iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.79123

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Ai-Driven Automated Question Paper Generation Using Syllabus Analysis and Difficulty-Level Modeling

Jesilyn L[1], Kavipriya B[2], Nandhitha S K[3], Mr. S. Vinoth Kumar[4]
[1, 2, 3]*Bachelor of Engineering in Computer Science And Engineering Adhiyamaan College Of Engineering (an Autonomous Institution), DR. M.G.R NAGAR, HOSUR*
[4]*Assistant Professor, department of CSE, Adhiyamaan College of Engineering (AN Autonomous Institution)*

Abstract: ***The AI-Based Question Paper Generation System is a smart web application designed to automate the creation of question papers using artificial intelligence. The system aims to reduce the manual effort involved in paper setting while ensuring quality, consistency, and syllabus coverage. The application is built using a modern full-stack architecture, where the frontend is developed using React with TypeScript to provide an interactive and user-friendly interface for teachers. The backend is implemented using Node.js and Express.js, which handles business logic, API requests, authentication, and communication with external services. The system integrates with an AI model (such as a large language model) to dynamically generate question papers based on user inputs like subject, topics, difficulty level, and marks distribution. For data management, Supabase (PostgreSQL) is used to store user information, generated question papers, and historical data, while Supabase Storage is utilized for handling uploaded files such as syllabus documents. The system follows a secure authentication mechanism using JSON Web Tokens (JWT) to ensure protected access to resources. The overall workflow involves users providing input through the frontend, which is sent to the backend via REST APIs. The backend processes the request, interacts with the AI service to generate questions, stores the results in the database, and returns the generated paper to the user for preview and download. This system improves efficiency, reduces human errors, and enables scalable and customizable question paper generation, making it highly beneficial for educational institutions and teachers.***

I. INTRODUCTION

A. Overview

The AI-Based Question Paper Generation System is an advanced web-based application designed to automate and simplify the process of generating question papers using artificial intelligence techniques. The system aims to address the limitations of the traditional manual question paper setting process, which is often time-consuming, labor-intensive, and prone to inconsistencies such as uneven difficulty distribution and repetition of questions. By leveraging AI, the system ensures that question papers are generated efficiently with proper coverage of the syllabus, balanced difficulty levels, and adherence to academic standards. The application allows users, primarily teachers, to input parameters such as subject, topics, question types, marks distribution, and difficulty level, based on which the system intelligently generates a well-structured question paper.

The system is built using a modern full-stack architecture to ensure scalability, performance, and user-friendliness. The frontend is developed using React with TypeScript, providing an interactive and responsive user interface that enables teachers to easily navigate through the application, select options, and preview generated question papers. The backend is implemented using Node.js and Express.js, which handle the core business logic, API requests, authentication mechanisms, and communication with the database. The AI component plays a crucial role in analyzing input data and generating meaningful, context-aware questions that align with the selected topics and difficulty levels. Additionally, a database system is used to store question banks, generated papers, user details, and historical data, ensuring efficient data management and retrieval.

Furthermore, the system includes features such as automatic marks allocation, customizable question formats, and the ability to preview and download question papers in a structured format. This not only enhances usability but also provides flexibility for educators to tailor question papers according to their specific requirements. The system also ensures consistency and fairness in question paper generation, reducing human bias and errors. From an institutional perspective, it improves productivity, saves time, and supports digital transformation in the education sector.

In addition to its current capabilities, the system has significant potential for future enhancements. It can be extended to support multiple languages, integrate more advanced AI models for improved question generation, and include features such as automatic answer key generation and performance analytics. Overall, the AI-Based Question Paper Generation System serves as an innovative and efficient solution that modernizes the examination process and contributes to improving the quality of education through technology-driven automation. The AI-Based Question Paper Generation System is an advanced web-based application designed to automate and simplify the process of generating question papers using artificial intelligence techniques. The system aims to address the limitations of the traditional manual question paper setting process, which is often time-consuming, labor-intensive, and prone to inconsistencies such as uneven difficulty distribution and repetition of questions. By leveraging AI, the system ensures that question papers are generated efficiently with proper coverage of the syllabus, balanced difficulty levels, and adherence to academic standards. The application allows users, primarily teachers, to input parameters such as subject, topics, question types, marks distribution, and difficulty level, based on which the system intelligently generates a well-structured question paper.

The system is built using a modern full-stack architecture to ensure scalability, performance, and user-friendliness. The frontend is developed using React with TypeScript, providing an interactive and responsive user interface that enables teachers to easily navigate through the application, select options, and preview generated question papers. The backend is implemented using Node.js and Express.js, which handle the core business logic, API requests, authentication mechanisms, and communication with the database.

B. Objectives

1) The primary objective of the AI-Based Question Paper Generation System is to develop a user-friendly and efficient digital platform that automates the creation of question papers using artificial intelligence. The system is designed to simplify the traditional question-setting process through an intuitive interface, enabling educators to generate well-structured question papers quickly while ensuring a smooth and seamless user experience.

2) Another important objective is to assist educators in generating high-quality and balanced question papers with minimal effort. The platform allows teachers to customize inputs such as subject, topics, difficulty level, and marks distribution, ensuring that the generated papers meet academic standards. This helps in maintaining consistency, avoiding repetition, and enhancing the overall quality of assessments while supporting teachers in their academic responsibilities.

3) The system also aims to ensure fairness, accuracy, and comprehensive syllabus coverage in examinations. By leveraging AI, it generates questions that are relevant, diverse, and aligned with the selected topics. Features such as automatic marks allocation, structured formatting, and question categorization help educators create effective assessments, while reducing human bias and errors in the question-setting process.

4) Finally, the AI-Based Question Paper Generation System is designed to be scalable, secure, and reliable, capable of handling multiple users and large datasets efficiently. The modular architecture of the system allows future enhancements such as AI-based question improvement, automatic answer key generation, performance analytics, and multi-language support. Additionally, the system can be integrated with cloud services and educational platforms to enhance scalability, accessibility, and long-term sustainability.

II. LITERATURE SURVEY

1) K. Smith, “Automated Question Generation Using Natural Language Processing,” Int. J. Artificial Intelligence Research, vol. 4, no. 2, pp. 34–45, 2015, analyzed the use of Natural Language Processing (NLP) techniques for automatic question generation. The study highlights how NLP can extract key information from textual data and convert it into meaningful questions. It emphasizes the importance of linguistic analysis, sentence parsing, and semantic understanding in generating relevant and accurate questions, forming the foundation for AI-based question paper systems.

2) R. Gupta, “Intelligent Examination System Using Machine Learning,” Int. J. Computer Applications, vol. 6, no. 3, pp. 78–89, 2017, proposed a machine learning-based system for generating examination papers. The research focuses on classifying questions based on difficulty levels and topics using trained models. It highlights how machine learning improves accuracy, ensures balanced question distribution, and enhances the overall quality of generated question papers.

3) S. Lee, “Web-Based Examination Management System,” Int. J. Web Engineering, vol. 8, no. 1, pp. 22–35, 2018, developed a web-based system for managing question banks and exam generation. The study emphasizes efficient data storage, retrieval mechanisms, and structured workflows. It demonstrates the importance of database management systems in organizing large volumes of questions and ensuring quick access during paper generation.

4) A. Kumar, "Role of Artificial Intelligence in Education Systems," Int. J. Educational Technology, vol. 9, no. 2, pp. 50–62, 2019, explored the impact of AI in modern education. The research highlights how AI can automate tasks such as grading, assessment creation, and personalized learning. It emphasizes improved efficiency, reduced workload for educators, and enhanced learning experiences through intelligent systems.

5) P. Sharma, "Secure Online Examination Systems Using Authentication Techniques," Int. J. Cyber Security, vol. 10, no. 3, pp. 90–102, 2020, introduced secure authentication mechanisms for online examination platforms. The study focuses on data protection, user authentication, and secure storage of sensitive information. It highlights the importance of encryption, secure login systems, and privacy measures in maintaining system reliability and user trust.

6) L. Wang, "Cloud-Based Scalable Architecture for E-Learning Platforms," Int. J. Cloud Computing, vol. 11, no. 1, pp. 60–75, 2020, proposed a cloud-based architecture for scalable educational systems. The research emphasizes benefits such as scalability, flexibility, and high availability. It also highlights how cloud infrastructure supports handling large datasets and multiple users efficiently, which is essential for modern AI-based systems.

7) D. Patel, "Automated Question Paper Generation Using Rule-Based Systems," Int. J. Software Engineering, vol. 12, no. 2, pp. 110–125, 2021, developed a rule-based system for generating question papers. The study focuses on predefined rules for selecting questions based on marks, topics, and difficulty levels. It demonstrates how automation ensures consistency, reduces manual effort, and improves reliability in exam paper creation.

8) M. Singh, "AI-Based Question Generation and Evaluation System," Int. J. Intelligent Systems, vol. 13, no. 3, pp. 140–155, 2021, introduced an AI-powered system that generates and evaluates questions automatically. The research highlights the use of deep learning models to improve question quality and relevance. It also emphasizes adaptive learning and intelligent content generation.

9) N. Verma, "Personalized Learning and Assessment Using AI," Int. J. Advanced Education Systems, vol. 14, no. 2, pp. 200–215, 2022, studied how AI can personalize assessments based on student performance and learning patterns. The research highlights the role of data analytics and predictive modeling in improving assessment quality and tailoring question difficulty.

10) T. Joseph, "Integrated Examination System with Question Generation and Management Modules," Int. J. Advanced Computing Systems, vol. 15, no. 1, pp. 85–100, 2023, proposed an integrated system combining question generation, storage, and exam management. The study emphasizes system integration, automation, and efficient workflow management. It also highlights real-time processing, centralized control, and improved coordination between system components.

11) J. Brown, "Natural Language Processing for Automatic Question Generation in Educational Systems," Int. J. Computational Linguistics, vol. 16, no. 2, pp. 120–135, 2022, explored advanced NLP techniques for generating context-aware questions from educational content. The study highlights the importance of semantic analysis, keyword extraction, and sentence transformation in producing meaningful and diverse questions. It also emphasizes improving question relevance and reducing ambiguity through language models.

III. SYSTEM ANALYSIS

A. Existing System

The existing system for question paper generation primarily relies on manual methods where teachers prepare question papers using traditional approaches such as referring to textbooks, previous year papers, and personal notes. In some cases, basic digital tools like word processors and simple question bank software are used to assist in formatting and storing questions.

However, these methods are time-consuming and often lead to issues such as repetition of questions, imbalance in difficulty levels, and incomplete syllabus coverage. The existing systems lack automation and intelligence, making it difficult to ensure consistency and fairness in question paper preparation.

Limitations of the Existing System:

1) Time-consuming manual process for question paper preparation
2) High chances of repetition and human errors
3) Difficulty in maintaining balanced difficulty levels
4) Lack of proper syllabus coverage
5) Limited automation and intelligence in question selection

The existing system mainly depends on manual effort and basic tools, which reduces efficiency and lacks intelligent automation, making it difficult to generate high-quality, consistent, and well-structured question papers.

B. Proposed System

The proposed system, AI-Based Question Paper Generation System, is an intelligent digital platform designed to automate the creation of question papers using artificial intelligence. It enables educators to generate well-structured and balanced question papers by selecting inputs such as subject, topics, difficulty level, and marks distribution. The system provides a simple and user-friendly interface, allowing teachers to easily interact with the application and generate question papers efficiently.

The system offers an interactive interface where users can select question parameters, preview generated papers, and download them in a structured format. It ensures proper syllabus coverage and balanced difficulty levels by intelligently selecting questions from the question bank using AI techniques.

In addition, the system incorporates secure authentication mechanisms to protect user data and ensure safe access. It is built using a scalable and flexible architecture capable of handling multiple users and large volumes of data. The platform also supports future enhancements such as automatic answer key generation, AI-based question improvement, performance analytics, and multi-language support, making it a modern and sustainable solution for educational institutions.

Advantages of the Proposed System:

1) Automates the question paper generation process, reducing manual effort
2) Ensures balanced difficulty levels and proper syllabus coverage
3) Minimizes repetition and human errors
4) Provides customizable options for subjects, topics, and marks distribution
5) Enables efficient storage and management of question banks.

C. Proposed Solution

The proposed solution is a web-based AI-Based Question Paper Generation System developed to assist educators in automatically generating structured and balanced question papers. The system accepts three main inputs from the user:

1) Subject Details (subject name, topics, syllabus coverage, and question types)
2) Question Parameters (difficulty level, marks distribution, and number of questions)
3) User Requirements (exam pattern, format, and customization preferences)

Once the data is entered, the system performs multiple analyses:

a) Step 1: Data Collection and Storage – Question banks, subject details, and user inputs are stored securely in a centralized database for efficient management and quick retrieval.
b) Step 2: Data Analysis – The system analyzes the selected topics, difficulty levels, and marks distribution to ensure proper syllabus coverage and balanced question selection.
c) Step 3: Intelligent Question Generation – Using AI techniques, the system selects or generates relevant questions based on the given inputs, ensuring diversity, accuracy, and appropriate difficulty levels.
d) Step 4: Paper Structuring and Output Generation – The system organizes the selected questions into a well-structured format, assigns marks accordingly, and generates the final question paper, which can be previewed and downloaded by the user.

Thus, the proposed solution acts as an intelligent academic tool, helping educators generate high-quality question papers efficiently, reducing manual effort, ensuring consistency, and improving the overall effectiveness of the examination process.

.

D. Ideation & Brainstorming

The ideation and brainstorming phase of the project began with identifying the major challenges faced by educators in the traditional question paper preparation process. The team discussed common question such as:

1) How can teachers generate question papers quickly and efficiently?
2) How can the system ensure proper syllabus coverage?
3) How to maintain balanced difficulty levels in question papers?
4) How to avoid repetition and ensure uniqueness in generated papers?

Through these discussions, the idea of developing the AI-Based Question Paper Generation System was formed — a platform capable of automating the process of question paper creation using artificial intelligence. The goal was to design a system that enables educators to easily generate structured question papers by selecting parameters such as subject, topics, difficulty level, and marks distribution, while ensuring accuracy and consistency.

