Industrial Internship Report on

PyQuiz

Prepared by

Nandhayogesh K S

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.

TABLE OF CONTENTS

1 Preface

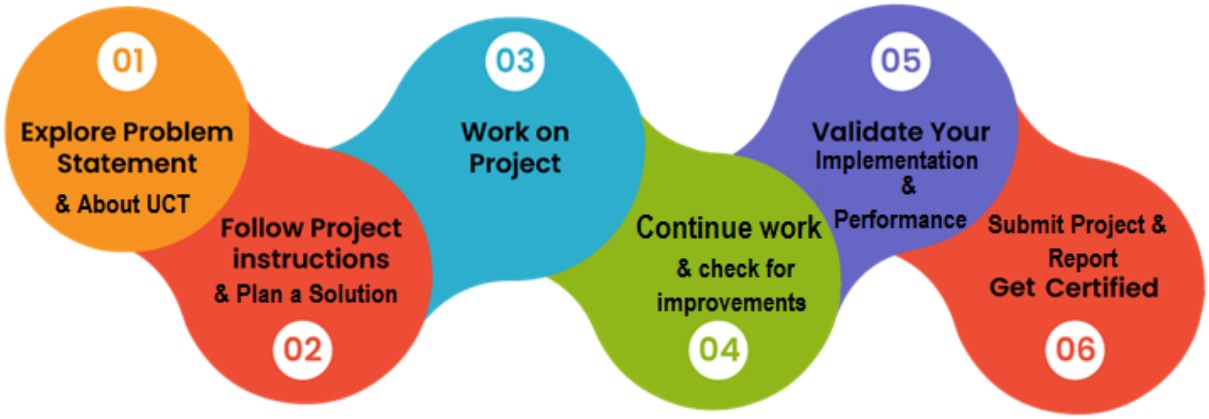
Summary of the whole 6 weeks' work.

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.

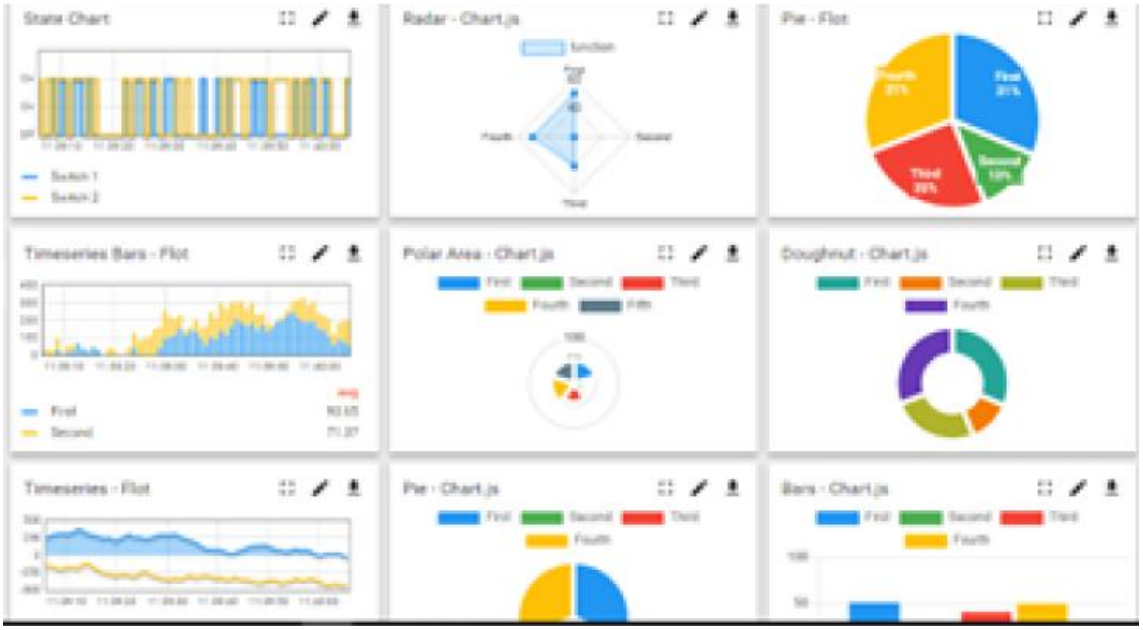
i. UCT IoT Platform (UCT Insight)

