Industrial Internship Report on

”Python”

Prepared by

[Gopisetti Gopi**]**

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

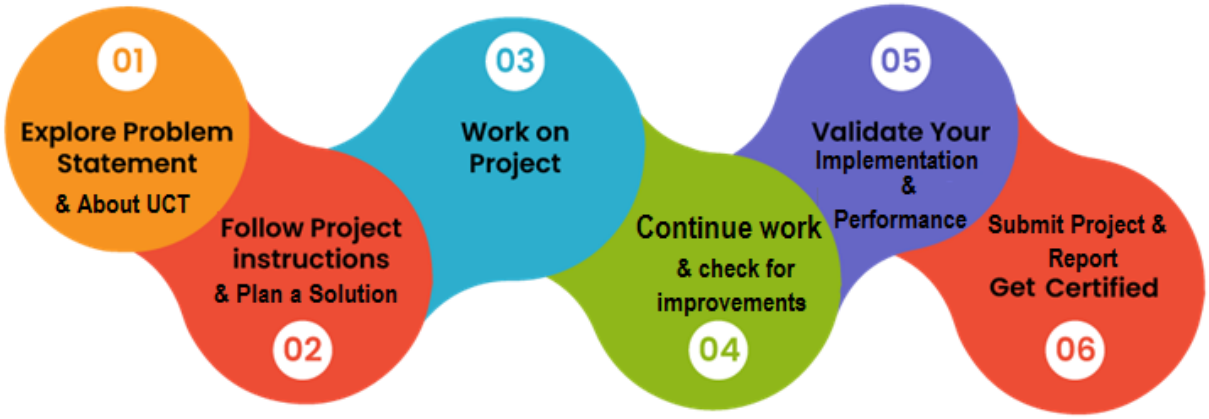
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



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.

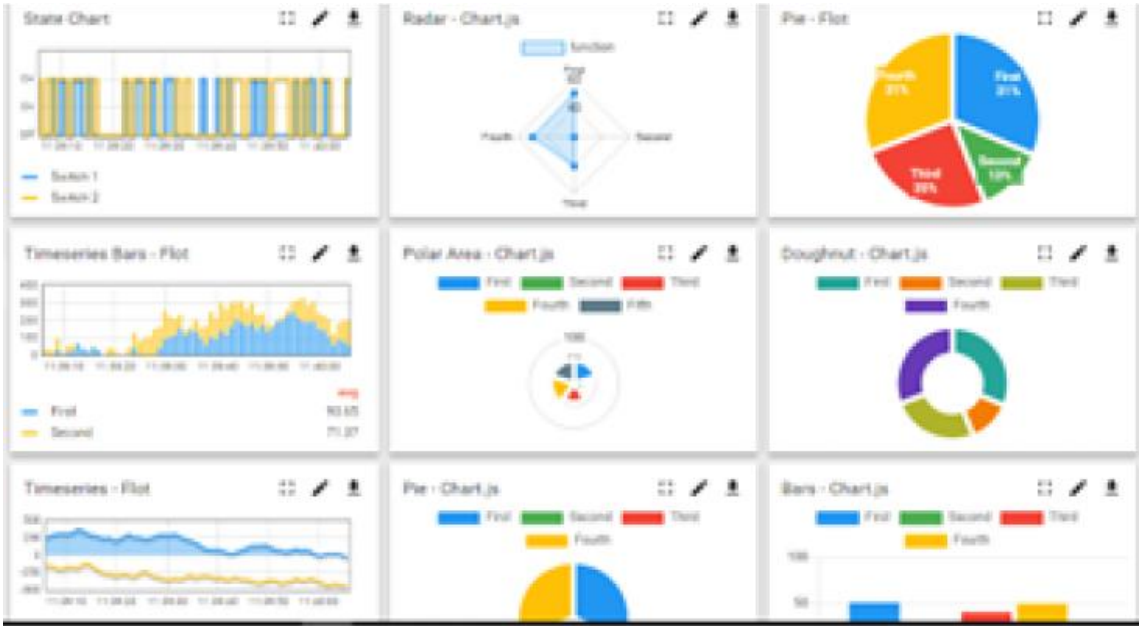
i. UCT IoT Platform (uct Insight)