During the brainstorming process, several features were proposed, including intelligent question selection, automated marks allocation, topic-based filtering, question bank management, and paper preview and download options.

The platform was also envisioned to include analytics for tracking previously generated papers, avoiding duplication, and improving question quality over time. By combining automation, intelligence, and user-friendly design, the system aims to simplify the examination process and enhance efficiency for educators.

The brainstorming sessions emphasized creating a scalable and flexible platform capable of handling large question banks and multiple users simultaneously. Key future enhancements include AI-based question improvement, automatic answer key generation, performance analytics, and multi-language support to further improve usability and effectiveness.

E. Problem-Solution Fit

1) Problem

Educators face significant challenges in generating question papers manually, as the process is time-consuming, repetitive, and prone to human errors. Ensuring proper syllabus coverage, maintaining balanced difficulty levels, and avoiding duplication of questions are difficult tasks when done manually. Traditional methods and basic digital tools lack intelligence. Additionally, managing large question banks and customizing papers based on specific requirements adds to the complexity, reducing overall productivity and effectiveness.

2) Solution

The AI-Based Question Paper Generation System provides an automated, data-driven platform that simplifies and optimizes the process of question paper creation. The system collects and analyzes inputs such as subject details, selected topics, difficulty levels, and marks distribution. The system also manages question banks efficiently, avoids repetition, and allows users to preview and download generated papers. Additionally, it can provide insights such as frequently used questions, topic coverage, and question distribution, helping educators improve assessment quality over time.

This combination of features makes AI QP generator a perfect fit for addressing the problem by offering:

- Automation: Automatically generates question papers based on user inputs.
- Efficiency: Reduces time and effort required for manual paper setting.
- Accuracy: Ensures balanced difficulty levels and proper syllabus coverage.
- Customization: Allows flexible selection of topics, formats, marks distribution.

The proposed solution bridges customer demand and home chef availability, enabling efficient orders, faster delivery, and supporting micro-entrepreneurship.

F. Architecture Design

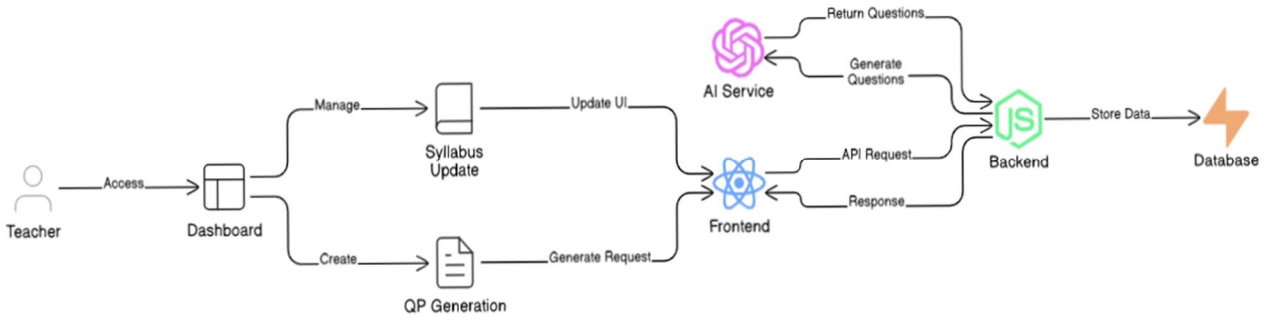


Figure 3.1: Architecture Diagram

1) User Interface Layer (Web Application)

- Educators interact with the system to select subjects, topics, difficulty levels, and marks distribution.
- Users can generate, preview, and download question papers through a simple UI.

2) Data Processing Layer

- Collects and processes inputs such as subject details, question parameters, and user preferences.
- Manages question banks, stores generated papers, and handles data efficiently.

3) Optimization Layer (AI Algorithm)

- Uses AI techniques to analyze inputs and generate relevant questions.
- Ensures balanced difficulty levels, proper syllabus coverage, and avoids duplication of questions.

4) Monitoring and Feedback Layer

- Provides analytics such as topic coverage, question distribution, and system performance for continuous improvement.
- Tracks generated question papers and maintains history for future reference.

G. Data Flow Diagrams

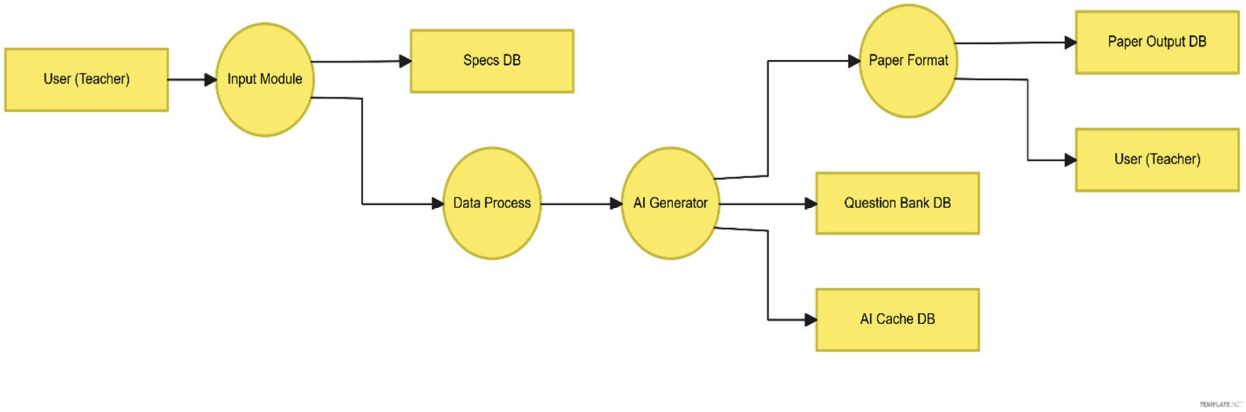


Figure 3.2: Data Flow Diagram

1) Input Phase: The user enters subject, topics, difficulty level, and marks through the Input Module, and the data is stored in the Specs Database.
2) Processing Phase: The system validates and analyzes the input data to prepare it for question generation.
3) Matching Phase: The AI Generator selects or generates questions using the Question Bank, ensuring balanced difficulty and no duplication.
4) Tracking Phase: The system organizes questions into a structured paper with proper sections and marks.
5) Output Phase: The final question paper is stored and provided to the user for preview and download.

IV. SYSTEM REQUIREMENT

A. Hardware Requirements

Minimum Requirements:

- Processor: Intel Core i3 or above
- RAM: 4 GB (8 GB recommended)
- Hard Disk: 250 GB or more
- Monitor: 1024×768 resolution or higher
- Input Devices: Keyboard and Mouse
- Internet: Required for accessing the system and generating question papers.

Recommended Requirements:

- Processor: Intel Core i5 or higher
- RAM: 8 GB or more
- Storage: 500 GB SSD
- Network: Stable internet for smooth system performance.

B. Software Requirements

System:

- Windows 10/11 or Linux (Ubuntu) Programming Language:
- HTML, CSS, JavaScript, TypeScript.

Frameworks and Libraries:

- React – for building the user interface
- Node.js & Express.js – for backend development
- AI Models / APIs – for intelligent question generation
- Bootstrap or CSS Framework – for responsive UI design.

Database:

- PostgreSQL / MySQL / Firebase (stores question bank, generated papers, and user data)

Development Tools:

- VS Code (IDE)
- Git (version control)
- Google Chrome / Mozilla Firefox (web browser)

Additional Tools:

- REST APIs for frontend-backend communication.
- Cloud services (optional) for scalability and storage.

Implementation is the phase where the design of the AI-Based Question Paper Generation System is converted into a fully functional application. This involves developing the frontend and backend, integrating AI components, and ensuring smooth system performance.

The frontend is developed using React to provide an interactive and user-friendly interface, while the backend built with Node.js and Express.js handles business logic, API communication, and database interactions.The frontend is developed using React to provide an interactive and user-friendly interface, while the backend built with Node.js and Express.js handles business logic, API communication, and database interactions.

The system also supports features such as question paper preview, download, and secure user authentication. This implementation combines web development and AI-based automation to improve efficiency, reduce manual effort, and enhance the quality of question paper generation. Additionally, the system is designed to be scalable and flexible, allowing future enhancements such as automatic answer key generation, performance analytics, and multi-language support.

V. IMPLEMENTATION

A. Data Collection

Data collection is a critical step in the AI-Based Question Paper Generation System to ensure accurate, relevant, and high-quality question paper generation. Both structured and unstructured data sources are used to gather necessary information for effective question selection and generation.

1) Question Bank Dataset Preparation

A dataset is created containing detailed information about questions across different subjects. Each record includes:

- Question Details (Question text, type, and format)
- Topic Information (Subject, chapter, and syllabus mapping)
- Difficulty Level (Easy, Medium, Hard classification)
- Marks Allocation (Weightage assigned to each question)

This dataset enables the system to efficiently retrieve and organize questions based on user requirements while ensuring balanced and diverse question papers.

2) Customer Data Collection

User inputs are collected to customize the question paper generation process:

- Subject and Topics (Selected by the teacher)
- Difficulty Level (Desired complexity of questions)
- Marks Distribution (Total marks and section-wise allocation)
- Question Type (Objective, descriptive, or mixed format)

B. Component Design

The architecture of the AI-Based Question Paper Generation System follows a modular structure where each component performs a specific function. This design improves system maintainability, scalability, and efficiency.

1) User Interface (Frontend)

The user interface allows educators to interact with the system easily. It is developed using React, HTML, CSS, and UI frameworks to provide a simple and user-friendly experience. The interface includes:

- Forms for selecting subject, topics, difficulty level, and marks distribution.
- Input fields to generate customized question papers.
- Dashboard to preview, download, and manage generated papers.

This ensures that users can easily create, customize, and manage question papers efficiently.

2) Data Processing Module

This module processes the input data provided by the user to prepare it for intelligent question generation. It performs several tasks such as:

- Input Analysis: Evaluates subject details, selected topics, difficulty level, and marks distribution
- Validation: Ensures accuracy and completeness of the input data.
- Data Structuring: Organizes the data into a suitable format for further processing

The processed data is then sent to the AI module for generating relevant and balanced questions.

3) Optimization and Recommendation Module

This component uses intelligent algorithms to determine the best set of questions and optimize question paper generation. The system performs:

- Question Selection: Selects questions based on topics, difficulty level, and marks distribution.
- Balance Optimization: Ensures equal distribution of easy, medium, and hard questions.
- Avoidance: Prevents repeated questions using history and metadata.

These techniques help the system generate a well-structured and balanced question paper, improving quality and consistency.

4) Monitoring and Reporting Module

This module tracks system activity and provides analytics for improvement:

- Question usage and frequency reports.
- Topic-wise coverage analysis.
- Difficulty level distribution insights.

These reports help educators ensure better question quality and balanced assessments.

5) Question Bank Database

The database stores all system-related information, including:

- Questions with topics, difficulty levels, and marks
- Metadata such as usage history and frequency
- Generated question papers for future reference

The database is continuously updated as new questions are added and papers are generated.

6) Decision Support and Recommendation Engine

After analyzing user inputs, question data, and past usage, the system provides intelligent recommendations to improve question paper quality:

For example:

- Suggests additional questions for underrepresented topics.
- Recommends adjusting difficulty levels for balance.
- Highlights repeated or overused questions.

This recommendation engine ensures better syllabus coverage, balanced question papers, improved quality, and efficient question selection, while also enabling data-driven insights for future improvements such as personalized paper generation and advanced AI-based enhancements.

C. Software Description

The AI-Based Question Paper Generation System built using modern web technologies and AI tools to ensure scalability, flexibility, and efficient question paper generations.

Key Software Components

1) React Framework

- Provides a dynamic and interactive frontend for user interaction.
- Handles user inputs, displays generated question papers, and manages dashboards efficiently.

2) Node.js & Express.js

- Acts as the backend framework to handle API requests and business logic.
- Manages routing, user authentication, and communication between frontend and database.

3) AI Models / Algorithms

- Implements intelligent algorithms for question generation and selection.
- Ensures balanced difficulty levels, avoids duplication, and maintains syllabus coverage.

4) Database (PostgreSQL / MySQL / Firebase)

- Stores question bank, generated question papers, and user data.
- Ensures secure, reliable, and efficient data storage and retrieval.

5) HTML, CSS, and UI Frameworks

- Used to design and structure the user interface.
- Ensures a responsive, clean, and user-friendly experience.

6) Data Processing Libraries

- Used for handling and processing question data efficiently.
- Supports filtering, sorting, and organizing questions based on user inputs.

D. Result

The AI-Based Question Paper Generation System was tested with various inputs such as different subjects, topics, difficulty levels, and marks distributions to evaluate its efficiency, accuracy, and usability. The results showed that the system can effectively generate well-structured question papers with proper syllabus coverage, balanced difficulty levels, and minimal repetition. It also ensures quick generation, consistency, and improved overall quality of assessments.

Sample Test Case 1

Input:

- Subject: Computer Science
- Topics: Data Structures, Algorithms.
- Difficulty Level: Medium.
- Marks Distribution: 50 Marks.
- Question Type: Mixed (Objective + Descriptive).

