VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

An Internship Report On

UPSC QUIZ GENERATION

Submitted in partial fulfillment of the requirement for the award of the

Degree
in
Master of Computer Applications

Submitted by

SUDEEP D R

1CE23MC023

Internship Carried Out At
Scontinet Technologies, Bangalore

Guide
Ms. Astha Tiwari
Assistant Professor,
MCA, CEC

CITY ENGINEERING COLLEGE

Approved by AICTE New Delhi & Affiliated by VTU, Belagavi
Doddakallasandra, Off Kanakapura Main Road,
Next to Gokulam Apartment, Bangalore - 560 062.

CITY ENGINEERING COLLEGE

Approved by AICTE New Delhi & Affiliated by VTU, Belagavi
Doddakallasandra, Off Kanakapura Main Road,
Next to Gokulam Apartment, Bangalore - 560 062.

Department of Master of Computer Applications

CERTIFICATE

This is to certify that the Internship work entitled

UPSC QUIZ GENERATION

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF COMPUTER APPLICATIONS

Of
Visvesvaraya Technological University, Belgaum, Karnataka
Is a result of the bonafide work carried out

By

SUDEEP D R
1CE23MC023

During the academic year 2024-2025

Coordinator Signature
Mr. Naseerhusen. A
Assistant Professor
Dept. of MCA

Guide Signature
Ms. Astha Tiwari
Assistant Professor
Dept. of MCA

Signature of HOD
Dr. Puja Shashi
Prof & HOD
Dept of MCA

Head of Department
Master in Computer Applications
CITY ENGINEERING COLLEGE
Doddakallasandra
Off Kanakapura Main Road
Bangalore 560...

CITY ENGINEERING COLLEGE

Approved by AICTE New Delhi & Affiliated by VTU, Belagavi
Doddakallasandra, Off Kanakapura Main Road,
Next to Gokulam Apartment, Bangalore - 560 062.

Department of Computer Applications (MCA)

22MCA39 Internship

Internship Report Evaluation Format

Student USN: 1CE23MC023

Student Name: SUDEEP D R

Sl. No	Particulars	Review1 (10)	Review 2 (10)	Review3 (10)	CIE (20)	Total (50)
1	NPTEL/Swayam/Moocs Certification Courses (Mandatory)					
2	Internship Scope and Objective					
3	Relevance of the Technology used					
4	Out Come of the Internship					
5	Assignment Using Internship					
Total Marks						

DECLARATION

I, SUDEEP D R student of third semester MCA, City Engineering College, Bangalore, hereby declare that the internship entitle "**UPSC QUIZ GENERATION**" has been independently carried out by me under the supervision of External Guide **Aishwarya, Team lead, SCONTINENT TECHNOLOGIES PVT LTD, Bangalore** and Internal Guide **Ms. Astha Tiwari, Assistant.Professor Dept. of MCA, CEC, Bangalore** and submitted in partial fulfillment of the requirements for the award of degree in **MASTER OF COMPUTER APPLICATIONS** by the Visvesvaraya Technological University during the academic year 2024-2025.

Place: Bangalore
Date:

Name: SUDEEP D R
USN:1CE23MC023
3rd Semester and 2nd year

ACKNOWLEDGEMENT

The satisfaction and euphoria of the successful completion of any task would be incomplete without the mention of the people who made it possible. The constant guidance of these people and encouragement provided by them crowned my efforts with success and glory. I consider it as a privilege to express my gratitude to all those who led and guided me in the course of his project.

First and foremost, I would like to express my gratitude to my institution **City Engineering College, Dr. S Karunakara** for making it possible for me to be a part of this project in the MCA program.

I express my sincere gratitude to **Dr. Puja Shashi, Professor and HOD**, Department of MCA , **City Engineering College**, for providing us the opportunity to materialized this project.

I wish to place my grateful thanks to **Ms. Astha Tiwari,** Assistant. Professor**,** Project coordinator, Department of Computer Application. **City Engineering College**, for her valuable suggestion and necessary guidance during the course of this project.

I express my sincere thanks to my External Guide **Aishwarya, Team lead** ,for guiding me throughout the project.

I am also grateful to **SCONTINENT TECHNOLOGIES PVT LTD, Bangalore** for providing me with an opportunity to work with them and undertake a project of such importance.

Name: SUDEEP D R
USN: 1CE23MC023

SCONTINENT TECHNOLOGIES PVT LTD

BENGALURU, KARNATAKA -560079

In *Recognition* with Ministry Of MSME

INTERNSHIP COMPLETION CERTIFICATE

Date- 07/01/2025,

To Whom It May Concern,

This is to certify that **SUDEEP D R, USN- 1CE23MC023** from **City Engineering College,** has successfully completed his internship with **SCONTNENT TECHNOLOGIES PVT. LTD,** in Offline mode. Throughout their internship, he demonstrated exceptional commitment,professionalism, and enthusiasm.

SUDEEP D R completed internship from **18/11/2024 to 04/01/2025** , successfully.

During **SUDEEP D R's** tenure with our organization, he has actively engaged as **"AI/DS"** intern on **"UPSC QUIZ GENERATION,"** .

In recognition of **SUDEEP D R,** outstanding performance and dedication, we are pleased to present this Internship Completion Certificate. This certificate confirms the successful fulfillment of our internship program.

We extend our best wishes to **SUDEEP D R's** for all his future endeavors and excel in chosen career path.

Regards,

Naveen Chetu..

Chief Marketing Officer,
Scontinent Technologies Pvt Ltd
Date- 22/11/2024

Scontinent Technologies Pvt Ltd
State Bank of India
Bangalore North Bangalore KA 562157

No. 23, 2nd Block, Old Outer Ring Rd, 2 Stage, Nagarabhavi, Bengaluru, Karnataka.
Pin Code- 560072. Contact Us: + 91 9964473910, +91 6363601896
Email ID- office@studentcontinent.com, Visit- https://www.scontinent.com/

EXECUTIVE SUMMARY

This internship at Network City Solutions Bangalore was an internship program. It aimed in training the interns and making them ready to face the corporate world. It included a well-prepared structure of the training program that included few basic concepts from Machine Learning followed by the programming language Python. Soon after this training we were given our streams to continue with the internship which included projects.

This internship further aimed at providing a complete in-depth knowledge about the various technologies used by their company which would make their tasks easier to take the interns further into their company as employees. Various technologies that are a hot topic and used in completing their company projects are taught. Educators from Network City Solutions would handle the classes and train the interns to understand the topic better and apply it during hands-on sessions. Various new techniques and easier way to learn and get the complete knowledge about the topic was imparted.

This internship included not only the technical benefits but added on to contribute further in my personal development and gave in a lifelong memory to cherish.

TABLE OF CONTENTS

1 INTRODUCTION

UPSC Quiz Generation is the process of creating quizzes tailored for aspirants preparing for the Union Public Service Commission (UPSC) examinations. Given the extensive and dynamic nature of the UPSC syllabus, an automated quiz generation system is essential for efficient learning and assessment. This introduction explores the significance of UPSC quiz generation in aiding aspirants through structured revision, self-assessment, and personalized learning experiences. It also outlines the objectives of the study, including examining various quiz generation techniques, addressing challenges, and discussing applications in modern educational systems.

Methods:

Several approaches are used for UPSC quiz generation, including machine learning, natural language processing (NLP), and rule-based techniques. Machine learning algorithms, such as decision trees and reinforcement learning, are commonly employed to generate adaptive quizzes based on user performance. NLP techniques, such as keyword extraction, named entity recognition, and question-answer generation, help in automatically formulating relevant questions from textual resources. Deep learning models, such as transformer-based models (e.g., BERT and GPT), offer advanced capabilities in understanding and generating questions from UPSC-related content.

Challenges:

Despite advancements in technology, UPSC quiz generation still faces challenges. One major challenge is dealing with the vast and evolving nature of the UPSC syllabus, which requires continuous updates to keep the quizzes relevant. The complexity of question framing, including multiple-choice questions (MCQs), comprehension-based questions, and analytical reasoning, adds to the difficulty. Additionally, ensuring fairness and accuracy in question difficulty levels is crucial to providing a balanced learning experience. Another challenge is maintaining engagement and motivation among aspirants while avoiding repetition in quiz content.

Applications:

UPSC quiz generation finds applications in various domains, including e-learning platforms, test series programs, and personalized study plans. In e-learning platforms, automatically generated quizzes can help users systematically revise and test their knowledge. Test series programs leverage quiz generation to offer structured mock

exams that mimic the actual UPSC examination pattern. Personalized study plans use quiz generation to recommend questions based on individual strengths and weaknesses, enhancing learning efficiency and performance.

Key Features:

1.Automatic Quiz Generation: The system automatically generates quiz questions from UPSC-related content without requiring manual intervention, saving time and effort.

2.High Accuracy: Ensures high accuracy in generating relevant and syllabus-aligned questions, allowing aspirants to practice effectively.

3.Scalability: Capable of handling large volumes of UPSC study material efficiently, scaling to accommodate increased user demand and syllabus updates.

4.Real-time Processing: Provides real-time or near-real-time quiz generation, enabling aspirants to access the latest and most relevant questions instantly.

5.Customization: Offers customization options to tailor quizzes based on user preferences, difficulty levels, and specific subjects, enhancing learning outcomes.

1.1 ABSTRACT

In the era of information overload, generating high-quality quizzes for UPSC (Union Public Service Commission) aspirants plays a crucial role in enhancing learning and assessment. This study explores various approaches to automated quiz generation, focusing on machine learning (ML), natural language processing (NLP), and deep learning techniques. The goal is to develop an intelligent system that automatically generates multiple-choice questions (MCQs) from structured and unstructured textual content related to UPSC subjects such as history, polity, economy, geography, and current affairs.

Traditional quiz generation methods rely on manual question creation, which is time-consuming and prone to biases or inconsistencies. To address this, automated quiz generation models are trained using text summarization, named entity recognition, and question-answer pair extraction techniques. However, generating high-quality, contextually relevant questions presents challenges due to the complexity and variability of textual data.

Furthermore, the study highlights the importance of automated quiz generation in e-learning platforms, online test series, and personalized learning systems, ensuring aspirants receive dynamic and well-structured assessments. By leveraging advanced AI techniques, this project aims to enhance the automation and reliability of quiz generation, contributing to the broader field of education technology (EdTech) and artificial intelligence (AI).

Keywords:

UPSC Quiz Generation, Machine Learning, Natural Language Processing, Deep Learning, Question Generation, Information Retrieval, Automated Assessment, Content Personalization, Text Summarization, Named Entity Recognition, Feature Extraction, Transformer Models, AI in Education, EdTech, MCQ Generation, Competitive Exam Preparation.

1.2 OVERVIEW OF THE PROJECT

The project focuses on developing an automated UPSC quiz generation system that creates multiple-choice questions (MCQs) from structured and unstructured textual content. Leveraging machine learning and natural language processing techniques, the system aims to enhance learning, assessment, and preparation for UPSC aspirants by generating high-quality, contextually relevant questions.

Key components and functionalities of the project include:

1. Data Collection: Gathering relevant textual content from various sources such as NCERT books, government reports, UPSC syllabus materials, and online resources to create a diverse dataset for training and testing.
2. Preprocessing: Cleaning and processing the collected text by removing noise, tokenizing, normalizing, and structuring the content to ensure it is suitable for question generation.
3. Feature Extraction: Extracting relevant features from the preprocessed text using techniques such as named entity recognition (NER), keyword extraction, text summarization, and topic modeling to generate meaningful question-answer pairs.
4. Model Development: Building machine learning or deep learning models to generate quiz questions, training them on labeled datasets to learn patterns in question formation. Transformer-based models like T5, GPT, or BERT can be employed to enhance question generation accuracy.

5. Evaluation: Assessing the performance of the quiz generation system using evaluation metrics such as question relevance, grammatical correctness, answer distractor quality, and Bloom's taxonomy compliance to ensure high-quality assessments.

1.3 PROJECT OBJECTIVE

The main objective of a project on UPSC quiz generation is to develop an efficient system that can automatically generate multiple-choice questions (MCQs) from relevant textual content. This involves:

1. Accuracy: Ensuring that the generated quiz questions are contextually relevant, factually correct, and aligned with the UPSC syllabus. High accuracy is crucial for providing aspirants with high-quality assessments.
2. Scalability: Designing a system that can handle large volumes of textual data efficiently, particularly in real-time scenarios where new study materials, government reports, and current affairs updates are continuously being published.
3. Automation: Minimizing human intervention by automating the quiz generation process using machine learning, natural language processing, or deep learning techniques. This improves efficiency and reduces manual effort in question creation.
4. Adaptability: Building a system that can adapt to changing exam patterns, evolving syllabus topics, and updates in current affairs. This involves periodically updating the question-generation models to maintain relevance and accuracy.
5. User Experience: Enhancing the user experience by enabling access to well-structured quizzes through e-learning platforms, test series applications, and personalized learning environments, helping aspirants assess their knowledge effectively.
6. Performance: Optimizing the performance of the quiz generation system in terms of speed, resource usage, and overall efficiency. This ensures smooth operation, quick question generation, and scalability in production environments.

By achieving these objectives, a UPSC quiz generation project aims to facilitate better learning, assessment, and exam preparation for aspirants, ultimately improving their overall knowledge retention and performance.

1.4 PROJECT OUTLINE

1. **Introduction**
    - Overview of the project
    - Importance of UPSC quiz generation
    - Objectives of the project
2. **Literature Review**
    - Review of existing methods and techniques for automated question generation
    - Analysis of challenges and limitations in current approaches
    - Discussion of recent advancements and trends in the field
3. **Data Collection and Preprocessing**
    - Identification of sources for UPSC-related content (NCERT books, government reports, news articles, previous exam papers)
    - Scraping or accessing study material from online databases and APIs
    - Preprocessing steps:
        - Text cleaning
        - Tokenization
        - Stop word removal
        - Lemmatization/Stemming
4. **Feature Extraction**
    - Selection of relevant features from the preprocessed text
    - Exploration of techniques such as named entity recognition (NER), text summarization, and topic modeling for feature representation
5. **Model Development**
    - Selection of appropriate machine learning or deep learning models

- Implementation of the chosen models using libraries like scikit-learn, TensorFlow, PyTorch
- Training the models on labeled datasets for question generation

6. Model Evaluation

- Evaluation metrics: question relevance, grammatical correctness, answer distractor quality, Bloom's taxonomy compliance
- Cross-validation techniques
- Analysis of model performance and potential areas for improvement

7. Deployment and Integration

- Integration of the trained model into an e-learning platform, mobile app, or API
- Development of a user interface for interacting with the quiz generation system
- Deployment on a server or cloud platform

8. Testing and Validation

- Testing the deployed system with real-world UPSC study material
- Validation of generated MCQs against expert annotations or UPSC question patterns

9. Performance Optimization

- Optimization techniques to improve speed and efficiency of the quiz generation system
- Fine-tuning model hyperparameters
- Exploration of techniques for handling scalability issues

10. Future Enhancements

- Discussion of potential future enhancements and extensions to the project
- Incorporation of additional features such as adaptive learning, AI-driven difficulty scaling, and real-time current affairs integration

- Exploration of advanced techniques for improving question diversity and accuracy

1.5 CONTRIBUTION TO THE PROJECT

UPSC quiz generation can be broadly classified into two types: manual question creation and automated question generation.

1. Manual Question Creation:

- In manual question creation, subject matter experts or educators manually draft multiple-choice questions (MCQs) based on study materials and exam patterns.

Significance:

- Human Expertise: Manual question creation allows educators to apply their subject knowledge and expertise to design high-quality, well-structured quiz questions.
- Pedagogical Control: Experts can align questions with UPSC syllabus and exam patterns, ensuring relevance and effectiveness in assessment.
- Contextual Understanding: Manual question creation captures complex reasoning, contextual depth, and analytical aspects that automated systems may overlook.
- Critical Thinking: Subjective questions requiring deep analysis can be crafted manually to enhance aspirants' critical thinking skills.

2. Automated Question Generation:

- Automated quiz generation utilizes machine learning (ML), natural language processing (NLP), and artificial intelligence (AI) to automatically create MCQs from textual content without human intervention.

2 LITERATURE REVIEW

1. Introduction to UPSC Quiz Generation

UPSC quiz generation is an essential application of Natural Language Processing (NLP) and Machine Learning (ML), aiming to automatically generate multiple-choice questions (MCQs) from textual content relevant to the UPSC syllabus. With the increasing volume of study materials, automated quiz generation is crucial for efficient exam preparation, adaptive learning, and personalized assessment.

2. Traditional Approach to Quiz Generation

Early methods of quiz generation relied on manual question creation, where educators and subject matter experts drafted questions based on textbooks, reports, and previous exam papers. However, this method was:

- Time-consuming: Creating a large number of high-quality MCQs required extensive human effort.
- Inconsistent: Variability in difficulty level and question quality depended on the expertise of question setters.
- Limited in scalability: Generating quizzes for new topics and current affairs required continuous manual updates.

3. Machine Learning-Based Approach

The rise of Machine Learning (ML) brought significant advancements in automated question generation by enabling statistical models to learn patterns from UPSC-related textual data. Popular ML techniques include:

- Naïve Bayes (NB): A probabilistic model used for text classification, aiding in selecting relevant text passages for question generation.
- Support Vector Machines (SVMs): A robust classification model that identifies key concepts from study materials for question formation.
- Decision Trees and Random Forests: These models help classify and structure questions based on topic relevance and difficulty level.

ML-based approaches rely on feature engineering techniques such as:

- TF-IDF (Term Frequency-Inverse Document Frequency): Identifies important words from study material for question formation.

- Bag-of-Words (BoW): Converts textual data into numerical representations to train question generation models.

4. Deep Learning for UPSC Quiz Generation

Recent advancements in deep learning have significantly improved the accuracy of automated question generation. Neural networks can capture complex linguistic patterns in textual data. Some popular models include:

- Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTM): Effective for handling sequential text data, ensuring grammatically and contextually correct question generation.
- Convolutional Neural Networks (CNNs): Used for extracting hierarchical textual features to improve question clarity and coherence.
- Transformer-Based Models (BERT, GPT):
    - BERT (Bidirectional Encoder Representations from Transformers): Generates context-aware MCQs by understanding the relationships between words.
    - GPT (Generative Pretrained Transformer): Automatically formulates coherent and diverse UPSC quiz questions based on textual data.

5. Challenges in UPSC Quiz Generation

Despite progress, several challenges remain:

- Handling Ambiguity: UPSC study materials cover multiple subjects and interpretations, making question extraction complex.
- Scalability Issues: Generating large numbers of high-quality questions in real-time requires significant computational resources.
- Bias in Training Data: ML models can inherit biases from historical data, affecting question quality and fairness.
- Multilingual Question Generation: Many UPSC aspirants study in different languages, and automated systems struggle with cross-lingual question creation.
- Mock Test Generation: Creating structured practice tests for aspirants.
- Adaptive Learning Systems: Personalizing quiz difficulty based on learner performance.

3.SYSTEM ANALYSIS

1. Requirements Gathering

- Identify stakeholders' needs and expectations for the UPSC Quiz Generation System.
- Gather requirements related to quiz accuracy, scalability, user interface, and integration with other learning platforms.

2. Feasibility Study

- Assess the technical feasibility of implementing the UPSC Quiz Generation System.
- Evaluate factors such as available technology, expertise, budget, and time constraints.

3. System Design

- Define the architecture of the UPSC Quiz Generation System, including components such as data collection, preprocessing, feature extraction, model development, and deployment.
- Determine the interaction between these components and the flow of data within the system.

4. Data Collection and Preprocessing

- Identify sources of UPSC-related study material and establish mechanisms for data collection from textbooks, news articles, and government reports.
- Preprocess the collected data to remove noise, clean text, tokenize, and normalize content for effective question generation.

5. Feature Extraction

- Select appropriate features from the preprocessed text data, such as TF-IDF vectors, word embeddings, or topic representations.
- Design feature extraction mechanisms that capture the most relevant information for question formation and difficulty categorization.

6. Model Development

- Choose suitable machine learning or deep learning models for question generation and classification, considering factors like performance, scalability, and ease of implementation.
- Train the selected models using labeled question-answer pairs to learn the relationship between textual content and MCQ formation.

7. Evaluation and Testing

- Assess the performance of the system using evaluation metrics such as accuracy, precision, recall, and F1-score.
- Conduct thorough testing to ensure the functionality, robustness, and reliability of generated questions.

8. Deployment and Integration

- Deploy the trained model into a production environment, either on-premises or in the cloud.
- Integrate the quiz generation system with learning management systems (LMS), mobile apps, or web applications for seamless accessibility.

3.1 REQUIREMENT SPECIFICATION

1. Functional Requirements

a) Data Collection

- The system should collect UPSC-related study materials from various sources, including websites, RSS feeds, PDFs, and APIs.

b) Preprocessing

- The system should perform text preprocessing steps such as:
    - Cleaning: Removing unnecessary characters and symbols.
    - Tokenization: Splitting text into individual words.
    - Stopword Removal: Filtering out common but unimportant words.
    - Lemmatization: Reducing words to their base forms.

c) Feature Extraction

- The system should extract relevant features from the preprocessed study material, such as:
    - TF-IDF Vectors: Identifying important terms.
    - Word Embeddings: Converting words into numerical vectors.
    - Topic Representations: Understanding subject-wise relevance.

d) Question Generation

- The system should generate multiple-choice questions (MCQs) from UPSC-related content using:
    - Machine Learning Models (Naïve Bayes, SVM).
    - Deep Learning Models (LSTM, Transformers).

e) Difficulty Level Classification

- The system should classify generated questions into Easy, Medium, and Hard categories using supervised learning techniques.

f) Evaluation

- The system should evaluate the quality and accuracy of generated questions using:
    - Accuracy, Precision, Recall, and F1-score.
    - Manual review and feedback collection from UPSC aspirants.

g) Deployment

- The trained model should be deployed in a production environment, allowing users to:
    - Generate quizzes.
    - Access quizzes via a web or mobile interface.

h) Integration

- The system should integrate with LMS platforms, mobile apps, and recommendation systems to enhance accessibility.

i) Maintenance

- The system should perform regular updates and maintenance, including:
    - Model retraining with new data.
    - Bug fixes and security updates.
    - Handling system patches.

2. Non-Functional Requirements

a) Accuracy

- The system should ensure high accuracy in generating relevant and well-structured UPSC questions.

b) Scalability

- The system should be scalable to handle large volumes of study materials and support multiple users simultaneously.

c) Performance

- The quiz generation process should be fast and efficient, providing results in real-time or near-real-time.

d) Robustness

- The system should be robust to handle:
    - Noisy or ambiguous text data.
    - Changing UPSC syllabus and question patterns.

e) User Interface

- The system should have a user-friendly interface for easy interaction, allowing users to:
    - Customize quiz difficulty.
    - Choose specific subjects.
    - Receive instant feedback.

f) Security

- Implement security measures to protect:
    - Sensitive user data.
    - Prevent unauthorized access.

g) Compliance

- Ensure compliance with data privacy regulations, such as GDPR, HIPAA, and PCI DSS.

h) Documentation

- Provide comprehensive documentation for:
    - Users: Guide on how to use the system.
    - Administrators: Technical specifications, maintenance instructions, and troubleshooting support.

3. Constraints

a) Resource Limitations

- Consider constraints like:
    - Computing power for training deep learning models.
    - Storage capacity for large datasets.
    - Internet bandwidth for API-based data collection.

b) Data Availability

- Performance may depend on:
    - The availability of quality labeled training data.
    - Access to updated UPSC study materials.

c) Regulatory Compliance

- Ensure compliance with legal and regulatory requirements, including:
    - Copyright laws for educational content.

3.2 PYTHON LIBRARIES USED

Several Python libraries are commonly used in projects that involve fetching data from APIs, handling JSON responses, and generating randomized outputs. These libraries provide functionalities for tasks such as data retrieval, processing, and randomization. Below are some key Python libraries used in such projects:

1. Requests

- A popular library for making HTTP requests to web services and APIs.
- Used to fetch data from online sources, including quiz databases, news APIs, and other external platforms.
- Supports GET, POST, PUT, DELETE requests for interacting with web APIs.
- Example Use Case: Fetching quiz questions or news articles from an API.

2. Random

- A built-in Python library for generating random numbers and selecting random elements from lists.
- Used to shuffle multiple-choice options in a quiz.
- Helps in randomizing question selection from a dataset.
- Example Use Case: Shuffling the order of answer choices to make the quiz more challenging.

3. json

- A built-in Python module for parsing and manipulating JSON (JavaScript Object Notation) data.
- Used for converting API responses into a structured format that Python can process.
- Helps in reading, writing, and modifying JSON files for data storage.
- Example Use Case: Extracting quiz questions from API responses formatted in JSON.

4 TYPES & USES

Quiz categorization helps in organizing questions based on different criteria, ensuring better user engagement, difficulty management, and relevance. Below are some key types of quiz categorization and their uses:

01. TOPICAL CATEGORIZATION

- Quizzes are categorized based on their subject matter or topic.
- Example categories include History, Geography, Science, Economy, Politics, Environment, Technology, Sports, and Literature.
- Use Case: Helps learners focus on specific subjects based on their interests or study requirements.

02. DIFFICULTY-BASED CATEGORIZATION

- Questions are classified according to their difficulty level.
- Categories may include Beginner, Intermediate, Advanced, and Expert.
- Use Case: Ensures that users get questions suited to their knowledge level, preventing frustration or lack of challenge.

03. FORMAT-BASED CATEGORIZATION

- Quizzes are grouped based on their question format.
- Formats may include Multiple Choice Questions (MCQs), True/False, Fill in the Blanks, Match the Following, and Case Study-Based Questions.
- Use Case: Allows users to practice different types of questions, especially for exam preparation.

04. EXAM-SPECIFIC CATEGORIZATION

- Quizzes are classified based on specific examinations or certifications.
- Categories may include UPSC, SSC, NEET, JEE, GRE, TOEFL, IELTS, Banking Exams, and Competitive Tests.
- Use Case: Helps aspirants prepare for standardized tests with relevant practice questions.

4.1 USE CASE DIAGRAM

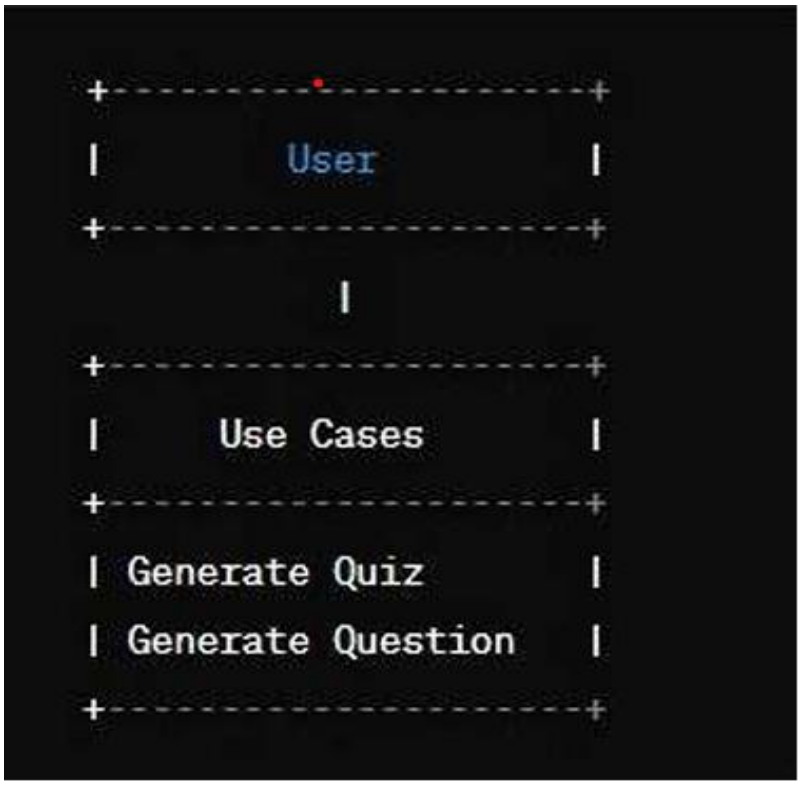


Figure1.1 use case

4.1.1 ACTIVITY DIAGRAM

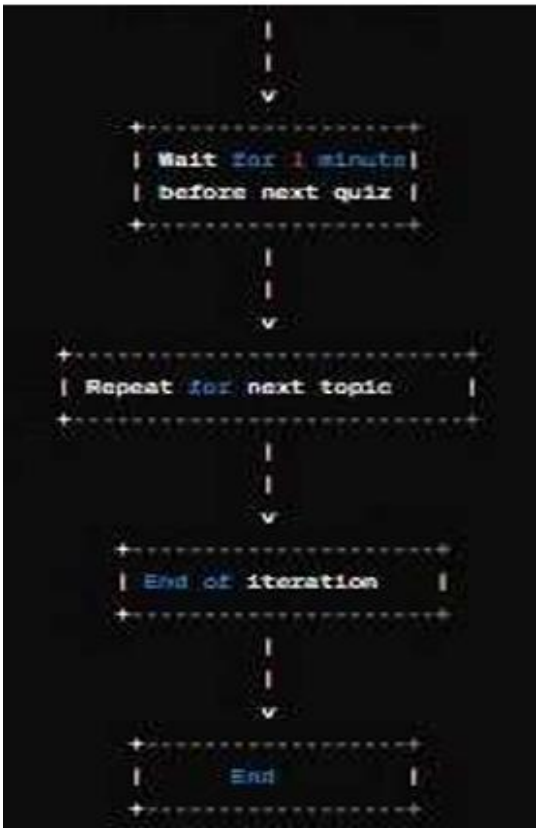


FIGURE1.2 ACTIVITY DIAGRAM

4.2 DATASET OF PROJECT

1. Description

The dataset used in this project is not a pre-stored file (such as CSV, JSON, or SQL database); instead, it is dynamically retrieved from an external API (Open Trivia Database - OpenTDB). The dataset consists of multiple-choice quiz questions categorized into different subjects, such as:

- Politics
- History
- Geography
- Economy
- Science

Each record in the dataset represents a single quiz question, while each column represents an attribute of that question, such as the text of the question, multiple-choice answers, the correct answer, and difficulty level.

2. Features

The dataset contains the following key attributes related to quiz questions:

- Question Text → The actual question displayed to the user.
- Options (A, B, C, D) → Four multiple-choice answer options.
- Correct Answer → The right answer among the four choices.
- Category → The subject of the question (e.g., History, Science, Politics).
- Difficulty Level → Defines the complexity of the question (Easy, Medium, or Hard).
- Type → Specifies whether the question format is multiple-choice (MCQ) or true/false.

3. Target Variable

- The correct answer is the target variable in this dataset.
- The system checks user responses against the correct answer to calculate the final score.
- Unlike datasets used for predictive modeling (e.g., fraud detection, maintenance failure prediction), this dataset is used for knowledge assessment.

4. Data Collection & Usage

- The dataset is fetched dynamically using the requests library, retrieving real-time questions from the OpenTDB API.
- The data is formatted and randomized before being presented to the user in quiz format.
- The program provides instant feedback on whether the user's response is correct or incorrect.

5. Alternative Dataset Storage

If a static dataset were used instead of an API, it could be stored as:

1. CSV File → With columns such as Question, Option A, Option B, Option C, Option D, Correct Answer, Category, Difficulty Level.
2. JSON File → Storing the questions and answers in a structured format, similar to an API response.
3. SQL Database → Storing quiz questions in a relational database, allowing efficient querying based on subject and difficulty.

EXPLANATION OF CHALLENGES ENCOUNTERED DURING IMPLEMENTATION

1. Data Quality and Availability

Challenge

- The OpenTDB API sometimes returns limited or repetitive questions.
- Some responses may contain special characters or HTML entities that affect readability.
- Questions may be too easy or irrelevant for a UPSC-level quiz.

Solution

- Implemented real-time validation to check if the received data is complete and relevant.
- Used text preprocessing techniques (e.g., html.unescape()) to remove unwanted special characters.
- If insufficient questions are fetched, the system can retry the API request or use backup datasets for quiz generation.

2. Feature Engineering

Challenge

- Converting raw API responses into a structured quiz format with shuffled answer options.
- Ensuring that each question has a unique and randomized option order to avoid biases.

Solution

- Used the random.shuffle() function to randomly arrange answer choices for every question.
- Designed a mapping system (A, B, C, D) to associate options with their respective answers.
- Ensured that the correct answer is always included in the options and not lost during shuffling.

3. API Reliability & Rate Limits

Challenge

- The OpenTDB API has rate limits, meaning excessive requests may be blocked.
- If the API server is down, users may not receive quiz questions.

Solution

- Implemented a fallback mechanism where the system retries fetching questions if an API request fails.
- If API requests exceed the limit, the system can delay requests to avoid getting blocked.
- Can be enhanced by caching previously fetched questions to reduce dependency on external APIs.

4. User Interaction & Input Handling

Challenge

- Users may enter invalid responses (e.g., entering "E" when only options "A-D" exist).
- Handling case insensitivity (e.g., "a" vs. "A").
- Providing real-time feedback while keeping the interface simple.

Solution

- Implemented input validation to accept only A, B, C, D (case-insensitive).
- If an invalid input is detected, the system prompts the user to enter a valid option.
- Enhanced user experience with instant feedback on correct/incorrect answers.

5. Lack of Static Dataset for Offline Use

Challenge

- The system relies on an API for quiz generation, meaning it won't work without an internet connection.
- No option to store past quiz questions for repeated practice.

Solution

- A local dataset (CSV, JSON, or SQL database) can be implemented for offline quiz generation.
- Periodically cache fetched questions so they can be reused when the API is unavailable.

6. Scalability and Performance

Challenge

- Handling large numbers of quiz participants at the same time could slow down API requests.
- Need for efficient data retrieval and question formatting.

Solution

- Used optimized data structures (dictionaries, lists) to store and retrieve questions efficiently.
- For a web-based implementation, asynchronous API calls (using asyncio or threading) can improve performance.
- Implementing database storage for frequently asked questions can reduce API dependency.

5. CONCLUSION

The UPSC Quiz Generation project plays a crucial role in automating the creation of multiple-choice questions (MCQs) based on various UPSC-related topics such as politics, history, geography, economy, and science. With the increasing demand for efficient exam preparation tools, an automated quiz system enhances learning efficiency, engagement, and knowledge retention for UPSC aspirants.

This project leverages Python programming and API-based data retrieval to fetch real-time quiz questions from the OpenTDB database. It utilizes libraries such as requests, random, and json to fetch, process, and present quiz questions in an interactive format. The system ensures that questions are shuffled and formatted correctly, making it more dynamic and effective for learners.

The project's workflow involves several critical steps, including data retrieval, question formatting, answer option randomization, user input handling, and score evaluation. Special emphasis is placed on input validation, error handling, and user-friendly interaction, ensuring a smooth experience. Techniques such as randomization of answer options and input validation improve quiz integrity and usability.

One of the key strengths of this project is its ability to automate quiz generation in real time, making it suitable for UPSC aspirants, educators, and online learning platforms. The system can be further enhanced by integrating offline question datasets, graphical interfaces, and AI-driven question difficulty classification. Additionally, tracking user performance can help in personalized learning experiences.

By leveraging automation, real-time question retrieval, and interactive learning techniques, this project provides a valuable tool for UPSC exam preparation, improving the way candidates practice and assess their knowledge.

6 REFERENCES

1. Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach (4th ed.).* Pearson.
2. Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing (3rd ed.).* Pearson.
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). "Distributed Representations of Words and Phrases and Their Compositionality." *Advances in Neural Information Processing Systems.*
4. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *arXiv preprint arXiv:1810.04805.*
5. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). "Attention Is All You Need." *Advances in Neural Information Processing Systems.*
6. Joachims, T. (1998). "Text Categorization with Support Vector Machines: Learning with Many Relevant Features." *European Conference on Machine Learning.*
7. McCallum, A., & Nigam, K. (1998). "A Comparison of Event Models for Naïve Bayes Text Classification." *AAAI Workshop on Learning for Text Categorization.*
8. OpenTDB API. (2024). "The Open Trivia Database – A Free to Use, User-Contributed Trivia API." *Retrieved from https://opentdb.com*
9. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research, 12, 2825-2830.*
10. Chollet, F. (2017). *Deep Learning with Python.* Manning Publications.

7 APPENDIX

Importing Libraries:

- requests: A Python library used for making HTTP requests to fetch data from APIs, such as Open Trivia Database (OpenTDB).
- random: Used for shuffling answer choices in the quiz to ensure a randomized order for each attempt.
- json: Helps parse JSON responses from APIs, making it easier to extract and structure data.

Fetching Quiz Questions:

- The requests.get() function is used to retrieve multiple-choice questions (MCQs) from OpenTDB.
- The fetched data is stored in JSON format, which is then processed to extract questions, correct answers, and incorrect answer choices.

Handling and Formatting Data:

- Extracted question text, correct answer, and incorrect answers are merged into a shuffled list of options using random.shuffle().
- Options are assigned labels (A, B, C, D) to match the multiple-choice quiz format.

User Interaction and Answer Validation:

- The user is prompted to select an answer from (A, B, C, D).
- The response is validated, and immediate feedback (Correct / Incorrect) is provided.
- If the answer is incorrect, the correct answer is displayed for learning purposes.

Scoring System:

- A score counter keeps track of the user's performance throughout the quiz.
- At the end of the quiz, the total score is displayed along with the number of correct answers.

Ensuring Randomization:

- random.shuffle() ensures that answer choices are displayed in different orders for each quiz attempt, preventing users from memorizing fixed patterns.

Printing Final Results:

- At the end of the quiz, the final score is displayed as "Your Final Score: X/5", summarizing the user's performance.

This appendix provides a structured overview of the key components of the UPSC Quiz Generation project, detailing how the libraries are used and how quiz questions are fetched, processed, and presented to users.

7.1 CODE SNIPPET

```
import requests
import random


def fetch_questions(category, num_questions=5):
    """Fetches MCQ questions from a free API."""
    url = f"https://opentdb.com/api.php?amount={num_questions}&category={category}&type=multiple"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        return data.get("results", [])
    return []


def format_question(data):
    """Formats API response into a standard MCQ format."""
    question = data["question"]
    incorrect_answers = data["incorrect_answers"]
    correct_answer = data["correct_answer"]
    options = incorrect_answers + [correct_answer]
    random.shuffle(options)
    option_map = {chr(65 + i): opt for i, opt in enumerate(options)}
    correct_option = [k for k, v in option_map.items() if v == correct_answer][0]
```

```
    return question, option_map, correct_option


def generate_quiz(category, num_questions=5):
    """Generates a UPSC quiz with real-time questions."""
    print(f"\n 🔹 UPSC Quiz on {category} 🔹 \n")
    score = 0
    questions = fetch_questions(category, num_questions)

    for i, q_data in enumerate(questions, 1):
        question, options, correct_option = format_question(q_data)
        print(f" ❓ Question {i}: {question}\n")
        for key, value in options.items():
            print(f"{key}) {value}")

        user_answer = input("\nYour answer (A/B/C/D): ").strip().upper()

        if user_answer == correct_option:
            print(" ✅ Correct!\n")
            score += 1
        else:
            print(f"❌ Incorrect! The correct answer was {correct_option}\n")

    print(f"\n🎯 Your Final Score: {score}/{num_questions}\n")
```

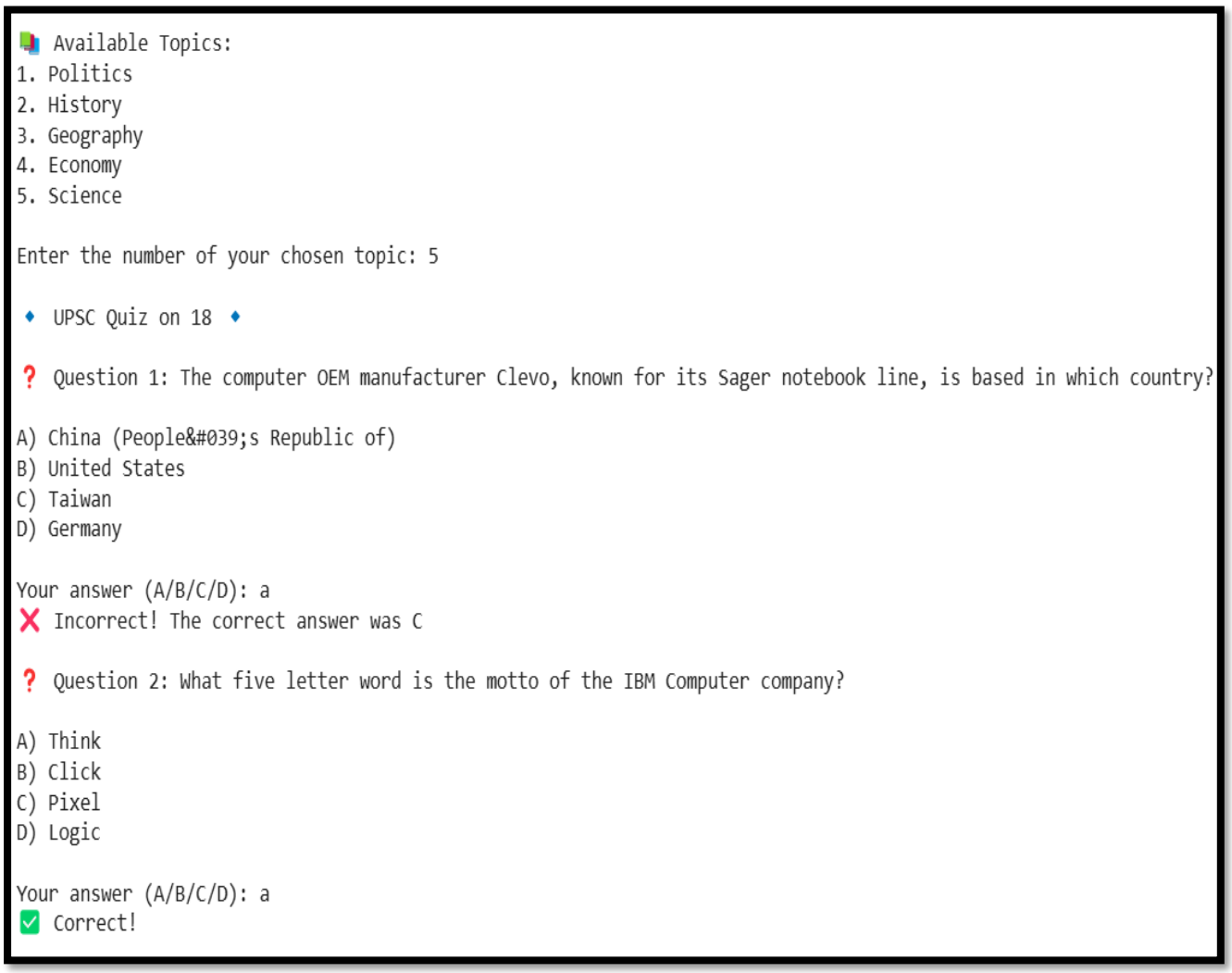
```
def main():
    """Allows the user to select a topic and take a quiz."""
    categories = {
        "1": ("Politics", 24),  # General Politics (OpenTDB Category ID)
        "2": ("History", 23),   # History
        "3": ("Geography", 22), # Geography
        "4": ("Economy", 17),   # Science & Nature (closest match)
        "5": ("Science", 18)    # Science: Computers
    }

    print("\n📚 Available Topics:")
    for key, (name, _) in categories.items():
        print(f"{key}. {name}")

    choice = input("\nEnter the number of your chosen topic: ").strip()
    if choice in categories:
        category_name, category_id = categories[choice]
        generate_quiz(category_id, num_questions=5)
    else:
        print("❌ Invalid choice! Please restart and enter a valid number.")

if __name__ == "__main__":
    main()
```

7.2 SNAPSHOTS



```
❓ Question 3: Which computer language would you associate Django framework with?

A) Python
B) C#
C) C++
D) Java

Your answer (A/B/C/D): a
✅ Correct!

❓ Question 4: The numbering system with a radix of 16 is more commonly referred to as

A) Octal
B) Duodecimal
C) Hexidecimal
D) Binary

Your answer (A/B/C/D): c
✅ Correct!

❓ Question 5: How many bits make up the significand portion of a single precision floating point number?

A) 15
B) 8
C) 53
D) 23

Your answer (A/B/C/D): d
✅ Correct!


🎯 Your Final Score: 4/5
```