**Industrial Internship Report on**

**”Quiz game python project”**

**Prepared by**

**Navanshu Maheshwari**

|  |
| --- |
| *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. I had to finish the project including the report in 6 weeks’ time.  My project was Quiz game python project which is created to develop an interactive and educational quiz game that would provide users with an enjoyable experience while testing their knowledge. The Quiz Game Python project utilized various technologies and libraries to achieve its objectives. MongoDB was chosen as the database system to store and manage the quiz data. PyWebIO, a Python library, was used as the web framework to create the user interface for the quiz game. The project's main features included displaying questions to the user, allowing them to select answers, providing feedback on their responses, and keeping track of their scores.  Throughout the development process, careful consideration was given to creating a seamless and intuitive user experience. The integration of MongoDB allowed for efficient retrieval and manipulation of quiz data, enabling users to access a wide range of questions and answer options. PyWebIO proved to be an excellent choice for the web framework, as it provided the necessary tools and functionalities to create an engaging and interactive user interface.  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

This report presents the development and implementation of a Quiz Game Python project aimed at creating an interactive web-based quiz application. The project focuses on designing a user-friendly interface that allows users to answer a series of questions and receive instant feedback on their performance. The application incorporates a database system using MongoDB to efficiently store quiz data and user information, enabling seamless retrieval and management of quiz-related information.

The Quiz Game Python project serves as an excellent opportunity to explore web-based application development, database integration, and user interaction using PyWebIO and MongoDB. Through this project, valuable insights have been gained into creating interactive and engaging Python applications that cater to a wide range of users.

The main objectives of the project were to:

1. Design a User Interface: Develop an intuitive and visually appealing user interface using PyWebIO, a powerful Python library for web-based user interfaces. The interface should display questions, collect user answers, and provide necessary instructions throughout the quiz.
2. Implement a Database System: Utilize MongoDB database, to store quiz questions, user data, and quiz scores. The integration with MongoDB allows for efficient data storage, retrieval, and management.
3. Develop a Scoring Algorithm: Create a scoring algorithm to track the user's progress during the quiz and calculate their final score based on the correctness of their answers. The scoring system should provide an accurate assessment of the user's performance.
4. User Authentication and Management: Implement functions for user login and registration to ensure a secure and personalized quiz experience. Users should be able to track their quiz history and scores.
5. Timer Functionality: Incorporate a timer function that automatically submits the quiz if the user does not complete it within a specified time. The timer feature adds an element of excitement and challenge to the quiz game.
6. Result Page: Design a result page to display the user's final score and provide feedback on their performance. The result page should also include correct answers for each question to help users learn from their mistakes.

# 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.uniconvergetech.in/about-us

[2] https://pywebio.readthedocs.io/en/latest/

[3] https://pymongo.readthedocs.io/en/stable/

# 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.

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.

The main objectives of this project is to develop:

**User Interface Design**: Create an intuitive and visually appealing user interface using PyWebIO, a Python library for web-based user interfaces. The interface should present quiz questions, allow users to select answers, and provide appropriate instructions and feedback throughout the quiz.

**Database Implementation**: Integrate a database system, preferably MongoDB, to store quiz questions, options, correct answers, user data, and quiz scores.

**Scoring Algorithm**: Develop a scoring algorithm to track the user's progress during the quiz and calculate their final score based on the correctness of their answers.

**User Authentication and Management**: Implement functions for user login and registration to ensure a secure and personalized quiz experience.

# Existing and Proposed solution

Our proposed solution to develop quiz game python project by using pywebio and mongodb python library. The project's main features included displaying questions to the user, allowing them to select answers, providing feedback on their responses, and keeping track of their scores.

The user will be able to access the quiz game after login input to the website through username and password which is saved in the database. The user will be directed to home page from where the user can access the quiz game. The result of the quiz game will be saved in the database which is used for calculation of the score.

One of the main objectives of this project was to ensure scalability and flexibility. By using MongoDB, the quiz game can accommodate a growing collection of questions and handle a large number of users simultaneously. Additionally, PyWebIO's adaptability allowed for easy customization of the user interface, enabling future enhancements and improvements to the game.

