A
Project Report
On
" AI-Powered Question Bank Generator with Adaptive Bloom's Taxonomy Integration "

Prepared by

Varad Uttarwar(612203181)
Vedant Shenave(612203188)
Vedanti Watekar(612203189)

Under the guidance of

Dr. Jibi Abraham

Submitted to

COEP Technological University

Department of CS &IT

Software Engineering-II

Of 6th Semester of B.Tech

Jan-May 2025

CERTIFICATE

This is to certify that the report entitled "**AI-Powered Question Bank Generator with Adaptive Bloom's Taxonomy Integration**" is a bonafied work carried out by **612203181(Varad Uttarwar),612203188(Vedant Shenave),612203189(Vedanti Watekar)** under the guidance and supervision of **Dr. Jibi Abraham** for the subject **SE-II** of 6th Semester of Bachelor of Technology in **Computer Engineering** at COEP Technological University.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Guide Name	Dr. Jibi Abraham
Dept	Computer Science and Engineering

ABSTRACT

Effective academic assessments require well-structured question banks that test students across different cognitive levels. Automating this process can help educators generate diverse and well-balanced questions efficiently, ensuring comprehensive evaluation. Bloom’s Taxonomy is a framework used in education to classify learning objectives into six levels: Remember, Understand, Apply, Analyze, Evaluate, and Create. It helps in designing questions that assess not only basic knowledge but also higher-order thinking skills. By incorporating Bloom’s Taxonomy, assessments become more structured, promoting deeper learning and critical thinking among students.

Traditional question paper creation at COEP is repetitive, inconsistent in difficulty, and often lacks structured variation. Existing tools either fail to adapt to different subjects or do not provide comprehensive question formats. Moreover, students face unpredictable question patterns and a shortage of structured practice materials, making exam preparation difficult. Without an efficient and adaptive system, both educators and students struggle with assessment quality and exam readiness.

To solve these issues, our Question Bank Generator provides an AI-driven, syllabus-adaptive solution tailored for COEP. It dynamically generates structured question sets, reducing faculty workload while ensuring balanced difficulty levels through Bloom’s Taxonomy. The tool supports multiple question formats, offers an export feature for easy distribution, and ensures consistency in assessments. By integrating automation and adaptability, this system enhances the academic process at COEP, making exam preparation more efficient for students and question creation seamless for educators.

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Introduction to the area of work

The education sector is undergoing a digital transformation, with assessment processes being a key area for innovation. Traditional methods of creating question banks and exams are being reimagined through artificial intelligence. Our project focuses on developing an intelligent system that automates the generation of academic questions while incorporating Bloom's Taxonomy - a well-established framework for categorizing educational objectives. This approach ensures the creation of balanced assessments that evaluate students across different cognitive levels, from basic knowledge recall to higher-order thinking skills like analysis and evaluation. The system aims to support educators by providing a reliable tool that maintains academic rigor while significantly reducing the time and effort required for exam preparation.

1.2 Gap in the current system

Current assessment creation methods present several challenges that our project addresses. Faculty members typically spend excessive amounts of time manually developing questions, often leading to inconsistent quality and difficulty levels across different exams. While some digital tools exist, they frequently lack the flexibility to adapt to various subjects or the capability to generate a comprehensive range of question types. Many existing solutions focus only on basic multiple-choice questions without considering the full spectrum of cognitive skills that assessments should evaluate. Additionally, there is often no systematic way to ensure questions properly align with course learning objectives or cover the entire syllabus proportionally. These limitations result in assessments that may not accurately measure student learning or provide meaningful feedback.

1.2 Problem Statement

The core challenge we address is developing an automated question generation system that overcomes the limitations of current methods while maintaining educational quality. The system must be capable of producing diverse, syllabus-aligned questions that cover all levels of Bloom's Taxonomy. It needs to generate questions that are not only factually accurate but also pedagogically sound, properly assessing different cognitive skills. The solution must maintain consistency in question quality while providing enough variety to create comprehensive assessments. Additionally, the system should significantly reduce the time and effort required for exam preparation without compromising academic standards. This requires careful balancing of automation with educational expertise to create a tool that educators can trust and rely on for their assessment needs.

1.4 Project Objectives

- Develop an AI-powered question generation system that understands curriculum content and produces educationally valid questions automatically.

- Implement Bloom's Taxonomy integration to systematically create questions at different cognitive levels, ensuring balanced assessments that test both foundational knowledge and critical thinking skills.

- Support multiple question formats including multiple-choice, short answer, and long answer questions, each designed to assess different types of knowledge and skills.

- Enable quiz generation directly from PDF course materials, allowing educators to quickly transform existing content into assessable formats.

- Provide flexible export functionality so generated question banks can be easily incorporated into existing examination systems or learning management platforms.

Chapter 2 Project Management Plan

2.1 Lifecycle Model Used

For developing our Question Bank Generator, we chose the **Agile methodology** because it works well for projects that need flexibility and regular improvements. Here's how we used it:

1. **Short Work Cycles (Sprints)**
   - We divided the work into 2-week periods called "sprints"
   - Each sprint focused on completing specific features

2. **Continuous Improvements**
   - After each sprint, we tested what we built
   - We got feedback from teachers and made changes accordingly

3. **Regular Team Check-ins**
   - We had quick daily meetings to discuss progress
   - Problems were solved immediately

4. **Flexible Planning**
   - We could adjust our plans as we learned more
   - New requirements could be added easily

5. **Early and Frequent Testing**
   - Teachers could try out features early
   - Their suggestions helped improve the system

Chapter 3 Software Requirements Specification

3.1 Software Requirement Specification (SRS)

3.1.1. Introduction

3.1.1.1 Purpose

The Question Bank Generator is an automated system that generates multiple-choice questions (MCQs), short-answer questions, and long-answer questions using an LLM (Large Language Model). The tool will assist educators and examiners in efficiently generating question sets for various subjects.

3.1.1.2 Document Conventions

This document follows the IEEE standard for Software Requirements Specification. All requirements are labeled with unique identifiers for reference.

3.1.1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, quality assurance teams, and potential users (educators and examiners) who will interact with the system.