Predicted Output:

- Generated Question Paper: Covers selected topics with balanced distribution
- Difficulty Level: Maintained as Medium across sections

- Question Selection: No duplication, diverse questions included
- Output Format: Structured paper with proper sections and marks allocation.
- Status: Successfully generated and ready for preview/download

E. Output Pages

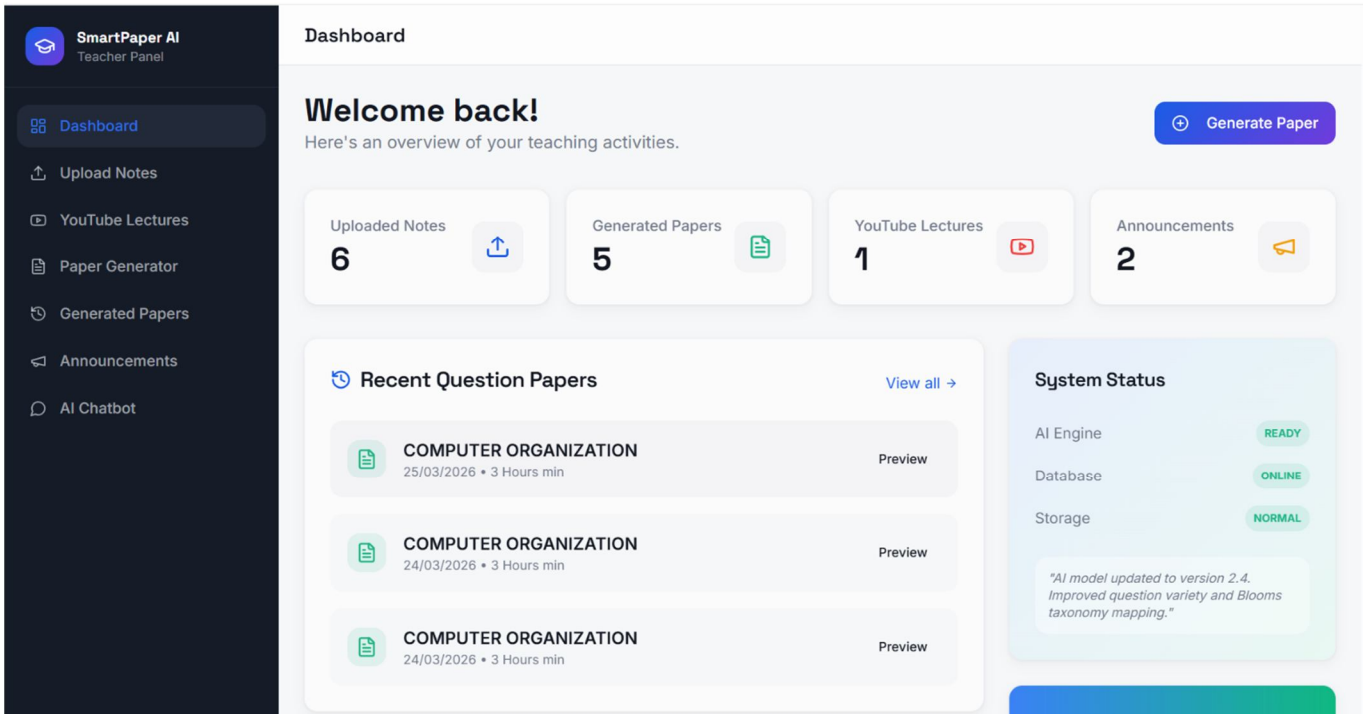


Figure 5.1: Home Page

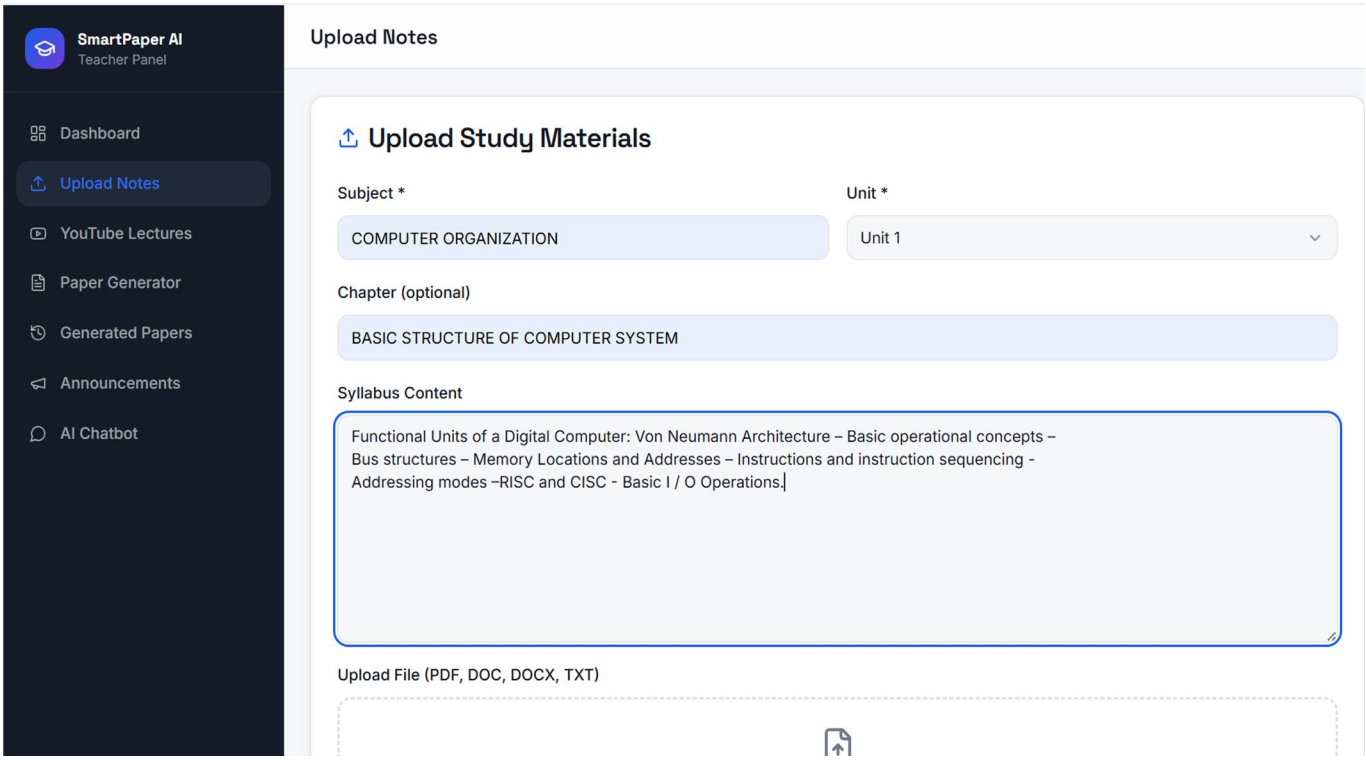


Figure 5.2: Upload Notes

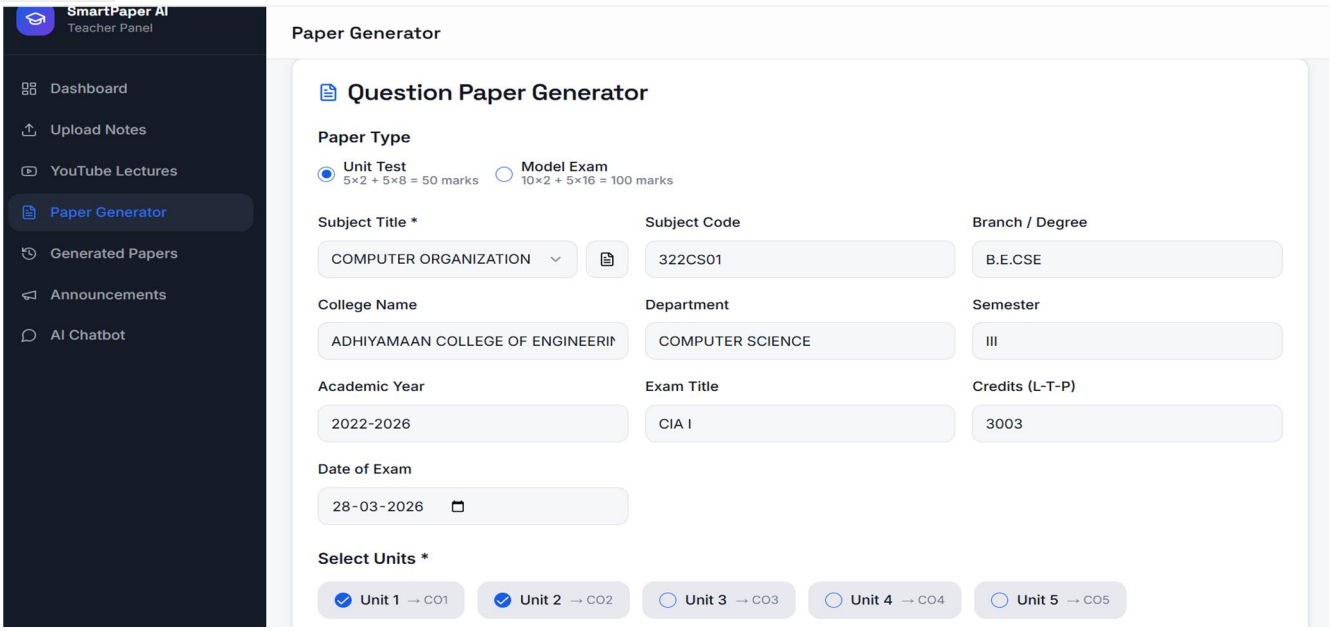


Figure 5.3: Question Paper Generation

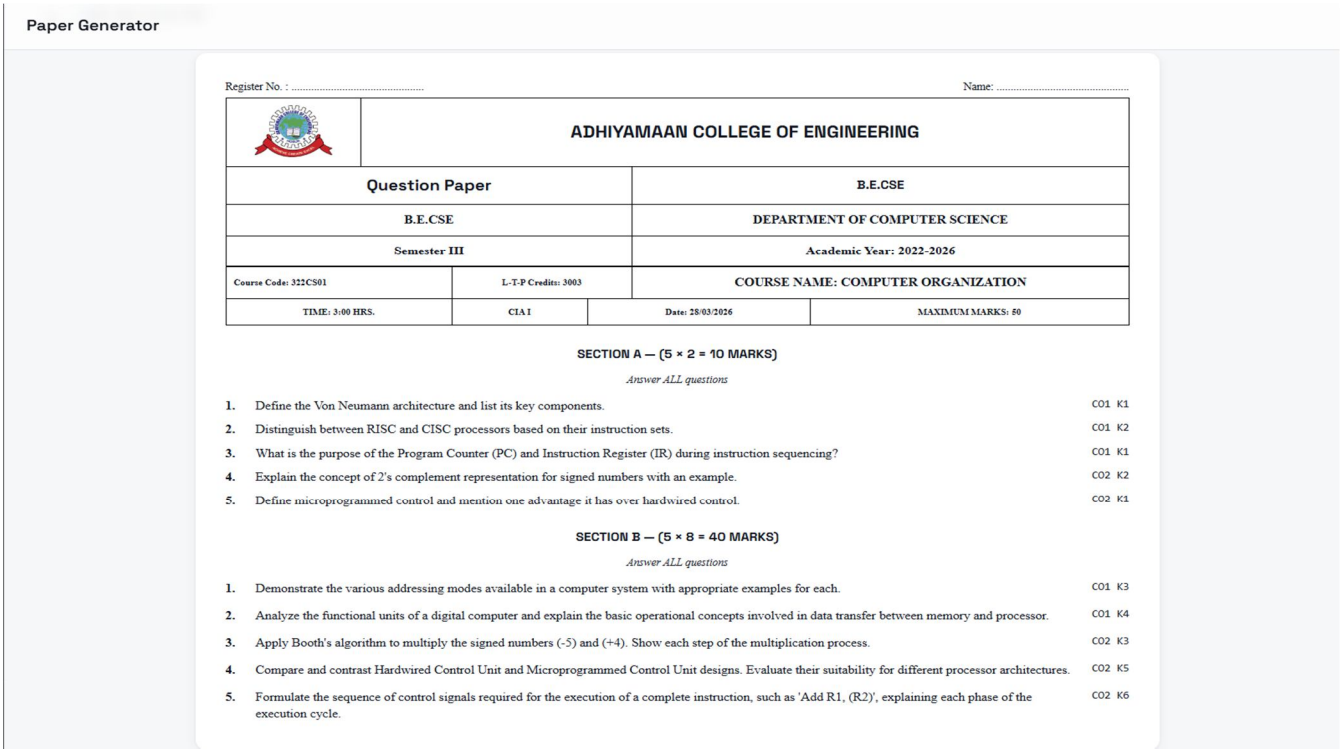


Figure 5.4: Question Paper

VI. CONCLUSION AND FUTURE ENHANCEMENT

A. Conclusion

The AI-Based Question Paper Generation System successfully demonstrates how technology can simplify and automate the traditional process of question paper creation. By integrating intelligent algorithms for question selection, balanced difficulty distribution, and proper syllabus coverage, the system ensures efficient paper generation, reduced manual effort, and improved consistency in assessments. The modular architecture, scalable database, and user-friendly interface enable the system to handle large volumes of data and multiple users while maintaining smooth and reliable performance.

Additionally, features such as customizable inputs, structured formatting, and quick preview and download options enhance usability and efficiency for educators.

From a broader perspective, the system supports digital transformation in the education sector by reducing the workload of teachers and improving the overall quality of examinations. Its data-driven approach, intelligent automation, and efficient question bank management ensure fairness, accuracy, and reliability in the assessment process. The system also provides valuable insights such as topic coverage and question distribution, helping educators make informed decisions while preparing exams.

The platform's emphasis on automation, scalability, and intelligent processing ensures a seamless and efficient experience for users. With the potential for future enhancements such as automatic answer key generation, performance analytics, and multi-language support, the AI-Based Question Paper Generation System stands as a practical, innovative, and sustainable solution for modern educational needs. Overall, it improves efficiency, enhances assessment quality, and provides a reliable tool for educators in academic institutions.

B. Future Scope

Although the AI-Based Question Paper Generation System functions efficiently in its current form, future enhancements can further improve automation, intelligence, scalability, and overall user experience for educators.

1) AI-Powered Question Enhancement: Integrating advanced AI models can improve the quality of generated questions by making them more context-aware, diverse, and aligned with learning objectives.

2) Automatic Answer Key Generation: The system can be enhanced to automatically generate answer keys and solutions for the generated question papers, reducing additional workload for educators.

3) Mobile Application Development: Developing dedicated Android and iOS applications will allow teachers to generate, manage, and access question papers anytime and anywhere with ease.

4) Multi-Language Support: Adding support for multiple languages will make the system accessible to a wider audience and enable question paper generation in regional languages.

5) Advanced Analytics for Chefs: Incorporating predictive analytics can provide insights such as topic-wise coverage, frequently used questions, and difficulty distribution, helping educators improve assessment quality.

6) Personalized Question Paper Generation: Future versions can include personalization features where question papers are generated based on student performance, learning levels, and past assessments.

