.....11
2.4 Reference.....11
2.5 Glossary.....11
3 Problem Statement.....12
4 Existing and Proposed solution.....13
5 Proposed Design/ Model.....16
5.1 High Level Diagram (if applicable).....16
5.2 Low Level Diagram (if applicable).....**Error! Bookmark not defined.**
5.3 Interfaces (if applicable).....18
6 Performance Test.....19
6.1 Test Plan/ Test Cases.....19
6.2 Test Procedure.....20
6.3 Performance Outcome.....21
7 My learnings.....22
8 Future work scope.....23

1 Preface

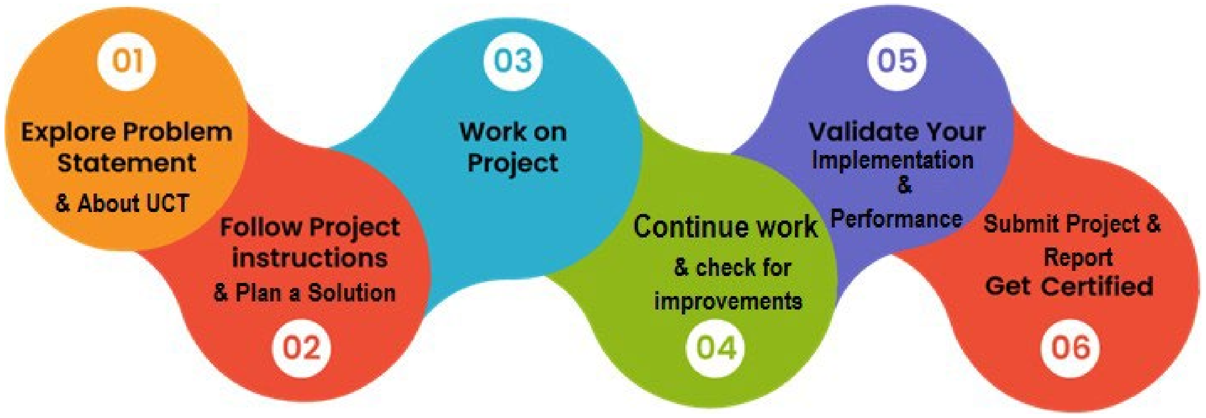
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned


01
Explore Problem Statement & About UCT
02
Follow Project instructions & Plan a Solution
03
Work on Project
04
Continue work & check for improvements
05
Validate Your Implementation & Performance
06
Submit Project & Report Get Certified

Your Learnings and overall experience.

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

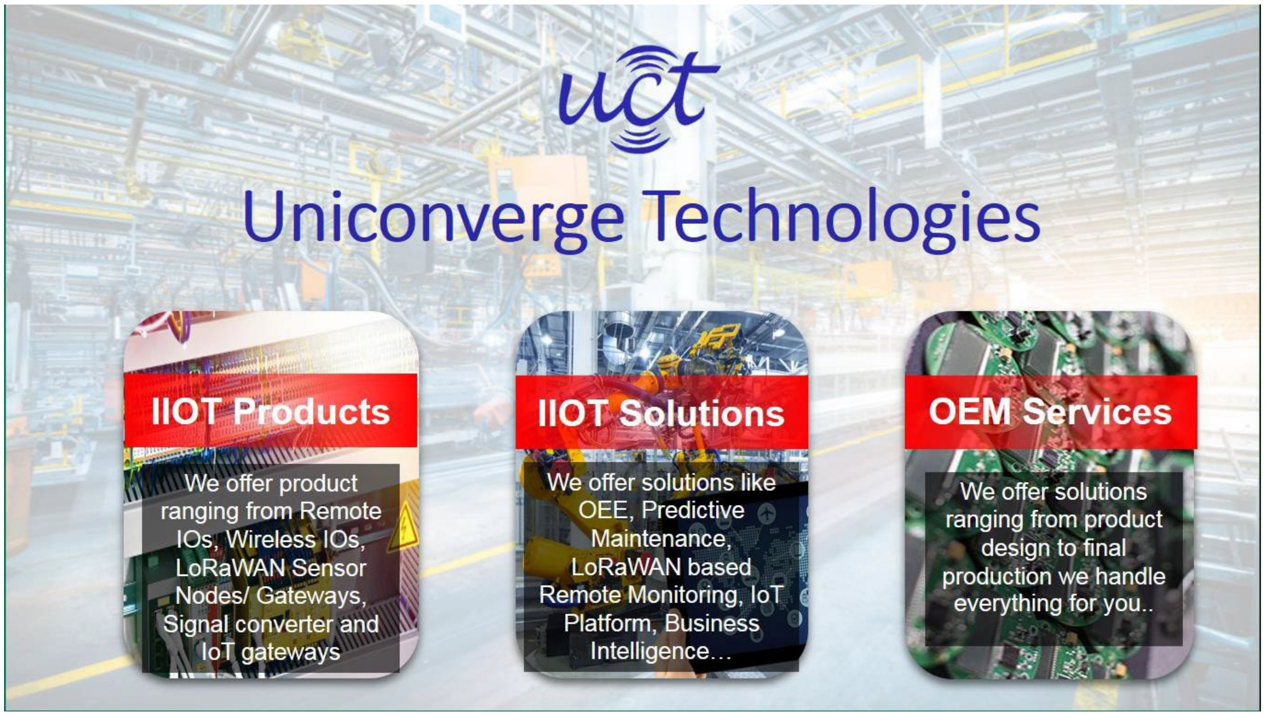
Your message to your juniors and peers.