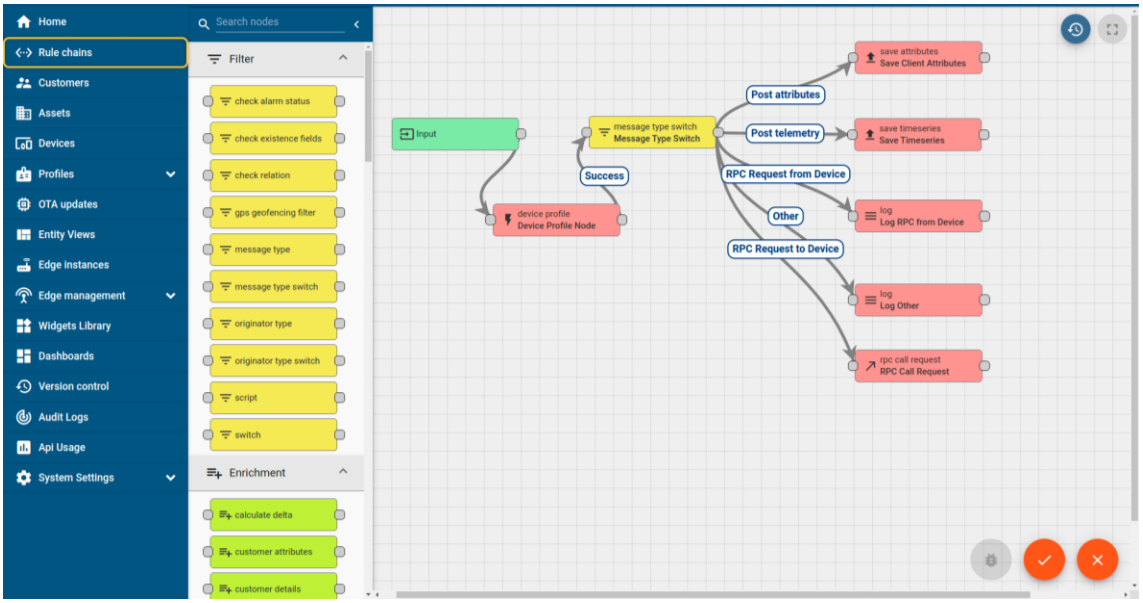
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

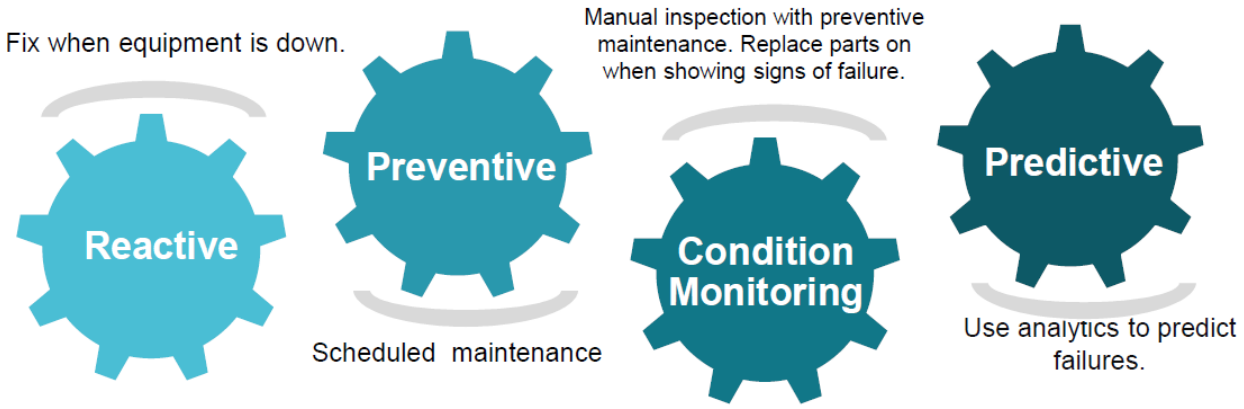
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