The proposed solution for the Quiz Game Python project involves the following key components:

1. **User Interface Design**: Utilizing PyWebIO, a Python library for web-based user interfaces, to design an interactive and visually appealing user interface. The interface will display quiz questions to users, present answer options, and allow for user input through radio buttons or text inputs.
2. **Database Integration**: Implementing a database system, MongoDB, to store quiz questions, options, correct answers, user data, and quiz scores. The database will facilitate efficient data storage, retrieval, and management, ensuring the seamless functioning of the application.
3. **Scoring Algorithm**: Developing a scoring algorithm that tracks the user's progress during the quiz and calculates their final score based on the correctness of their answers. The scoring system will provide instant feedback on each question and offer a comprehensive score at the end of the quiz.
4. **User Authentication and Management**: Creating functions for user login and registration to ensure secure access and personalized quiz experiences. Users will be able to view their quiz history and scores.
5. **Timer Functionality**: Incorporating a timer function that automatically submits the quiz if the user does not complete it within a specified time. The timer will add a sense of urgency to the quiz game and enhance the user experience.
6. **Result Page**: Designing a result page to display the user's final score and provide feedback on their performance. The result page will also include correct answers for each question, allowing users to learn from their mistakes.

The successful implementation of the proposed solution will result in a fully functional quiz game Python project that offers an engaging and educational experience for users. It will provide an effective platform for users to test their knowledge, track their progress, and gain valuable insights into web-based application development and database integration.

## Code submission (Github link):

https://github.com/navanshu1234/pythonproject/tree/master/quizgamepp

## Report submission (Github link) :

https://github.com/navanshu1234/pythonproject/tree/master/quizgamepp

# Proposed Design/ Model

The quiz game project design consists of the following components:

**a) User Interface Design:** The project involved designing a user interface using PyWebIO, a Python library that simplifies web application development. The user interface needed to display questions, present answer options, and allow users to select their answers.

**b) Data Storage:** To store the quiz data, a database or file system is implemented. MongoDB database, is chosen for its flexibility and scalability. It provided a robust solution for storing and managing the quiz questions and answer options.

**c) Scoring Algorithm:** A scoring algorithm is developed to track the user's progress and calculate their final score. The algorithm needed to evaluate the correctness of the user's answers and update the score accordingly.

The proposed model to develop the quiz game web application consists of following functionalities:

* **Login Page function:** The existing user can login to quiz game by entering user name and password. The user name and password details will be stored in the database. The user can only be able to access quiz game after entering correct username and password. The purpose of creating username field is to store the score of quiz game in database which can easily be identified based on username.
* **Registration Page function:** If a new user want to access quiz game then they can register through registration page by providing their details. Through registration page, the user will be able to create username field and choose a password.
* **Home Screen Page function:** The user can start a new game from home screen page. The home screen page consists of buttons for starting quiz game and option for logout. The new quiz game will redirect the user to quiz game screen from where the user can answer the quiz questions.
* **Quiz game page function:** This function is created to display the quiz game which is in form of multiple choice question answer. After successful user login and starting new quiz from home screen, the user can access the quiz. The quiz function display multiple choice question from which the user can choose one of the options. The quiz game function can display the quiz question either through using pywebio output components or through retrieving the question set stored in mongodb. In the implementation of quiz game page function, an empty list is used to store the correct answer choice selected by the user. This empty list is appended everytime user select an answer. This list is created to help in deciding the score of quiz game
* **Score calculation function:** This function is created to calculate the score of the quiz game based on the no. of correct answer selected by the user. This function uses a score variable with initial value assign to zero. The score variable value is incremented by one for every correct answer. This is done by using a ‘for loop’ to match the correct answer between actual answer and user selected answer through ‘empty list’ created in quiz game page function to store the user selected answers.

The Algorithm to calculate the user score is as follows:

* Initialize a score variable value assign to zero such as:
* score = 0
* Create a ‘for loop’ to iterate ith answer of question set.
* Use ‘If’ condition to compare actual answer with user selected answer such as:
* if qus[ans] == user\_ans[i]
* Increment the value of score by one for every ‘if’ condition satisfied such as:
* if qus[ans] == user\_ans[i]:
* score += 1
* **Result Page Function :** A function to implement a page to show user score is implemented. The purpose of this page is to show score of multiple user on the screen. The result page function also contain the button for retry, back and logout through which the user can reattempt the quiz game, go back to home screen or logout directly.The user can check the score of other user and previous score by entering the username in the input field present on the result page screen.The result page provide the user score by retrieving it from database based on the input username.The result page shows the user scores in form of table field. This is done to make the application more appealing.
* **Timer Function :** This functionality is implemented in the quiz function part of the application. The timer function is used to implement a time limit to complete the quiz in the given amount of time. This timer function set a timer for counting the amount of time from input time to zero to complete the quiz. When the time run out it automatically submit the quiz.