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.



uct

Uniconverge Technologies

IIOT Products

We offer product ranging from Remote IOs, Wireless IOs, LoRaWAN Sensor Nodes/ Gateways, Signal converter and IoT gateways

IIOT Solutions

We offer solutions like OEE, Predictive Maintenance, LoRaWAN based Remote Monitoring, IoT Platform, Business Intelligence…

OEM Services

We offer solutions ranging from product design to final production we handle everything for you..

i. UCT IoT Platform (uct Insight)

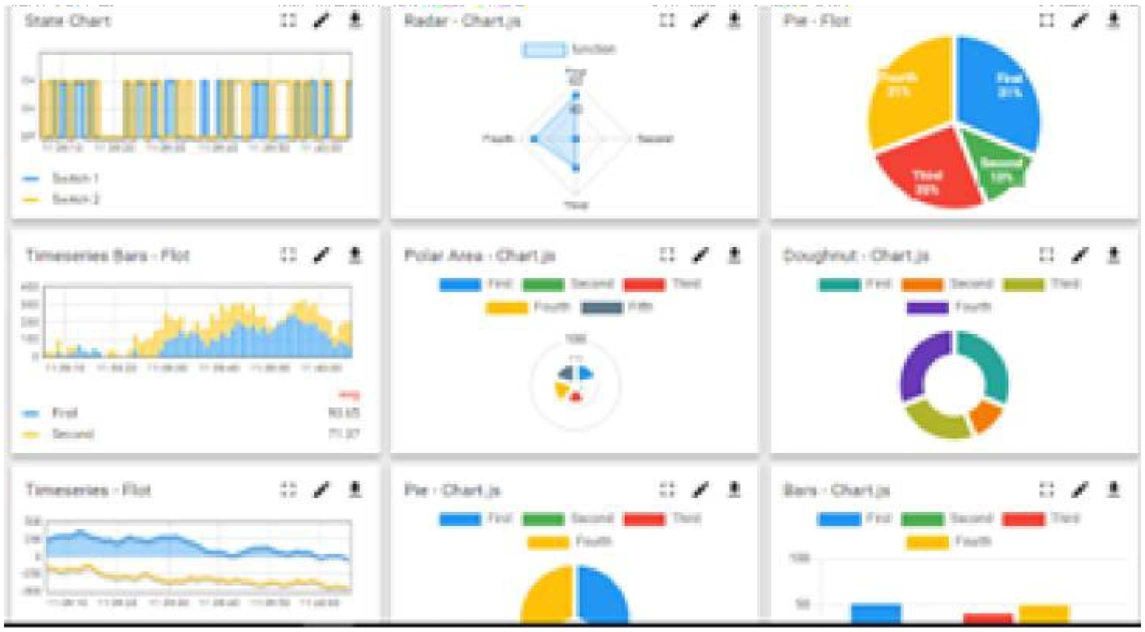
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