3.1.1.4 Product Scope

The system is designed as a web-based application using Streamlit and LangChain to generate various types of questions. Users can provide a subject, syllabus, and example questions, and the system will generate new questions accordingly.

3.1.1.5 References

- IEEE Software Requirements Specification Template
- LangChain and Streamlit Documentation

3.1.2. Overall Description

3.1.2.1 Product Perspective

The Question Bank Generator is a standalone application with an intuitive UI for educators to input syllabus details and receive automatically generated questions.

3.1.2.2 Product Functions

- Generate MCQs, short-answer, and long-answer questions
- Export generated questions as a DOCX file
- Allow user input for subject and syllabus details
- Support multiple question formats

3.1.2.3 User Classes and Characteristics

- **Educators:** Need question generation for exams.
- **Examiners:** Require structured question banks.
- **Students:** May use the system for practice questions.

3.1.2.4 Operating Environment

The system will run in a browser and requires:

- Python 3.8+
- Streamlit for UI
- LangChain with LLM backend

3.1.2.5 Design and Implementation Constraints

- Requires an internet connection for LLM API calls.
- Uses Groq's Llama3-8b-8192 model.
- Limited to text-based question generation.

3.1.2.6 User Documentation

- User Guide for input fields and output format
- FAQ section

3.1.2.7 Assumptions and Dependencies

- Users will have a valid API key for LLM integration.
- System performance depends on the response time of the LLM.

3.1.3. External Interface Requirements

3.1.3.1 User Interfaces

- A web-based UI using Streamlit
- Sidebar for user input
- Main panel for displaying generated questions

3.1.3.2 Hardware Interfaces

- Compatible with any desktop or laptop with internet access.

3.1.3.3 Software Interfaces

- Integration with LangChain and OpenAI/Groq LLM APIs.
- Uses Streamlit for web-based rendering.

3.1.3.4 Communications Interfaces

- HTTP-based API calls to the LLM.
- Downloadable DOCX format output.

3.1.4. System Features

3.1.4.1 MCQ Question Generation

Description: Generates MCQs based on user-input subject and syllabus.

- **Priority:** High
- **Functional Requirements:**
    - Accepts subject name, syllabus, and example MCQs as input.

- Generates a specified number of MCQs in a formatted structure.

3.1.4.2 Short Question Generation

Description: Generates short-answer questions based on the syllabus.

- **Priority:** High
- **Functional Requirements:**
  - Accepts syllabus and example short-answer questions as input.
  - Generates structured short-answer questions.

3.1.4.3 Long Question Generation

Description: Generates long-answer questions.

- **Priority:** Medium
- **Functional Requirements:**
  - Accepts example long-answer questions.
  - Outputs structured long-answer questions.

3.1.4.4 Question Export to DOCX

Description: Allows users to download generated questions.

- **Priority:** High
- **Functional Requirements:**
  - Converts generated questions into a DOCX file.
  - Provides a download button for users.

3.1.5. Other Nonfunctional Requirements

3.1.5.1 Performance Requirements

- Response time should be under 10 seconds for question generation.
- Downloadable files should be generated in under 5 seconds.

3.1.5.2 Safety Requirements

- Ensure secure handling of user inputs.

3.1.5.3 Security Requirements

- API keys should not be hardcoded.
- Secure connection (HTTPS) required.

3.1.5.4 Software Quality Attributes

- **Usability:** Simple UI with clear input fields.
- **Scalability:** Should handle large question sets efficiently.
- **Reliability:** Should function without crashes.

3.1.5.5 Business Rules

- The system must not generate explicit, harmful, or unethical content.
- Users must agree to the terms before using the application.

3.1.6. Other Requirements

- Localization support for multiple languages (future scope).
- Logging mechanism for tracking API usage.

Appendices

Appendix A: Glossary

- **LLM:** Large Language Model
- **MCQ:** Multiple-Choice Question
- **API:** Application Programming Interface

Appendix B: Analysis Models

- (To be determined)

Appendix C: To Be Determined List

- Future integrations with Learning Management Systems (LMS).

Chapter 4 System Analysis and Design

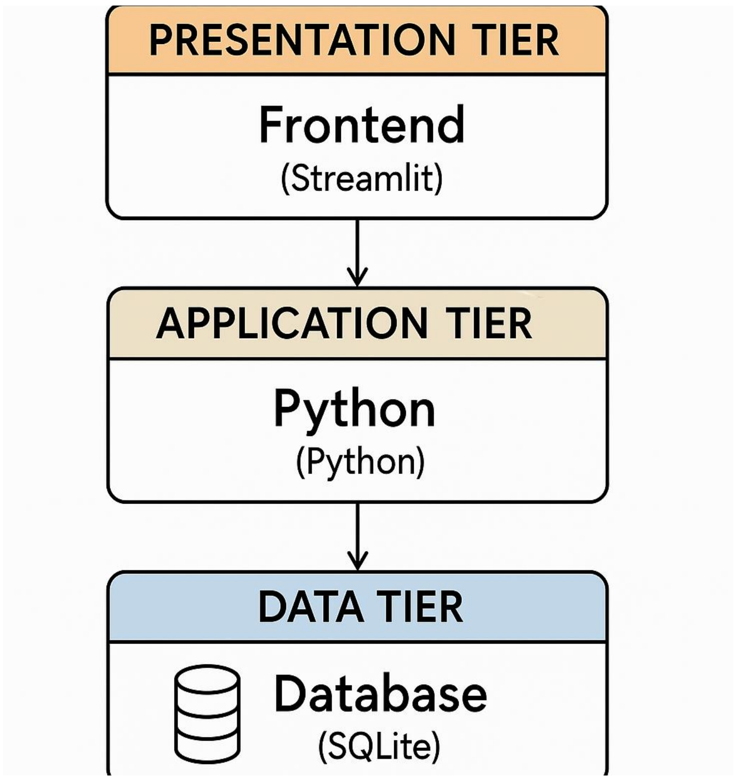
4.1 Proposed Architecture

The system follows a **three-tier architecture**, which separates the application into three main logical layers:

1. **Presentation Tier (Frontend - Streamlit)**
   - This is the **user interface** that educators interact with.
   - Built using **Streamlit**, a Python framework for creating web applications.
   - Handles **user input** (e.g., syllabus upload, question generation requests) and **displays outputs** (generated questions, quiz results).