## Interfaces

User Interface:

Data flow diagram:

Quiz Game App

Login Page

Registration Page

Store User Information

Home Screen

Retrieve Quiz Question from MongoDB

Quiz Options

Collect User Answer

Store user quiz response in MongoDB

Calculate final Score

Store user Quiz score in MongoDB

User Authentication

Select new quiz

Answer Questions

Calculate Score

Display result Page

Show user score

# Performance Test

ome performance test constraints for the quiz game project:

* Concurrent Users: Test the system with multiple concurrent users to determine how well it handles simultaneous requests. Start with a small number of concurrent users and gradually increase the load to stress-test the system.
* Response Time: Measure the response time for different operations such as login, registration, retrieving quiz questions, submitting answers, and displaying results. Ensure that response times remain within acceptable limits even under heavy load.
* Database Performance: Evaluate the database's performance in terms of read and write operations. Test the time taken to fetch quiz questions and store user responses in the database.
* Error Handling: Intentionally introduce errors and exceptions to see how the system handles them. Measure the system's ability to recover from errors gracefully and provide meaningful error messages to users.
* Timer Function Accuracy: Test the timer function to ensure that it accurately triggers the automatic submission of the quiz when the time runs out. Verify that it works consistently and reliably.

## Test Plan/ Test Cases

**Test Case Name:** User Registration

**Test Case Description:** Verify that a user can successfully register for the quiz game.

**Test Steps:**

1. Open the quiz game application.
2. Click on the "REGISTER" option on the home screen.
3. Enter valid details for Name, Age, User ID, and Password.
4. Check the "All details are Correct" checkbox.
5. Click on the "Submit" button to register.
6. Verify that the user is successfully registered and a confirmation message is displayed.
7. Check the MongoDB database to ensure that the user's information is stored correctly.

**Expected Results:**

* The user should be able to register successfully with valid details.
* After registration, the user's information should be stored in the MongoDB database.

**Test Case Name:** User Login

**Test Case Description:** Verify that a registered user can log in to the quiz game.

**Test Steps:**

1. Open the quiz game application.
2. Click on the "LOGIN" option on the home screen.
3. Enter the correct User ID and Password of a registered user.
4. Click on the "Login" button to log in.
5. Verify that the user is successfully logged in and redirected to the quiz page.
6. Check the MongoDB database to ensure that the user's login activity is recorded.

**Expected Results:**

* The registered user should be able to log in successfully using valid credentials.
* After login, the user should be redirected to the quiz page.

**Test Case Name:** Quiz Gameplay **Test Case Description:** Verify that the quiz game displays questions and collects user answers.

**Test Steps:**

1. Log in as a registered user.
2. Click on the "Start Quiz" option on the quiz page.
3. Verify that the first quiz question is displayed with multiple options.
4. Select an answer and click on the "Next" button.
5. Repeat steps 3-4 for all quiz questions.
6. Verify that the user's answers are recorded in the MongoDB database.

**Expected Results:**

* The quiz game should display questions and options to the user.
* The user's selected answers should be recorded in the MongoDB database.

## Test Procedure

**Test Objective:** The test aims to verify the functionality of the quiz game Python project, including user registration, login, quiz gameplay, timer function, and result page.

**Test Setup:**

1. Install the necessary Python packages: PyWebIO, pymongo.
2. Make sure the MongoDB server is running locally.

**Test Steps:**

1. Open the quiz game application.
2. Perform the following sub-tests:

**Sub-Test 1: User Registration**

* Click on the "REGISTER" option on the home screen.
* Enter valid details for Name, Age, User ID, and Password.
* Check the "All details are Correct" checkbox.
* Click on the "Submit" button to register.
* Verify that the user is successfully registered and a confirmation message is displayed.
* Check the MongoDB database to ensure that the user's information is stored correctly.

**Sub-Test 2: User Login**

* Click on the "LOGIN" option on the home screen.
* Enter the correct User ID and Password of a registered user.
* Click on the "Login" button to log in.
* Verify that the user is successfully logged in and redirected to the quiz page.
* Check the MongoDB database to ensure that the user's login activity is recorded.