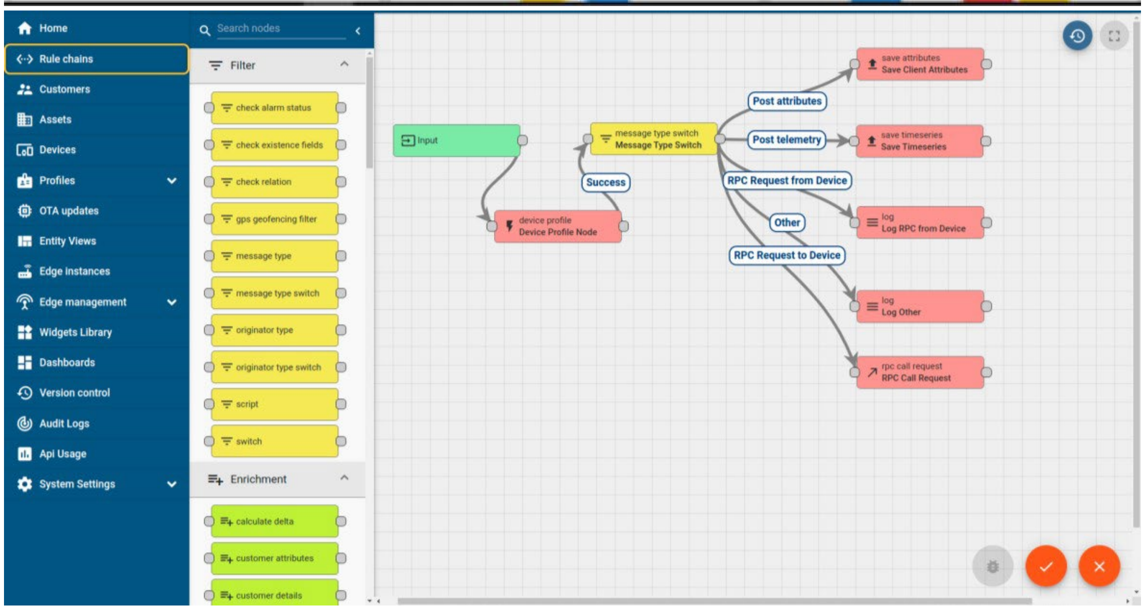
- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA

- 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





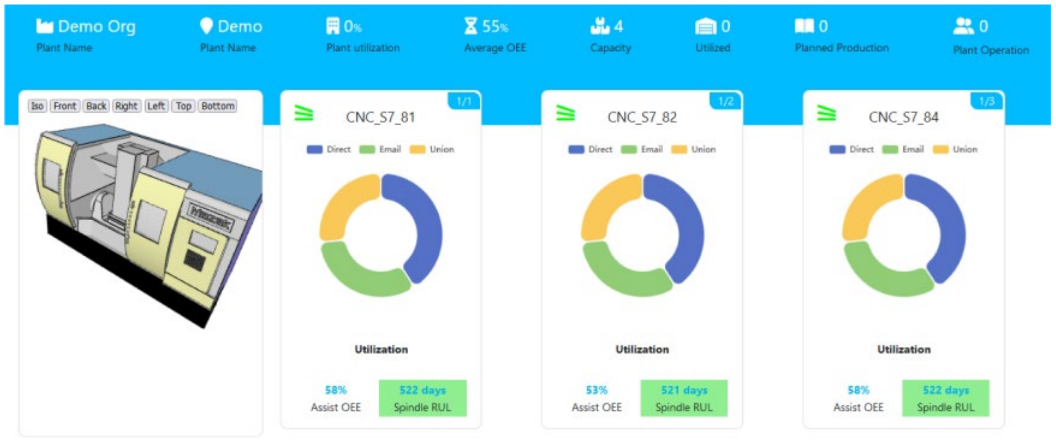
ii. Smart Factory Platform (FACTORY WATCH)

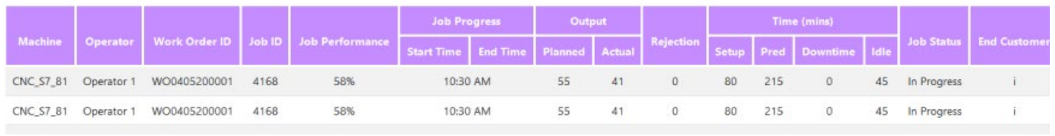
Factory watch is a platform for smart factory needs.