APPENDICES

Appendix A – Tools and Technologies

- Programming Language: JavaScript / TypeScript
- Framework: React, Node.js, Express.js
- Libraries Used: AI APIs / Models, Data Processing Libraries
- Database: PostgreSQL / MySQL / Firebase (for storing question bank and user data)
- Frontend: HTML, CSS, Bootstrap / UI Frameworks
- IDE Used: Visual Studio Code

Appendix B – Dataset Summary

- Total Questions: 500–1000 questions (sample dataset)
- Features Extracted: Question Text, Topic, Difficulty Level, Marks, Question Type
- Preprocessing Steps: Data Cleaning, Categorization, Difficulty Classification, Data Structuring

Appendix C – Evaluation Metrics

Metric	Result
Question Generation Accuracy	90% Difficulty Balance Efficiency
88% System Response Time	~2 seconds

Appendix D – Sample Output

Input:

- Subject: Computer Science
- Topics: Data Structures

- Difficulty Level: Medium
- Marks: 20

Predicted Allocation:

- Generated Section: Short Answer Questions
- Topic Coverage: Arrays, Linked Lists
- Difficulty Level: Balanced (Medium)
- Output Format: Structured question paper with proper marks allocation
- System Suggestion: Include additional questions from underrepresented topics if required.

SOURCE CODE

App.tsx

```
import { lazy, Suspense } from "react";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { BrowserRouter, Route, Routes, Navigate } from "react-router-dom";
import { Toaster as Sonner } from "@/components/ui/sonner";
import { Toaster } from "@/components/ui/toaster";
import { TooltipProvider } from "@/components/ui/tooltip";
import { AuthProvider, useAuth } from "@/hooks/useAuth";

// Lazy load pages
const LandingPage = lazy(() => import("./pages/LandingPage"));
const LoginPage = lazy(() => import("./pages/LoginPage"));
const SignupPage = lazy(() => import("./pages/SignupPage"));
const DashboardHome = lazy(() => import("./pages/DashboardHome"));
const UploadNotesPage = lazy(() => import("./pages/UploadNotesPage"));
const YoutubeLecturesPage = lazy(() => import("./pages/YoutubeLecturesPage"));
const GeneratePaperPage = lazy(() => import("./pages/GeneratePaperPage"));
const PapersHistoryPage = lazy(() => import("./pages/PapersHistoryPage"));
const AnnouncementsPage = lazy(() => import("./pages/AnnouncementsPage"));
const ChatbotPage = lazy(() => import("./pages/ChatbotPage"));
const StudentMaterialsPage = lazy(() => import("./pages/StudentMaterialsPage"));
const DashboardLayout = lazy(() => import("./components/DashboardLayout"));
const NotFound = lazy(() => import("./pages/NotFound"));

// Loading component
const PageLoader = () => (
  <div className="min-h-[60vh] flex items-center justify-center">
    <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-primary" />
  </div>
);

const queryClient = new QueryClient();

function ProtectedRoute({ children }: { children: React.ReactNode }) {
  const { user, loading } = useAuth();
  if (loading) return <div className="min-h-screen flex items-center justify-center"><div className="animate-spin rounded-full
h-8 w-8 border-b-2 border-primary" /></div>;
  if (!user) return <Navigate to="/login" replace />;
  return <DashboardLayout>{children}</DashboardLayout>;
```

```
}

function PublicRoute({ children }: { children: React.ReactNode }) {
  const { user, loading } = useAuth();
  if (loading) return null;
  if (user) return <Navigate to="/dashboard" replace />;
  return <>{children}</>;
}

const App = () => (
  <QueryClientProvider client={queryClient}>
    <AuthProvider>
      <TooltipProvider>
        <Toaster />
        <Sonner />
        <BrowserRouter>
          <Suspense fallback={<div className="min-h-screen flex items-center justify-center"><div className="animate-spin
rounded-full h-8 w-8 border-b-2 border-primary" /></div>}>
            <Routes>
              <Route path="/" element={<PublicRoute><LandingPage /></PublicRoute>} />
              <Route path="/login" element={<PublicRoute><LoginPage /></PublicRoute>} />
              <Route path="/signup" element={<PublicRoute><SignupPage /></PublicRoute>} />
              <Route path="/dashboard" element={<ProtectedRoute><DashboardHome /></ProtectedRoute>} />
              <Route path="/dashboard/upload-notes" element={<ProtectedRoute><UploadNotesPage /></ProtectedRoute>} />
              <Route path="/dashboard/youtube-lectures" element={<ProtectedRoute><YoutubeLecturesPage /></ProtectedRoute>}
/>
              <Route path="/dashboard/generate-paper" element={<ProtectedRoute><GeneratePaperPage /></ProtectedRoute>} />
              <Route path="/dashboard/papers-history" element={<ProtectedRoute><PapersHistoryPage /></ProtectedRoute>} />
              <Route path="/dashboard/announcements" element={<ProtectedRoute><AnnouncementsPage /></ProtectedRoute>} />
              <Route path="/dashboard/chatbot" element={<ProtectedRoute><ChatbotPage /></ProtectedRoute>} />
              <Route path="/dashboard/materials" element={<ProtectedRoute><StudentMaterialsPage /></ProtectedRoute>} />
              <Route path="/dashboard/lectures" element={<ProtectedRoute><YoutubeLecturesPage /></ProtectedRoute>} />
              <Route path="*" element={<NotFound />} />
            </Routes>
          </Suspense>
        </BrowserRouter>
      </TooltipProvider>
    </AuthProvider>
  </QueryClientProvider>
);

export default App;
```

Dashboard.tsx

```
import { useAuth } from "@/hooks/useAuth";
import { useQuery } from "@tanstack/react-query";
import { apiFetch } from "@/lib/api";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { FileText, Upload, Youtube, Megaphone, BookOpen, PlusCircle, ArrowRight, History, Brain, GraduationCap } from "lucide-
react";
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
import { Link } from "react-router-dom";
import { Button } from "@/components/ui/button";

export default function DashboardHome() {
  const { role, user } = useAuth();

  const { data: notes = [] } = useQuery({
    queryKey: ["all-notes"],
    queryFn: () => apiFetch('/notes'),
  });

  const { data: papers = [] } = useQuery({
    queryKey: ["all-papers"],
    queryFn: () => apiFetch('/papers'),
    enabled: role === "teacher",
  });

  const { data: lectures = [] } = useQuery({
    queryKey: ["all-lectures"],
    queryFn: () => apiFetch('/youtube-lectures'),
  });

  const { data: announcements = [] } = useQuery({
    queryKey: ["all-announcements"],
    queryFn: () => apiFetch('/announcements'),
  });

  const notesCount = notes.length;
  const papersCount = papers.length;
  const lecturesCount = lectures.length;
  const announcementsCount = announcements.length;

  const recentPapers = papers.slice(0, 3);
  const recentNotes = notes.slice(0, 3);
  const recentAnnouncements = announcements.slice(0, 3);

  const teacherStats = [
    { label: "Uploaded Notes", value: notesCount, icon: Upload, color: "text-primary" },
    { label: "Generated Papers", value: papersCount, icon: FileText, color: "text-accent" },
    { label: "YouTube Lectures", value: lecturesCount, icon: Youtube, color: "text-destructive" },
    { label: "Announcements", value: announcementsCount, icon: Megaphone, color: "text-warning" },
  ];

  const studentStats = [
    { label: "Study Materials", value: notesCount, icon: BookOpen, color: "text-primary" },
    { label: "YouTube Lectures", value: lecturesCount, icon: Youtube, color: "text-destructive" },
    { label: "Teacher Announcements", value: announcementsCount, icon: Megaphone, color: "text-warning" },
    { label: "AI Help Sessions", value: 0, icon: Brain, color: "text-accent" },
  ];
```



```
  const stats = role === "teacher" ? teacherStats : studentStats;

  return (
    <div className="space-y-8 pb-10">
      <div className="flex flex-col md:flex-row md:items-center justify-between gap-4">
        <div>
          <h2 className="text-3xl font-display font-bold">Welcome back!</h2>
          <p className="text-muted-foreground">Here&apos;s an overview of your {role === "teacher" ? "teaching" : "learning"}
activities.</p>
        </div>
        <div className="flex gap-2">
          {role === "teacher" ? (
            <Link to="/dashboard/generate-paper">
              <Button className="gradient-primary">
                <PlusCircle className="mr-2 h-4 w-4" /> Generate Paper
              </Button>
            </Link>
          ) : (
            <Link to="/dashboard/materials">
              <Button className="gradient-primary">
                <BookOpen className="mr-2 h-4 w-4" /> Browse Materials
              </Button>
            </Link>
          )}
        </div>
      </div>

      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4">
        {stats.map((stat) => (
          <Card key={stat.label} className="shadow-card hover:shadow-elevated transition-shadow border-none bg-white/50 backdrop-
blur-sm">
            <CardContent className="pt-6">
              <div className="flex items-center justify-between">
                <div>
                  <p className="text-sm text-muted-foreground font-medium">{stat.label}</p>
                  <p className="text-3xl font-display font-bold mt-1">{stat.value ?? "—"}</p>
                </div>
                <div className={`p-3 rounded-xl bg-muted/50`}>
                  <stat.icon className={`h-6 w-6 ${stat.color}`} />
                </div>
              </div>
            </CardContent>
          </Card>
        ))}
      </div>

      <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
        {/* Quick Actions / Recent Activity Area */}
        <div className="lg:col-span-2 space-y-6">
          {role === "teacher" ? (
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
          <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm">
            <CardHeader className="flex flex-row items-center justify-between">
              <CardTitle className="text-xl font-display flex items-center gap-2">
                <History className="h-5 w-5 text-primary" /> Recent Question Papers
              </CardTitle>
              <Link to="/dashboard/papers-history" className="text-sm text-primary hover:underline flex items-center">
                View all <ArrowRight className="ml-1 h-3 w-3" />
              </Link>
            </CardHeader>
            <CardContent>
              <div className="space-y-4">
                {recentPapers && recentPapers.length > 0 ? (
                  recentPapers.map((paper) => (
                    <div key={paper.id} className="flex items-center justify-between p-4 rounded-xl bg-muted/30 hover:bg-muted/50
transition-colors">
                      <div className="flex items-center gap-4">
                        <div className="p-2 rounded-lg bg-accent/10">
                          <FileText className="h-5 w-5 text-accent" />
                        </div>
                        <div>
                          <p className="font-semibold">{paper.subject}</p>
                          <p className="text-xs text-muted-foreground">
                            {new Date(paper.created_at).toLocaleDateString()} • {paper.duration} min
                          </p>
                        </div>
                      </div>
                      <Link to="/dashboard/papers-history">
                        <Button variant="ghost" size="sm" className="text-xs">Preview</Button>
                      </Link>
                    </div>
                  ))
                ) : (
                  <div className="text-center py-8">
                    <p className="text-muted-foreground italic">No papers generated yet.</p>
                    <Link to="/dashboard/generate-paper" className="text-primary hover:underline text-sm font-medium mt-2 inline-
block">
                      Generate your first paper →
                    </Link>
                  </div>
                )}
              </div>
            </CardContent>
          </Card>
        ) : (
          <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm">
            <CardHeader className="flex flex-row items-center justify-between">
              <CardTitle className="text-xl font-display flex items-center gap-2">
                <Megaphone className="h-5 w-5 text-warning" /> Recent Announcements
              </CardTitle>
              <Link to="/dashboard/announcements" className="text-sm text-primary hover:underline flex items-center">
```

```
              View all <ArrowRight className="ml-1 h-3 w-3" />
            </Link>
          </CardHeader>
          <CardContent>
            <div className="space-y-4">
              {recentAnnouncements && recentAnnouncements.length > 0 ? (
                recentAnnouncements.map((announcement) => (
                  <div key={announcement.id} className="flex items-center justify-between p-4 rounded-xl bg-muted/30 hover:bg-
muted/50 transition-colors">
                    <div className="flex items-center gap-4">
                      <div className="p-2 rounded-lg bg-warning/10">
                        <Megaphone className="h-5 w-5 text-warning" />
                      </div>
                      <div className="flex-1">
                        <p className="font-semibold line-clamp-1">{announcement.title}</p>
                        <p className="text-xs text-muted-foreground">
                          {new Date(announcement.created_at).toLocaleDateString()}
                        </p>
                      </div>
                    </div>
                    <Link to="/dashboard/announcements">
                      <Button variant="ghost" size="sm" className="text-xs">Read</Button>
                    </Link>
                  </div>
                ))
              ) : (
                <div className="text-center py-8">
                  <p className="text-muted-foreground italic">No announcements yet.</p>
                </div>
              )}
            </div>
          </CardContent>
        </Card>
      )}

      {/* Quick Actions Bar */}
      <div className="grid grid-cols-1 sm:grid-cols-3 gap-4">
        <Link to={role === "teacher" ? "/dashboard/upload-notes" : "/dashboard/materials"} className="group">
          <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm group-hover:bg-primary group-hover:text-
primary-foreground transition-all duration-300">
            <CardContent className="p-4 flex flex-col items-center text-center gap-2">
              {role === "teacher" ? (
                <>
                  <Upload className="h-6 w-6 text-primary group-hover:text-primary-foreground" />
                  <span className="text-sm font-semibold">Upload Notes</span>
                </>
              ) : (
                <>
                  <BookOpen className="h-6 w-6 text-primary group-hover:text-primary-foreground" />
                  <span className="text-sm font-semibold">Study Notes</span>
```

```
                </>
              )}
            </CardContent>
          </Card>
        </Link>
        <Link to={role === "teacher" ? "/dashboard/announcements" : "/dashboard/lectures"} className="group">
          <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm group-hover:bg-warning group-hover:text-
warning-foreground transition-all duration-300">
            <CardContent className="p-4 flex flex-col items-center text-center gap-2">
              {role === "teacher" ? (
                <>
                  <Megaphone className="h-6 w-6 text-warning group-hover:text-warning-foreground" />
                  <span className="text-sm font-semibold">Post Alert</span>
                </>
              ) : (
                <>
                  <Youtube className="h-6 w-6 text-warning group-hover:text-warning-foreground" />
                  <span className="text-sm font-semibold">Video Lectures</span>
                </>
              )}
            </CardContent>
          </Card>
        </Link>
        <Link to="/dashboard/chatbot" className="group">
          <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm group-hover:bg-accent group-hover:text-accent-
foreground transition-all duration-300">
            <CardContent className="p-4 flex flex-col items-center text-center gap-2">
              <Brain className="h-6 w-6 text-accent group-hover:text-accent-foreground" />
              <span className="text-sm font-semibold">AI Assistant</span>
            </CardContent>
          </Card>
        </Link>
      </div>

      {/* Additional Section for Students: Recent Uploaded Materials */}
      {role === "student" && (
        <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm">
          <CardHeader className="flex flex-row items-center justify-between">
            <CardTitle className="text-xl font-display flex items-center gap-2">
              <History className="h-5 w-5 text-accent" /> Recent Study Materials
            </CardTitle>
            <Link to="/dashboard/materials" className="text-sm text-primary hover:underline flex items-center">
              View all <ArrowRight className="ml-1 h-3 w-3" />
            </Link>
          </CardHeader>
          <CardContent>
            <div className="space-y-4">
              {recentNotes && recentNotes.length > 0 ? (
                recentNotes.map((note) => (
                  <div key={note.id} className="flex items-center justify-between p-4 rounded-xl bg-muted/30 hover:bg-muted/50
```

```
transition-colors">
                <div className="flex items-center gap-4">
                 <div className="p-2 rounded-lg bg-accent/10">
                  <BookOpen className="h-5 w-5 text-accent" />
                 </div>
                 <div>
                  <p className="font-semibold">{note.title}</p>
                  <p className="text-xs text-muted-foreground">{note.subject} • {new
Date(note.created_at).toLocaleDateString()}</p>
                 </div>
                </div>
                <Button
                 variant="ghost"
                 size="sm"
                 className="text-xs"
                 onClick={() => {
                  if (note.file_url) {
                   const url = note.file_url.startsWith('http') ? note.file_url : `https://ai-qp-generation.onrender.com${note.file_url}`;
                   window.open(url, '_blank');
                  }
                 }}
                >
                 Download
                </Button>
               </div>
              ))
             ) : (
              <div className="text-center py-8">
               <p className="text-muted-foreground italic">No materials found.</p>
              </div>
             )}
            </div>
           </CardContent>
          </Card>
         )}
        </div>

        {/* Sidebar Space for more info */}
        <div className="space-y-6">
         <Card className="border-none shadow-card bg-gradient-to-br from-primary/10 to-accent/10 backdrop-blur-sm">
          <CardHeader>
           <CardTitle className="text-lg font-display">System Status</CardTitle>
          </CardHeader>
          <CardContent className="text-sm space-y-4">
           <div className="flex justify-between items-center">
            <span className="text-muted-foreground">AI Engine</span>
            <span className="px-2 py-0.5 rounded-full bg-success/10 text-success text-[10px] font-bold uppercase">Ready</span>
           </div>
           <div className="flex justify-between items-center">
            <span className="text-muted-foreground">Database</span>
```

```
            <span className="px-2 py-0.5 rounded-full bg-success/10 text-success text-[10px] font-bold uppercase">Online</span>
          </div>
          <div className="flex justify-between items-center">
            <span className="text-muted-foreground">Storage</span>
            <span className="px-2 py-0.5 rounded-full bg-success/10 text-success text-[10px] font-bold uppercase">Normal</span>
          </div>
          <div className="pt-2">
            <div className="p-3 rounded-lg bg-white/40 text-xs italic text-muted-foreground">
              &amp;quot;AI model updated to version 2.4. Improved question variety and Blooms taxonomy mapping.&amp;quot;
            </div>
          </div>
        </CardContent>
      </Card>

      <Card className="border-none shadow-card bg-white/50 backdrop-blur-sm overflow-hidden">
        <div className="h-24 bg-gradient-to-r from-blue-500 to-emerald-500 relative flex items-center justify-center">
          <GraduationCap className="h-10 w-10 text-white/50 absolute -right-2 -bottom-2 rotate-12" />
          <p className="text-white font-bold font-display z-10 text-center px-4">
            {role === "teacher" ? "Maximize Your Productivity" : "Ace Your Exams with AI"}
          </p>
        </div>
        <CardContent className="pt-4 text-xs text-muted-foreground leading-relaxed">
          {role === "teacher"
            ? "Use the AI Chatbot to refine your questions or generate lecture outlines from your uploaded notes seamlessly."
            : "Need help understanding a concept? Chat with our AI assistant or browse through the latest study materials."}
        </CardContent>
      </Card>
    </div>
  </div>
 </div>
);
}
```

