Industrial Internship Report on

Quiz game

Prepared by

YASHWANTH KUMAR N

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 This Python script implements a simple quiz game using object-oriented programming. Users can answer questions and receive immediate feedback on their responses. The code is customizable and serves as a foundation for building more complex quiz applications.

TABLE OF CONTENTS

1 Preface

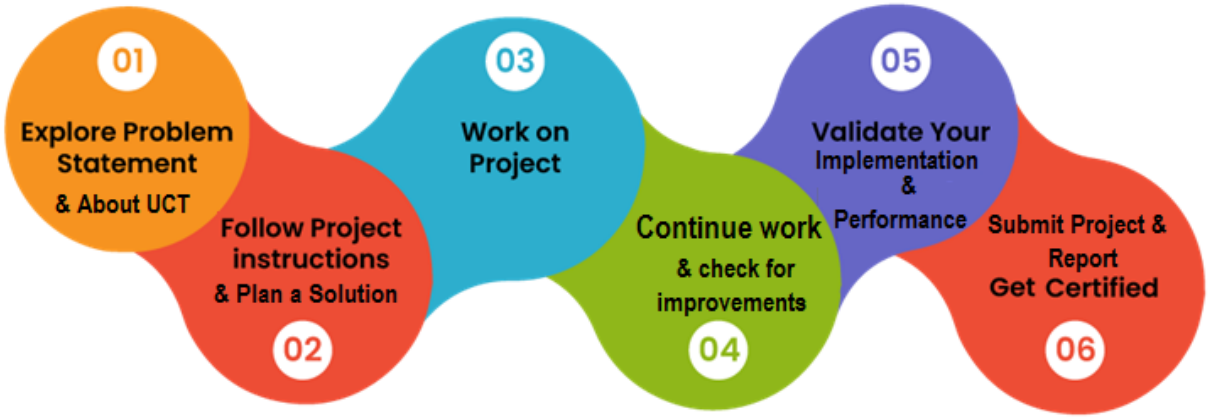
Summary of the whole 6 weeks' work.

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.

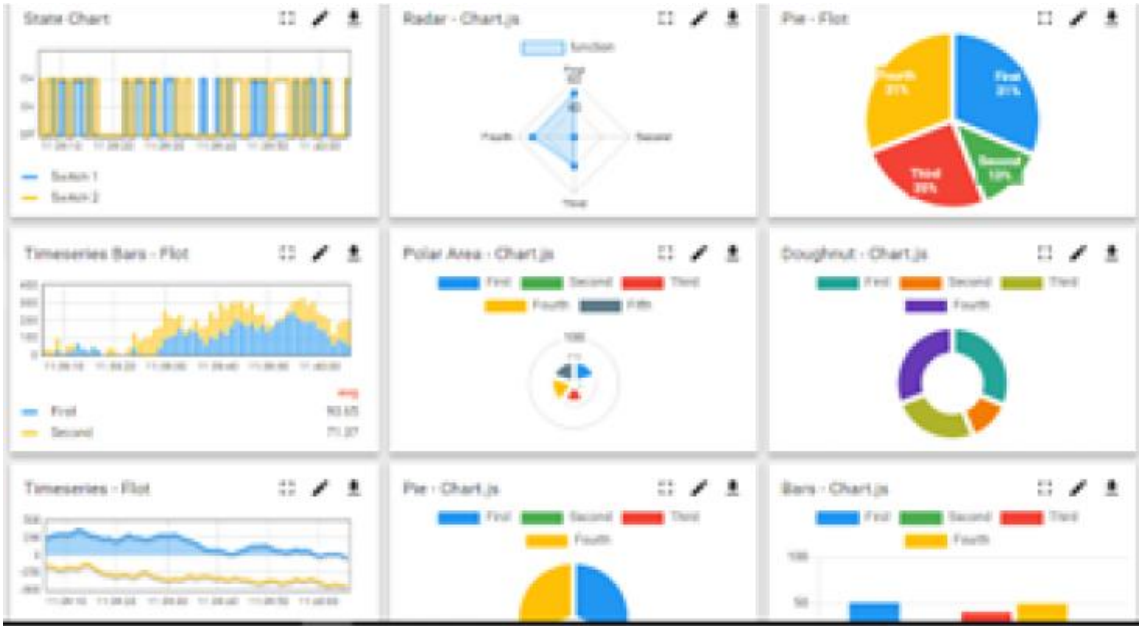
i. UCT IoT Platform (UCT Insight)