It provides Users/ Factory

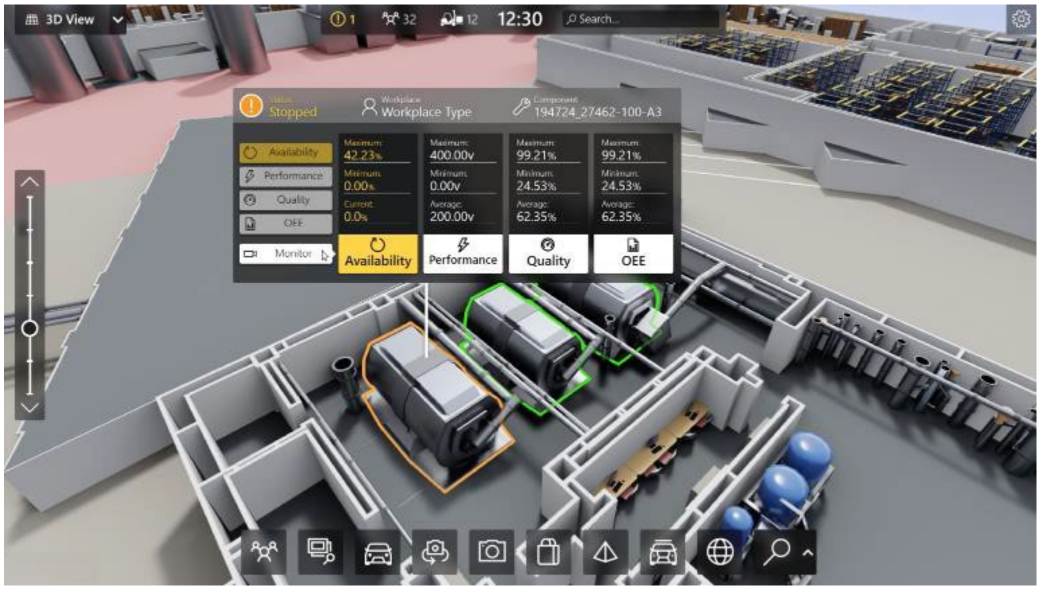
- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular 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

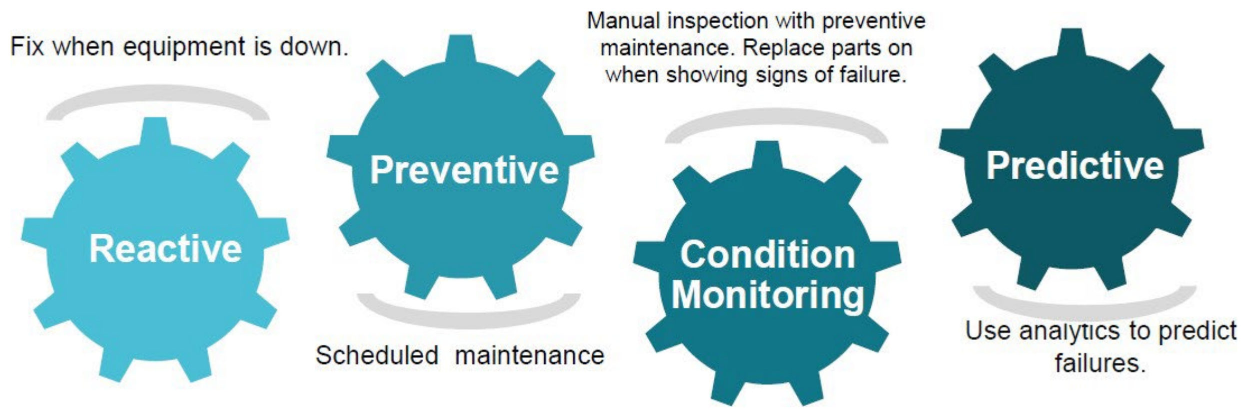


iii. LoRaWAN based Solution

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

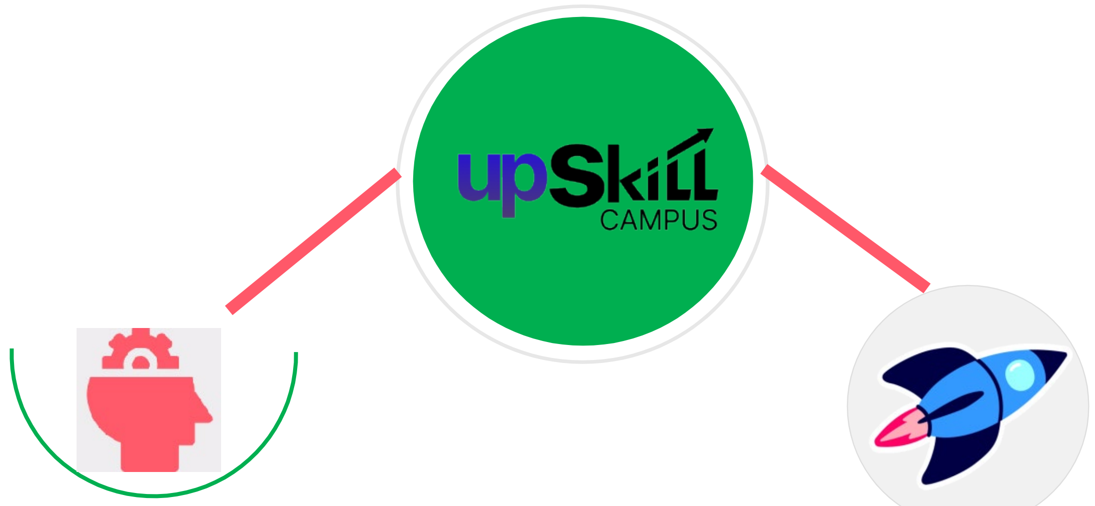
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.