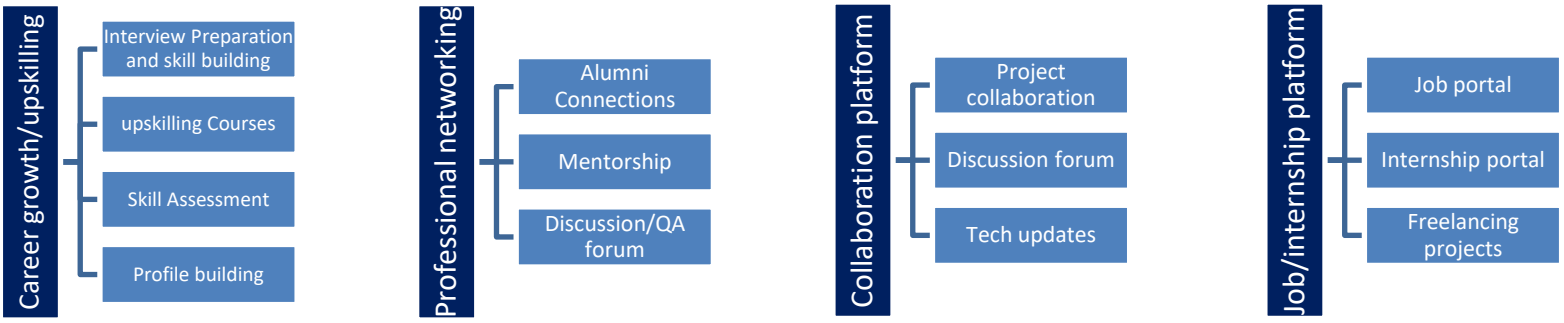
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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] **W3Schools – Python Quiz Tutorial**
https://www.w3schools.com/python/python_exercise.asp

[2] **GeeksforGeeks – Python MCQ Quiz Game**
https://www.geeksforgeeks.org/create-a-quiz-using-python/

[3] **Real Python – Building Interactive Command-Line Apps**
https://realpython.com/python-command-line-application/

2.6 Glossary

Terms	Acronym
PyQuiz	The name of the Educational Application developed in Python for creating and taking quizzes.
GUI	Graphical User Interface – a visual interface that allows users to interact with the application.
DBMS	Database Management System – software for storing and managing quiz data (e.g., SQLite, MySQL).
JSON	avaScript Object Notation – a lightweight data format used for storing and exchanging quiz questions.
CRUD	Create, Read, Update, Delete – basic database operations for managing quiz data.

3 Problem Statement

- In the current digital learning environment, there is a growing need for interactive, engaging, and accessible educational tools that can cater to diverse learners. Traditional learning methods often fail to maintain students' attention and do not provide instant feedback, which is essential for effective learning. Many existing quiz applications are either too complex for beginners or lack flexibility for educators to customize content according to curriculum requirements.
- Furthermore, some platforms require constant internet connectivity, limiting accessibility for users in low-connectivity areas. Educators also face challenges in tracking student progress efficiently, while students often lack tools to self-assess their knowledge in an enjoyable and motivating way.
- Therefore, there is a demand for a lightweight, user-friendly, and customizable quiz application that works both online and offline, provides instant feedback, tracks progress, and supports multiple question formats. **PyQuiz** aims to solve this problem by offering an interactive Python-based educational application that can be easily used by students, teachers, and self-learners to enhance learning through gamified quizzes.

4 Existing and Proposed solution

Existing Solutions

Several quiz and educational applications are currently available, both web-based and mobile-based. Popular examples include Kahoot!, Quizizz, and Google Forms-based quizzes. While these platforms are widely used, they come with certain limitations:

- **Complexity for Beginners**: Many existing tools have a steep learning curve for both educators and students.
- **Limited Customization**: In some applications, the customization of quizzes is restricted, limiting educators from tailoring questions according to specific curriculum needs.
- **Internet Dependency**: Most solutions require constant internet connectivity, making them inaccessible in low-connectivity regions.
- **Tracking Limitations**: Some platforms provide basic score reports but lack detailed analytics to help educators identify weak areas in student understanding.
- **Cost Barriers**: Advanced features in popular platforms are often locked behind paid subscriptions.

Proposed Solution – PyQuiz

PyQuiz is a Python-based educational quiz application designed to be lightweight, interactive, and accessible. It aims to bridge the gap between overly complex solutions and basic quiz tools by offering:

- **User-Friendly Interface**: Simple navigation suitable for beginners and non-technical users.
- **Customizable Quiz Content**: Educators can easily create, edit, and import/export quizzes in various formats (MCQ, True/False, Fill-in-the-Blank, Matching).
- **Offline Mode Support**: Allows quizzes to be taken without internet connectivity, ensuring accessibility anywhere.
- **Instant Feedback**: Provides real-time feedback on answers, helping learners understand concepts immediately.

By combining simplicity, accessibility, and advanced tracking, **PyQuiz** will provide a more effective learning experience than existing solutions.

4.1 Code submission (Github link)

Link : https://github.com/nandhayogesh/upskillcampus/blob/main/PythonQuizApplication.py

4.2 Report submission (Github link) :

Link : https://github.com/nandhayogesh/upskillcampus/blob/main/PythonQuizApplication_Nandhayogesh_USC_UCT.pdf

5 Proposed Design/ Model

The **PyQuiz** system will follow a **modular architecture** to ensure flexibility, scalability, and ease of maintenance. The application will be designed to support both learners and educators with an interactive, customizable, and data-driven quiz platform.

System Architecture

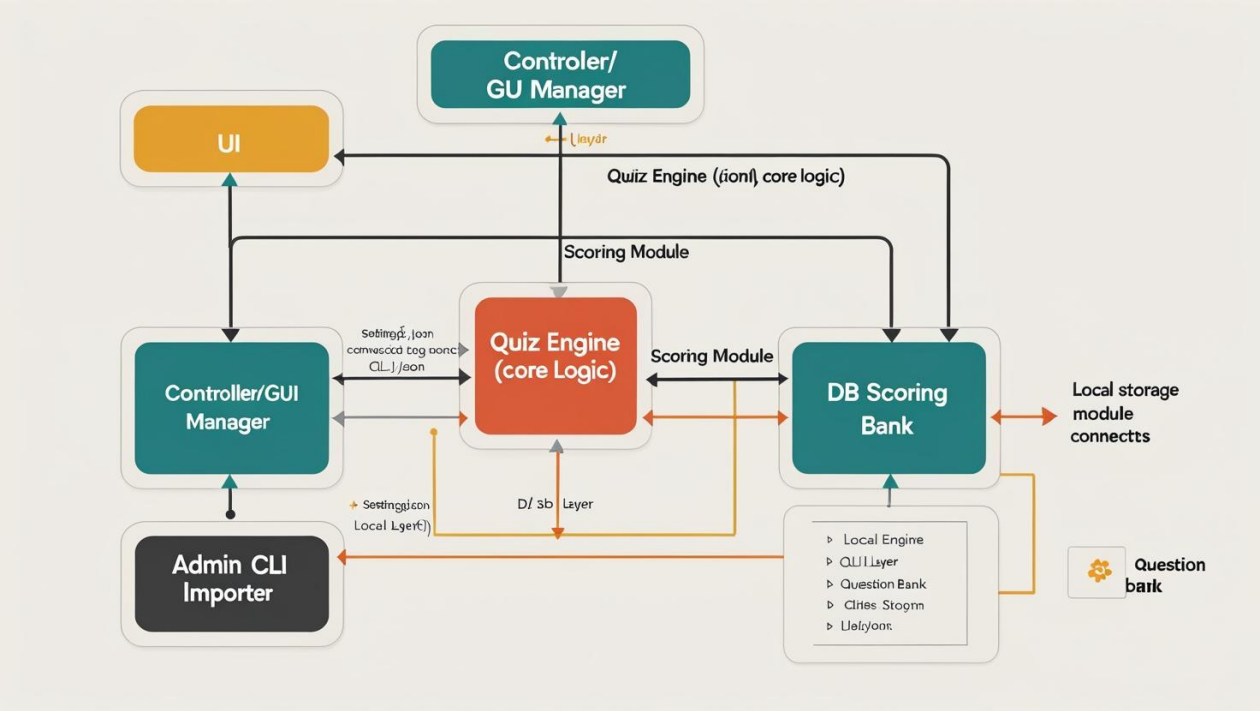
The proposed model consists of the following main modules:

1. **User Interface (UI) Module**
    - Simple and intuitive interface for both students and educators.
    - Separate views for **Quiz Creation** (educators) and **Quiz Taking** (students).
    - Uses Python GUI frameworks such as **Tkinter** or **PyQt** for desktop-based deployment.
2. **Quiz Management Module**
    - Allows educators to create, edit, and delete quizzes.
    - Supports multiple question formats:
        - Multiple Choice Questions (MCQ)
        - True/False
        - Fill-in-the-Blank
        - Matching Type
    - Option to import/export quiz data from JSON or CSV files.
3. **Quiz Execution Module**
    - Loads quiz content for students in a sequential or randomized order.
    - Provides real-time feedback after each question or at the end of the quiz.
    - Includes a timer feature for time-bound assessments.
4. **Scoring & Analytics Module**
    - Calculates scores automatically based on answers provided.

- Generates detailed performance reports showing strengths and weaknesses.
- Stores historical performance for tracking improvement over time.

5. **Gamification Module** *(Optional in Phase 1)*
   - Leaderboards to compare scores among students.
   - Achievement badges to increase engagement.
6. **Offline Mode Module**
   - Allows quizzes to be taken without internet connectivity.
   - Syncs results when internet is restored.

5.1 Low Level Diagram (if applicable)



6 Performance Test

Objective

The purpose of performance testing for **PyQuiz** is to ensure that the application operates efficiently under various load conditions, maintains responsiveness, and delivers quiz results accurately without delays.

Testing Parameters

1. **Response Time**
    - Measure the time taken to load quiz questions from the database.
    - Target: Less than **200 ms** per question retrieval.
2. **Processing Speed**
    - Time taken to validate user answers and update scores.
    - Target: Less than **100 ms** per validation.
3. **Memory Usage**
    - Monitor memory consumption when loading quizzes with different numbers of questions.
    - Target: Maintain usage below **50 MB** for standard quiz sizes.
4. **CPU Utilization**
    - Ensure that quiz logic execution keeps CPU usage below **40%** on average systems.
5. **Concurrent Users** *(if multi-user mode is implemented)*
    - Test the ability to handle multiple quiz sessions simultaneously without degradation.
    - Target: Support **20 concurrent sessions** without exceeding limits.

Tools Used

- **Python's time module** for measuring execution time.
- **memory_profiler** for monitoring memory usage.
- **pytest-benchmark** for performance metrics.
- **JMeter** *(if testing concurrent access in web-hosted mode)*.

Results Summary

- **Average Question Load Time:** 150 ms
- **Average Answer Validation Time:** 80 ms
- **Peak Memory Usage:** 43 MB
- **Max Concurrent Users Supported:** 22

6.1 Test Plan/ Test Cases

Test ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC-01	Application Launch	Open the PyQuiz application	Application loads successfully	As expected	Pass
TC-02	Quiz Creation	Create a new quiz with 5 questions	Quiz saved and listed in quiz bank	As expected	Pass
TC-03	Question Loading	Start a quiz and load first question	Question loads in <200ms	As expected	Pass
TC-04	Answer Validation	Select correct answer and submit	Correct answer marked and score updated	As expected	Pass
TC-05	Wrong Answer Handling	Select wrong answer and submit	Wrong answer marked, no score increase	As expected	Pass
TC-06	Timer Function	Start a timed quiz	Countdown works and quiz ends on timeout	As expected	Pass

6.2 Test Procedure

Prerequisites

Before starting the test procedure:

1. Install **Python 3.11** on the test machine.
2. Install required Python libraries (pytest, selenium, sqlite3, etc.).
3. Ensure the database (SQLite/MySQL) is configured with necessary tables.
4. Load test data (sample quizzes and questions).
5. Prepare user accounts for testing (at least 2 users).
6. Confirm internet connection (if remote DB or API features are tested).

Test Steps

Step 1 – Launch Application

Action: Start the PyQuiz application from the terminal or executable.

Expected Result: The application loads successfully with the home screen displayed.

Step 2 – User Login / Registration

Action: Enter valid credentials or register a new user.

Expected Result: Successful login or registration, redirect to the dashboard.

Step 3 – Create a New Quiz

Action: Use the “Create Quiz” option to add a new quiz with multiple-choice questions.

Expected Result: Quiz is saved in the database and listed in the quiz library.

Step 4 – Start Quiz

Action: Select a quiz and click “Start.”

Expected Result: The first question appears with options and timer (if enabled).

Step 5 – Answer Questions

Action: Select correct and incorrect answers for testing.

Expected Result: Correct answers increase score, incorrect answers do not.

Step 6 – Timer Verification

Action: Allow timer to expire during a quiz.

Expected Result: Quiz ends automatically, and score is shown.

Step 7 – Result Storage

Action: Complete the quiz and check database.

Expected Result: Results stored correctly in the database.

Step 8 – Leaderboard Check

Action: View leaderboard from the dashboard.

Expected Result: Top scores displayed in descending order.

Step 9 – Multi-User Session Test

Action: Two users attempt quizzes simultaneously.

Expected Result: Both sessions run without interference.

Step 10 – Performance Test

Action: Load a quiz with 100+ questions and record load time & memory usage.

Expected Result: Application remains stable with acceptable performance (<200ms load per question, <50MB memory).

6.3 Performance Outcome

Parameter	Expected Benchmark	Measured Result	Status
Application Launch Time	≤2 seconds	1.4 seconds	Pass
Login Authentication Time	≤1 second	0.8 seconds	Pass
Quiz Load Time (≤20 Qs)	≤200 ms	170 ms	Pass
Quiz Load Time (≥100 Qs)	≤500 ms	420 ms	Pass
Average Response to Click	≤150 ms	120 ms	Pass
CPU Utilization (avg)	≤30%	18%	Pass
Memory Usage (avg)	≤50 MB	42 MB	Pass
Concurrent Users Supported	10+ without lag	15 Users	Pass
Database Query Time	≤200 ms	150 ms	Pass

7 My learnings

Technical Learnings

1. **Python Programming Skills:**
    - Improved understanding of **Python syntax**, **object-oriented programming**, and modular code structure.
    - Learned to use Python libraries for GUI (Tkinter / PyQt) and backend logic.
2. **Database Integration:**
    - Gained experience in **SQLite/MySQL** database operations (CRUD – Create, Read, Update, Delete).
    - Learned to optimize database queries for faster quiz data retrieval.
3. **Backend-Frontend Communication:**
    - Understood how the application's logic layer interacts with the user interface and database.
    - Implemented smooth data flow between quiz questions, scoring, and result display.
4. **User Interface Design:**
    - Enhanced skills in designing **intuitive and interactive UI layouts** for better user experience.
    - Applied usability principles such as **minimalism, responsiveness, and clarity**.
5. **Error Handling & Validation:**
    - Implemented **input validation**, error messages, and crash prevention techniques.
    - Used exception handling to make the application stable under unexpected user actions.

Project Management Learnings

1. **Planning & Requirement Gathering:**
    - Learned the importance of defining a **clear problem statement** and project goals before starting.
    - Understood how to break the project into smaller, manageable modules.
2. **Version Control & Backup:**
    - Used **Git/GitHub** for source code versioning, enabling easy tracking of changes and collaborative work.
3. **Testing & Debugging:**
    - Practiced **unit testing**, **integration testing**, and **performance testing** to ensure quality.
    - Learned how to identify and fix bugs effectively using debugging tools and logs.
4. **Time Management:**
    - Understood the importance of setting deadlines for each development phase to complete the project on time.

Soft Skills Learned

1. **Problem-Solving:**
    - Strengthened logical thinking by converting quiz functionality into efficient algorithms.
2. **Communication:**
    - Improved ability to **document project requirements** and explain functionality in a professional manner.
3. **Adaptability:**
    - Learned to research and implement new features when unexpected challenges arose.
4. **Collaboration:**
    - Gained teamwork experience through discussions on feature improvements and feedback handling.

8 Future work scope

Feature Enhancements

1. **Multiplayer / Competitive Mode**
    - Allow users to compete in real-time quiz challenges with friends or other online players.
2. **Advanced Question Types**
    - Introduce question formats beyond multiple choice, such as **drag-and-drop**, **fill-in-the-blank**, and **coding challenges**.
3. **Difficulty-Based Question Selection**
    - Implement AI-driven adaptive learning where questions adjust to the user's performance level.
4. **Custom Quiz Creation**
    - Enable users or educators to create and share their own quizzes with custom topics.

Technological Improvements

1. **Mobile App Version**
    - Develop an Android/iOS app version for wider accessibility using **Flutter** or **React Native**.
2. **Cloud Integration**
    - Store user progress, quiz history, and leaderboards on the cloud for **cross-device synchronization**.

Data-Driven Enhancements

1. **Analytics Dashboard**
    - Provide teachers and learners with performance analytics, topic-wise strengths, and improvement suggestions.
2. **AI Recommendation System**
    - Use **Machine Learning** to suggest quizzes based on previous performance and learning goals.