**Industrial Internship Report on**

**”Quiz Game”**

**Prepared by**

**[Bhumi dabhi]**

|  |
| --- |
| *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 about Quiz Game . to develop a system that will show the quiz and after all the result will declare the score.  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)

[3 Problem Statement 10](#_Toc139702813)

[4 Existing and Proposed solution 10](#_Toc139702814)

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

[6 Performance Test 15](#_Toc139702819)

[6.1 Test Plan/ Test Cases 17](#_Toc139702820)

[6.2 Test Procedure 19](#_Toc139702821)

[7 My learnings 22](#_Toc139702823)

[8 Future work scope](#_Toc139702824) 23

# 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 , who have helped you directly or indirectly.

Your message to your juniors and peers.

# 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.

## Reference

[1] https://www.kaggle.com/

[2]  https://www.annalsofrscb.ro/index.php/journal/article/view/7386

# Problem Statement

* Quiz Game:

Description: 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.

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.

# Existing And Proposed System

**Existing System:**

Mobile Quiz Apps: There are numerous quiz game apps available on mobile platforms (iOS and Android). These apps offer a wide range of quizzes on various topics, from general knowledge to specific subjects like history, science, movies, and more. Users can select quizzes, answer questions, and compete with others for high scores. Some apps also incorporate gamification elements, such as time limits, power-ups, and leaderboards.

Online Quiz Websites: Many websites offer online quiz games that can be accessed through web browsers. These websites often have a large database of quiz questions and provide options for users to choose categories and difficulty levels. Players can take quizzes individually or compete in multiplayer modes.

Educational Platforms: Educational platforms and learning management systems (LMS) often incorporate quiz games as part of their interactive learning modules. These quizzes are designed to assess the knowledge and progress of students in a specific subject or course.

Quiz Shows and Game Shows: Traditional quiz shows and game shows have been popular on television for many years. These shows feature contestants answering questions to win prizes or money. Some well-known quiz shows include "Who Wants to Be a Millionaire?" and "Jeopardy!"

**Proposed System:**

**Overview:**

The proposed system aims to develop an interactive and engaging quiz game that allows users to test their knowledge on various topics. The game will be designed to provide an enjoyable and challenging experience while promoting learning and entertainment.

**Key Features:**

1. User-Friendly Interface: The system will have an intuitive and user-friendly interface to ensure that players of all ages and backgrounds can easily navigate and participate in the quiz game.

2. Diverse Quiz Categories: The system will offer a wide range of quiz categories, including general knowledge, history, science, geography, literature, movies, sports, and more. Users can select their preferred categories to take quizzes.

3. Multiple Difficulty Levels: Each quiz category will have multiple difficulty levels, catering to users with varying levels of knowledge and expertise. Beginners can start with easy quizzes, while experts can challenge themselves with harder ones.

4. Time-Based Challenges: To add excitement and urgency, the system will incorporate time-based challenges, where players need to answer questions within a set time limit. This feature will make the game more competitive and engaging.

5. Points and Leaderboard: Players will earn points for each correct answer and bonus points for completing quizzes within the time limit. A real-time leaderboard will display the top scores, encouraging healthy competition among users.

6. Personalized Profiles: Users will have the option to create personalized profiles with avatars and usernames. Their profiles will track their quiz progress, scores, and achievements.

7. Social Sharing: The system will allow users to share their quiz scores and achievements on social media platforms, such as Facebook and Twitter, to challenge friends and invite them to join the game.

**Conclusion:**

The proposed interactive quiz game system aims to provide an engaging and educational experience for users across different platforms. By offering a diverse range of quiz categories and difficulty levels, incorporating time-based challenges and leaderboard features, and providing educational resources, the system seeks to create an enjoyable and enriching quiz game for users of all ages and backgrounds.

## Code submission : [Repository of code](https://github.com/bhumi32-source/Upskill_Campus)

## Report submission (Github link) : [Submission\_file](https://github.com/bhumi32-source/Upskill_Campus)

# Proposed Design/ Model

Certainly! When designing a software application, various design models can be used to guide the development process. One commonly used design model is the "Model-View-Controller" (MVC) pattern. The MVC pattern separates the application into three interconnected components:

1. Model:

- The Model represents the application's data and business logic. It is responsible for managing and processing data, as well as implementing the application's core functionalities.

- In the context of a quiz game system, the Model would handle tasks such as loading quiz data from a database or file, calculating scores, and managing user profiles.

2. View:

- The View is responsible for presenting the user interface to the application's users. It receives data from the Model and displays it in a user-friendly format.

- For the quiz game system, the View would be the graphical user interface (GUI) where users interact with the quiz questions, answer options, and receive feedback on their quiz performance.

3. Controller:

- The Controller acts as an intermediary between the Model and the View. It receives user input from the View, processes it, and updates the Model accordingly. It also receives updates from the Model and updates the View accordingly.