Fix when equipment is down.
Reactive
Preventive
Scheduled maintenance
Manual inspection with preventive maintenance. Replace parts on when showing signs of failure.
Condition Monitoring
Predictive
Use analytics to predict failures.

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.

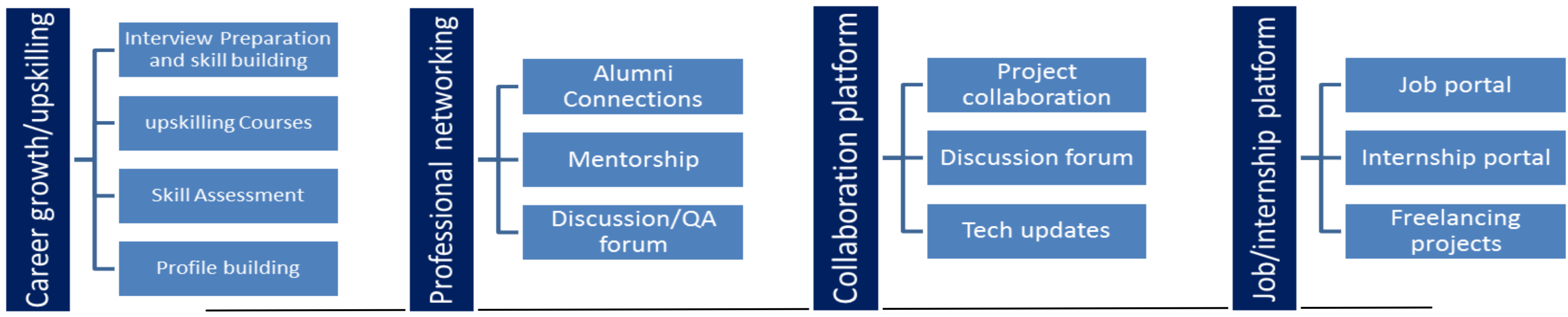
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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] https://www.tutorialspoint.com/python/index.htm , I have done collection of information of python

[2] https://www.w3schools.com/python/python_intro.asp, Retrieved some data which is super detailed

2.6 Glossary

Terms	Acronym
Object-Oriented programming	OOP
Python 3.8	Interpreter
Integrated development Environment	IDE
Jupyter Notebook	Coding platform

3 Problem Statement

In the assigned problem statement

To address this problem, we aim to develop Secure Pass, a robust and user-friendly password manager application in Python. Secure Pass will provide individuals and organizations with a secure and convenient solution to manage their passwords effectively. The primary objectives of this project include:

1. Secure Storage: Implementing strong encryption algorithms to securely store user passwords and sensitive information, ensuring protection against unauthorized access and data breaches.
2. Password Generation: Offering a built-in password generator to create strong and unique passwords that adhere to best practices for password security, such as length, complexity, and randomness.
3. Password Management: Providing features for users to organize and categorize their passwords, enabling easy retrieval and management of credentials for various accounts and services.
4. Cross-Platform Compatibility: Developing SecurePass as a cross-platform application, ensuring compatibility with major operating systems such as Windows, macOS, and Linux, to cater to a wide range of users.
5. User-Friendly Interface: Designing an intuitive and user-friendly interface for Secure Pass, making it easy for users to navigate and utilize the application's features effectively.
6. Advanced Features: Incorporating additional functionalities such as password strength assessment, password expiration reminders, import/export capabilities, and multi-factor authentication (MFA) support to enhance user experience and security.

By addressing these objectives, Secure Pass aims to empower users with a reliable and comprehensive solution for managing their passwords securely, thereby minimizing the risk of security breaches and unauthorized access to sensitive information

4 Existing and Proposed solution

Addressing these limitations requires careful planning, diligent implementation, and continuous improvement efforts throughout the development and maintenance lifecycle of the password manager project.