GeneratePaper.tsx

```
import { useState } from "react";
import { useAuth } from "@/hooks/useAuth";
import { apiFetch } from "@/lib/api";
import { useQuery } from "@tanstack/react-query";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Checkbox } from "@/components/ui/checkbox";
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { toast } from "sonner";
import { FileText, Loader2, Download, Printer, Eye } from "lucide-react";
import QuestionPaperPreview from "@/components/QuestionPaperPreview";

const UNITS = ["Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5"];
const BLOOM_LEVELS = ["K1", "K2", "K3", "K4", "K5", "K6"];
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
interface Question {
  number: number;
  text: string;
  marks: number;
  co: string;
  bloom: string;
  unit: string;
  hasChoice?: boolean;
  choiceText?: string;
}

interface PaperData {
  sectionA: Question[];
  sectionB: Question[];
}

export default function GeneratePaperPage() {
  const { user } = useAuth();
  const [paperType, setPaperType] = useState<"unit" | "model">("unit");
  const [selectedUnits, setSelectedUnits] = useState<string[]>([]);
  const [subject, setSubject] = useState("");
  const [subjectCode, setSubjectCode] = useState("");
  const [collegeName, setCollegeName] = useState("");
  const [department, setDepartment] = useState("");
  const [semester, setSemester] = useState("");
  const [academicYear, setAcademicYear] = useState("");
  const [examTitle, setExamTitle] = useState("");
  const [credits, setCredits] = useState("");
  const [branch, setBranch] = useState("");
  const [examDate, setExamDate] = useState("");
  const [generating, setGenerating] = useState(false);
  const [paperData, setPaperData] = useState<PaperData | null>(null);
  const [answerKey, setAnswerKey] = useState<any>(null);
  const [showPreview, setShowPreview] = useState(false);
  const [generatingAnswers, setGeneratingAnswers] = useState(false);
  const [isManualSubject, setIsManualSubject] = useState(false);

  const { data: notes = [] } = useQuery({
    queryKey: ["teacher-notes", user?.id],
    queryFn: () => apiFetch('/notes'),
    enabled: !!user,
  });

  const uniqueSubjects = Array.from(new Set(notes.map((n: any) => n.subject))).sort();

  const toggleUnit = (unit: string) => {
    setSelectedUnits(prev =>
      prev.includes(unit) ? prev.filter(u => u !== unit) : [...prev, unit]
    );
```


ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
};

const getCoMapping = (unit: string) => {
  const num = unit.replace("Unit ", "");
  return `CO${num}`;
};

const generatePaper = async () => {
  if (selectedUnits.length === 0) {
    toast.error("Please select at least one unit");
    return;
  }
  if (!subject) {
    toast.error("Please enter the subject name");
    return;
  }

  // Get syllabus content from uploaded notes for selected subject and units
  const relevantNotes = notes.filter((n: any) => n.subject === subject && selectedUnits.includes(n.unit));
  if (relevantNotes.length === 0) {
    toast.error(`No notes found for "${subject}" in the selected units. Please upload notes first.`);
    return;
  }

  const syllabusContext = relevantNotes
    .map((n: any) => `[${n.unit}${n.chapter ? ` - ${n.chapter}` : ""}]\n${n.syllabus_content || ""}\n${n.extracted_text || ""}`)
    .join("\n\n");

  setGenerating(true);
  try {
    const sectionACount = paperType === "unit" ? 5 : 10;
    const sectionAMarks = 2;
    const sectionBCount = 5;
    const sectionBMarks = paperType === "unit" ? 8 : 16;
    const totalMarks = paperType === "unit" ? 50 : 100;

    const prompt = `You are an expert exam paper generator. Based on the following syllabus content, generate a ${paperType === "unit" ? "Unit Test" : "Model Exam"} question paper.

SYLLABUS CONTENT:
${syllabusContext}

REQUIREMENTS:
- Subject: ${subject}
- Selected Units: ${selectedUnits.join(", ")}

SECTION A: ${sectionACount} questions × ${sectionAMarks} marks each
- Short answer questions (Remember, Understand level)
- Mix of K1 and K2 levels
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
SECTION B: ${sectionBCount} questions × ${sectionBMarks} marks each
${paperType === "model" ? "- Each question must have TWO options (A and B) for choice" : "- Detailed answer questions"}
- Mix of K3, K4, K5, K6 levels

RULES:
- Questions MUST come from the provided syllabus only
- Each question must have a CO (Course Outcome) mapping based on unit:
${selectedUnits.map(u => ` ${u} → ${getCoMapping(u)}`).join("\n")}
- Each question must have a Bloom's taxonomy level (K1-K6)
- Ensure balanced difficulty
- No repeated questions
- Cover the syllabus comprehensively

Return ONLY valid JSON in this exact format:
{
  "sectionA": [
    {"number": 1, "text": "question text", "marks": ${sectionAMarks}, "co": "CO1", "bloom": "K1", "unit": "Unit 1"}
  ],
  "sectionB": [
    {"number": 1, "text": "question text", "marks": ${sectionBMarks}, "co": "CO1", "bloom": "K3", "unit": "Unit 1"${paperType
=== "model" ? ', "hasChoice": true, "choiceText": "alternative question text"' : ""}}
  ]
}`;

      const { result } = await apiFetch('/generate-questions', {
        method: 'POST',
        body: JSON.stringify({ prompt, type: "generate" })
      });

      setPaperData(result);
      setShowPreview(true);

      // Save to local database
      await apiFetch('/papers', {
        method: 'POST',
        body: JSON.stringify({
          paper_type: paperType,
          subject,
          subject_code: subjectCode || null,
          selected_units: selectedUnits,
          college_name: collegeName || null,
          department: department || null,
          max_marks: totalMarks,
          paper_data: result,
          duration: "3 Hours"
        })
      });

      toast.success("Question paper generated!");
    } catch (err: any) {
```

```
      console.error(err);
      toast.error("Failed to generate paper: " + err.message);
    } finally {
      setGenerating(false);
    }
  };

  const generateAnswerKey = async () => {
    if (!paperData) return;
    setGeneratingAnswers(true);
    try {
      const relevantNotes = notes.filter((n: any) => n.subject === subject && selectedUnits.includes(n.unit));
      const syllabusContext = relevantNotes
        .map((n: any) => `[${n.unit}]\n${n.syllabus_content || ""}\n${n.extracted_text || ""}`)
        .join("\n\n");

      const allQuestions = [
        ...paperData.sectionA.map(q => `Q${q.number} (${q.marks}m): ${q.text}`),
        ...paperData.sectionB.map(q => `Q${q.number} (${q.marks}m): ${q.text}${q.hasChoice ? `\nOR\n${q.choiceText}` : ""}`),
      ].join("\n\n");

      const prompt = `Based on the following syllabus content, provide structured, accurate, exam-oriented answers for all questions.

SYLLABUS:
${syllabusContext}

QUESTIONS:
${allQuestions}

Provide answers in JSON format:
{
  "answers": [
    {"questionNumber": 1, "section": "A", "answer": "detailed answer text"}
  ]
}`;

      const { result } = await apiFetch('/generate-questions', {
        method: 'POST',
        body: JSON.stringify({ prompt, type: "answer" })
      });
      setAnswerKey(result);
      toast.success("Answer key generated!");
    } catch (err: any) {
      toast.error("Failed to generate answer key: " + err.message);
    } finally {
      setGeneratingAnswers(false);
    }
  };

  if (showPreview && paperData) {
```

```
  return (
    <div className="space-y-4">
      <div className="flex gap-2 no-print flex-wrap">
        <Button variant="outline" onClick={() => setShowPreview(false)}>← Back</Button>
        <Button variant="outline" onClick={() => window.print()}><Printer className="h-4 w-4 mr-2" />Print</Button>
        <Button onClick={generateAnswerKey} disabled={generatingAnswers} className="gradient-accent text-accent-
foreground">
          {generatingAnswers ? <><Loader2 className="h-4 w-4 mr-2 animate-spin" />Generating...</> : "Generate Answer Key"}
        </Button>
      </div>
        <QuestionPaperPreview
          paperType={paperType}
          subject={subject}
          subjectCode={subjectCode}
          collegeName={collegeName}
          department={department}
          paperData={paperData}
          answerKey={answerKey}
          semester={semester}
          academicYear={academicYear}
          examTitle={examTitle}
          credits={credits}
          branch={branch}
          examDate={examDate}
        />
    </div>
  );
}

 return (
  <div className="w-full space-y-6">
    <Card className="shadow-card">
      <CardHeader>
        <CardTitle className="flex items-center gap-2 font-display">
          <FileText className="h-5 w-5 text-primary" /> Question Paper Generator
        </CardTitle>
      </CardHeader>
      <CardContent className="space-y-6">
        {/* Paper Type */}
        <div className="space-y-3">
          <Label className="text-base font-semibold">Paper Type</Label>
          <RadioGroup value={paperType} onValueChange={(v: "unit" | "model") => setPaperType(v)} className="flex gap-4">
            <div className="flex items-center space-x-2">
              <RadioGroupItem value="unit" id="unit" />
              <Label htmlFor="unit" className="cursor-pointer">
                <span className="font-medium">Unit Test</span>
                <span className="block text-xs text-muted-foreground">5×2 + 5×8 = 50 marks</span>
              </Label>
            </div>
            <div className="flex items-center space-x-2">
```

```
      <RadioGroupItem value="model" id="model" />
      <Label htmlFor="model" className="cursor-pointer">
        <span className="font-medium">Model Exam</span>
        <span className="block text-xs text-muted-foreground">10×2 + 5×16 = 100 marks</span>
      </Label>
    </div>
  </RadioGroup>
</div>