- In the quiz game system, the Controller would handle user interactions, such as selecting answer options, progressing to the next question, and managing the scoring.

Proposed Design Model for Quiz Game System:

Based on the proposed quiz game system discussed earlier, here's an overview of how the MVC pattern can be applied:

1. Model:

- The Model would handle quiz data management. It would include functions for loading quiz questions, options, and correct answers from a database or CSV file.

- The Model would also handle scoring logic, calculating the user's final score, and tracking quiz completion.

2. View:

- The View would be the graphical user interface presented to the users. It would include elements such as question labels, answer option buttons, and a display for the user's score.

- The View would also provide visual feedback to the user, such as highlighting the selected answer option and displaying whether the answer is correct or incorrect.

3. Controller:

- The Controller would manage user interactions with the View. It would handle events such as button clicks (selecting answer options) and manage the flow of the quiz (progressing to the next question).

- The Controller would communicate with the Model to retrieve quiz data and update the user's score based on their responses.

Benefits of Using the MVC Pattern:

- Separation of Concerns: The MVC pattern promotes a clear separation of concerns between the data (Model), user interface (View), and user interactions (Controller). This makes the application easier to understand, maintain, and extend.

- Reusability: By separating the application into three components, each part can be developed independently and potentially reused in other parts of the application or in different projects.

- Scalability: The modular design of the MVC pattern allows for scalability as the application grows in complexity and size. It facilitates adding new features or modifying existing ones without affecting other components.

- Collaboration: The MVC pattern enables collaboration among development teams. Different teams can work on different components (Model, View, Controller) concurrently without conflicts.

It's important to note that while the MVC pattern is a widely used and effective design model, there are other design patterns and architectural styles that can also be considered depending on the specific requirements and complexity of the quiz game system. The key is to choose a design model that best fits the application's needs and allows for maintainability and scalability over time.

# Performance Test

To conduct performance testing for the quiz game system, you would evaluate how well the application performs under various load conditions. The objective is to identify potential performance bottlenecks, assess the system's responsiveness, and ensure that it can handle the expected number of concurrent users without degrading user experience. Here's an outline of the performance test plan for the quiz game system:

1. Performance Test Objectives:

- Determine the maximum number of concurrent users the system can handle without performance degradation.

- Measure the response time of the system for different user loads.

- Identify and address any performance bottlenecks.

2. Performance Test Scenarios:

- Define different test scenarios that represent the expected real-world usage of the quiz game system. For example:

- Scenario 1: Simulate 100 concurrent users taking quizzes continuously.

- Scenario 2: Simulate 500 concurrent users accessing the system simultaneously.

- Scenario 3: Simulate a mix of concurrent users accessing quizzes and leaderboard functionality.

- Scenario 4: Simulate user interactions during peak hours.

3. Performance Metrics:

- Response Time: Measure the time taken by the system to respond to user interactions, such as answering questions and progressing to the next question.

- Throughput: Measure the number of quiz requests that the system can handle per unit of time.

- Concurrent User Load: Determine the maximum number of concurrent users that the system can handle while maintaining acceptable performance.

4. Test Environment:

- Set up a dedicated test environment that closely resembles the production environment but isolated from actual users. This environment should have similar hardware, software, and network configurations.

5. Load Generation:

- Use performance testing tools (e.g., JMeter) to generate the desired number of concurrent users and simulate real-world usage scenarios.

6. Test Execution:

- Run each performance test scenario multiple times to ensure consistent results.

- Monitor system resource utilization (CPU, memory, network) during test execution.

7. Performance Analysis:

- Analyze the test results to identify performance bottlenecks and areas for improvement.

- Check for any abnormalities in response times, error rates, or system resource usage.

8. Scalability Testing:

- Assess the system's scalability by gradually increasing the user load to determine its breaking point and limitations.

9. Load Testing with Database:

- Include load testing scenarios that put stress on the database by simulating multiple concurrent quiz data access.

10. Stress Testing:

- Conduct stress testing to determine the system's stability and performance under extreme conditions.

11. Reporting:

- Summarize the performance test results and provide recommendations for optimizing system performance.

12. Tuning and Optimization:

- Based on the test results, implement optimizations to address any identified performance bottlenecks.

13. Regression Testing:

- After making optimizations, conduct regression testing to ensure that performance improvements did not introduce new issues.

14. Documentation:

- Document the performance test plan, test scenarios, results, and any changes made for future reference.

By conducting thorough performance testing, the quiz game system can be fine-tuned to handle a large number of concurrent users while maintaining a smooth and responsive user experience. Regular performance testing and optimization are essential to ensure that the system continues to perform well as the user base and data volume grow over time.