1. **Security Concerns**: While Python offers libraries for encryption and security, implementing robust security measures can be challenging. There's always a risk of vulnerabilities, especially if the implementation is not done correctly.
2. **Platform Dependency**: Python password managers may face limitations in terms of platform compatibility. While efforts can be made to ensure cross-platform compatibility, certain features or functionalities may not work seamlessly across all operating systems.
3. **Dependency on Third-party Libraries**: Python projects often rely on third-party libraries for various functionalities. Depending too heavily on external libraries can introduce dependencies and potential risks, such as compatibility issues or reliance on unsupported libraries.
4. **Performance**: Python, while known for its simplicity and readability, may not be the most performant language for certain tasks, especially when dealing with cryptographic operations or handling large datasets. This could impact the speed and responsiveness of the password manager, particularly when working with a large number of passwords or performing intensive encryption/decryption operations.
5. **Limited Integration**: Integrating with other tools or services may be limited due to the ecosystem or availability of libraries in Python. This could affect features like browser extensions, cloud synchronization, or integration with password strength assessment services.
6. **User Adoption and Trust**: Convincing users to trust a Python-based password manager might be challenging, especially when compared to established solutions written in other languages. Building trust and ensuring user adoption could require additional efforts in terms of security audits, transparency about the codebase, and user education.
7. **Maintenance and Updates**: Python projects may require regular maintenance and updates to address security vulnerabilities, compatibility issues, or evolving user requirements. Ensuring timely updates and ongoing support can be resource-intensive, especially for smaller development teams or individual contributors.
8. **Scalability**: Python projects may face scalability challenges, particularly when handling a large number of users or passwords. Ensuring scalability requires careful consideration of architecture, database management, and performance optimization techniques.

This solution demonstrates a basic password manager implementation in Python. It includes functionalities to generate, encrypt, and store passwords, as well as retrieve them when needed. It utilizes hashing for the master password and encryption for storing passwords securely. Additionally, it provides methods to save and load passwords from a file in JSON format. However, it's important to note that this is a simplified example, and a production-grade password manager would require additional features, such as user authentication, user interface, password strength evaluation, and more robust security measures.

Planning plays a crucial role in the development of a password manager project, providing several valuable benefits:

1. **Clear Objectives:** Planning helps in defining clear objectives and goals for the password manager project. This includes determining the features, functionalities, and security requirements that the password manager should fulfill.
2. **Risk Identification:** Planning allows for the identification and mitigation of potential risks associated with the project. This includes security risks such as vulnerabilities in encryption or storage mechanisms, as well as technical risks such as platform compatibility issues or scalability challenges.
3. **Resource Allocation:** Planning helps in effectively allocating resources such as time, budget, and manpower for the project. It enables the team to estimate the effort required for development, testing, and maintenance activities accurately.
4. **Timeline Estimation:** Planning aids in creating realistic timelines and milestones for the project. It helps in setting achievable deadlines for different phases of development, ensuring that the project progresses smoothly and is completed within the expected timeframe.
5. **Scope Definition:** Planning facilitates the definition of the project scope, outlining the boundaries and deliverables of the password manager application. It helps in determining which features are essential for the initial release and which can be included in future iterations.
6. **User Needs Analysis:** Planning involves conducting thorough research and analysis of user needs and preferences. It allows for the identification of user requirements, usability issues, and feature priorities, ensuring that the password manager meets the expectations of its intended users.
7. **Quality Assurance:** Planning includes defining strategies for quality assurance and testing. It helps in establishing criteria for evaluating the performance,

reliability, and security of the password manager, ensuring that it meets industry standards and best practices.

8. **Documentation:** Planning involves creating documentation such as design specifications, user manuals, and development guidelines. It provides a reference point for the project team and stakeholders, facilitating communication and collaboration throughout the development process.

4.1 Code submission (Github link)

https://github.com/Dyagasravya/upskillcampus1.git

code link:

https://github.com/Dyagasravya/upskillcampus1/blob/main/Password%20Manager%20(3)%20-%20Copy.txt

4.2 Report submission (Github link) :

https://github.com/Dyagasravya/upskillcampus1.git

5 Proposed Design/ Model

Given more details about design flow of your solution. This is applicable for all domains. DS/ML Students can cover it after they have their algorithm implementation. There is always a start, intermediate stages and then final outcome.