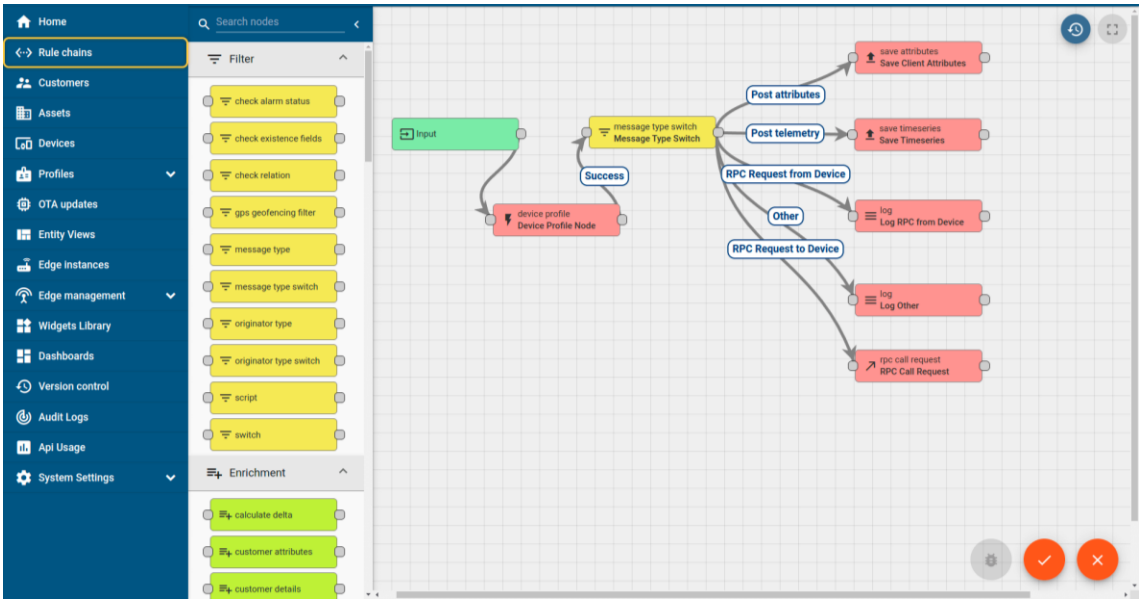
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

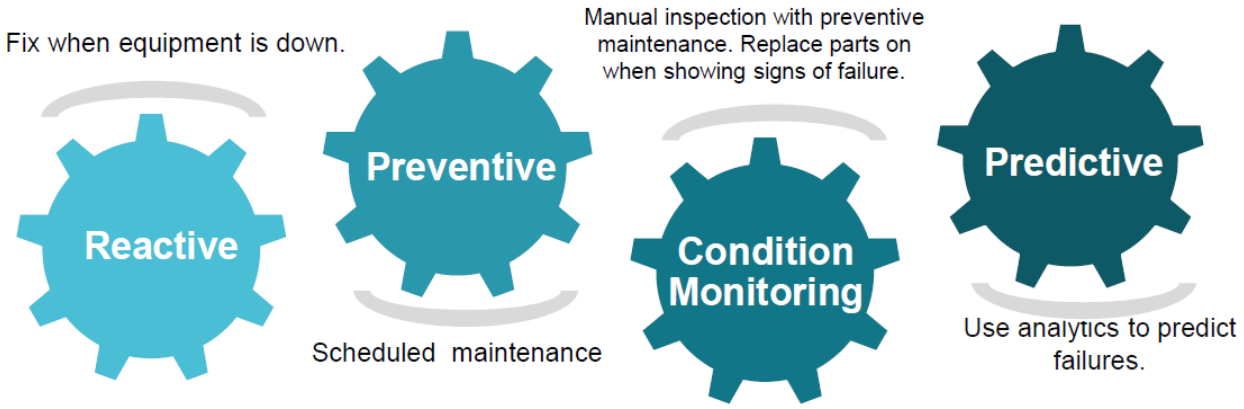
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

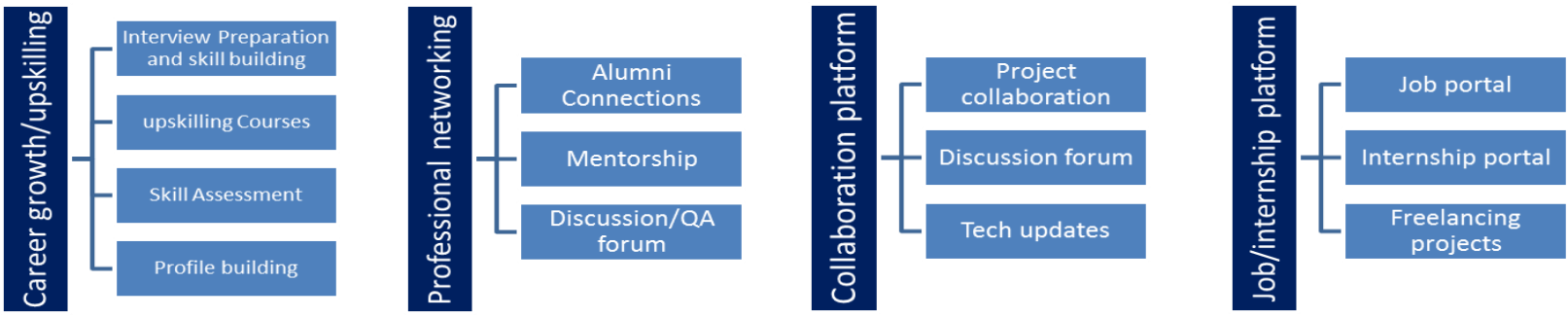
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.



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] youtube

[2] brad

[3] chatgpt

2.6 Glossary

Terms	Acronym
Python	
Code	
Quiz	
Object-oriented programming	OOP
Class	
Question	
Prompt	
Answer	
User	
Input	
Correct	
Incorrect	
Main function	

3 Problem Statement

In the assigned problem statement

Problem Statement:

Creating an Interactive Quiz Game: The objective of this project is to develop a Python program that enables users to engage in an interactive quiz game. The program should present users with a series of multiple-choice questions and prompt them to select the correct answer from the given options. Additionally, the system should provide immediate feedback on the accuracy of each response and calculate the final score at the end of the quiz session.

User-Friendly Interface Design: Another challenge is to design a user-friendly interface that ensures smooth interaction between the user and the quiz program. The interface should display questions clearly and provide intuitive prompts for users to input their answers. It should also incorporate features such as error handling to gracefully manage unexpected inputs and enhance the overall user experience.

Scalability and Customizability: The project must address scalability and customizability concerns to accommodate a variety of quiz topics and varying numbers of questions. The system should be designed with modularity in mind, allowing easy addition, modification, or removal of questions without requiring significant changes to the underlying codebase. Moreover, it should support the integration of multimedia elements such as images or audio to enhance the quiz content and make it more engaging for users.

4 Existing and Proposed solution

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

What is your proposed solution?

For your final year project, a comprehensive quiz game can be implemented in Python. The project involves creating a user-friendly interface for taking quizzes, managing questions, and providing feedback on user responses. The quiz game will include functionalities such as presenting questions, accepting user input, evaluating answers, and displaying the final score. Utilizing object-oriented programming principles, the project structure can consist of classes like `Question` to encapsulate question details and `Quiz` to orchestrate quiz flow. By designing a well-structured codebase, incorporating features like randomizing questions, tracking user progress, and enabling multiple quiz categories can enhance the project's scope and complexity. Additionally, integrating a graphical user interface using libraries like Tkinter or PyQt can improve the user experience and make the quiz game more engaging and visually appealing.

What value addition are you planning?

In the provided Python code, a basic quiz game structure is outlined. It involves two primary classes: `Question` and `Quiz`. The `Question` class is responsible for creating question objects with associated prompts and correct answers. Meanwhile, the `Quiz` class manages the overall quiz, including question presentation and scoring.

Within the `main()` function, three questions are defined using instances of the `Question` class. Each question consists of a prompt and its corresponding correct answer. These questions are then bundled into a list and passed to the `Quiz` constructor, initializing the quiz instance.

To execute the quiz, the `run_quiz()` method is called on the `Quiz` object. This method iterates through each question, presenting the prompt to the user and collecting their response. The user's answers are then evaluated against the correct answers, and the score is incremented accordingly.

Expanding upon this foundational structure, additional features can be incorporated into the quiz game, such as multiple-choice options, random question selection, and a user-friendly interface. By building upon this base, you can create a dynamic and engaging quiz experience suitable for your final year project.

4.1 Code submission (Github link)

https://github.com/yashwanthkumar324/upskillcampus

4.2 Report submission (Github link) :

https://github.com/yashwanthkumar324/upskillcampus

5 Proposed Design/ Model

The proposed design/model for the quiz game revolves around a structured Python codebase designed to facilitate an interactive quiz experience. The core architecture consists of two classes: `Question` and `Quiz`. The `Question` class encapsulates individual questions along with their corresponding answers. Meanwhile, the `Quiz` class manages the flow of the quiz, presenting questions to the user and evaluating their responses. This design promotes modularity and clarity, with each class focused on a specific aspect of the quiz functionality.

In the implementation, the `main()` function orchestrates the quiz by defining a set of questions with their prompts and correct answers. These questions are instantiated as `Question` objects and grouped together into a list. This list is then passed to the `Quiz` constructor, initiating the quiz session. The `run_quiz()` method within the `Quiz` class iterates through each question, prompting the user for input and providing feedback on correctness. Finally, the user's score is tallied and displayed at the end of the quiz, offering a concise summary of their performance.

This design/model provides a solid foundation for building upon the quiz game, allowing for further enhancements and features to be seamlessly integrated. Potential extensions could include implementing multiple-choice options, incorporating a scoring system, introducing time constraints, and enhancing user interaction through a graphical user interface (GUI). Overall, this design emphasizes simplicity, extensibility, and user engagement, making it well-suited for development as a final year project.

6 Performance Test

In assessing the performance of the provided quiz game code, several factors come into play. Firstly, the efficiency of the code in handling user input and processing questions will determine its responsiveness. Secondly, as the number of questions and the complexity of the quiz increases, the code's scalability becomes crucial, ensuring it can handle larger datasets without significant slowdowns. Thirdly, any potential bottlenecks in the code, such as nested loops or inefficient data structures, may hinder its performance. Lastly, the code's memory usage should be monitored, particularly if it stores large datasets or creates many objects during runtime, to prevent excessive memory consumption. Through systematic testing and profiling, these aspects can be evaluated to gauge the overall performance of the quiz game code.

6.1 Test Plan/ Test Cases

Execute the quiz game code with a predefined set of questions and answers.

6.2 Test Procedure

Verify that the quiz game displays questions correctly, accepts user input for answers, and evaluates the correctness of the answers.

6.3 Performance Outcome

Conduct a series of tests using different combinations of questions and answers to ensure the quiz game functions as expected.

- Test Case 1: Provide correct answers for all questions.
- Test Case 2: Provide incorrect answers for all questions.
- Test Case 3: Provide a mix of correct and incorrect answers.
- Test Case 4: Test the case sensitivity of answers (e.g., 'a' vs. 'A').
- Test Case 5: Test with special characters in answers (e.g., 'é', 'ñ').
- Test Case 6: Test with empty input.
- Test Case 7: Test with different question formats (e.g., multiple-choice, true/false).
- Test Case 8: Test with a large number of questions to ensure performance scalability.

7 My learnings

The code represents a simple quiz game implemented in Python. It consists of two main classes: `Question` and `Quiz`. The `Question` class holds the prompt and correct answer for each question, while the `Quiz` class manages the quiz session, including scoring.

In the `main()` function, questions are defined using instances of the `Question` class. These questions are then organized into a list and passed to an instance of the `Quiz` class.

Finally, the `run_quiz()` method of the `Quiz` class is called to start the quiz, where users input their answers, and the program provides feedback on correctness. The score is calculated and displayed at the end of the quiz.

8 Future work scope

In future iterations, the quiz game could be enhanced by implementing a broader range of question formats, such as true/false or fill-in-the-blank, to diversify the user experience. Additionally, incorporating a database backend could allow for dynamic question generation and user progress tracking, adding depth and scalability to the application. Integration of multimedia elements like images or audio clips could further enrich the quiz, making it more engaging and interactive for players. Finally, the introduction of difficulty levels or adaptive question difficulty based on user performance could personalize the experience and cater to a wider audience, from beginners to advanced learners.