**Sub-Test 3: Quiz Gameplay**

* Log in as a registered user.
* Click on the "Start Quiz" option on the quiz page.
* Verify that the first quiz question is displayed with multiple options.
* Select an answer and click on the "Next" button.
* Repeat the previous step for all quiz questions.
* Verify that the user's answers are recorded in the MongoDB database.

**Sub-Test 4: Result Page**

* Log in as a registered user.
* Complete the quiz by answering all the questions.
* Verify that the user is redirected to the result page.
* Check the MongoDB database to ensure that the quiz responses and score are recorded correctly.

**Test Pass Criteria:**

* All sub-tests should pass without any errors or failures.
* The user should be able to register, log in, take the quiz, and view the result without any issues.
* The quiz responses and scores should be accurately recorded in the MongoDB database.

Top of Form

## Performance Outcome

The performance of the quiz game Python project can be measured based on several key performance indicators. Here are the performance outcomes for the quiz game project:

1. **User Registration Time:**
   * Measure the time taken for a user to complete the registration process.
   * Aim for fast registration times to provide a seamless user experience.
2. **User Login Time:**
   * Measure the time taken for a user to log in after providing their credentials.
   * Aim for quick login times to minimize user wait times.
3. **Quiz Load Time:**
   * Measure the time taken to load the quiz page and display the first question.
   * Aim for quick quiz load times to engage users promptly.
4. **Quiz Completion Time:**
   * Measure the time taken by the user to complete the entire quiz.
   * Aim for reasonable quiz completion times to keep users engaged.
5. **Database Operations Performance:**
   * Measure the time taken for database operations, such as storing user details, quiz responses, and scores in MongoDB.
   * Aim for efficient database operations to minimize delays in data retrieval and storage.
6. **Scoring Algorithm Efficiency:**
   * Evaluate the efficiency of the scoring algorithm to calculate the user's score based on quiz responses.
   * Aim for a quick and accurate scoring algorithm to provide instant feedback to users.
7. **Result Page Load Time:**
   * Measure the time taken to load the result page after the quiz is completed.
   * Aim for fast result page load times to display the user's score promptly.
8. **Error Handling and Invalid Login:**
   * Evaluate the performance of error handling when users enter invalid login credentials.
   * Ensure that users receive appropriate error messages and are redirected to the registration page if login is invalid.

# My learnings

1. **Designing User Interface with PyWebIO:**
   * Learned how to use PyWebIO library to create interactive and user-friendly web-based user interfaces for the quiz game.
   * Gained experience in displaying questions, collecting user answers, and providing input validation using PyWebIO input functions.
2. **Working with MongoDB:**
   * Acquired knowledge of connecting to MongoDB and setting up a database to store quiz data, user details, and quiz responses.
3. **Implementing User Authentication:**
   * Developed a user login and registration system to allow only registered users to access the quiz game.
4. **Scoring Algorithm and Result Calculation:**
   * Designed and implemented a scoring algorithm to calculate the user's score based on their quiz responses.
   * Gained knowledge into evaluating and comparing user answers with the correct answers to determine the score.
5. **Testing and Performance Evaluation:**
   * Explored various testing approaches, including unit testing, functional testing, and performance testing.
   * Gained experience in evaluating the application's performance based on factors like load times, database operations, and user experience.
6. **Project Management:**
   * Developed skills in project planning for successful project implementation.
   * Explored version control using tools like Git for code management and collaboration.

# Future work scope

Future Scope for Quiz Game Python Project:

1. **Multi-Player Support:** One exciting future scope is to introduce multi-player support, allowing users to compete against each other in real-time quizzes. Implementing a live leaderboard and tracking individual scores during a multi-player quiz will enhance user engagement.
2. **Categories and Levels:** Introducing quiz categories and different difficulty levels will add variety to the quiz game. Users can choose their preferred category and difficulty level, making the quizzes more personalized and enjoyable.
3. **Dynamic Quiz Generation:** Instead of a fixed set of questions, dynamically generating quizzes from a pool of questions stored in the database will ensure that users get different quizzes each time they play.
4. **Time-Based Quizzes:** Expanding the timer function to have variable quiz durations for different quizzes will make the game more challenging. Users will have to answer questions within a specified time.
5. **Interactive Result Page:** Enhancing the result page with graphical representations, detailed performance analysis, and feedback on correct/incorrect answers will provide users with valuable insights into their performance and areas for improvement