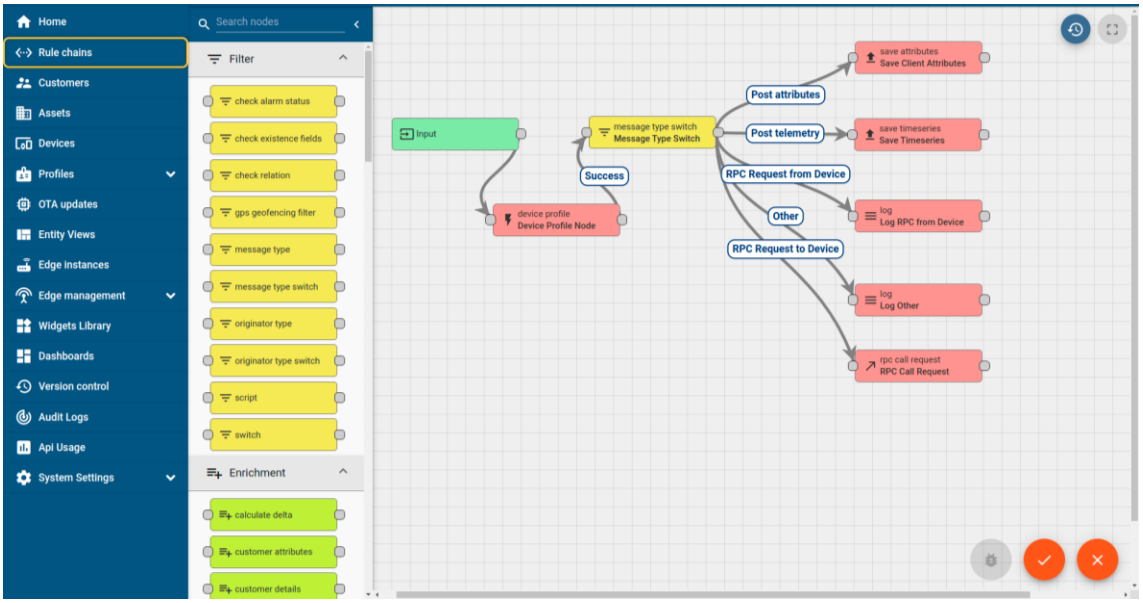
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

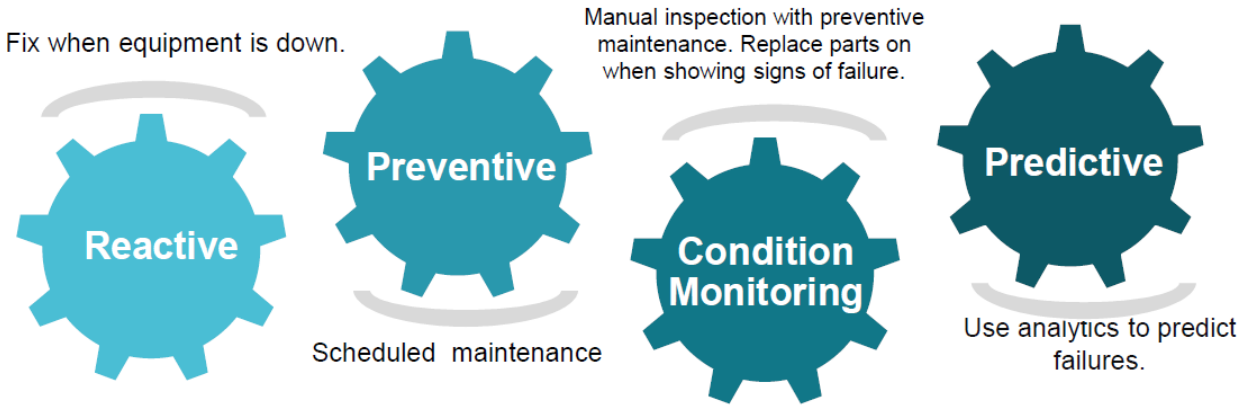
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