2. **Application Tier (Backend - Python)**
   - Contains the **core logic** of the system.
   - Processes requests from the frontend (e.g., generating questions via AI, analyzing PDFs).
   - Uses **LangChain and LLM (Large Language Model)** for AI-powered question generation.
   - Manages **data flow** between the frontend and database.

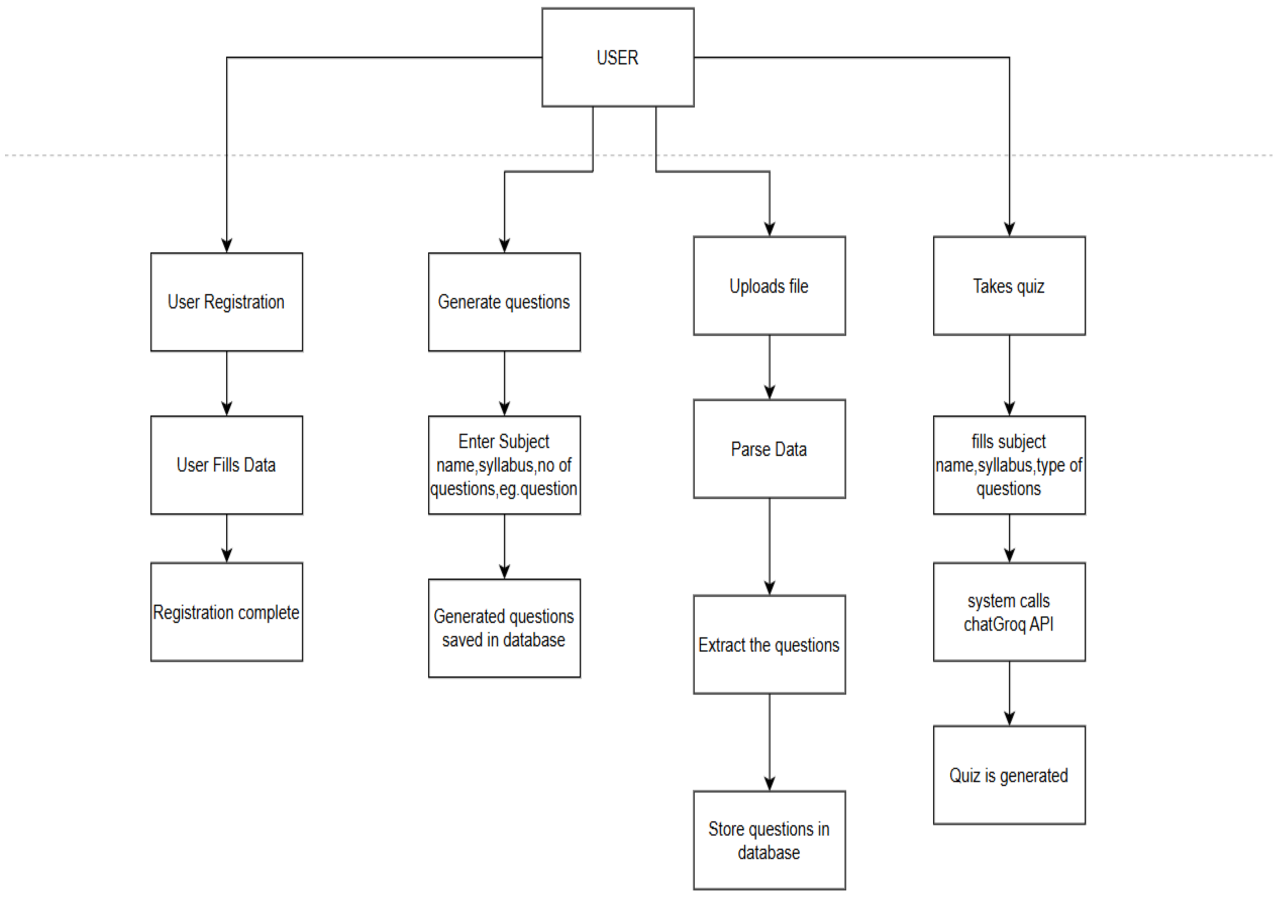
3. **Data Tier (Database - SQLite)**
   - Stores all **question banks, user preferences, and quiz results**.
   - **SQLite** provides a lightweight, file-based database solution.
   - Supports **CRUD operations** (Create, Read, Update, Delete) for managing questions.



4.2 Structured and Data Oriented Design

4.2.1 DFD

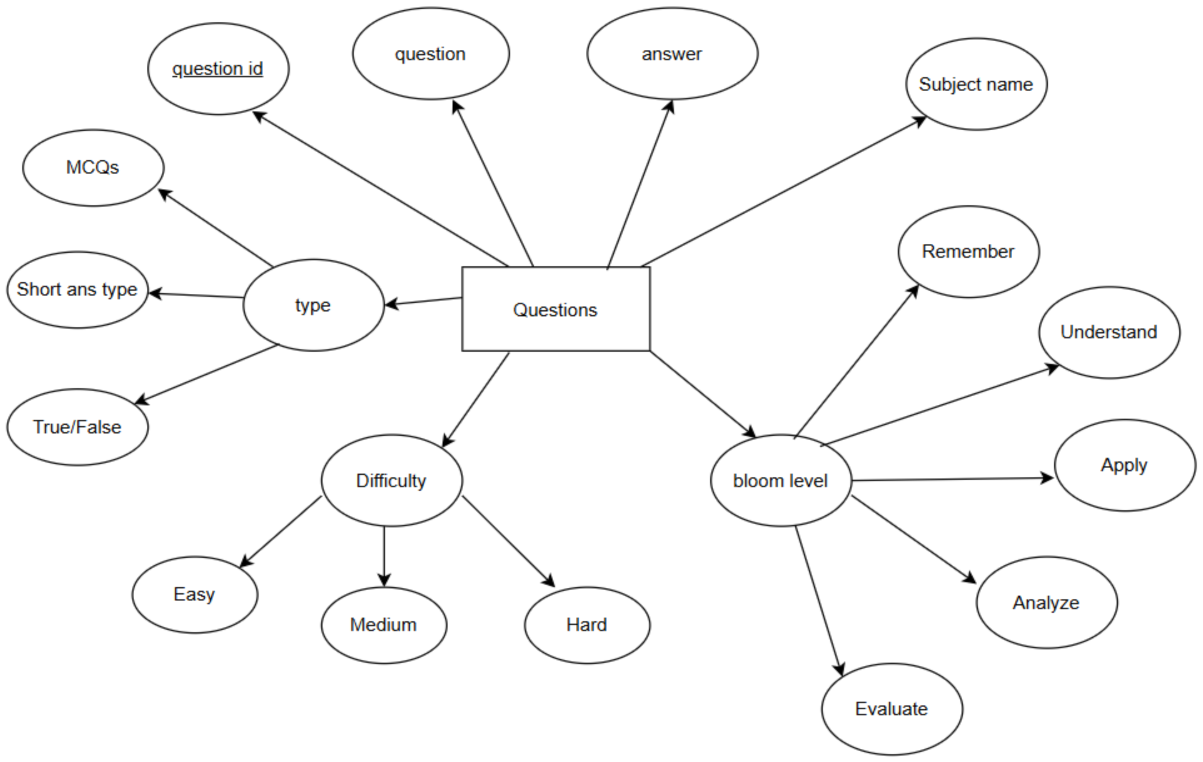
The Data Flow Diagram (DFD) in Figure-1 illustrates the overall flow of user interaction within the quiz system. It outlines key operations including user registration, question generation or uploading, and quiz taking. Each process captures user input, processes data accordingly, and stores or retrieves information from the database or external APIs like chatGroq.


USER
User Registration
User Fills Data
Registration complete
Generate questions
Enter Subject name,syllabus,no of questions,eg.question
Generated questions saved in database
Uploads file
Parse Data
Extract the questions
Store questions in database
Takes quiz
fills subject name,syllabus,type of questions
system calls chatGroq API
Quiz is generated

(Figure – 1)

4.2.2 ER Diagram

The ER diagram in Figure-2 represents a question bank system for educational assessments. The central entity is "Questions", which stores details like question id, question, answer, subject name, and is categorized by type (e.g., MCQs, Short Answer, True/False), difficulty level (Easy, Medium, Hard), and Bloom's Taxonomy level (e.g., Remember, Understand, Apply, Analyze, Evaluate). This helps in organizing and retrieving questions based on educational objectives and complexity

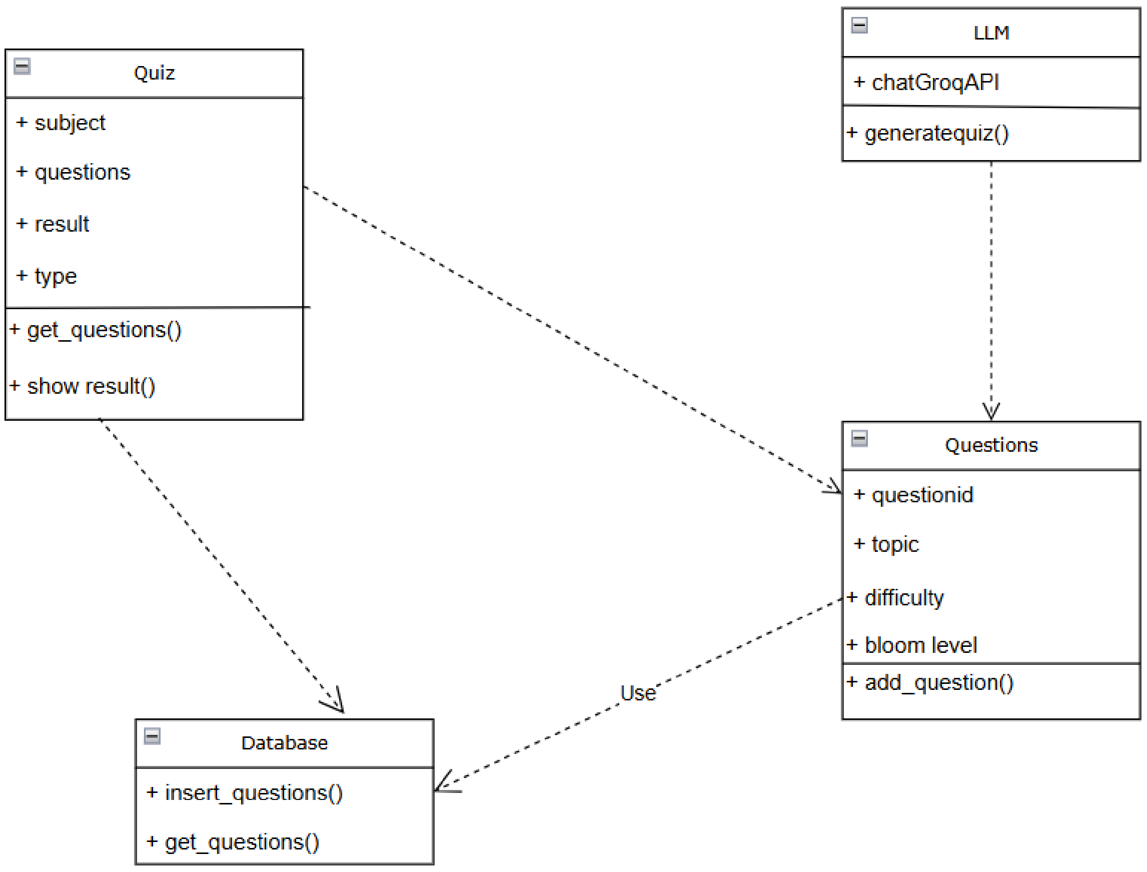


(Figure – 2)

4.3 Object Oriented Design

4.3.1 Class Diagram

The UML Class Diagram in Figure-3 represents the core components of the quiz generation system. The Quiz class manages quiz generation and result handling by retrieving questions from the Database, which provides insertion and retrieval functionality through insert_questions() and get_questions(). The Questions class defines the structure of a question and interacts with both the Database and the LLM (chatGroq API) to generate and store questions using generatequiz() and add_question() methods.

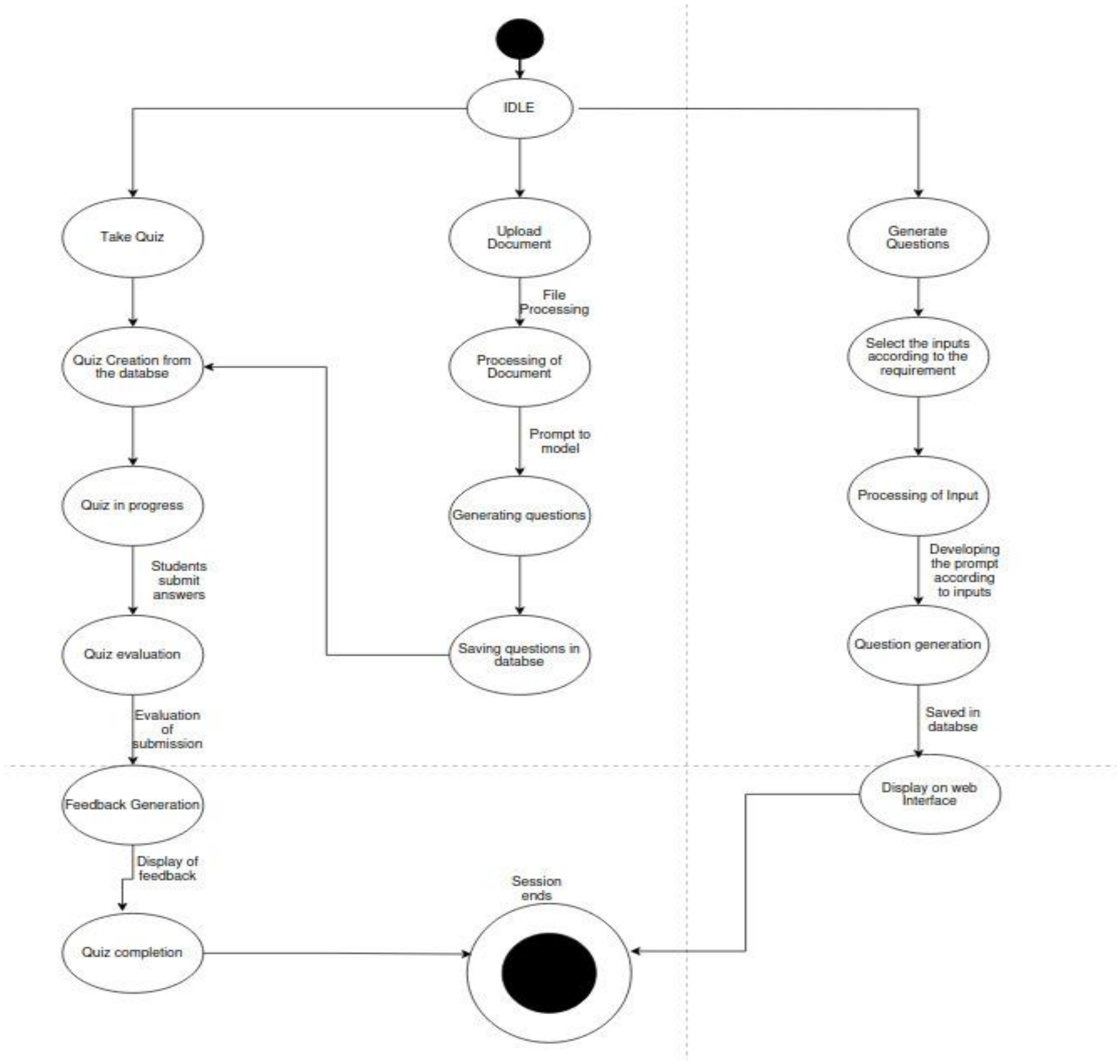


(Figure – 3)