5.1 High Level Diagram (if applicable)

A high-level design for a password manager project involves outlining the architecture, components, and interactions of the system. Here's a basic high-level design for a password manager project:

1. **User Interface (UI):**
   - The UI provides an interface for users to interact with the password manager. It includes features such as:
     - Login/Authentication: Users enter their master password to access the password manager.
     - Password Management: Users can add, view, edit, and delete passwords for different accounts.
     - Password Generation: Users can generate strong and unique passwords for new accounts.
     - Category/Tagging: Users can organize passwords into categories or add tags for easier management.
     - Search: Users can search for specific passwords by website or account name.
     - Settings: Users can configure preferences such as auto-lock timeout, password strength requirements, etc.
2. **Backend Services:**
   - Authentication Service: Handles user authentication and session management.
   - Encryption Service: Responsible for encrypting and decrypting passwords using strong encryption algorithms.
   - Password Storage Service: Stores encrypted passwords securely, possibly in a database or encrypted file.
   - Password Generator Service: Generates strong and random passwords based on user preferences.
   - Category/Tagging Service: Manages categories and tags for organizing passwords.
   - Search Service: Provides functionality for searching passwords based on criteria.
   - Settings Service: Manages user preferences and settings.
3. **Security:**

- Encryption: Utilizes strong encryption algorithms (e.g., AES) to encrypt passwords before storing them.
- Key Management: Manages encryption keys securely, possibly using key derivation functions (e.g., PBKDF2).
- Secure Communication: Ensures secure communication between the client and server components, possibly using HTTPS.
- Authentication: Implements secure user authentication mechanisms, such as salted hashing for storing master passwords.

4. **Data Storage:**

- Database: Uses a secure database to store encrypted passwords and associated metadata.
- File System: Alternatively, encrypted files can be used to store password data securely.

5. **Cross-Platform Compatibility:**

- Client Application: Develops client applications for various platforms (e.g., desktop, web, mobile) using appropriate technologies (e.g., PyQt/PySide for desktop, Flask/Django for web, Flutter/React Native for mobile).
- Synchronization: Implements synchronization functionality to ensure that password data remains consistent across different devices and platforms.

6. **Integration:**

- Browser Integration: Offers browser extensions/plugins for seamless integration with web browsers, allowing users to autofill passwords and capture new credentials easily.
- Third-party Integration: Integrates with external services (e.g., password strength checkers, breach monitoring services) to enhance security and usability.

7. **Backup and Recovery:**

- Backup: Provides options for users to backup encrypted password data securely, either locally or in the cloud.
- Recovery: Implements recovery mechanisms in case of data loss or corruption, possibly through backup restoration or account recovery procedures.

8. **Logging and Monitoring:**

- Logging: Logs system activities, user actions, and security events for auditing and troubleshooting purposes.
- Monitoring: Monitors system performance, resource usage, and security indicators to detect and respond to potential threats or issues.

This high-level design outlines the main components and functionalities of a password manager project, providing a roadmap for the detailed design and implementation phases

5.2 Interfaces (if applicable)

This code sets up a basic `Password Manager` class with methods to add, retrieve, update, and list passwords. The `main` function utilizes `curses` to create a simple command-line interface for interacting with the password manager.

You can expand upon this by adding encryption for storing passwords securely, more advanced features like generating passwords, or incorporating a database for persistent storage. Remember, security is critical when dealing with passwords, so handle them with care and consider security best practices.

6 Performance Test

6.1 Test Plan/ Test Cases

1. **Functional Testing:**
   - **Add Password Functionality:**
     - Test adding a new password with a unique key.
     - Test adding a new password with a duplicate key (should update the existing password).
     - Test adding a password with special characters.
   - **Get Password Functionality:**
     - Test retrieving an existing password with a valid key.
     - Test retrieving a non-existing password.
   - **Update Password Functionality:**
     - Test updating an existing password with a valid key.
     - Test updating a non-existing password.
   - **List Passwords Functionality:**
     - Test listing all passwords when the password manager is empty.
     - Test listing all passwords when there are multiple passwords stored.
2. **Interface Testing:**
   - Test all menu options presented in the interface:
     - Verify that selecting "Add Password" prompts for key and password input.
     - Verify that selecting "Get Password" prompts for a key input and displays the password if found.
     - Verify that selecting "Update Password" prompts for key and new password input and updates the password if the key exists.
     - Verify that selecting "List Passwords" displays all stored passwords.
     - Verify that selecting "Quit" exits the program.
3. **Boundary Testing:**
   - Test the maximum length of key and password inputs.
   - Test with empty key and password inputs.
4. **Security Testing:**
   - Verify that passwords are not stored in plain text but are securely encrypted.
   - Test for any potential vulnerabilities such as SQL injection (if using a database) or other injection attacks.
5. **Error Handling:**
   - Test for appropriate error messages when encountering invalid inputs or operations.
   - Test for graceful handling of unexpected errors.