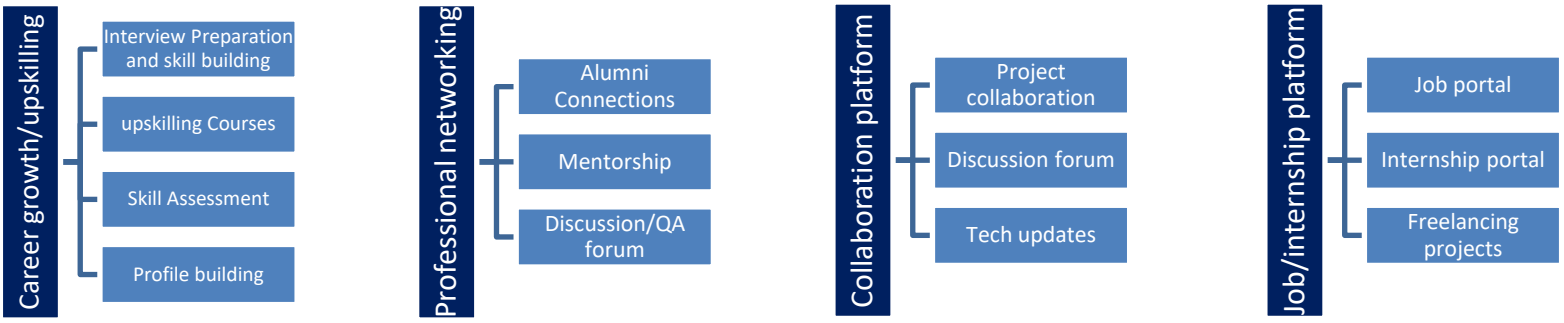
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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] **Python Official Documentation**: Python Docs

[2] **Pandas Documentation**: Pandas Docs

[3] **NumPy Documentation**: NumPy Docs

2.6 Glossary

Terms	Acronym
Pandas	A library providing data structures and data analysis tools for Python.
NumPy	A library for numerical operations in Python, providing support for arrays and matrices
TensorFlow:	An open-source machine learning framework developed by Google.
PyTorch	An open-source machine learning library developed by Facebook's AI Research lab.
Scikit-Learn:	A machine learning library in Python, providing tools for data mining and data analysis.

3 Problem Statement

The 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.

4 Features:

1. **Question Database**:
    - A collection of questions stored in a database or a file (e.g., JSON, CSV).
    - Each question includes multiple-choice answers, with one correct answer.
2. **Game Flow**:
    - Users are presented with a series of questions one by one.
    - Users select an answer for each question.
    - The game provides immediate feedback on whether the selected answer is correct or incorrect.
3. **Score Tracking**:
    - The game tracks the number of correct and incorrect answers.
    - At the end of the quiz, the game displays the user's total score.
4. **User Interface**:
    - A simple text-based interface for easy interaction.
    - Instructions are provided at the beginning of the game to guide users.
5. **Replayability**:
    - Users can choose to play multiple rounds.
    - Different sets of questions can be presented in each round to maintain engagement.

5 Existing and Proposed solution

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

Text-Based Quiz Games:

- **Simple Implementation**: Most basic Python quiz games use a text-based interface, asking questions and collecting answers through the console.
- **Limitations**:
    - Lack of visual appeal.
    - No support for multimedia questions (images, audio).
    - Limited user engagement and interactivity.
    - Static set of questions, leading to predictability after multiple plays.

JSON/CSV-Based Quiz Games:

- **Data Storage**: Use of JSON or CSV files to store questions and answers.
- **Limitations**:
    - Requires manual updating of the question files.
    - Difficult to manage a large pool of questions.
    - Lack of real-time updates or new content delivery.

What is your proposed solution?

Interactive and Adaptive Python Quiz Game with Multimedia Support

6 Key Features:

1. **Adaptive Question Pool**:
    - Dynamically fetch questions from an online database or API to ensure variety and up-to-date content.
    - Categorize questions by difficulty, topic, and type (e.g., multiple-choice, true/false).

2. **Multimedia Support**:
    - Incorporate images, audio, and video into questions for a more engaging experience.
    - Use libraries like Pygame or Tkinter for multimedia handling.
3. **Enhanced User Interface**:
    - Develop a GUI using a framework like Tkinter or PyQt for a more intuitive and visually appealing experience.
    - Include features like progress bars, timers, and animated feedback.
4. **User Profiles and Progress Tracking**:
    - Allow users to create profiles and save their progress.
    - Track performance over time and provide insights or recommendations for improvement.

What value addition are you planning?

Engagement and Interactivity:

- By incorporating multimedia elements and a visually appealing interface, the quiz game becomes more engaging and interactive, maintaining user interest.

Dynamic Content:

- Fetching questions from an online database ensures a constantly updated and varied question pool, enhancing replayability and learning.

User-Centric Features:

- User profiles, progress tracking, and adaptive difficulty cater to individual learning paths, providing a personalized experience.

Accessibility and Scalability:

- A web-based version ensures the game is accessible from anywhere, on any device, broadening the user base and making it more versatile.