## Test Plan/ Test Cases

A test plan is a document that outlines the objectives, scope, approach, resources, and schedule for testing a software application or system. It provides a roadmap for the testing process and ensures that all aspects of the application are thoroughly tested to deliver a high-quality product. Here are the key components and details typically included in a test plan:

1. Introduction:

- Purpose: Describe the purpose of the test plan, including the goals and objectives of testing.

- Scope: Specify the scope of testing, including the features, modules, and platforms to be tested.

- Test Environment: List the hardware, software, and network configurations needed for testing.

2. Test Objectives:

- List the specific objectives and goals of testing, such as validating functionality, performance, security, etc.

3. Test Strategy:

- Testing Levels: Define the different levels of testing to be performed (e.g., unit, integration, system, acceptance).

- Testing Types: Identify the testing types to be conducted (e.g., functional, performance, security, usability).

- Testing Techniques: Describe the testing techniques to be employed (e.g., black-box, white-box, exploratory).

- Entry and Exit Criteria: Specify the conditions for entering and exiting each testing phase.

4. Test Schedule:

- Provide a timeline for each testing phase, including start and end dates, milestones, and deliverables.

5. Test Deliverables:

- List the test deliverables to be produced, such as test cases, test data, test scripts, defect reports, etc.

6. Test Environment and Data:

- Detail the test environment setup, including hardware, software, databases, and network configurations.

- Specify the test data to be used during testing, including sample data and data generation processes.

7. Test Cases and Test Scripts:

- Describe the approach for creating and executing test cases and test scripts.

- Define the criteria for passing and failing test cases.

8. Test Execution:

- Assign responsibilities for test execution and test management.

- Identify the roles and responsibilities of the testing team members.

9. Defect Management:

- Explain the process for reporting, tracking, and resolving defects.

- Define severity and priority levels for defects.

10. Risks and Mitigation:

- Identify potential risks and challenges related to testing.

- Provide mitigation strategies and contingency plans.

11. Test Completion Criteria:

- Define the criteria for determining when testing is complete and the application is ready for release.

12. Approvals:

- Identify stakeholders who need to review and approve the test plan.

13. Glossary:

- Provide definitions of key testing terms and acronyms used in the test plan.

The test plan is a critical document that helps ensure a structured and systematic approach to testing. It serves as a reference for the testing team and other stakeholders, providing a clear roadmap for testing activities throughout the software development lifecycle. Regularly updating the test plan as the project progresses and changes occur is essential to maintain its accuracy and effectiveness.

## Test Procedure

Test Procedure for Precision Weed Spraying System:

Weed Detection Accuracy Test:

Prepare a dataset of images with labelled ground truth (weeds and crops).

Feed the images into the weed detection model. Compare the predicted weed labels with the ground truth labels.

Calculate the accuracy of the weed detection model by dividing the number of correctly detected weeds by the total number of weeds in the dataset.

Crop Identification Accuracy Test: Prepare a dataset of images with labeled ground truth (different crop types).

Feed the images into the crop identification model. Compare the predicted crop labels with the ground truth labels.

Calculate the accuracy of the crop identification model by dividing the number of correctly identified crops by the total number of crops in the dataset.

Precision Spraying Mechanism Test:

Set up a controlled test environment with simulated weed patches and crops.

Activate the precision spraying mechanism.

Observe and document the precision spraying mechanism's performance in accurately targeting the weed patches while avoiding the crops.

Measure the percentage of pesticide application on weeds versus crops.

Real-time Decision-making Test:

Simulate various scenarios with different weed densities and crop types.

Monitor the system's decision-making process in determining where and when to apply pesticides.

Evaluate the system's decision accuracy and timeliness based on the actual weed distribution and crop types.

System Robustness and Durability Test:

Conduct tests in different environmental conditions, such as rain, wind, or extreme temperatures.

Observe the system's performance and functionality under varying environmental conditions.

Ensure that the system remains operational and reliable without any hardware or software failures.

Power Consumption Optimization Test:

Measure power consumption during different operational modes of the system, such as detection, decision-making, and spraying.

Evaluate the effectiveness of power-saving techniques, such as sleep modes or dynamic power scaling, in reducing energy consumption.

Monitor and record the system's power usage and battery life under different scenarios.

Integration and Communication Test:

Verify the seamless integration and communication between different components of the system, such as sensors, decision-making algorithms, and the spraying mechanism.

Conduct data exchange tests to ensure smooth and accurate data flow between the components.

Monitor the system's performance and document any issues related to integration or communication.

Test Reporting:

Document the test procedures, test results, and any issues encountered during the testing process.

Report the accuracy rates of the weed detection and crop identification models.

Provide feedback on the precision spraying mechanism's performance and its ability to avoid spraying crops.

Evaluate the system's decision-making accuracy and timeliness based on real-world scenarios.

Summarize the system's robustness, power consumption, and integration performance.