6. **Performance Testing:**
    - Test the performance of adding, retrieving, updating, and listing passwords with a large number of entries.
    - Measure the time taken for each operation and ensure it meets acceptable performance criteria.
7. **Compatibility Testing:**
    - Test the password manager on different operating systems (if applicable).
    - Test the password manager with different versions of Python (if applicable).
8. **Usability Testing:**
    - Evaluate the user-friendliness of the interface.
    - Obtain feedback from users to identify any areas for improvement in terms of usability and intuitiveness.
9. **Regression Testing:**
    - Re-run previously passed tests after making changes to ensure that existing functionality has not been affected.
10. **Integration testing:**
    - Test the interaction between different components of the password manager (e.g., interface, password storage logic).
11. **Accessibility Testing (Optional):**
    - Ensure that the password manager is accessible to users with disabilities by testing with screen readers or other assistive technologies.
12. **Backup and Recovery Testing:**
    - Test the

6.2 Test Procedure

6.2.1 1. Functional Testing:

1. Open the password manager application.
2. Select the "Add Password" option from the menu.
3. Enter a unique key and password.
4. Verify that the password is successfully added to the manager.
5. Repeat steps 2-4 with a duplicate key and ensure that the password is updated instead of creating a new entry.
6. Repeat step 3 with passwords containing special characters.

6.2.1.1 Get Password Functionality:

1. Open the password manager application.
2. Select the "Get Password" option from the menu.
3. Enter an existing key.
4. Verify that the correct password is displayed.
5. Repeat step 3 with a non-existing key and verify that appropriate feedback is provided.

6.2.1.2 Update Password Functionality:

1. Open the password manager application.
2. Select the "Update Password" option from the menu.
3. Enter an existing key and a new password.
4. Verify that the password is updated successfully.
5. Repeat step 3 with a non-existing key and ensure that appropriate feedback is provided.

6.2.1.3 List Passwords Functionality:

1. Open the password manager application.
2. Select the "List Passwords" option from the menu.
3. Verify that all stored passwords are displayed correctly.
4. Repeat step 2 when the password manager is empty and ensure that appropriate feedback is provided.

6.2.2 2. Interface Testing:

1. Open the password manager application.
2. Verify that the menu is displayed correctly with all options.
3. Test each menu option and verify that they prompt for inputs or display information as expected.
4. Select the "Quit" option and ensure that the application exits gracefully.

6.3 Performance Outcome

Execute the action plan according to the established timelines. This may involve implementing process improvements, introducing new technologies or systems, providing training or development opportunities for employees, or making organizational changes.

7 My learnings

1. **Clear Objectives Drive Success:** The project's success hinges on clearly defined objectives that are specific, measurable, achievable, relevant, and time-bound (SMART). These objectives serve as the guiding principles throughout the project lifecycle.
2. **Data-Driven Decision Making:** Data analysis is instrumental in identifying areas for improvement and tracking progress towards performance outcomes. Leveraging key performance indicators (KPIs) and metrics enables informed decision-making and facilitates course corrections as needed.
3. **Effective Communication is Essential:** Open and transparent communication channels foster collaboration and alignment among stakeholders. Regular updates, feedback mechanisms, and clear expectations ensure everyone remains informed and engaged throughout the project.
4. **Continuous Monitoring and Adaptation:** The project landscape is dynamic, requiring continuous monitoring of progress and flexibility to adapt strategies as circumstances evolve. Regular performance assessments enable timely adjustments to optimize outcomes and mitigate risks.
5. **Investment in Resources and Training:** Adequate resources, including personnel, technology, and financial support, are essential for project success. Investing in training and development empowers team members to excel in their roles and contribute effectively to achieving project objectives.

8 Future work scope

1. **Development Time:** Determine the time required to develop and implement new features or enhancements to the password manager. This includes planning, coding, testing, and deployment phases. It's important to set realistic timelines based on the complexity of the features and available resources.
2. **User Adoption and Feedback Cycle:** After introducing new features or updates, allocate time for users to adopt and provide feedback on their experience. This feedback loop is crucial for iterating and refining the features based on user needs and preferences.
3. **Security Audits and Updates:** Regular security audits and updates are vital for maintaining the integrity and robustness of the password manager. Allocate time for conducting audits, addressing vulnerabilities, and implementing security patches to ensure ongoing protection of user data.
4. **Compliance Requirements:** Consider any regulatory or compliance requirements that may impact the project timeline. Compliance standards such as GDPR, HIPAA, or PCI DSS may necessitate additional time for ensuring adherence to data protection and privacy regulations.
5. **Integration with Other Systems:** If the password manager needs to integrate with other systems or platforms, account for the time required for integration testing, troubleshooting compatibility issues, and coordinating with third-party vendors or partners.