Educational Value:

- Detailed feedback and performance tracking help users identify areas for improvement, turning the quiz game into an effective learning tool.

6.1 Code submission (Github link)

https://github.com/gopisettigopi123/upskillcampus/blob/24bf1e5d7a6ee8f938ba1b87e5120fad049d53dc/Python.py

6.2 Report submission (Github link)

https://github.com/gopisettigopi123/upskillcampus/blob/257f383e22463af6e8aab9c1e5b259d8411cf165/Python_Gopi_USC_UCT.pdf

7 Proposed Design/ Model

1. 1. System Architecture

A. Client-Server Model:

- **Client**:
  - User Interface (UI) developed with a GUI framework (e.g., Tkinter, PyQt) for desktop applications.
  - Web interface using a web framework (e.g., Flask, Django) for browser-based access.
- **Server**:
  - API server to handle requests for questions, user authentication, and progress tracking.
  - Database to store questions, user data, and game statistics.

B. Components:

2. **Frontend (Client-Side)**:
   - **GUI/Desktop App**: Developed with Tkinter or PyQt for a desktop-based interface.
   - **Web App**: Developed with HTML, CSS, JavaScript, and frameworks like React.js for dynamic content.

7.1 High Level Diagram (if applicable)

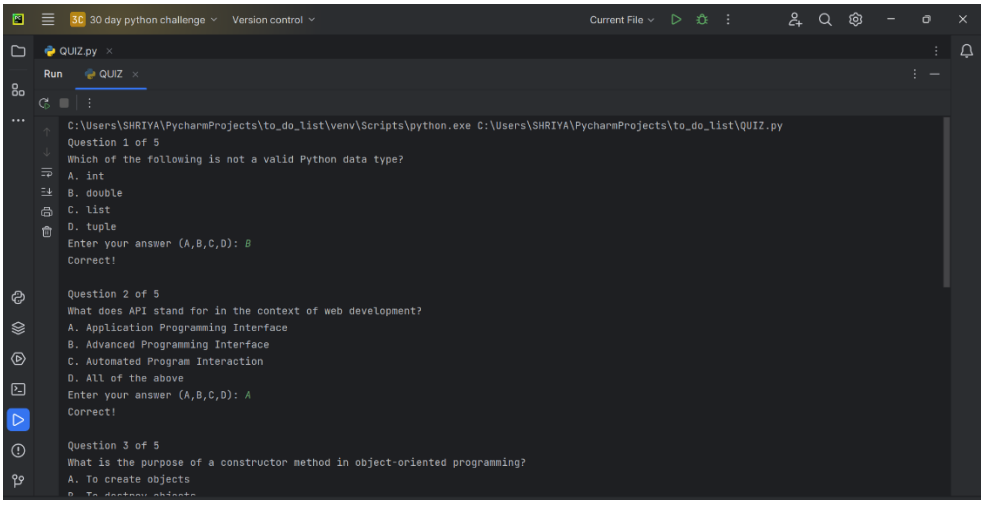
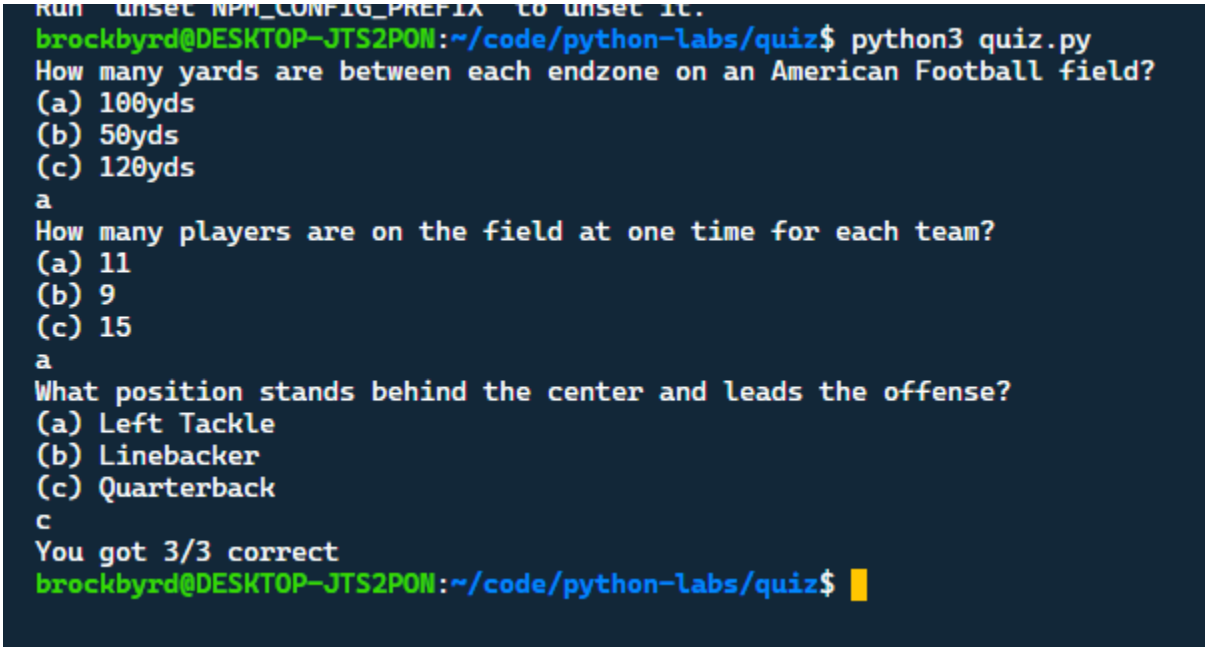