4.4 Standardization using UML

4.4.1 State Diagram

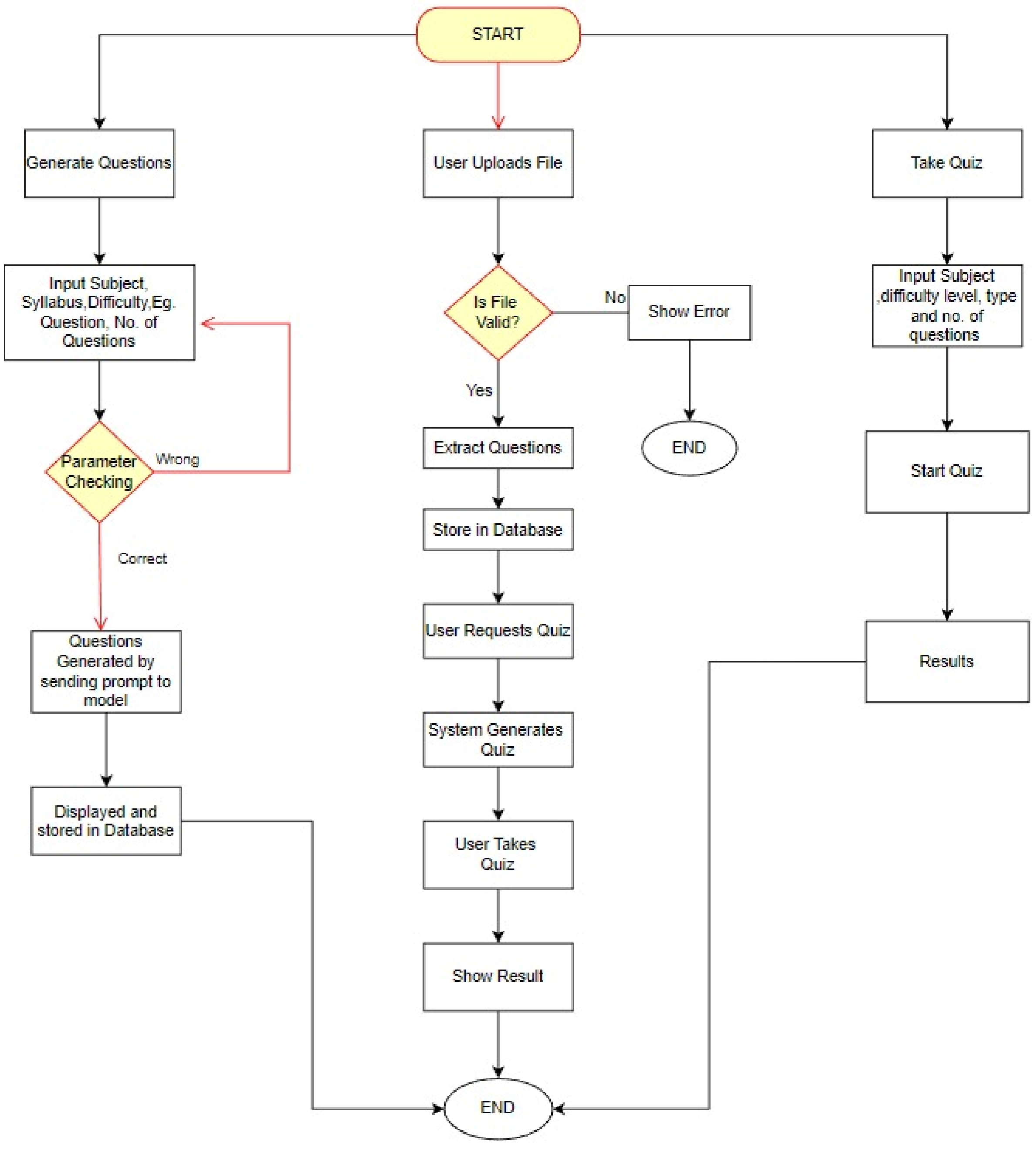
The state diagram in Figure-4 illustrates the workflow of the quiz generation and evaluation system. It begins in the IDLE state, from where the user can:
Take a Quiz: Questions are fetched from the database, the quiz is conducted, responses are evaluated, feedback is generated, and the process ends with quiz completion.
Upload a Document: The document is processed and used to generate questions via an LLM. These questions are then stored in the database.
Generate Questions: Users input parameters to generate questions through the LLM, which are then saved and displayed on the web interface.The system concludes with the Session Ends state, completing the cycle of question generation, quiz delivery, and evaluation.



(Figure – 4)

4.4.3 Activity Diagram

The activity diagram in Figure-5 illustrates the workflow of a quiz generation and evaluation system. It supports two main flows: automatic question generation through prompts or file upload, and taking a quiz. After parameter validation or question extraction, data is stored in a database. Users can then request and take quizzes, and view results, completing the cycle of automated assessment.



(Figure -5)

Chapter 5 Implementation and Testing

5.1 Experimentation setup

The system was developed using the following technologies and frameworks:

1. Python 3.10
- Primary programming language for backend logic and AI integration.
- Provides extensive libraries for natural language processing (NLP), data handling, and web development.
- Ensures cross-platform compatibility (Windows, Linux, macOS).

2. Streamlit (Frontend/UI)
- A lightweight Python-based web framework for creating interactive dashboards.
- Used to build the **user interface** where educators:
  - Upload syllabi/PDFs
  - Select question formats (MCQ, short/long answer)
  - Generate and export question banks
- Enables real-time updates without full page reloads.

3. LangChain (LLM Integration)
- A framework for AI-powered applications using large language models (LLMs).
- Connects the system with Groq’s Llama3-8b-8192 model for question generation.
- Handles:
  - Prompt engineering (ensuring questions align with Bloom’s Taxonomy).
  - Context-aware processing (generating questions from uploaded PDFs/syllabi).

4. Groq’s Llama3-8b-8192 Model
- A high-speed, open-weight LLM optimized for performance.
- Generates diverse, curriculum-aligned questions in real time.
- Supports:
  - Multiple question formats (MCQs, descriptive answers).
  - Difficulty scaling (Remembering → Creating, per Bloom’s Taxonomy).

5. SQLite (Database)
- A **serverless,** file-based database for storing:
  - Generated question banks
  - User preferences (e.g., difficulty settings)
  - Quiz results and feedback

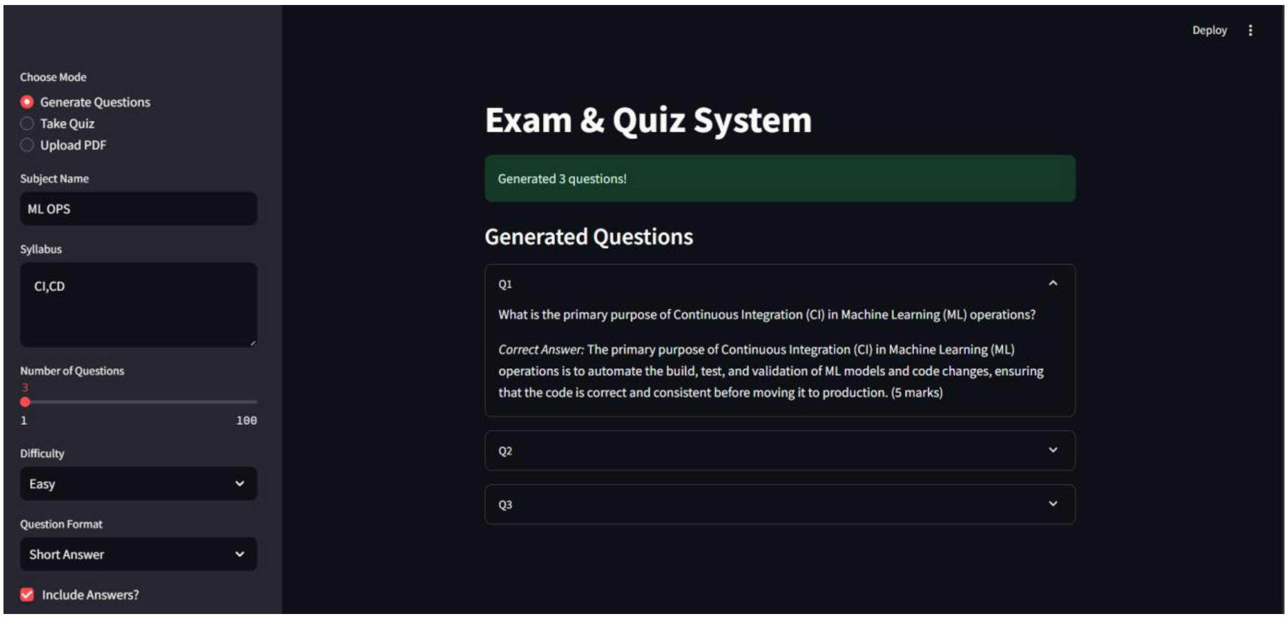
5.2 Test Cases

Test Case ID	Test Steps	Input Data	Expected Results	Actual Results	Test Environment	Execution Status	Bug Severity	Bug Priority	Notes
TC_01	1. Select "Generate Questions" mode 2. Enter subject, syllabus, number of questions 3. Set format to MCQ 4. Click "Generate Questions"	Valid subject, syllabus, number of questions	System generates specified number of MCQs with 4 options each	As Expected	Web Browser	Pass	High	High	Valid
TC_02	1. Select "Generate Questions" mode 2. Set format to "Short Answer" 3. Click "Generate Questions"	Valid subject, syllabus, number of questions	System generates questions requiring short text answers	As Expected	Web Browser	Pass	High	High	Valid
TC_03	1. Leave subject field empty 2. Click "Generate Questions"	Incomplete Input	Error message appears about missing fields	As Expected	Web Browser	Pass	Medium	Medium	Valid
TC_04	1. Upload valid PDF 2. Specify number of questions 3. Click "Generate Quiz from PDF"	Valid PDF file	System extracts text and generates quiz questions	As Expected	Web Browser	Pass	High	High	Valid
TC_05	1. Upload corrupted PDF file 2. Attempt to generate quiz	Corrupted PDF file	Error message about invalid PDF appears	As Expected	Web Browser	Pass	Medium	Medium	Valid

TC_06	1. Click on “take quiz” 2. Select ques- Tion type, syll- abus, no.of qu- estions. 3. give quiz	Generated quiz	System records selected answers	As Expected	Web Browser	Pass	High	High	Valid
TC_07	1. Start quiz 2. Submit without selecting answers	Unanswered quiz	System shows unanswered questions	As Expected	Web Browser	Pass	Medium	Medium	Valid
TC_08	1. Complete quiz with known correct answers 2. Submit	Completed quiz with correct answers	Score matches number of correct answers	As Expected	Web Browser	Pass	High	High	Valid
TC_09	1. Submit completed quiz 2. View results	Submitted quiz	Shows correct/incorrect for each question with explanations	As Expected	Web App / Browser	Pass	Medium	Medium	Valid
TC_10	1. Generate questions 2. Check database	Generated question set	Questions appear in database with all fields	As Expected	Database Backend	Pass	High	High	Valid
TC_11	1. Generate quiz with known answers 2. Check database	Quiz with correct answers	Correct answers stored accurately	As Expected	Database Backend	Pass	High	High	Valid
TC_12	1. Switch between Generate/Take /Upload modes	User clicks on different modes	Interface updates correctly for each mode	As Expected	Web Browser	Pass	Medium	Medium	Valid
TC_13	1. Resize browser window	Window resized to different dimensions	All components remain accessible and properly formatted	As Expected	Web Browser	Pass	Low	Low	Valid

5.3 Screenshots with description

The UI in Figure- 6 illustrates the "Generate Questions" mode of the Exam & Quiz System. Users can input parameters such as Subject Name, Syllabus Topics, Number of Questions, Difficulty Level, Question Format (e.g., Short Answer) and include example questions and Bloom's Taxonomy Level (e.g., Understand, Apply). Based on these inputs, the system generates relevant questions using an AI model. The generated questions, along with their correct answers, are displayed on the right side, making it easy for educators to quickly create customized and cognitively diverse assessments.



Subject Name: ML OPS

Syllabus: CI,CD

Number of Questions: 3 (1–100)

Difficulty: Easy

Question Format: Short Answer

Include Answers? ✓

Exam & Quiz System

Generated 3 questions!

Generated Questions

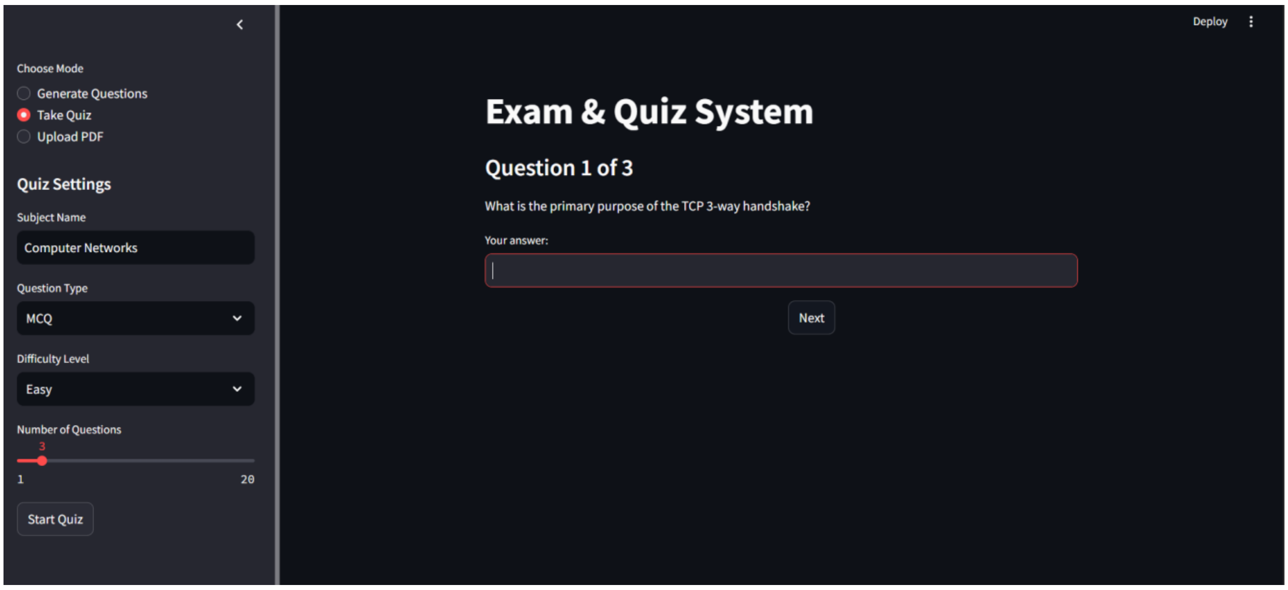
Q1

What is the primary purpose of Continuous Integration (CI) in Machine Learning (ML) operations?

Correct Answer: The primary purpose of Continuous Integration (CI) in Machine Learning (ML) operations is to automate the build, test, and validation of ML models and code changes, ensuring that the code is correct and consistent before moving it to production. (5 marks)

Figure-6

The UI in Figure-7 displays the "Take Quiz" mode of the Exam & Quiz System. Users begin by selecting the Subject Name, Question Type (e.g., MCQ), Difficulty Level, and the desired Number of Questions. After clicking "Start Quiz", questions are presented one at a time, and users can input their answers in the provided field. The interface ensures a clean and focused quiz-taking experience, enabling users to test their knowledge interactively on the selected topic and parameters



Choose Mode

Generate Questions [✓ Take Quiz] Upload PDF

Quiz Settings

Subject Name: Computer Networks

Question Type: MCQ

Difficulty Level: Easy

Number of Questions: 3 (1–20)

Exam & Quiz System

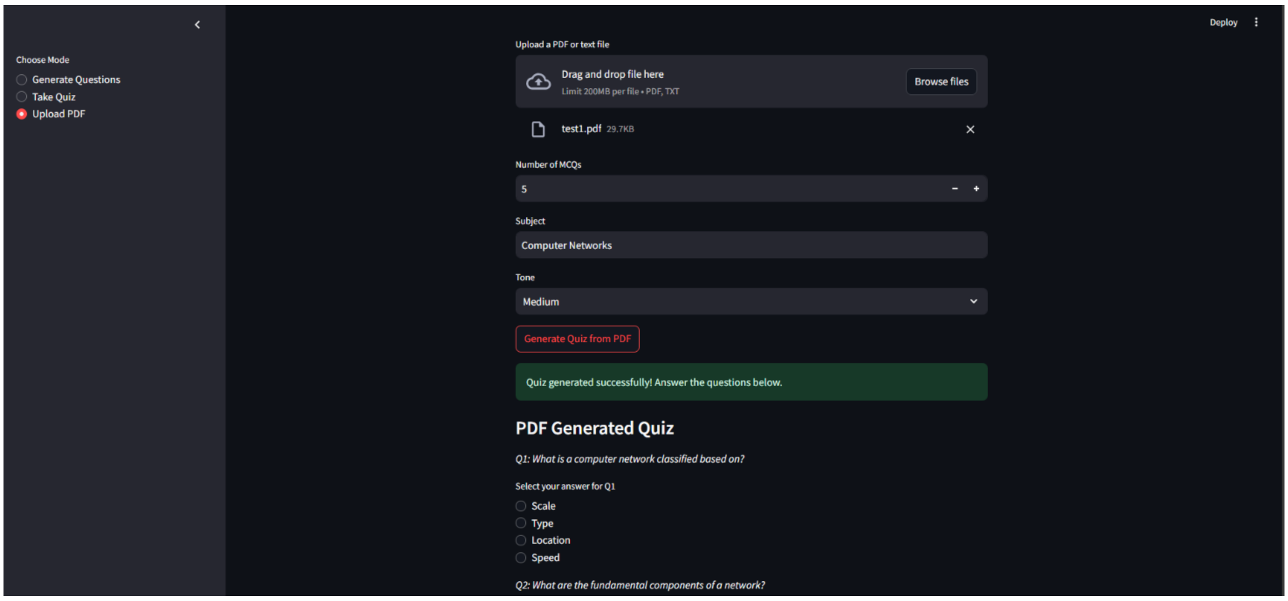
Question 1 of 3

What is the primary purpose of the TCP 3-way handshake?

Your answer:

Figure-7

Figure-8 shows a quiz generation interface where users can upload a PDF, select subject, tone, and number of MCQs. After uploading, the system generates a quiz with multiple-choice questions based on the document content. Also after Giving the quiz the result is generated.



Choose Mode

Generate Questions Take Quiz [✓ Upload PDF]

Upload a PDF or text file

Drag and drop file here
Limit 200MB per file • PDF, TXT

test1.pdf 29.7KB

Number of MCQs: 5

Subject: Computer Networks

Tone: Medium

Quiz generated successfully! Answer the questions below.

PDF Generated Quiz

Q1: What is a computer network classified based on?

Select your answer for Q1

- Scale
- Type
- Location
- Speed

Q2: What are the fundamental components of a network?

Figure-8

Chapter 6: Conclusion and Future Works

6.1: Conclusion:

This project demonstrates a robust AI-powered quiz generation system leveraging modern NLP (Llama-3 via Groq) and database integration. It successfully bridges automated content creation with structured assessment workflows, offering flexibility through PDF processing, adaptive question formats, and Bloom's taxonomy alignment. The modular architecture ensures maintainability, while Streamlit provides an accessible interface for educators and learners. Current limitations include parsing reliability for complex PDFs and scalability constraints in handling large user bases.

6.2: Future Works:

COEP-specific Training:
Fine-tune the model on COEP question papers for targeted preparation.

Image Analysis:
Enable document image processing for better content extraction.

Leaderboard:
Add a leaderboard to encourage user engagement and competition.

Quiz Timer:
Include a timer to simulate real exam conditions.

Multi-language Support:
Support various languages for broader accessibility.

References

1. Bloom, B.S. (1956). Taxonomy of Educational Objectives
2. IEEE Software Requirements Specification Standards
3. LangChain Documentation
4. Streamlit Documentation
5. Groq API Documentation