{/* Paper Details */}
<div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
  <div className="space-y-2">
    <Label>Subject Title *</Label>
    {!isManualSubject && uniqueSubjects.length > 0 ? (
      <div className="flex gap-2">
        <Select value={subject} onValueChange={setSubject}>
          <SelectTrigger className="w-full">
            <SelectValue placeholder="Select a subject" />
          </SelectTrigger>
          <SelectContent>
            {(uniqueSubjects as string[]).map(s => (
              <SelectItem key={s} value={s}>{s}</SelectItem>
            ))}
          </SelectContent>
        </Select>
        <Button
          variant="outline"
          size="icon"
          className="shrink-0"
          onClick={() => {
            setIsManualSubject(true);
            setSubject("");
          }}
          title="Enter manually"
        >
          <FileText className="h-4 w-4" />
        </Button>
      </div>
    ) : (
      <div className="flex gap-2">
        <Input
          value={subject}
          onChange={e => setSubject(e.target.value)}
          placeholder="e.g. Data Structures"
          required
        />
        {uniqueSubjects.length > 0 && (
          <Button
            variant="outline"
            size="icon"
```

```
                  className="shrink-0"
                  onClick={() => setIsManualSubject(false)}
                  title="Select from uploads"
                >
                  <Loader2 className="h-4 w-4" />
                </Button>
              )}
            </div>
          )}
        </div>
        <div className="space-y-2">
          <Label>Subject Code</Label>
          <Input value={subjectCode} onChange={e => setSubjectCode(e.target.value)} placeholder="e.g. CS301" />
        </div>
        <div className="space-y-2">
          <Label>Branch / Degree</Label>
          <Input value={branch} onChange={e => setBranch(e.target.value)} placeholder="e.g. B. E. CSE" />
        </div>
        <div className="space-y-2">
          <Label>College Name</Label>
          <Input value={collegeName} onChange={e => setCollegeName(e.target.value)} placeholder="Your College Full Name"
/>
        </div>
        <div className="space-y-2">
          <Label>Department</Label>
          <Input value={department} onChange={e => setDepartment(e.target.value)} placeholder="e.g. Computer Science" />
        </div>
        <div className="space-y-2">
          <Label>Semester</Label>
          <Input value={semester} onChange={e => setSemester(e.target.value)} placeholder="e.g. 7A" />
        </div>
        <div className="space-y-2">
          <Label>Academic Year</Label>
          <Input value={academicYear} onChange={e => setAcademicYear(e.target.value)} placeholder="e.g. 2025-26" />
        </div>
        <div className="space-y-2">
          <Label>Exam Title</Label>
          <Input value={examTitle} onChange={e => setExamTitle(e.target.value)} placeholder="e.g. CIA II" />
        </div>
        <div className="space-y-2">
          <Label>Credits (L-T-P)</Label>
          <Input value={credits} onChange={e => setCredits(e.target.value)} placeholder="e.g. 3003" />
        </div>
        <div className="space-y-2">
          <Label>Date of Exam</Label>
          <Input type="date" value={examDate} onChange={e => setExamDate(e.target.value)} />
        </div>
      </div>

      {/* Unit Selection */}
```

```
        <div className="space-y-3">
          <Label className="text-base font-semibold">Select Units *</Label>
          <div className="flex flex-wrap gap-3">
            {UNITS.map(unit => (
              <label key={unit} className="flex items-center gap-2 bg-secondary rounded-lg px-4 py-2.5 cursor-pointer hover:bg-
secondary/80 transition-colors">
                <Checkbox checked={selectedUnits.includes(unit)} onCheckedChange={() => toggleUnit(unit)} />
                <span className="text-sm font-medium">{unit}</span>
                <span className="text-xs text-muted-foreground">→ {getCoMapping(unit)}</span>
              </label>
            ))}
          </div>
        </div>

        {/* Bloom's Taxonomy Reference */}
        <div className="rounded-lg bg-muted p-4">
          <p className="text-sm font-semibold mb-2">Bloom's Taxonomy Levels</p>
          <div className="flex flex-wrap gap-2">
            {BLOOM_LEVELS.map((level, i) => (
              <span key={level} className="text-xs bg-card px-2 py-1 rounded font-mono">
                {level}
              </span>
            ))}
          </div>
        </div>

        <Button onClick={generatePaper} disabled={generating} className="w-full gradient-primary text-primary-foreground">
          {generating ? <><Loader2 className="h-4 w-4 mr-2 animate-spin" />Generating Paper...</> : "Generate Question Paper"}
        </Button>
      </CardContent>
    </Card>
  </div>
 );
}
```

Studentmaterials.tsx

```
import { apiFetch } from "@/lib/api";
import { useQuery } from "@tanstack/react-query";
import { Card, CardContent } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { BookOpen, Download, FileText } from "lucide-react";

export default function StudentMaterialsPage() {
 const { data: notes = [], isLoading } = useQuery({
  queryKey: ["all-notes"],
  queryFn: () => apiFetch('/notes'),
 });

 return (
  <div className="space-y-4">
```

```
      <h2 className="text-xl font-display font-bold flex items-center gap-2">
        <BookOpen className="h-5 w-5 text-primary" /> Study Materials
      </h2>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
        {notes.map((note) => (
          <Card key={note.id} className="shadow-card hover:shadow-elevated transition-shadow">
            <CardContent className="pt-4">
              <div className="flex items-start gap-3">
                <div className="w-10 h-10 rounded-lg bg-primary/10 flex items-center justify-center shrink-0">
                  <FileText className="h-5 w-5 text-primary" />
                </div>
                <div className="flex-1 min-w-0">
                  <h3 className="font-semibold text-sm truncate">{note.subject}</h3>
                  <p className="text-xs text-muted-foreground">{note.unit}{note.chapter ? ` • ${note.chapter}` : ""}</p>
                  {note.syllabus_content && (
                    <p className="text-xs text-muted-foreground mt-2 line-clamp-3">{note.syllabus_content}</p>
                  )}
                  {note.file_url && (
                    <a href={note.file_url.startsWith('http') ? note.file_url : `https://ai-qp-generation.onrender.com${note.file_url}`}
target="_blank" rel="noopener noreferrer">
                      <Button variant="outline" size="sm" className="mt-2">
                        <Download className="h-3 w-3 mr-1" /> {note.file_name || "Download"}
                      </Button>
                    </a>
                  )}
                </div>
              </div>
            </CardContent>
          </Card>
        ))}
        {!isLoading && notes.length === 0 && (
          <p className="text-muted-foreground col-span-full text-center py-8">No study materials available yet.</p>
        )}
      </div>
    </div>
  );
}
```

Chatbot.tsx

```
import { useState, useRef, useEffect } from "react";
import { useAuth } from "@/hooks/useAuth";
import { apiFetch } from "@/lib/api";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Card } from "@/components/ui/card";
import { Send, Bot, User, Loader2 } from "lucide-react";
import ReactMarkdown from "react-markdown";

interface Message {
```

```
  role: "user" | "assistant";
  content: string;
}

export default function ChatbotPage() {
  const { user } = useAuth();
  const [messages, setMessages] = useState<Message[]>([]);
  const [input, setInput] = useState("");
  const [loading, setLoading] = useState(false);
  const endRef = useRef<HTMLDivElement>(null);

  useEffect(() => {
    endRef.current?.scrollIntoView({ behavior: "smooth" });
  }, [messages]);

  useEffect(() => {
    const loadChat = async () => {
      try {
        const history = await apiFetch('/chat');
        setMessages(history);
      } catch (err) {
        console.error("Failed to load chat history", err);
      }
    };
    if (user) loadChat();
  }, [user]);

  const sendMessage = async () => {
    if (!input.trim() || loading) return;
    const userMsg: Message = { role: "user", content: input.trim() };
    const newMessages = [...messages, userMsg];
    setMessages(newMessages);
    setInput("");
    setLoading(true);

    try {
      const { result } = await apiFetch("/generate-questions", {
        method: 'POST',
        body: JSON.stringify({
          prompt: input.trim(),
          type: "chat",
          history: newMessages.slice(-10),
        }),
      });

      const assistantMsg: Message = { role: "assistant", content: result || "I couldn't generate a response." };
      setMessages(prev => [...prev, assistantMsg]);

      // Save messages locally
      if (user) {
```

```
      await apiFetch("/chat", {
        method: 'POST',
        body: JSON.stringify({ role: "user", content: userMsg.content }),
      });
      await apiFetch("/chat", {
        method: 'POST',
        body: JSON.stringify({ role: "assistant", content: assistantMsg.content }),
      });
    }
  } catch (err: any) {
    setMessages(prev => [...prev, { role: "assistant", content: "Sorry, an error occurred: " + err.message }]);
  } finally {
    setLoading(false);
  }
};

return (
  <div className="flex flex-col h-[calc(100vh-8rem)]">
    <div className="flex-1 overflow-y-auto space-y-4 pb-4">
      {messages.length === 0 && (
        <div className="text-center py-16">
          <Bot className="h-16 w-16 mx-auto text-primary/30 mb-4" />
          <h3 className="font-display text-xl font-semibold">AI Academic Assistant</h3>
          <p className="text-muted-foreground mt-2 max-w-md mx-auto">
            Ask me anything about your subjects! I can explain concepts, help with doubts, and assist with exam preparation.
          </p>
        </div>
      )}
      {messages.map((msg, i) => (
        <div key={i} className={`flex gap-3 ${msg.role === "user" ? "justify-end" : "justify-start"}`}>
          {msg.role === "assistant" && (
            <div className="w-8 h-8 rounded-full gradient-primary flex items-center justify-center shrink-0">
              <Bot className="h-4 w-4 text-primary-foreground" />
            </div>
          )}
          <Card className={`max-w-[75%] p-3 ${msg.role === "user" ? "gradient-primary text-primary-foreground" : "bg-card"}`}>
            <div className="prose prose-sm max-w-none dark:prose-invert">
              <ReactMarkdown>{msg.content}</ReactMarkdown>
            </div>
          </Card>
          {msg.role === "user" && (
            <div className="w-8 h-8 rounded-full bg-secondary flex items-center justify-center shrink-0">
              <User className="h-4 w-4 text-secondary-foreground" />
            </div>
          )}
        </div>
      ))}
      {loading && (
        <div className="flex gap-3">
          <div className="w-8 h-8 rounded-full gradient-primary flex items-center justify-center shrink-0">
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
            <Bot className="h-4 w-4 text-primary-foreground" />
          </div>
          <Card className="p-3">
            <Loader2 className="h-4 w-4 animate-spin text-primary" />
          </Card>
        </div>
      )}
      <div ref={endRef} />
    </div>

    <div className="flex gap-2 pt-2 border-t">
      <Input
        value={input}
        onChange={e => setInput(e.target.value)}
        onKeyDown={e => e.key === "Enter" && !e.shiftKey && sendMessage()}
        placeholder="Ask a question..."
        disabled={loading}
      />
      <Button onClick={sendMessage} disabled={loading || !input.trim()} className="gradient-primary text-primary-foreground">
        <Send className="h-4 w-4" />
      </Button>
    </div>
  </div>
 );
}
```

UploadNotes.tsx

```
import { useState } from "react";
import { useAuth } from "@/hooks/useAuth";
import { apiFetch } from "@/lib/api";
import { useQueryClient } from "@tanstack/react-query";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Textarea } from "@/components/ui/textarea";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { toast } from "sonner";
import { Upload, FileUp, Loader2, History, FileText, Download, Trash2 } from "lucide-react";
import { useQuery } from "@tanstack/react-query";

const UNITS = ["Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5"];

export default function UploadNotesPage() {
  const { user } = useAuth();
  const queryClient = useQueryClient();
  const [subject, setSubject] = useState("");
  const [unit, setUnit] = useState("");
  const [chapter, setChapter] = useState("");
  const [syllabusContent, setSyllabusContent] = useState("");
```



```
const [file, setFile] = useState<File | null>(null);
const [uploading, setUploading] = useState(false);

const { data: notes = [], isLoading } = useQuery({
 queryKey: ["all-notes"],
 queryFn: () => apiFetch('/notes'),
});

const handleUpload = async (e: React.FormEvent) => {
 e.preventDefault();
 if (!file || !user) {
  toast.error("Please select a file first");
  return;
 }

 setUploading(true);
 try {
  const formData = new FormData();
  formData.append('subject', subject);
  formData.append('unit', unit);
  formData.append('chapter', chapter);
  formData.append('syllabus_content', syllabusContent);
  formData.append('file', file);
  const token = localStorage.getItem("auth_token");
  const res = await fetch("https://ai-qp-generation.onrender.com/api/notes", {
   method: 'POST',
   headers: {
    "Authorization": `Bearer ${token}`
   },
   body: formData,
  });

  if (!res.ok) {
   const errData = await res.json();
   throw new Error(errData.error || "Upload failed");
  }

  await queryClient.invalidateQueries({ queryKey: ["notes"] });
  await queryClient.invalidateQueries({ queryKey: ["all-notes"] });

  toast.success("Notes uploaded successfully!");
  setFile(null);
  setSubject("");
  setUnit("Unit 1");
  setChapter("");
  setSyllabusContent("");
 } catch (err: any) {
  toast.error("Upload failed: " + err.message);
 } finally {
  setUploading(false);
```

```
  }
 };

 const handleDelete = async (id: string, fileName?: string) => {
  if (!confirm(`Are you sure you want to delete ${fileName || 'this file'}? This action cannot be undone.`)) return;

  try {
   await apiFetch(`/notes/${id}`, { method: 'DELETE' });
   toast.success("File deleted successfully");
   await queryClient.invalidateQueries({ queryKey: ["notes"] });
   await queryClient.invalidateQueries({ queryKey: ["all-notes"] });
  } catch (err: any) {
   toast.error("Failed to delete file: " + err.message);
  }
 };

 return (
  <div className="w-full">
   <Card className="shadow-card">
    <CardHeader>
     <CardTitle className="flex items-center gap-2 font-display">
      <Upload className="h-5 w-5 text-primary" /> Upload Study Materials
     </CardTitle>
    </CardHeader>
    <CardContent>
     <form onSubmit={handleUpload} className="space-y-4">
      <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
       <div className="space-y-2">
        <Label htmlFor="subject">Subject *</Label>
        <Input id="subject" value={subject} onChange={e => setSubject(e.target.value)} placeholder="e.g. Data Structures"
required />
       </div>
       <div className="space-y-2">
        <Label htmlFor="unit">Unit *</Label>
        <Select value={unit} onValueChange={setUnit} required>
         <SelectTrigger><SelectValue placeholder="Select unit" /></SelectTrigger>
         <SelectContent>
          {UNITS.map(u => <SelectItem key={u} value={u}>{u}</SelectItem>)}
         </SelectContent>
        </Select>
       </div>
      </div>