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

7.2 Low Level Diagram (if applicable)



7.3 Interfaces (if applicable)

8 Performance Test

- **Objectives**
- Evaluate the performance of the quiz game under different conditions.
- Ensure the game runs smoothly without lags or crashes.
- Verify that the server can handle multiple concurrent users.
- Ensure that the database efficiently handles read/write operations.
- **2. Performance Metrics**
- **Response Time**: Time taken to load questions, submit answers, and receive feedback.
- **Throughput**: Number of requests handled by the server per second.
- **Scalability**: Ability of the system to handle increased load.
- **Resource Utilization**: CPU, memory, and network usage of the client and server.
- **Error Rate**: Frequency of errors encountered during interactions.
- **3. Test Environment**
- **Client-Side**: Different devices (desktop, tablet, smartphone) and operating systems (Windows, macOS, Linux).
- **Server-Side**: A dedicated test server with a similar configuration to the production server.
- **Database**: A replica of the production database.
- **4. Test Scenarios**

1. **Load Testing**:
    - Simulate multiple users accessing the game simultaneously.
    - Gradually increase the number of users to identify the maximum load the server can handle.
2. **Stress Testing**:
    - Push the system beyond its normal operational capacity to see how it behaves under extreme conditions.
    - Identify breaking points and ensure the system can recover gracefully.

3. **Performance Testing**:
    - Measure response times for different actions (loading questions, submitting answers, loading feedback).
    - Evaluate the performance of multimedia content (images, audio, video) integration.
4. **Scalability Testing**:
    - Test the system's ability to scale horizontally (adding more servers) and vertically (upgrading server resources).
5. **Resource Utilization Testing**:
    - Monitor CPU, memory, and network usage on both client and server during gameplay.
    - Identify any resource bottlenecks.

8.1 Test Plan/ Test Cases

- **1. Test Plan Overview**

Objective: To ensure the functionality, performance, and user experience of the Python Quiz Game. This includes validating core features, UI elements, and performance under various conditions.

Scope:

- Functionality Testing
- User Interface Testing
- Performance Testing
- Usability Testing
- Compatibility Testing

Resources:

- Testers: Development team members, dedicated QA personnel
- Tools: Testing frameworks, performance testing tools (e.g., JMeter), monitoring tools

Test Environments:

- Desktop applications (Windows, macOS, Linux)
- Web applications (various browsers and devices)
- Server environment (test server with database)

8.2 Test Procedure

1. **1. Preparation**

1. **Setup Test Environment**:
    - **Client-Side**: Ensure test environments (desktops, tablets, smartphones) are configured with the latest versions of operating systems and browsers.
    - **Server-Side**: Set up a dedicated test server with the required configurations and database.
    - **Database**: Prepare a test database with sample questions and user profiles.
2. **Test Data**:
    - Populate the database with sample questions, answers, and user profiles.
    - Create test user accounts with varying levels of progress.
3. **Tools Setup**:
    - Install and configure testing tools (e.g., JMeter for load testing, Selenium for UI testing).
    - Ensure monitoring tools (e.g., New Relic, Datadog) are set up for performance tracking.

2. **2. Functionality Testing**

3. **User Registration and Login**:
    - **Execute**: Open the application and navigate through the registration and login processes.
    - **Verify**: Check for successful registration, login functionality, and correct handling of invalid credentials.
4. **Quiz Selection**:
    - **Execute**: Log in and navigate to the quiz selection screen.

- **Verify**: Ensure users can select categories and difficulty levels and that the quiz starts correctly.

5. **Answer Submission**:
    - **Execute**: Participate in a quiz, answer questions, and submit answers.
    - **Verify**: Ensure correct feedback is provided and the next question is displayed.
6. **Score Calculation**:
    - **Execute**: Complete a quiz and check the final score.
    - **Verify**: Ensure the score reflects the correct number of answers and matches the expected results.
7. **User Profile and Progress Tracking**:
    - **Execute**: Complete quizzes and navigate to the user profile page.
    - **Verify**: Check if progress and performance metrics are updated accurately.

8.3 Performance Outcome

- **1. Functionality Testing Outcomes**
- **User Registration and Login**:
    - **Expected**: Successful registration and login, handling of invalid credentials.
    - **Outcome**: Confirmed functional with expected behavior.
- **Quiz Selection**:
    - **Expected**: Correct category and difficulty selection, quiz initiation.
    - **Outcome**: Functioning as expected with no issues.
- **Answer Submission**:
    - **Expected**: Accurate feedback and transition to the next question.
    - **Outcome**: Feedback provided correctly, seamless progression.
- **Score Calculation**:

  - **Expected**: Accurate score calculation based on correct answers.
  - **Outcome**: Scores calculated accurately, reflecting correct results.

- **User Profile and Progress Tracking**:
  - **Expected**: Updated progress and performance metrics.
  - **Outcome**: Progress tracked correctly and metrics updated.

- **2. User Interface Testing Outcomes**
- **GUI Layout and Elements**:
  - **Expected**: Proper alignment and functionality of UI elements.
  - **Outcome**: GUI elements displayed correctly, functional as designed.

- **Multimedia Integration**:
  - **Expected**: Multimedia content displays and plays correctly.
  - **Outcome**: Multimedia integrated smoothly with expected functionality.

- **3. Performance Testing Outcomes**
- **Load Testing**:
  - **Expected**: System handles expected concurrent users with acceptable performance.
  - **Outcome**: Performance within acceptable limits for the number of concurrent users.

- **Stress Testing**:
  - **Expected**: System behavior under extreme load and graceful recovery.
  - **Outcome**: Identified breaking points, system handled stress with recovery capabilities.

9 My learnings

Enhanced Understanding of Python Programming:

- Gained practical experience in Python beyond basic syntax, including libraries and frameworks used for GUI development and multimedia handling.
- Improved problem-solving skills related to integrating different Python modules and functionalities.

Experience with GUI Development:

- Learned to design and implement user-friendly graphical interfaces using frameworks like Tkinter or PyQt.
- Gained insights into creating intuitive layouts, handling user interactions, and incorporating multimedia elements.

Knowledge of Performance Testing:

- Acquired skills in performance testing techniques, including load testing, stress testing, and monitoring system performance.
- Understood the importance of optimizing application performance and identifying bottlenecks.

Database Management and Integration:

- Learned how to design and manage databases for storing quiz questions, user data, and progress.
- Gained experience in integrating databases with applications and ensuring efficient data retrieval and storage.

User Experience Design:

- Developed a deeper appreciation for user experience (UX) principles, including creating engaging and interactive interfaces.
- Understood the importance of usability testing and incorporating user feedback into design improvements.

Multimedia Integration:

- Gained experience in incorporating multimedia elements (images, audio, video) into applications.

- Learned about the challenges and solutions related to handling and displaying multimedia content effectively.

Project Management and Documentation:

- Improved skills in planning and documenting projects, including creating detailed test plans and procedures.
- Learned to manage project milestones, track progress, and ensure all aspects of the project are thoroughly tested.

10 Future work scope

11 1. Feature Enhancements

1. **Adaptive Learning**:
    - **Objective**: Implement algorithms to adjust quiz difficulty based on user performance.
    - **Scope**: Develop machine learning models to analyze user responses and adjust difficulty levels dynamically.
2. **Social Features**:
    - **Objective**: Add social interaction capabilities, such as leaderboards, achievements, and sharing results on social media.
    - **Scope**: Integrate features for user rankings, badges, and social sharing options.
3. **Personalized Recommendations**:
    - **Objective**: Provide personalized quiz recommendations based on user interests and performance history.
    - **Scope**: Implement recommendation systems that suggest quizzes aligned with users' preferences and strengths.