**Industrial Internship Report on**

**PYTHON QUIZ GAME**

**Prepared by**

**PALAK MAHAJAN**

|  |
| --- |
| *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 Python Quiz Game  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 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)

[2.3 Objective 9](#_Toc139702810)

[2.4 Reference 9](#_Toc139702811)

[2.5 Glossary 10](#_Toc139702812)

[3 Problem Statement 11](#_Toc139702813)

[4 Existing and Proposed solution 12](#_Toc139702814)

[5 Proposed Design/ Model 13](#_Toc139702815)

[5.1 High Level Diagram (if applicable) 13](#_Toc139702816)

[5.2 Low Level Diagram (if applicable) 13](#_Toc139702817)

[5.3 Interfaces (if applicable) 13](#_Toc139702818)

[6 Performance Test 14](#_Toc139702819)

[6.1 Test Plan/ Test Cases 14](#_Toc139702820)

[6.2 Test Procedure 14](#_Toc139702821)

[6.3 Performance Outcome 14](#_Toc139702822)

[7 My learnings 15](#_Toc139702823)

[8 Future work scope 16](#_Toc139702824)

# Preface

Quiz game is a Python project that quizzes users on various topics. It reads questions and answers from a file or database, presents them to the user, and keeps track of their score.

Scope: The scope of this project involves designing a user interface to display questions and collect user answers, implementing a database or file system to store quiz data, and developing a scoring algorithm to track the user's progress and calculate their final score.



It was a great learning experience for me. I learned how to write code to ask questions, accept user input, and keep score. It was really cool to see the game come to life and interact with the players. I'm excited to learn more.

I just wanted to say that learning Python is an awesome journey. Don't worry if things seem challenging at first, just keep practicing and experimenting. Remember to have fun along the way

I faced a few challenges while creating the project. One of the main ones was figuring out how to structure the code and organize the questions and answers. It took some time to design the logic and make sure everything flowed smoothly. I also had to debug some errors along the way, but that's all part of the learning process. Overall, it was a great experience and I learned a lot from it.

# Introduction

## 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.



1. UCT IoT Platform **(****)**

**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

1. **Smart Factory Platform (****)**

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.

1.  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.

1. 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.



## 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.

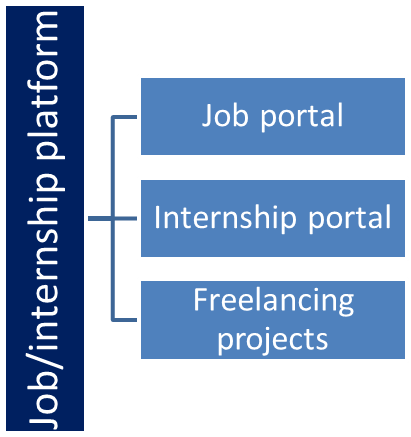
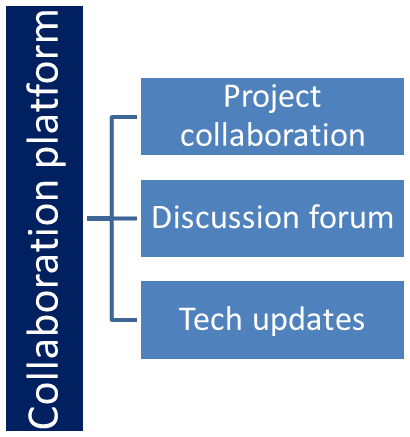
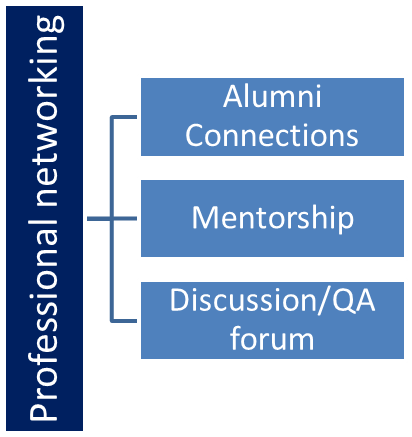
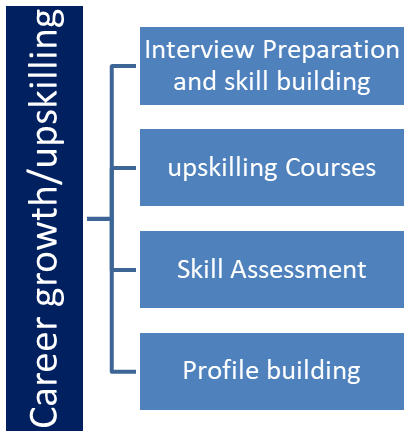
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

<https://www.upskillcampus.com/>

upSkill Campus aiming to upskill 1 million learners in next 5 year



## 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.

## 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.

# Problem Statement

Create a Python quiz game that asks the user a series of multiple-choice questions. The game should keep track of the user's score and display it at the end. Each question should have four options, and the user should be able to input their answer. The game should provide feedback on whether the answer is correct or incorrect. Make sure to include a variety of questions to challenge the player.

# Existing and Proposed solution

Existing Solution:

For my Python quiz game, I created a function that stores the questions, options, and correct answers in a list of dictionaries. The game prompts the user with each question, accepts their input, and checks if it matches the correct answer. The user's score is incremented for each correct answer, and at the end of the game, their final score is displayed.

Proposed Solution:

To make the quiz game more engaging, I could add a timer feature that limits the amount of time the user has to answer each question. Additionally, I could implement a leaderboard that stores and displays the highest scores achieved by different players. This would add a competitive element and encourage players to improve their scores. I could also consider adding different difficulty levels or categories of questions to cater to different skill levels and interests.

## Code submission (Github link)

## Report submission (Github link) : first make placeholder, copy the link.

# Proposed Design/ Model

1. User Interface:

- Display a welcome message and instructions to the player.

- Present each question with the corresponding options.

- Accept user input for their answer.

- Display feedback on whether the answer is correct or incorrect.

- Keep track of the user's score.

2. Question Bank:

- Create a list of dictionaries, where each dictionary represents a question.

- Each question dictionary should include the question itself, options, and the correct answer.

3. Game Logic:

- Randomly select questions from the question bank to present to the player.

- Compare the user's answer with the correct answer and update the score accordingly.

- Repeat the process until all questions have been asked.

4. Timer (Optional):

- Implement a timer feature to limit the time the player has to answer each question.

- Display a countdown timer for each question.

- If the player doesn't answer within the time limit, consider it as an incorrect answer.

5. Leaderboard (Optional):

- Create a leaderboard to store and display the highest scores achieved by different players.

- Update the leaderboard after each game and show the player's ranking.

# Performance Test

1. Load Testing: Simulate a high number of concurrent users accessing the game to evaluate how it handles the load. This test helps identify any performance, bottlenecks or limitations in the system.

2. Stress Testing: Push the game to its limits by overwhelming it with a significantly higher load than it is designed to handle. This test helps determine the breaking point of the game and how it recovers from failures.

3. Scalability Testing: Assess the game's ability to handle an increasing number of users by gradually increasing the load and monitoring its performance. This test helps identify if the game can scale effectively as the user base grows.

4. Response Time Testing: Measure the time it takes for the game to respond to user actions, such as answering a question or navigating between screens. This test helps evaluate the game's responsiveness and user experience.

5. Network Latency Testing: Introduce network delays or simulate different network conditions to assess how the game performs in real-world network scenarios. This test helps identify any issues related to network latency or connectivity.

Remember to monitor system resources during performance testing, such as CPU usage, memory consumption, and network bandwidth. This will help identify any resource constraints that may impact the game's performance.

## Test Plan/ Test Cases

1. User Interface:

a. Verify that the welcome message is displayed correctly.

b. Check if the instructions are clear and easy to understand.

c. Ensure that each question is presented with the corresponding options.

d. Validate that the user's input for their answer is accepted correctly.

e. Confirm that the feedback on the correctness of the answer is displayed accurately.

f. Test if the user's score is calculated correctly.

2. Question Bank:

a. Check if the questions are stored correctly in the question bank.

b. Validate that each question has the correct options and the correct answer.

c. Ensure that the questions are randomized or presented in a random order.

3. Game Logic:

a. Verify that questions are selected randomly from the question bank.

b. Validate that the user's answer is compared correctly with the correct answer.

c. Check if the score is updated accurately based on the user's answer.

d. Test if the game ends when all questions have been asked.

4. Timer (Optional):

a. Check if the timer starts correctly for each question.

b. Validate that the timer counts down accurately.

c. Test if the user's answer is accepted or rejected based on the timer expiration.

5. Leaderboard (Optional):

a. Check if the leaderboard is updated correctly after each game.

b. Validate that the player's score is recorded accurately.

c. Test if the leaderboard displays the highest scores and ranks the players correctly

## Test Procedure

1. Test Setup:

a. Install the game on the test environment.

b. Set up any necessary dependencies or configurations.

c. Ensure that the question bank is populated with test questions.

2. Test Execution:

a. Launch the game and verify that the welcome message is displayed correctly.

b. Follow the game instructions and navigate through the user interface.

c. Answer the questions using different options and validate the feedback.

d. Test the game logic by answering questions correctly and incorrectly.

e. If applicable, test the timer functionality by allowing it to expire.

f. Complete a full game session and verify that the score is calculated accurately.

g. If applicable, test the leaderboard functionality by checking if scores are recorded and displayed correctly.

3. Test Boundary Conditions:

a. Test the game with the minimum and maximum number of questions in the question bank.

b. Verify that the game handles invalid user input gracefully.

c. Test the game with a large number of concurrent users to assess its performance.

d. If applicable, test the game's scalability by gradually increasing the user load.

4. Test Compatibility:

a. Test the game on different devices and platforms (e.g., desktop, mobile).

b. Verify that the game functions properly on different screen resolutions.

c. Test the game on different operating systems and browsers, if applicable.

5. Test Error Handling:

a. Intentionally introduce errors in the game code or configuration and verify that appropriate error messages are displayed.

b. Test the game's ability to recover from errors and continue functioning properly.

6. Test Reporting:

a. Document any issues or defects encountered during testing.

b. Provide clear and detailed steps to reproduce any bugs or errors.

c. Include screenshots or videos if necessary to illustrate the issues.

## Performance Outcome

1. Responsiveness: Evaluate how quickly the game responds to user actions, such as selecting an answer or navigating through the interface. A fast and smooth response enhances the user experience.

2. Load Handling: Test the game's ability to handle a large number of concurrent users. This will help identify any performance bottlenecks or issues that arise when the game is under heavy load.

3. Resource Utilization: Monitor the game's resource usage, including CPU, memory, and network bandwidth. Ensure that the game operates efficiently without consuming excessive resources.

4. Scalability: Assess the game's ability to handle an increasing number of users without a significant degradation in performance. This is important if you expect the game to accommodate a growing user base.

5. Error Handling: Evaluate how the game handles errors and exceptions. It should gracefully handle unexpected situations and provide appropriate error messages to users.

6. Latency: Measure the time it takes for the game to respond to user actions, such as fetching questions, calculating scores, or displaying feedback. Lower latency leads to a more seamless and enjoyable gameplay experience.

# My learnings

During the project, I also learned about different ways to structure and organize code, like using functions and classes. It was fascinating to see how these concepts come together to create an interactive and engaging game experience.

I also learned a lot about error handling and how to make the game more robust by handling unexpected user inputs. It's always important to consider different scenarios and make sure the game can handle them gracefully.

Overall, the Python quiz game project was a great learning experience for me, and I'm excited to continue growing my knowledge and skills in Python and game development.

# Future work scope

1. Expand Question Database: Add more questions to the quiz game to increase its variety and replayability. You can categorize questions into different topics or difficulty levels to provide a more tailored experience for players.

2. Add a Timer: Implement a timer feature to add an element of challenge and excitement to the game. Players will have a limited amount of time to answer each question, increasing the sense of urgency and making the game more engaging.

3. Multiplayer Mode: Introduce a multiplayer mode where players can compete against each other in real-time. This could be done through online connectivity or by implementing a local multiplayer option for friends to play together.

4. Scoreboard and Leaderboard: Create a scoreboard to track and display players' scores during the game. Additionally, consider implementing a leaderboard feature that allows players to compare their scores with others and strive for the top spot.

5. Improve User Interface: Enhance the visual design and user interface of the game to make it more appealing and user-friendly. This includes using appropriate colors, fonts, and graphics to create an immersive and enjoyable experience.

6. Sound Effects and Music: Incorporate sound effects and background music to enhance the overall gaming experience. This can help create a more immersive atmosphere and add excitement to the gameplay.

7. Mobile Compatibility: Adapt the game for mobile devices by optimizing the user interface and controls for touchscreens. This will allow players to enjoy the quiz game on their smartphones or tablets.