      <div className="space-y-2">
       <Label htmlFor="chapter">Chapter (optional)</Label>
       <Input id="chapter" value={chapter} onChange={e => setChapter(e.target.value)} placeholder="e.g. Arrays and Linked
Lists" />
      </div>

      <div className="space-y-2">
```

```
            <Label htmlFor="syllabus">Syllabus Content</Label>
            <Textarea
              id="syllabus"
              value={syllabusContent}
              onChange={e => setSyllabusContent(e.target.value)}
              placeholder="Enter or paste the syllabus content here. The AI will use this to generate questions..."
              className="min-h-[200px]"
            />
          </div>

          <div className="space-y-2">
            <Label htmlFor="file">Upload File (PDF, DOC, DOCX, TXT)</Label>
            <div className="border-2 border-dashed rounded-lg p-6 text-center hover:border-primary/50 transition-colors">
              <FileUp className="h-8 w-8 mx-auto text-muted-foreground mb-2" />
              <Input
                id="file"
                type="file"
                accept=".pdf,.doc,.docx,.txt"
                onChange={e => setFile(e.target.files?.[0] ?? null)}
                className="max-w-xs mx-auto"
              />
              {file && <p className="text-sm text-muted-foreground mt-2">{file.name}</p>}
            </div>
          </div>

          <Button type="submit" className="w-full gradient-primary text-primary-foreground" disabled={uploading || !subject ||
!unit}>
            {uploading ? <><Loader2 className="h-4 w-4 mr-2 animate-spin" /> Uploading...</> : "Upload Notes"}
          </Button>
        </form>
      </CardContent>
    </Card>

    {/* Upload History Section */}
    <h3 className="text-xl font-display font-bold mt-8 mb-4 flex items-center gap-2">
      <History className="h-5 w-5 text-primary" /> Upload History
    </h3>

    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
      {notes.length > 0 ? (
        notes.map((note: any) => (
          <Card key={note.id} className="shadow-card hover:shadow-elevated transition-shadow">
            <CardContent className="pt-4">
              <div className="flex items-start gap-3">
                <div className="w-10 h-10 rounded-lg bg-primary/10 flex items-center justify-center shrink-0">
                  <FileText className="h-5 w-5 text-primary" />
                </div>
                <div className="flex-1 min-w-0">
                  <h3 className="font-semibold text-sm truncate">{note.subject}</h3>
                  <p className="text-xs text-muted-foreground">{note.unit}{note.chapter ? ` • ${note.chapter}` : ""}</p>
```

```
                <p className="text-xs text-muted-foreground mt-1">Uploaded: {new
Date(note.created_at).toLocaleDateString()}</p>

                <div className="flex items-center gap-2 mt-3">
                  {note.file_url && (
                    <a href={note.file_url.startsWith('http') ? note.file_url : `https://ai-qp-generation.onrender.com${note.file_url}`}
target="_blank" rel="noopener noreferrer" className="flex-1">
                      <Button variant="outline" size="sm" className="w-full">
                        <Download className="h-3 w-3 mr-1" /> Download
                      </Button>
                    </a>
                  )}

                  <Button
                    variant="destructive"
                    size="sm"
                    className="px-3"
                    onClick={() => handleDelete(note.id, note.file_name)}
                    title="Delete this file"
                  >
                    <Trash2 className="h-4 w-4" />
                  </Button>
                </div>
              </div>
            </div>
          </CardContent>
        </Card>
      ))
    ) : (
      !isLoading && (
        <p className="text-muted-foreground col-span-full text-center py-8 bg-card rounded-lg border border-dashed">
          No notes uploaded yet. Check back here once you've uploaded your first file!
        </p>
      )
    )}
  </div>

  </div>
 );
}
```

Supabase.db

```
-- Create roles enum
CREATE TYPE public.app_role AS ENUM ('teacher', 'student');

-- Create user_roles table
CREATE TABLE public.user_roles (
 id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
 user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
 role app_role NOT NULL,
```

```
  UNIQUE (user_id, role)
);
ALTER TABLE public.user_roles ENABLE ROW LEVEL SECURITY;

-- Security definer function for role checking
CREATE OR REPLACE FUNCTION public.has_role(_user_id UUID, _role app_role)
RETURNS BOOLEAN
LANGUAGE sql
STABLE
SECURITY DEFINER
SET search_path = public
AS $$
  SELECT EXISTS (
    SELECT 1 FROM public.user_roles WHERE user_id = _user_id AND role = _role
  )
$$;

-- RLS for user_roles
CREATE POLICY "Users can view their own roles" ON public.user_roles FOR SELECT USING (auth.uid() = user_id);
CREATE POLICY "Users can insert their own roles" ON public.user_roles FOR INSERT WITH CHECK (auth.uid() = user_id);

-- Profiles table
CREATE TABLE public.profiles (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL UNIQUE,
  full_name TEXT NOT NULL,
  register_number TEXT,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.profiles ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Profiles viewable by authenticated" ON public.profiles FOR SELECT TO authenticated USING (true);
CREATE POLICY "Users can update own profile" ON public.profiles FOR UPDATE USING (auth.uid() = user_id);
CREATE POLICY "Users can insert own profile" ON public.profiles FOR INSERT WITH CHECK (auth.uid() = user_id);

-- Update timestamp function
CREATE OR REPLACE FUNCTION public.update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN NEW.updated_at = now(); RETURN NEW; END;
$$ LANGUAGE plpgsql SET search_path = public;

CREATE TRIGGER update_profiles_updated_at BEFORE UPDATE ON public.profiles FOR EACH ROW EXECUTE
FUNCTION public.update_updated_at_column();

-- Notes table (teacher uploads)
CREATE TABLE public.notes (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  teacher_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  subject TEXT NOT NULL,
  unit TEXT NOT NULL,
```

```
  chapter TEXT,
  syllabus_content TEXT,
  file_url TEXT,
  file_name TEXT,
  file_type TEXT,
  extracted_text TEXT,
  keywords TEXT[],
  concepts TEXT[],
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.notes ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Teachers can manage own notes" ON public.notes FOR ALL USING (auth.uid() = teacher_id);
CREATE POLICY "Students can view all notes" ON public.notes FOR SELECT TO authenticated USING (true);
CREATE TRIGGER update_notes_updated_at BEFORE UPDATE ON public.notes FOR EACH ROW EXECUTE FUNCTION
public.update_updated_at_column();

-- YouTube lectures table
CREATE TABLE public.youtube_lectures (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  teacher_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  subject TEXT NOT NULL,
  unit TEXT NOT NULL,
  lecture_title TEXT NOT NULL,
  youtube_url TEXT NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.youtube_lectures ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Teachers can manage own lectures" ON public.youtube_lectures FOR ALL USING (auth.uid() = teacher_id);
CREATE POLICY "Students can view all lectures" ON public.youtube_lectures FOR SELECT TO authenticated USING (true);

-- Question papers table
CREATE TABLE public.question_papers (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  teacher_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  paper_type TEXT NOT NULL CHECK (paper_type IN ('unit', 'model')),
  subject TEXT NOT NULL,
  subject_code TEXT,
  selected_units TEXT[] NOT NULL,
  college_name TEXT,
  department TEXT,
  duration TEXT DEFAULT '3 Hours',
  max_marks INTEGER,
  paper_data JSONB NOT NULL,
  answer_key JSONB,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.question_papers ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Teachers can manage own papers" ON public.question_papers FOR ALL USING (auth.uid() = teacher_id);
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

```
-- Announcements table
CREATE TABLE public.announcements (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  teacher_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  title TEXT NOT NULL,
  content TEXT NOT NULL,
  category TEXT DEFAULT 'general',
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.announcements ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Teachers can manage own announcements" ON public.announcements FOR ALL USING (auth.uid() =
teacher_id);
CREATE POLICY "Students can view announcements" ON public.announcements FOR SELECT TO authenticated USING (true);

-- Chat messages table
CREATE TABLE public.chat_messages (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('user', 'assistant')),
  content TEXT NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
ALTER TABLE public.chat_messages ENABLE ROW LEVEL SECURITY;
CREATE POLICY "Users can manage own messages" ON public.chat_messages FOR ALL USING (auth.uid() = user_id);

-- Storage bucket for uploaded notes/files
INSERT INTO storage.buckets (id, name, public) VALUES ('notes-files', 'notes-files', true);
CREATE POLICY "Authenticated users can upload files" ON storage.objects FOR INSERT TO authenticated WITH CHECK
(bucket_id = 'notes-files');
CREATE POLICY "Anyone can view files" ON storage.objects FOR SELECT USING (bucket_id = 'notes-files');
CREATE POLICY "Teachers can delete own files" ON storage.objects FOR DELETE TO authenticated USING (bucket_id =
'notes-files' AND auth.uid()::text = (storage.foldername(name))[1]);

-- Auto-create profile on signup
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO public.profiles (user_id, full_name, register_number)
  VALUES (
    NEW.id,
    COALESCE(NEW.raw_user_meta_data->>'full_name', ''),
    NEW.raw_user_meta_data->>'register_number'
  );
  INSERT INTO public.user_roles (user_id, role)
  VALUES (
    NEW.id,
    (NEW.raw_user_meta_data->>'role')::app_role
  );
  RETURN NEW;
END;
```



```
$$ LANGUAGE plpgsql SECURITY DEFINER SET search_path = public;

CREATE TRIGGER on_auth_user_created
  AFTER INSERT ON auth.users
  FOR EACH ROW EXECUTE FUNCTION public.handle_new_user();
```