Make recommendations for improvements or further iterations based on the test results

.

## Performance Outcome

The performance outcome of a quiz game system can be measured based on several key metrics and factors. Here are some performance outcomes that can be evaluated:

User Engagement: One of the essential performance outcomes is the level of user engagement with the quiz game. It can be measured by the number of active users, the frequency of quiz sessions, and the average time spent per session.

Quiz Completion Rate: The percentage of users who start a quiz and complete it can indicate how interesting and well-designed the quizzes are. A higher completion rate suggests engaging content and a positive user experience.

User Retention: User retention measures the percentage of users who return to the quiz game system after their initial visit. High user retention indicates that users find the quiz game enjoyable and worth revisiting.

Quiz Success Rate: The percentage of correct answers submitted by users in quizzes reflects their knowledge and understanding of the topics. A high success rate may indicate that the quiz game system effectively tests and reinforces knowledge.

Load Time and Responsiveness: Performance is also assessed in terms of how quickly the system loads and responds to user interactions. A fast and responsive system enhances user satisfaction.

Error Rate and Bug Fixes: Monitoring the number of errors and bugs reported by users and implementing timely bug fixes is crucial to ensure a smooth user experience.

User Feedback and Ratings: User feedback and ratings can provide valuable insights into the strengths and areas of improvement for the quiz game system.

Conversion Rate (for Educational Platforms): For educational platforms, conversion rate measures the percentage of quiz players who progress to enroll in courses or additional educational content.

Social Sharing and Referrals: If the system supports social sharing, tracking the number of shares and referrals can help gauge user satisfaction and enthusiasm for the quiz game.

Scalability and Performance under Load: Evaluating how the system performs under heavy user load helps ensure it can handle increasing user traffic without significant performance degradation.

Accuracy of Scoring and Leaderboard: The accuracy of scoring and leaderboard updates is crucial for fair competition and user motivation.

# My learnings

Understanding of Quiz Game Concepts: You now have a better understanding of how quiz games work, including the basic mechanics, user interfaces, and scoring algorithms involved.

Python Programming: You learned how to use Python programming to build a quiz game system, implement functionalities such as user interfaces, scoring, and handling user input.

GUI Development: You explored GUI development in Python using the tkinter library, which allows you to create interactive and user-friendly interfaces for your applications.

File Handling and Data Storage: You learned how to read data from CSV files to populate the quiz questions and answers, which can be extended to handle data storage and retrieval in real-world applications.

Problem-Solving Skills: While working on the code and implementing functionalities, you likely encountered challenges and practiced problem-solving skills to overcome them.

Software Design and Architecture: You gained insights into how to design a quiz game system, breaking it down into smaller components and implementing the interactions between these components.

User Experience (UX): You may have learned about the importance of user experience in application design and how to create an intuitive and enjoyable user interface.

Proposed vs. Existing Systems: By discussing the proposed system and comparing it to existing quiz game systems, you explored how different systems cater to various user needs and requirements.

Future Work and Scope: You considered potential future work and enhancements for the quiz game system, broadening your understanding of iterative development and continuous improvement.

Learning by Doing: Through the process of building and discussing the quiz game system, you engaged in a hands-on learning experience, which is often one of the most effective ways to learn and retain new knowledge.

Overall, this interaction provided you with practical experience in Python programming, GUI development, data handling, and system design while exploring the world of quiz games and their applications. These learnings can serve as a foundation for further exploration and growth in software development and related fields. Remember, continuous learning and practice are key to becoming a proficient developer. Keep experimenting and building projects to further enhance your skills and knowledge.

# Future Work Space

The future work scope for the proposed interactive quiz game system in a workspace setting, there are several areas that could be explored to enhance the system's features, usability, and impact. Here are some potential future workspaces for the quiz game:

Gamification Enhancements: Implement additional gamification elements to make the quiz game more engaging and motivating for employees. This could include introducing badges, leveling systems, and rewards for achieving milestones or high scores.

Collaborative Quiz Mode: Create a collaborative quiz mode where employees can participate in team-based quizzes. This feature promotes teamwork, communication, and a sense of camaraderie among colleagues.

Integration with Learning Management Systems: Integrate the quiz game system with existing learning management systems (LMS) used in the workspace. This allows employees to take quizzes as part of their ongoing training and development programs.

Custom Quiz Creation: Enable employees to create their own custom quizzes on specific topics relevant to their roles or interests. This user-generated content can foster knowledge-sharing and community building within the workspace.

Analytics and Reporting: Implement analytics and reporting features to track employee quiz performance, identify knowledge gaps, and provide insights to L&D teams for targeted training initiatives.

Mobile App Support: Develop a mobile app version of the quiz game system, allowing employees to access quizzes on-the-go and participate in quizzes during their free time or while commuting.