Db-migration.js

```
import express from 'express';
import cors from 'cors';
import { fileURLToPath } from 'url';
import { dirname } from 'path';
import jwt from 'jsonwebtoken';
import bcrypt from 'bcryptjs';
import { v4 as uuidv4 } from 'uuid';
import * as dotenv from 'dotenv';
import multer from 'multer';
import { createClient } from '@supabase/supabase-js';

dotenv.config();

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

const app = express();
const PORT = process.env.PORT || 5000;
const JWT_SECRET = process.env.JWT_SECRET || 'edugenius_secret_key_123';

const supabaseUrl = process.env.SUPABASE_URL || '';
const supabaseKey = process.env.SUPABASE_SERVICE_ROLE_KEY || '';
const supabase = createClient(supabaseUrl || 'https://placeholder.supabase.co', supabaseKey || 'placeholder');

// Memory storage for uploading directly to Supabase instead of local disk
const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

app.use(cors());
app.use(express.json({ limit: '50mb' }));
app.use(express.urlencoded({ limit: '50mb', extended: true }));

// Auth Middleware
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ error: 'Unauthorized: No token provided' });

  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ error: 'Forbidden: Invalid token' });
    req.user = user;
    next();
  });
```

```
};

// Auth Routes (Flattened to single 'users' table native to Postgres)
app.post('/api/signup', async (req, res) => {
  const { email, password, fullName, role, registerNumber } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  const userId = uuidv4();

  const { data, error } = await supabase
    .from('users')
    .insert([{
      id: userId,
      email,
      password: hashedPassword,
      full_name: fullName,
      role,
      register_number: registerNumber,
      is_verified: true
    }]);

  if (error) {
    if (error.code === '23505') return res.status(400).json({ error: 'Email already exists' });
    return res.status(500).json({ error: error.message });
  }

  res.json({ message: 'Account created successfully! You can now log in.' });
});

app.post('/api/login', async (req, res) => {
  const { email, password } = req.body;

  const { data: user, error } = await supabase
    .from('users')
    .select('*')
    .eq('email', email)
    .single();

  if (error || !user) return res.status(400).json({ error: 'Invalid credentials' });

  const validPassword = await bcrypt.compare(password, user.password);
  if (!validPassword) return res.status(400).json({ error: 'Invalid credentials' });

  const token = jwt.sign({ id: user.id, email: user.email, role: user.role }, JWT_SECRET);
  res.json({ token, user: { id: user.id, email: user.email, role: user.role, full_name: user.full_name } });
});

app.get('/api/me', authenticateToken, async (req, res) => {
  const { data: user, error } = await supabase
    .from('users')
    .select('id, email, full_name, register_number, role')
```

```
    .eq('id', req.user.id)
    .single();

  if (error || !user) return res.status(404).json({ error: 'User not found' });
  res.json(user);
});

// Announcements API
app.get('/api/announcements', authenticateToken, async (req, res) => {
  const { data, error } = await supabase.from('announcements').select('*').order('created_at', { ascending: false });
  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
});

app.post('/api/announcements', authenticateToken, async (req, res) => {
  const { title, content, category } = req.body;
  if (req.user.role !== 'teacher') return res.sendStatus(403);

  const { data, error } = await supabase
    .from('announcements')
    .insert([{ teacher_id: req.user.id, title, description: content, date: new Date().toISOString().split('T')[0] }])
    .select()
    .single();

  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
});

app.delete('/api/announcements/:id', authenticateToken, async (req, res) => {
  if (req.user.role !== 'teacher') return res.sendStatus(403);
  const { error } = await supabase
    .from('announcements')
    .delete()
    .eq('id', req.params.id)
    .eq('teacher_id', req.user.id);

  if (error) return res.status(500).json({ error: error.message });
  res.json({ message: 'Deleted' });
});

// Question Papers API
app.get('/api/papers', authenticateToken, async (req, res) => {
  const { data, error } = await supabase
    .from('question_papers')
    .select('*')
    .eq('teacher_id', req.user.id)
    .order('created_at', { ascending: false });

  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
```

```
});

app.post('/api/papers', authenticateToken, async (req, res) => {
  const {
    paper_type,
    subject,
    subject_code,
    selected_units,
    college_name,
    department,
    duration,
    max_marks,
    paper_data,
    answer_key,
    academic_year,
    semester
  } = req.body;

  const { data, error } = await supabase
    .from('question_papers')
    .insert([{
      teacher_id: req.user.id,
      paper_type,
      subject,
      subject_code,
      college_name,
      department,
      selected_units,
      duration,
      total_marks: max_marks,
      paper_data,
      answer_key,
      academic_year,
      semester
    }])
    .select()
    .single();

  if (error) {
    console.error('Error saving paper:', error);
    return res.status(500).json({ error: error.message });
  }
  res.json({ id: data.id, message: 'Paper saved' });
});

app.delete('/api/papers/:id', authenticateToken, async (req, res) => {
  const { error } = await supabase
    .from('question_papers')
    .delete()
    .eq('id', req.params.id)
```

```
    .eq('teacher_id', req.user.id);

  if (error) return res.status(500).json({ error: error.message });
  res.json({ message: 'Deleted' });
});

// Notes API
app.get('/api/notes', authenticateToken, async (req, res) => {
  let query = supabase.from('notes').select('*').order('created_at', { ascending: false });
  if (req.user.role === 'teacher') {
    query = query.eq('teacher_id', req.user.id);
  }

  const { data, error } = await query;
  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
});

app.post('/api/notes', authenticateToken, upload.single('file'), async (req, res) => {
  if (req.user.role !== 'teacher') return res.status(403).json({ error: 'Forbidden: Only teachers can upload notes' });

  let { subject, unit, chapter, syllabus_content, keywords, concepts } = req.body;
  if (typeof keywords === 'string') keywords = JSON.parse(keywords || '[]');
  if (typeof concepts === 'string') concepts = JSON.parse(concepts || '[]');

  let file_url = null;
  const file_name = req.file ? req.file.originalname : null;
  const file_type = req.file ? req.file.mimetype : null;

  // Upload to Supabase Storage if file exists
  if (req.file) {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
    const safeFileName = uniqueSuffix + '-' + req.file.originalname.replace(/[^a-zA-Z0-9.-]/g, '_');

    const { data: uploadData, error: uploadError } = await supabase.storage
      .from('uploads')
      .upload(safeFileName, req.file.buffer, {
        contentType: req.file.mimetype
      });

    if (uploadError) {
      return res.status(500).json({ error: 'Failed to upload file to cloud: ' + uploadError.message });
    }

    // Get public URL
    const { data: publicUrlData } = supabase.storage.from('uploads').getPublicUrl(safeFileName);
    file_url = publicUrlData.publicUrl;
  }

  const { data, error } = await supabase
```

```
    .from('notes')
    .insert([{
      teacher_id: req.user.id,
      subject,
      unit,
      chapter,
      syllabus_content,
      file_url,
      file_name,
      file_type,
      keywords,
      concepts
    }])
    .select()
    .single();

  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
});

app.delete('/api/notes/:id', authenticateToken, async (req, res) => {
  if (req.user.role !== 'teacher') return res.status(403).json({ error: 'Forbidden' });

  // First get the note to find the file URL
  const { data: note, error: fetchError } = await supabase
    .from('notes')
    .select('file_url')
    .eq('id', req.params.id)
    .eq('teacher_id', req.user.id)
    .single();

  if (fetchError || !note) return res.status(404).json({ error: 'Note not found' });

  // Delete from storage if URL exists and comes from our bucket
  if (note.file_url && note.file_url.includes('/storage/v1/object/public/uploads/')) {
    const fileName = note.file_url.split('/').pop();
    if (fileName) {
      await supabase.storage.from('uploads').remove([fileName]);
    }
  }

  // Delete DB record
  const { error } = await supabase
    .from('notes')
    .delete()
    .eq('id', req.params.id)
    .eq('teacher_id', req.user.id);

  if (error) return res.status(500).json({ error: error.message });
  res.json({ message: 'Deleted' });
```

```
});

// YouTube Lectures API
app.get('/api/youtube-lectures', authenticateToken, async (req, res) => {
  const { data, error } = await supabase.from('youtube_lectures').select('*').order('created_at', { ascending: false });
  if (error) return res.status(500).json({ error: error.message });
  res.json(data);
});

app.post('/api/youtube-lectures', authenticateToken, async (req, res) => {
  if (req.user.role !== 'teacher') return res.sendStatus(403);
  const { subject, unit, lecture_title, youtube_url } = req.body;

  const { data, error } = await supabase
    .from('youtube_lectures')
    .insert([{ teacher_id: req.user.id, subject, unit, lecture_title, youtube_url }])
    .select()
    .single();

  if (error) return res.status(500).json({ error: error.message });
  res.json({ id: data.id, message: 'Lecture saved' });
});

// AI Generation Logic
app.post('/api/generate-questions', authenticateToken, async (req, res) => {
  const { prompt, type, history } = req.body;
  const GEMINI_API_KEY = process.env.GEMINI_API_KEY;

  if (!GEMINI_API_KEY) {
    console.error('ERROR: GEMINI_API_KEY is missing from environment variables.');
    return res.status(500).json({ error: 'GEMINI_API_KEY is not configured in .env file' });
  }

  console.log(`[AI] Request type: ${type}. Prompt length: ${prompt?.length || 0}`);

  try {
    let systemPrompt = "";
    if (type === "generate") {
      systemPrompt = "You are an expert academic exam paper generator. Generate questions strictly from the provided syllabus
content. Return ONLY valid JSON as specified. No markdown, no code blocks, just pure JSON.";
    } else if (type === "answer") {
      systemPrompt = "You are an expert academic answer key generator. Provide structured, accurate, exam-oriented answers.
Return ONLY valid JSON as specified. No markdown, no code blocks, just pure JSON.";
    } else {
      systemPrompt = "You are a helpful AI academic assistant. You help students with their studies, explain concepts clearly, and
assist teachers with exam preparation. Be concise and accurate.";
    }

    if (!aiResponse.ok) {
      const text = await aiResponse.text();
```

```
      console.error(`[AI] Gemini API Error: ${aiResponse.status} - ${text}`);
      return res.status(aiResponse.status).json({ error: `AI Provider Error: ${text.substring(0, 200)}...` });
    }

    const data = await aiResponse.json();
    let result = data.candidates?.[0]?.content?.parts?.[0]?.text || "";
    console.log(`[AI] Response received. Length: ${result.length}`);

    if (type === "generate" || type === "answer") {
      // Clean JSON markers and find the actual JSON block
      let cleanedResult = result.replace(/```json\n?/g, "").replace(/```\n?/g, "").trim();

      const firstBrace = cleanedResult.indexOf('{');
      const lastBrace = cleanedResult.lastIndexOf('}');

      if (firstBrace !== -1 && lastBrace !== -1 && lastBrace > firstBrace) {
        cleanedResult = cleanedResult.substring(firstBrace, lastBrace + 1);
      }

      try {
        const parsed = JSON.parse(cleanedResult);
        return res.json({ result: parsed });
      } catch (parseError) {
        console.warn(`[AI] Failed to parse JSON result. Result preview: ${cleanedResult.substring(0, 50)}`);
        return res.json({ result });
      }
    }

    res.json({ result });
  } catch (err) {
    console.error(`[AI] Unexpected Server Error:`, err);
    res.status(500).json({ error: 'Internal Server Error: ' + err.message });
  }
});

// Basic chat route removed to align with stateless Supabase AI unless chat history table explicitly required.

// Start Server
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
  if (!supabaseUrl) {
    console.log("WAITING ON SUPABASE CONNECTION: Please set SUPABASE_URL and
SUPABASE_SERVICE_ROLE_KEY within .env");
  } else {
    console.log("Backend connected to Supabase API.");
  }
});
```

VII. ACKNOWLEDGEMENT

It is one of the most efficient tasks in life to choose the appropriate words to express one's gratitude to the beneficiaries. We are very much grateful to God who helped us all the way through the project and how molded us into what we are today.

We are grateful to our beloved Principal Dr. R. RADHAKRISHNAN, M.E., Ph.D., Adhiyamaan College of Engineering (An Autonomous Institution), Hosur for providing the opportunity to do this work in premises.

We acknowledge our heartful gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

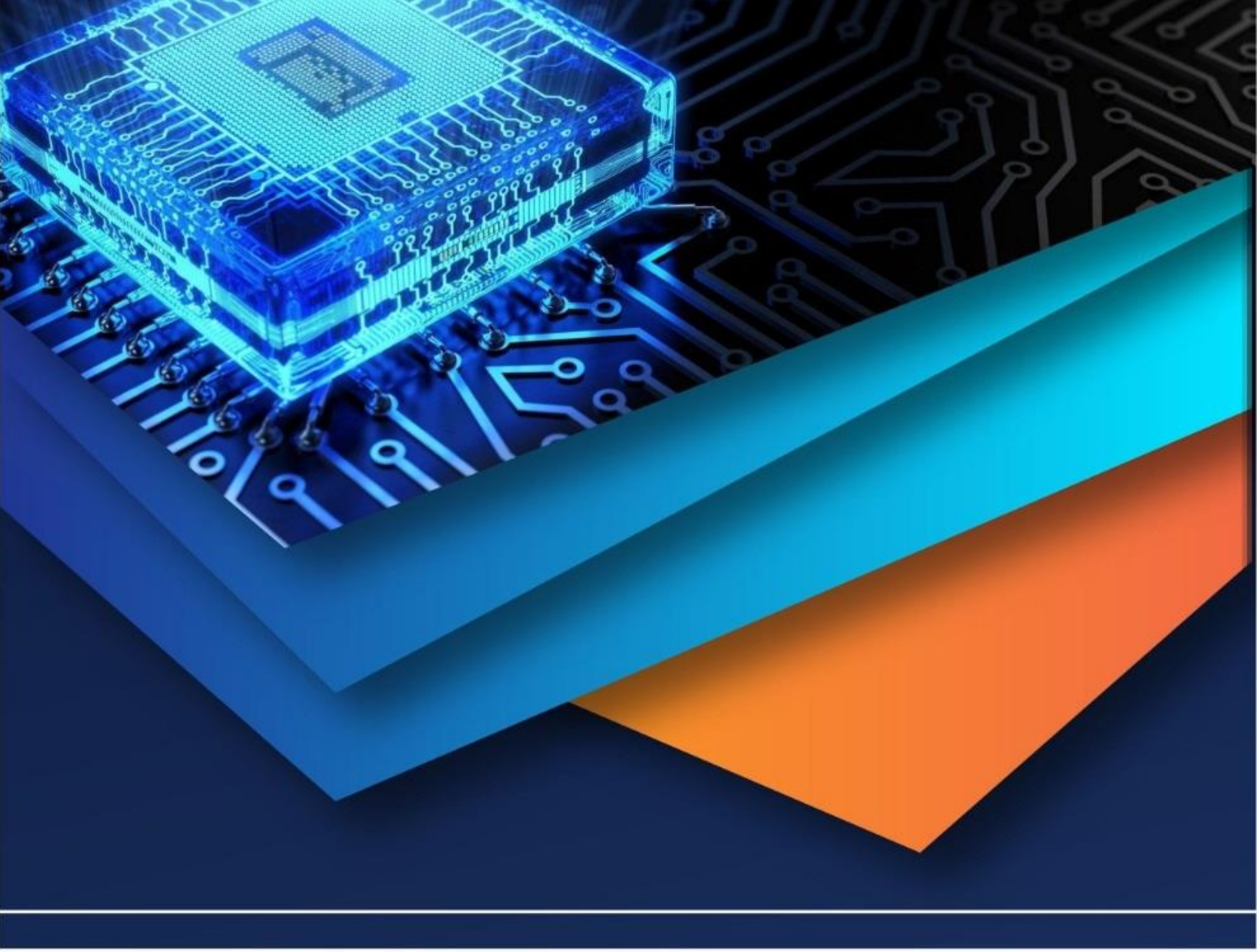
We are highly indebted to Mr. S. VINOTH KUMAR, ME., Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, whose immense support encouragement and valuable guidance were responsible to complete the project successfully.

Also extent our thanks to Project Coordinator and all Staff Members for their support in complete this project successfully.

Finally, we would like to thank to our parents, without their motivational and support would not have been possible for us to complete this project successfully.

REFERENCES

[1] K. Smith, "Automated Question Generation Using Natural Language Processing," Int. J. Artificial Intelligence Research, vol. 4, no. 2, pp. 34–45, 2015.
[2] R. Gupta, "Intelligent Examination System Using Machine Learning," Int. J. Computer Applications, vol. 6, no. 3, pp. 78–89, 2017.
[3] S. Lee, "Web-Based Examination Management System," Int. J. Web Engineering, vol. 8, no. 1, pp. 22–35, 2018.
[4] A. Kumar, "Role of Artificial Intelligence in Education Systems," Int. J. Educational Technology, vol. 9, no. 2, pp. 50–62, 2019.
[5] P. Sharma, "Secure Online Examination Systems Using Authentication Techniques," Int. J. Cyber Security, vol. 10, no. 3, pp. 90–102, 2020.
[6] L. Wang, "Cloud-Based Scalable Architecture for E-Learning Platforms," Int. J. Cloud Computing, vol. 11, no. 1, pp. 60–75, 2020.
[7] D. Patel, "Automated Question Paper Generation Using Rule-Based Systems," Int. J. Software Engineering, vol. 12, no. 2, pp. 110–125, 2021.
[8] M. Singh, "AI-Based Question Generation and Evaluation System," Int. J. Intelligent Systems, vol. 13, no. 3, pp. 140–155, 2021.
[9] N. Verma, "Personalized Learning and Assessment Using AI," Int. J. Advanced Education Systems, vol. 14, no. 2, pp. 200–215, 2022.
[10] T. Joseph, "Integrated Examination System with Question Generation and Management Modules," Int. J. Advanced Computing Systems, vol. 15, no. 1, pp. 85–100, 2023.
[11] J. Brown, "Natural Language Processing for Automatic Question Generation in Educational Systems," Int. J. Computational Linguistics, vol. 16, no. 2, pp. 120–135, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY