AI-BASED AUTOMATED QUESTION PAPER GENERATOR USING LLM

DESIGN THINKING REPORT

Submitted by

DHIVYA DHARSINI S (2303917724422009)

GOPIKA A (2303917724422015)

SUBHA RANI P (2303917724422056)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND BUSINESS SYSTEMS

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI – 15

(A Government Aided Autonomous Institution Affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2025

THIAGARAJAR COLLEGE OF ENGINEERING

MADURAI-15

(A Government Aided Autonomous Institution Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report **"AI- Based Automated Question Paper Generator Using LLM"** is the bonafide work of **"DHIVYA DHARSINI S (2303917724422009), GOPIKA A(2303917724422015), SUBHA RANI P (2303917724422056),"** who carried out the project work under my supervision during the Academic Year 2024–2025.

SIGNATURE

Dr.P.Chitra, M.E., Ph.D.,

FACULTY INCHARGE &

HEAD OF THE DEPARTMENT

COMPUTER SCIENCE AND BUSINESS SYSTEMS

THIAGARAJAR COLLEGE OF ENGG.

MADURAI – 625 015

Submitted for the VIVA VOCE Examination held at Thiagarajar College of Engineering on ………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Acknowledgement

I wish to express my deep sense of gratitude to **Dr.L.Ashok Kumar**, Principal of Thiagarajar College of Engineering for his support and encouragement throughout this project work.

I wish to express my sincere thanks to **Dr.P.Chitra,** Head of the Department of Computer Science and Business Systems for her support and ardent guidance.

I owe my special thanks and gratitude to the faculty incharge **Dr.P.Chitra** Professor, Department of Computer Science and Business Systems for her guidance and support throughout our project.

I am also indebted to all the teaching and non-teaching staff members of our college for helping us directly or indirectly by all means throughout the course of our study and project work.

I extremely thank my parents, family members and friends for their moral support and encouragement for my project.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
LLM	Large Language Model
UI	User Interface
PDF	Portable Document Format
DOCX	Microsoft Word Document Format
MCQ	Multiple Choice Question
CO	Course Outcome
FAISS	Facebook AI Similarity Search
API	Application Programming Interface
IDE	Integrated Development Environment
VS Code	Visual Studio Code
QA	Question Answering
Bloom's Tax.	Bloom's Taxonomy
REST API	Representational State Transfer - Application Programming Interface
LLaMA	Large Language Model Meta AI
UI/UX	User Interface / User Experience
venv	Virtual Environment (Python)
Regex	Regular Expression

Abstract

We’re living in a fast-changing world where technology and education are evolving side by side. As new tools and innovations reshape the academic landscape, the demand for efficient, high-quality assessment methods continues to grow. One critical task that remains a major challenge for educators—particularly in higher education—is the preparation of question papers. Traditionally, this has been a manual process that requires educators to spend hours reviewing course materials, aligning with learning outcomes, and ensuring that questions are balanced in difficulty and coverage. This repetitive and time-consuming task often limits their ability to focus on teaching and student engagement.

This project introduces a modern, AI-powered approach to question paper generation. Educators can upload syllabus, specify course outcomes, and choose preferred exam patterns. The system then intelligently processes the content, understands the core topics, and generates a set of questions categorized by Bloom’s Taxonomy. This ensures that the final paper covers various cognitive levels, from remembering and understanding to applying and analyzing.

The system is designed with user convenience in mind. It offers a clean and intuitive interface where teachers can preview generated papers, make modifications, and download the final version in Word formats. Features such as customizable templates, balanced difficulty levels make the platform a powerful academic tool. Behind the scenes, the platform stores both content and questions as vector embeddings in a FAISS index, enabling instant semantic search and future reuse. In this project, syllabus documents are processed into vector representations using Hugging Face embedding models. These vectors allow the system to retrieve and generate contextually relevant questions using Groq-hosted Large Language Models (LLMs).

Overall, this solution reduces the manual burden on educators, increases the quality and consistency of question papers. It reflects how artificial intelligence can be effectively leveraged to support teaching, not replace it—empowering educators with more time, flexibility, and confidence in their assessments.

1. INTRODUCTION

1.1 BACKGROUND

An amazing transformation for educational practices was brought by the digital transformation. It affected the most crucial areas concerned with assessments-the core measure for student learning and progress, where generally, preparing question papers is a manual process and takes much time. It requires a lot of investment by the teachers in time and effort and depends mostly on their domain expertise.

However, the time has now come with the generation of artificial intelligence (AI) to automate and streamline things. These AI systems understand, interpret, and generate human-like text; thus, they have a new way of performing assessments in the academic world. This project will mainly focus on utilizing these technologies for the development of an automated question paper generator which is efficient, high, and scalable.

1.1.1 CURRENT EDUCATIONAL ASSESSMENT LANDSCAPE

Manually creating question papers presents several challenges. The process is time-consuming, requiring teachers to spend significant hours developing and reviewing questions. There's often inconsistency in the quality, style, and difficulty of papers, especially when multiple educators are involved. Generating a wide range of questions that assess different cognitive levels can be difficult, leading to limited question diversity. Human errors—such as grammatical issues or factual mistakes—can also occur. On top of that, managing the secure storage, duplication, and version control of papers adds a considerable administrative burden.

1.2 PROBLEM STATEMENT

Traditional methods of question paper generation are often inefficient, time-consuming, and susceptible to human error, leading to inconsistencies in quality and difficulty. There is a growing need for a system that can automate this process while maintaining academic rigor and consistency. This system will automatically generate diverse question types, categorize them based on topic and difficulty, and export well-structured question papers along with their answer keys in a professional format.

1.2.1 CORE CHALLENGES

The development of an AI-driven question paper generator presents several core challenges. One of the primary difficulties lies in achieving semantic understanding of complex educational content across various subjects and formats. The system must also ensure context-aware generation, maintaining the relevance, coherence, and accuracy of the generated questions. Another key challenge is the ability to produce a variety of question types, including multiple-choice questions (MCQs), short and long answer questions, and fill-in-the-blanks. Additionally, user customization plays a critical role, as educators should be able to control the output format, question length, and complexity level based on their specific requirements.

1.3 OBJECTIVES

1.3.1 PRIMARY OBJECTIVES

The main goal of this project is to create a smart system that can automatically generate question papers using Artificial Intelligence. It also ensures that the questions match the academic syllabus and stay relevant to the topic. Another important aspect of the project is to give users more control over the quality and format of the questions. The system will aim to create well-structured papers with a mix of difficulty levels, follow recognized academic standards like Bloom's Taxonomy, and include answer keys to help with easy and accurate evaluation.

1.3.2 SECONDARY OBJECTIVES

The secondary objectives focus on making the system faster and easier to use. It's designed to generate question papers quickly and efficiently. To make the experience more user-friendly, the system will have a simple and accessible interface that's easy to navigate. Users will be able to export their question papers in formats like Word, choose from a set of ready-made templates, and even rearrange questions to suit their specific needs or exam patterns. Provide a semantic-search layer using vector embeddings so educators can quickly retrieve and remix previously generated questions.

1.4 PROJECT SCOPE

1.4.1 TECHNICAL SCOPE

This project will bring together the power of Artificial Intelligence (AI) to understand and analyze educational content, making it possible to automatically generate relevant questions. It will be able to process PDFs and other study materials to extract useful content. The system will include both a user-friendly frontend and a functional backend, allowing users to easily customize the questions and export them in the formats they need. Core components include a Groq-hosted LLM for generation, Hugging Face Transformers for embeddings, a FAISS vector store, and a Flask/FastAPI back-end with Streamlit front-end.

1.4.2 FUNCTIONAL SCOPE

The system is designed to handle a variety of content inputs like textbooks, notes, and syllabi, making it flexible for different types of study materials. It will allow teachers or students to generate questions based on specific chapters, units, or topics, ensuring that the questions are targeted and relevant. Once the questions are generated, users can export them in standard academic formats and even customize the type of questions—such as 1-mark, 2-mark, or essay-style questions. To make things easier, the system will have administrative tools for managing user roles and organizing the data effectively.

2. LITERATURE SURVEY

2.1 EVOLUTION OF QUESTION PAPER GENERATION

The process of question paper preparation has evolved significantly over the past few decades. Traditionally, teachers created papers manually, crafting questions based on their understanding of the syllabus and classroom priorities. While this allowed for flexibility and personal input, it was often time-consuming and inconsistent — especially when preparing assessments for multiple subjects or sections.

To reduce the manual burden, institutions later adopted question banks — repositories of pre-written questions organized by topic. This offered some efficiency but came with challenges: many banks became outdated over time, and often lacked alignment with newly updated syllabi. The resulting papers were frequently repetitive and failed to address varying difficulty levels.

Simple software tools emerged in the next wave, automating the selection of questions from these databases. While faster, these tools lacked the intelligence to understand context, assess question relevance, or ensure cognitive balance. Most relied on fixed templates or keyword tagging, making them more mechanical than meaningful in design.

With the rise of Artificial Intelligence, however, the landscape has begun to shift. Modern AI systems — particularly those using large language models (LLMs) — can analyze syllabus content, understand topic relationships, and generate well-structured, contextually relevant questions. This new direction holds the potential to significantly improve both the quality and efficiency of question paper generation.

2.2 EXISTING SYSTEMS AND THEIR LIMITATIONS

Several educational platforms offer automated question paper generation or assessment creation tools. While these solutions have simplified exam preparation, they still present significant gaps when compared to the needs of dynamic, syllabus-driven generation.

The following table benchmarks some popular solutions against the features proposed in this project:

Module Description	**State-of-the-Art Solution Available**	**Drawbacks of the Solution**	**Features Proposed to Overcome**
Question Paper.ai	- AI-generated question papers - PDF/DOC export	- Limited syllabus or course outcome (CO) mapping - No fine-grained cognitive level tagging - Limited customization options	- Dynamic generation from syllabus uploads - Bloom's Taxonomy-based difficulty control - Editable and customizable papers
ClassMarker	- Auto-grading for MCQs - Time-based online tests - Question banks	- Focus mainly on online MCQs - No syllabus/content mapping - No printable exam papers for offline settings	- Printable, structured exam papers - Support for both objective and descriptive types
Teachmint	- Online exam builder - Subject tagging for questions	- Static question pools - Limited higher-order thinking questions - Minimal syllabus interpretation	- Semantic embedding and retrieval for syllabus-aligned generation
MeritHub	- Custom quiz creation - Exam scheduling	- Limited question variety - Focused on quizzes, not full exams	- Balanced full-length exams with cognitive diversity (Remember, Understand, Apply)

Table 2.1 Existing Systems and Their Limitations

2.3 ROLE OF AI IN EDUCATION

Artificial Intelligence (AI) has made a noticeable impact in the education sector over the last decade. From adaptive learning platforms to automated grading tools, AI is helping both students and educators work more efficiently. One of the most promising applications is in the automation of academic content generation — including quizzes, summaries, assignments, and question papers.

For educators, AI reduces the time spent on repetitive tasks like question preparation, freeing up time for teaching and student engagement. It also introduces consistency in assessment design by ensuring a balanced mix of question types and cognitive difficulty levels.

Recent advancements in large language models (LLMs) have significantly expanded the potential of AI in this space. These models are capable of reading and understanding academic content, interpreting learning objectives, and generating questions that are contextually appropriate and linguistically accurate.

In our project, we build on this capability by using AI not just to select questions but to generate them dynamically from syllabus documents. This allows for truly customized, syllabus-aligned question papers with minimal manual input.

2.4 EMBEDDING-BASED SEMANTIC PROCESSING IN ASSESSMENT

Rather than using traditional Natural Language Processing (NLP) techniques, this project relies on embedding-based semantic processing to understand and utilize academic content.

The process involves:

- Extracting clean text from PDFs and study materials.
- Converting sections of content into high-dimensional embeddings using Hugging Face models.
- Storing embeddings in a FAISS vector database for efficient semantic retrieval.
- Using retrieved content as contextual input for Groq-hosted LLMs to dynamically generate questions.

This approach allows the system to interpret topics at a deeper level, match question generation with academic objectives, and produce assessments that are relevant, grammatically accurate, and aligned with Bloom’s Taxonomy.

3. PROBLEM DESCRIPTION

3.1 CURRENT CHALLENGES IN ASSESSMENT CREATION

Despite advancements in educational technology, assessment creation remains a significant challenge for educators. Manual preparation of question papers is a time-consuming and repetitive task, requiring careful attention to syllabus coverage, question diversity, and cognitive level balance. Key challenges include:

- Ensuring alignment with updated curriculum and learning outcomes.
- Balancing questions across cognitive levels (e.g., Remember, Understand, Apply) according to Bloom's Taxonomy.
- Preventing redundancy and repetition in exams across batches and academic years.

These challenges often result in increased workload for educators, inconsistencies in exam difficulty, and reduced time available for student-focused activities.

3.2 STAKEHOLDER ANALYSIS

3.2.1 PRIMARY STAKEHOLDERS

Stakeholder	**Role**	**Needs/Expectations**
Faculty Members (Teachers, Professors)	Direct users who prepare and customize question papers.	Require fast, accurate, customizable, and syllabus-aligned question generation.
Students	Indirect users who take assessments generated by the system.	Need balanced, syllabus-relevant, and fair exams that evaluate a range of cognitive skills.

Table 3.1 Primary Stakeholders: Roles and Needs

3.2.2 SECONDARY STAKEHOLDERS

Stakeholder	Role	Needs/Expectations
Academic Coordinators / HODs	Supervisors who review, validate, and approve assessments.	Require high-quality, curriculum-compliant, and error-free question papers.
Institutional Administrators	Policy makers responsible for academic quality and audits.	Need consistent documentation, standardization of assessments, and ease of academic reporting.

Table 3.2 Secondary Stakeholders: Roles and Needs

3.3 SYSTEM REQUIREMENTS

The proposed system must satisfy the following functional and non-functional requirements:

3.3.1 FUNCTIONAL REQUIREMENTS

- Upload and parse syllabus documents (PDFs).
- Allow section-wise or topic-wise content segmentation.
- Embed content using semantic models for retrieval.
- Generate questions dynamically using Groq-hosted LLMs.
- Categorize questions by cognitive level (Remember, Understand, Apply).
- Provide editing, reordering, and formatting options before final export.
- Support export to standard formats (DOCX).

3.3.2 NON-FUNCTIONAL REQUIREMENTS

- **Scalability**: The system should efficiently handle multiple users and document uploads without slowing down.
- **Usability**: A simple, easy-to-use interface should be provided for educators.
- **Performance**: Question papers should be generated within 15 minutes.

- **Security**: Study materials and generated papers must be stored safely and securely.
- **Accuracy**: Generated questions must be grammatically correct, syllabus-aligned, and match Bloom’s cognitive levels.

3.4 TECHNICAL FEASIBILITY

The technical feasibility of the project is strongly supported by the availability of modern AI and embedding technologies:

- **Content Extraction**: Tools like PyMuPDF and pdfplumber allow accurate extraction and cleaning of study material from PDFs.
- **Semantic Understanding**: Hugging Face transformer models enable the creation of vector embeddings that capture the meaning of academic content.
- **Fast Retrieval**: FAISS provides an efficient vector database for fast semantic search and content retrieval.
- **Dynamic Generation**: Groq-hosted LLMs (LLaMA-3 models) accessed via LangChain enable the generation of high-quality, contextually appropriate questions.

4. PROPOSED APPROACH

4.1 SYSTEM ARCHITECTURE

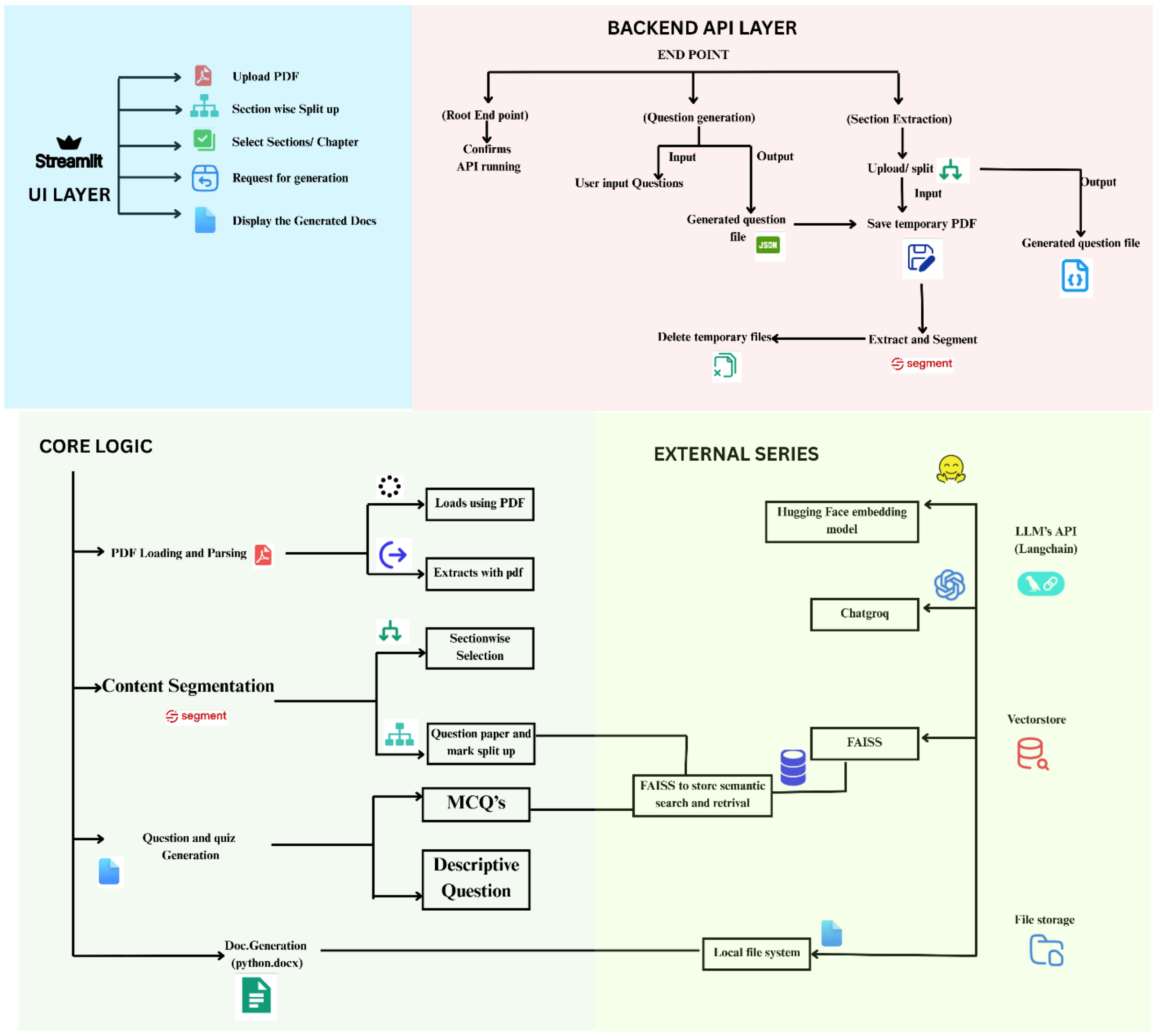


Figure 4.1 System Architecture

4.2 TECHNOLOGY STACK

COMPONENT	**TECHNOLOGY USED**
Language Model	GPT/Groq (LLM)
Embedding Model	Hugging Face, Transformers
Vector Search	FAISS Vector database
Text Extraction	Python (PyMuPDF, pdfplumber)
Data Processing	Python (Pandas, Regex)
Backend	Flask / FastAPI
Frontend	Streamlit

Table 4.1 Technology Stack

Language Model – GPT/Groq (LLM):

- The core engine for question generation is a large language model accessed through Groq's API.
- These LLMs can understand contextual information and generating Bloom’s taxonomy-aligned questions based on the provided textual content.
- Groq offers high-speed inference and cost-efficient querying for large-scale tasks.

Embedding Model – Hugging Face Transformers:

- HuggingFace Transformers are used to convert both the content and the generated questions into numerical embeddings (vector representations).
- These embeddings are crucial for storing, searching, and semantically comparing content efficiently.

Vector Search – FAISS:

- FAISS (Facebook AI Similarity Search) is used as the vector store for efficient similarity search and retrieval.
- It enables the system to quickly find related content or questions based on semantic similarity, making it valuable for educational insights or feedback.

Text Extraction – Python (PyMuPDF, pdfplumber):

- For processing raw PDFs, PyMuPDF and pdfplumber are used to extract clean, structured text.
- These libraries support accurate text layout detection, making it easier to identify chapters, sub-sections, or question banks within documents.

Data Processing – Python (Pandas, Regex):

- Python's Pandas library and regular expressions (Regex) are utilized for cleaning, parsing, and transforming the extracted data into a format suitable for model input.
- This step ensures content consistency and supports automated prompt preparation.

Backend – Flask / FastAPI :

- A lightweight backend using Flask or FastAPI can handle API requests, process uploads, communicate with the LLM, and serve results.
- FastAPI is particularly chosen for its speed, simplicity, and native support for asynchronous operations.

- ## 4.3 Data Flow and Processing

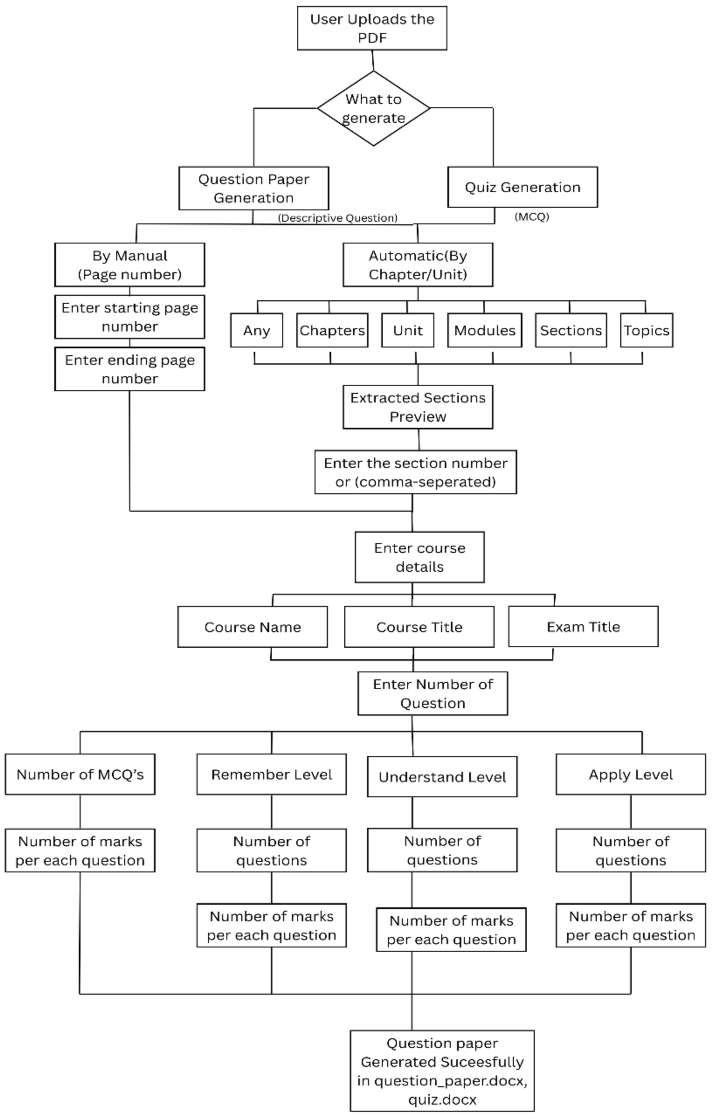


Figure 4.2 Data Flow and Processing

4.4 SECURITY CONSIDERATIONS

1. **Data Privacy**
    - Uploaded content is processed locally or in secure cloud instances (depending on deployment).
    - Sensitive academic material is not stored unnecessarily.

2. **API Key Protection**
    - LLM API keys (e.g., Groq or OpenAI) are stored securely using environment variables.
    - Backend access is restricted and monitored.

3. **Prompt Injection Protection**
    - Input prompts are sanitized to prevent injection attacks or misuse.

4. **Access Control (For Deployment)**
    - Authentication can be implemented to restrict usage to authorized educators or institutions.
5. **Rate Limiting**
    - To prevent abuse of LLM APIs and avoid incurring high costs.

5.IMPLEMENTATION

5.1 DEVELOPMENT ENVIRONMENT

Category	**Details**
Programming Language	Python 3.11
IDE (Integrated Development Environment)	Visual Studio Code (VS Code)
Backend Framework	FastAPI / Flask
Frontend Framework	Streamlit (for quick web interface prototyping)
AI Model Provider	Groq-hosted LLMs (LLaMA 3 series)
Embedding Model	Hugging Face Transformers ("all-MiniLM-L6-v2" model)
Vector Database	FAISS (Facebook AI Similarity Search)
PDF Parsing Libraries	PyMuPDF, pdfplumber
Data Processing Libraries	pandas, re (Regular Expressions)
Document Generation	python-docx library (for DOCX export)
Environment Management	Python Virtual Environment (venv)
API Integration	LangChain library (for Groq and embedding pipeline)

Table 5.1: Development Environment

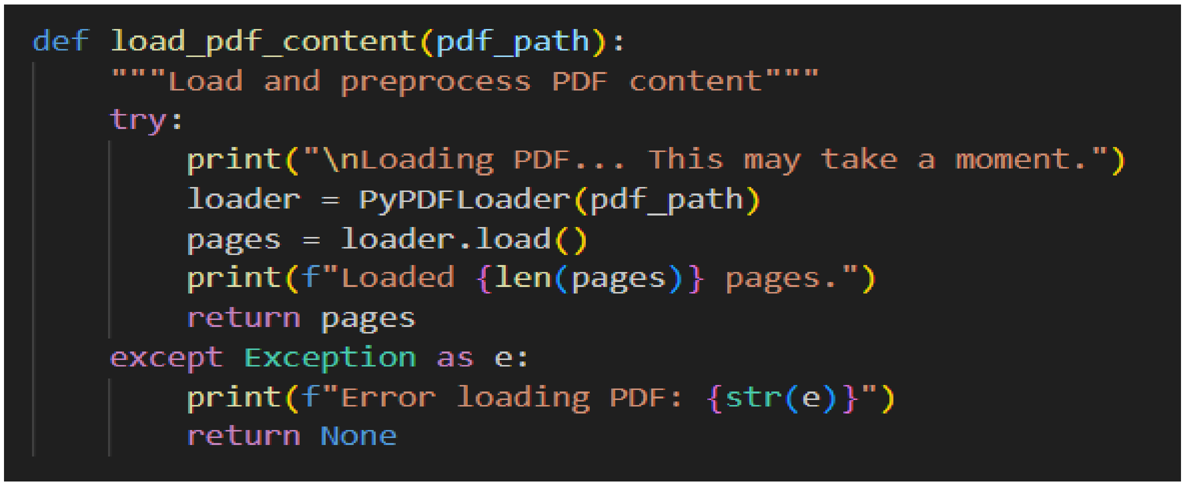
During development, environment variables were securely managed using the dotenv library to store sensitive credentials such as the Groq API key.Libraries were installed and managed using pip, and the codebase was version-controlled locally during the development phase.The system was primarily tested in a local server setup (FastAPI backend + Streamlit frontend) before being prepared for final deployment.

5.2 CORE COMPONENTS

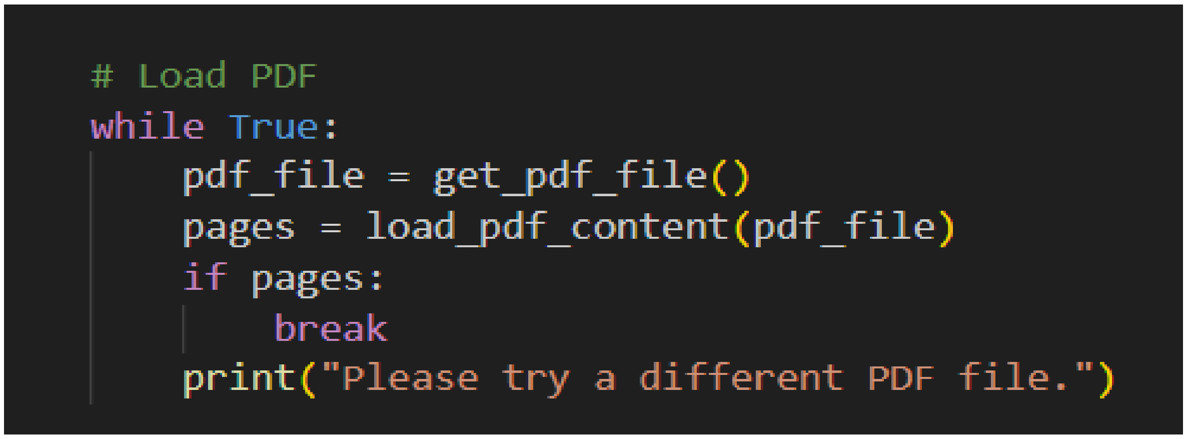
Our system is built using five main parts. Each part has a special job that helps the system understand the syllabus and generate high-quality questions automatically.

1.PDF Extraction:

- Extracts clean text from syllabus documents using PyMuPDF and pdfplumber.



```python
def load_pdf_content(pdf_path):
    """Load and preprocess PDF content"""
    try:
        print("\nLoading PDF... This may take a moment.")
        loader = PyPDFLoader(pdf_path)
        pages = loader.load()
        print(f"Loaded {len(pages)} pages.")
        return pages
    except Exception as e:
        print(f"Error loading PDF: {str(e)}")
        return None
```



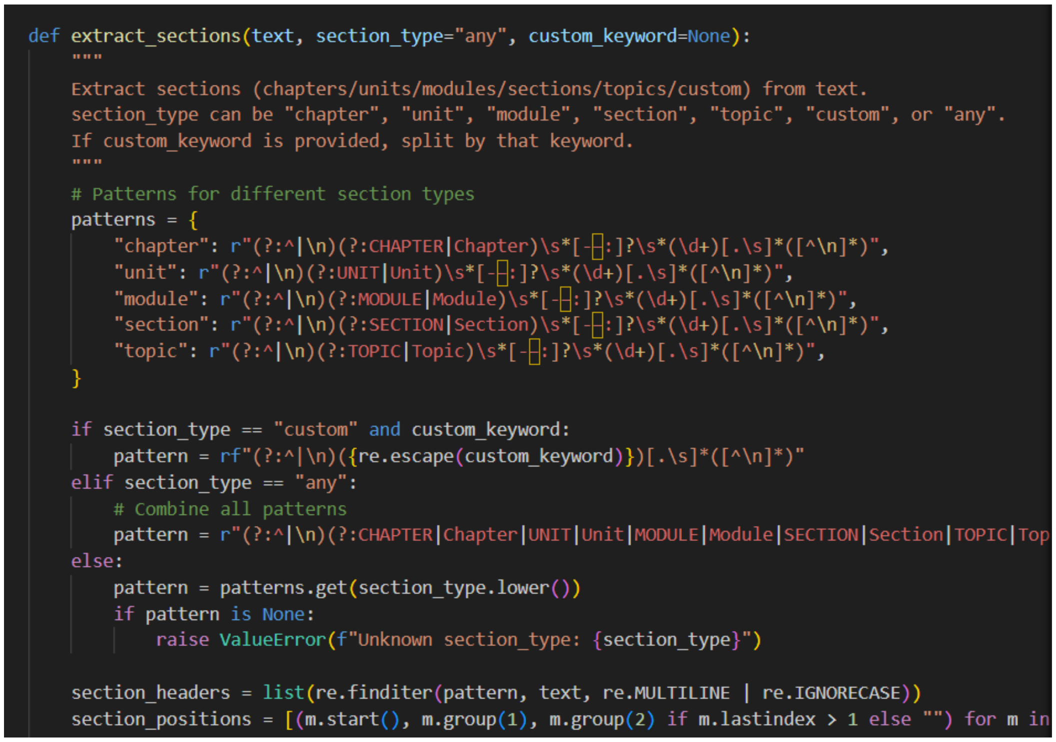
```python
# Load PDF
while True:
    pdf_file = get_pdf_file()
    pages = load_pdf_content(pdf_file)
    if pages:
        break
    print("Please try a different PDF file.")
```

Figure 5.2.1: PDF Extraction — Code for loading PDF content.

Figure 5.2.2: PDF Extraction — Code for loading PDF after file selection.

2.Content Segmentation:

- Splits extracted text into sections (Chapters/Units) using pandas and regular expressions.

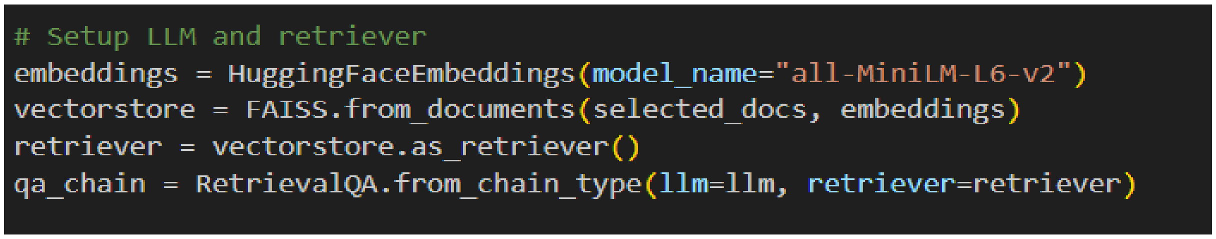


```python
def extract_sections(text, section_type="any", custom_keyword=None):
    """
    Extract sections (chapters/units/modules/sections/topics/custom) from text.
    section_type can be "chapter", "unit", "module", "section", "topic", "custom", or "any".
    If custom_keyword is provided, split by that keyword.
    """
    # Patterns for different section types
    patterns = {
        "chapter": r"(?:^|\n)(?:CHAPTER|Chapter)\s*[-–:]?\s*(\d+)[.\s]*([^\n]*)",
        "unit": r"(?:^|\n)(?:UNIT|Unit)\s*[-–:]?\s*(\d+)[.\s]*([^\n]*)",
        "module": r"(?:^|\n)(?:MODULE|Module)\s*[-–:]?\s*(\d+)[.\s]*([^\n]*)",
        "section": r"(?:^|\n)(?:SECTION|Section)\s*[-–:]?\s*(\d+)[.\s]*([^\n]*)",
        "topic": r"(?:^|\n)(?:TOPIC|Topic)\s*[-–:]?\s*(\d+)[.\s]*([^\n]*)",
    }

    if section_type == "custom" and custom_keyword:
        pattern = rf"(?:^|\n)({re.escape(custom_keyword)})[.\s]*([^\n]*)"
    elif section_type == "any":
        # Combine all patterns
        pattern = r"(?:^|\n)(?:CHAPTER|Chapter|UNIT|Unit|MODULE|Module|SECTION|Section|TOPIC|Top
    else:
        pattern = patterns.get(section_type.lower())
        if pattern is None:
            raise ValueError(f"Unknown section_type: {section_type}")

    section_headers = list(re.finditer(pattern, text, re.MULTILINE | re.IGNORECASE))
    section_positions = [(m.start(), m.group(1), m.group(2) if m.lastindex > 1 else "") for m in
```

Figure 5.2.3: Content Segmentation - Code for extracting sections (Chapter/Unit/Module) from text.

3.Embedding Generation:

- Creates semantic vector embeddings from syllabus content using Hugging Face models ("all-MiniLM-L6-v2").

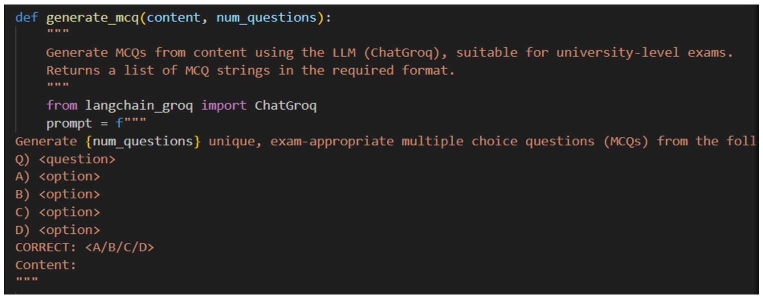


```python
# Setup LLM and retriever
embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
vectorstore = FAISS.from_documents(selected_docs, embeddings)
retriever = vectorstore.as_retriever()
qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)
```

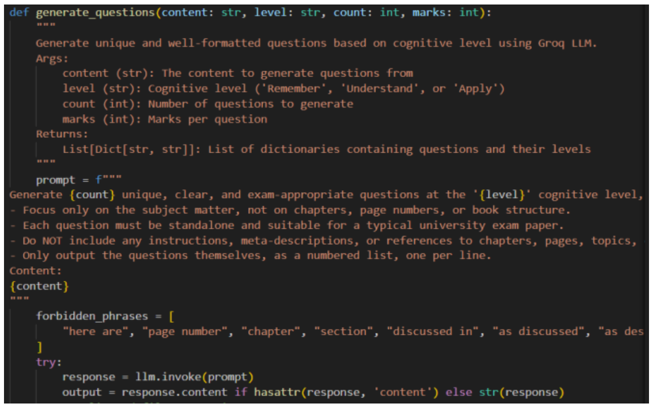
Figure 5.2.4: Embedding Generation — Code for setting up LLM and retriever using Hugging Face and FAISS.

4.Question Generation:

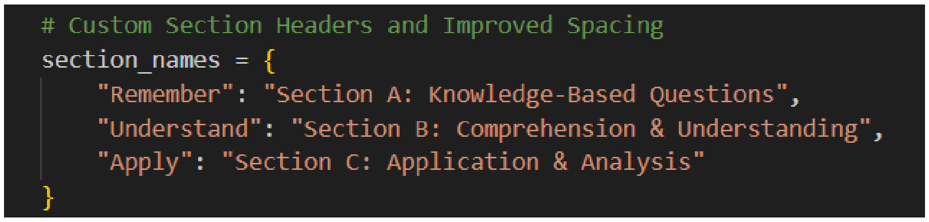
- Queries Groq-hosted LLMs through LangChain to generate contextually appropriate questions across Bloom's cognitive levels.



```python
def generate_mcq(content, num_questions):
    """
    Generate MCQs from content using the LLM (ChatGroq), suitable for university-level exams.
    Returns a list of MCQ strings in the required format.
    """
    from langchain_groq import ChatGroq
    prompt = f"""
Generate {num_questions} unique, exam-appropriate multiple choice questions (MCQs) from the foll
Q) <question>
A) <option>
B) <option>
C) <option>
D) <option>
CORRECT: <A/B/C/D>
Content:
"""
```



```python
def generate_questions(content: str, level: str, count: int, marks: int):
    """
    Generate unique and well-formatted questions based on cognitive level using Groq LLM.
    Args:
        content (str): The content to generate questions from
        level (str): Cognitive level ('Remember', 'Understand', or 'Apply')
        count (int): Number of questions to generate
        marks (int): Marks per question
    Returns:
        List[Dict[str, str]]: List of dictionaries containing questions and their levels
    """
    prompt = f"""
Generate {count} unique, clear, and exam-appropriate questions at the '{level}' cognitive level,
- Focus only on the subject matter, not on chapters, page numbers, or book structure.
- Each question must be standalone and suitable for a typical university exam paper.
- Do NOT include any instructions, meta-descriptions, or references to chapters, pages, topics,
- Only output the questions themselves, as a numbered list, one per line.
Content:
{content}
"""
    forbidden_phrases = [
        "here are", "page number", "chapter", "section", "discussed in", "as discussed", "as des
    ]
    try:
        response = llm.invoke(prompt)
        output = response.content if hasattr(response, 'content') else str(response)
```



```python
# Custom Section Headers and Improved Spacing
section_names = {
    "Remember": "Section A: Knowledge-Based Questions",
    "Understand": "Section B: Comprehension & Understanding",
    "Apply": "Section C: Application & Analysis"
}
```

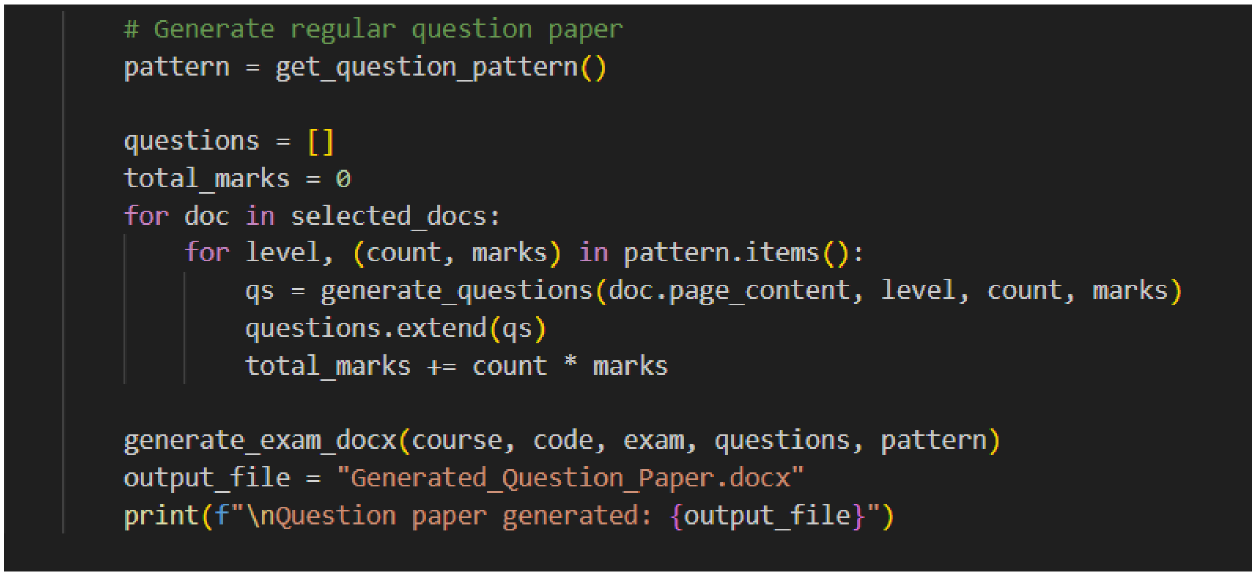
Figure 5.2.5: MCQ Generation using Groq LLM

Figure 5.2.6: Cognitive-Level Based Question Generation

Figure 5.2.7: Section Headers Based on Cognitive Levels

5.Document Assembly:

- Organizes generated questions and exports structured question papers or quizzes into DOCX format.



```
    # Generate regular question paper
    pattern = get_question_pattern()

    questions = []
    total_marks = 0
    for doc in selected_docs:
        for level, (count, marks) in pattern.items():
            qs = generate_questions(doc.page_content, level, count, marks)
            questions.extend(qs)
            total_marks += count * marks

    generate_exam_docx(course, code, exam, questions, pattern)
    output_file = "Generated_Question_Paper.docx"
    print(f"\nQuestion paper generated: {output_file}")
```

Figure 5.2.8: Document Assembly Code for Generating Question Paper

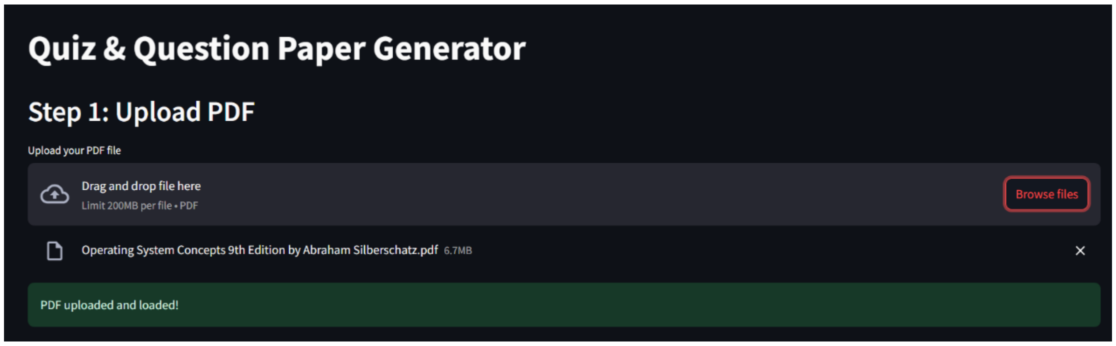
5.3 INTEGRATION AND TESTING

5.3.1 INTEGRATION

The system integrates multiple independent modules into a seamless pipeline:

Frontend (Streamlit UI):

- Allows users to upload PDF syllabus files, select sections, specify course details, and choose the type of assessment (Quiz or Question Paper).



Quiz & Question Paper Generator

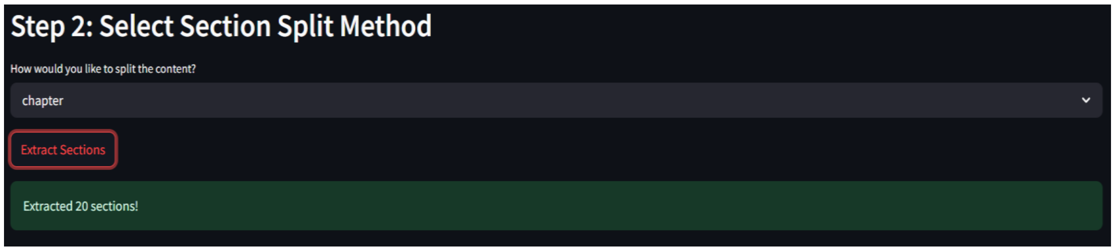
Step 1: Upload PDF

Upload your PDF file

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

Operating System Concepts 9th Edition by Abraham Silberschatz.pdf 6.7MB

PDF uploaded and loaded!

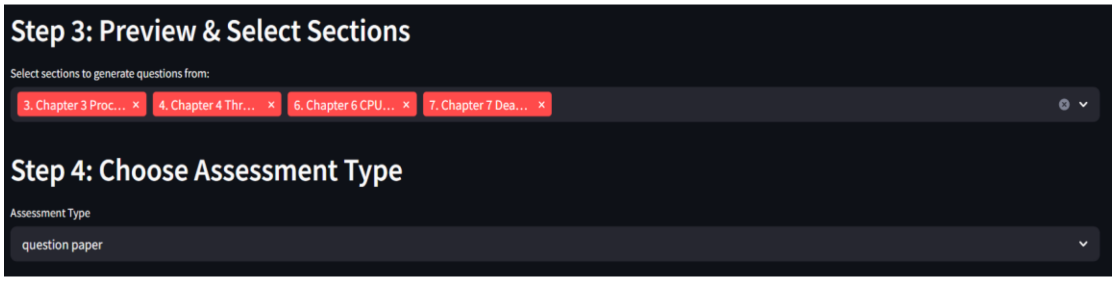


Step 2: Select Section Split Method

How would you like to split the content?

chapter

Extracted 20 sections!



Step 3: Preview & Select Sections

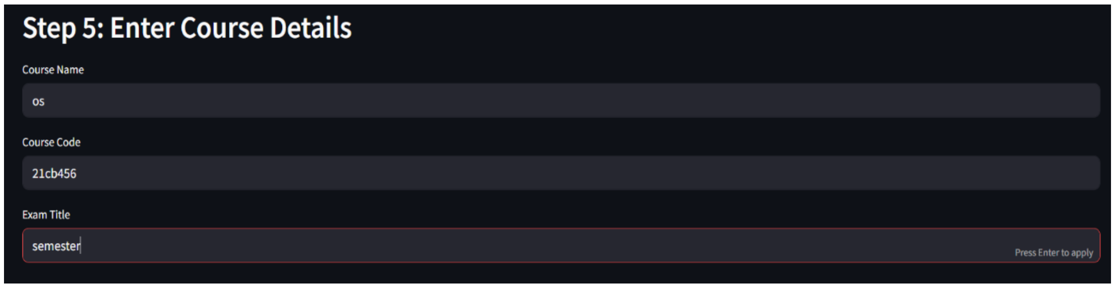
Select sections to generate questions from:

3. Chapter 3 Proc… × 4. Chapter 4 Thr… × 6. Chapter 6 CPU… × 7. Chapter 7 Dea… ×

Step 4: Choose Assessment Type

Assessment Type

question paper



Step 5: Enter Course Details

Course Name

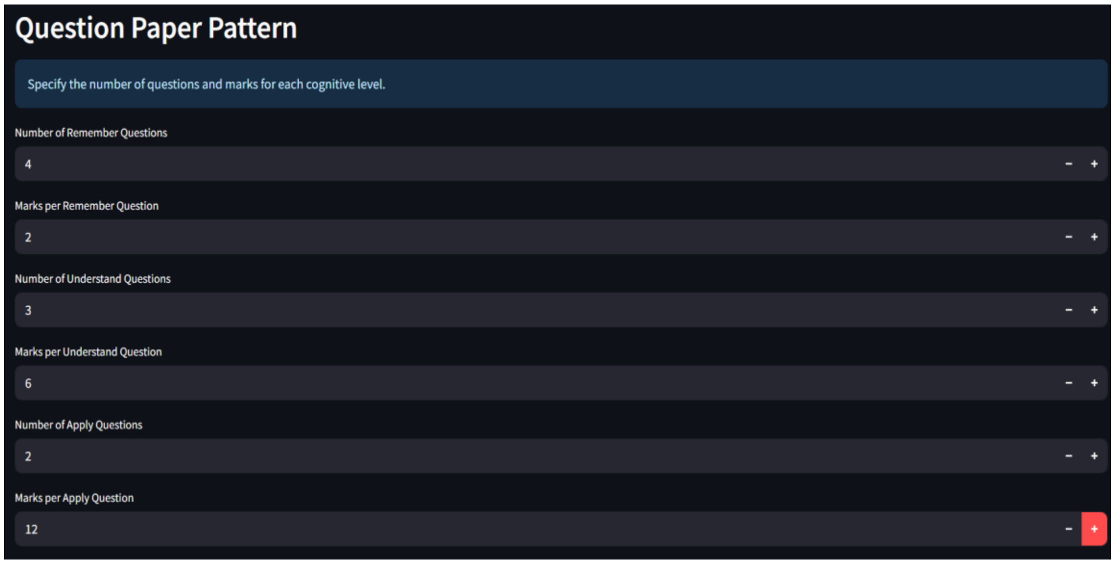
os

Course Code

21cb456

Exam Title

semester



Question Paper Pattern

Specify the number of questions and marks for each cognitive level.

Number of Remember Questions

4

Marks per Remember Question

2

Number of Understand Questions

3

Marks per Understand Question

6

Number of Apply Questions

2

Marks per Apply Question

12

Figure 5.3.1(a): Step 1 - Upload PDF (Streamlit UI)

Figure 5.3.1(b): Step 2 - Select Section Split Method

Figure 5.3.1(c): Step 3 - Preview & Select Sections

Figure 5.3.1(d): Step 4 - Choose Assessment Type

Figure 5.3.1(e): Step 5 - Enter Course Details

Figure 5.3.1(f): Step 6 - Question Paper Pattern

Backend (FastAPI/Flask Server):

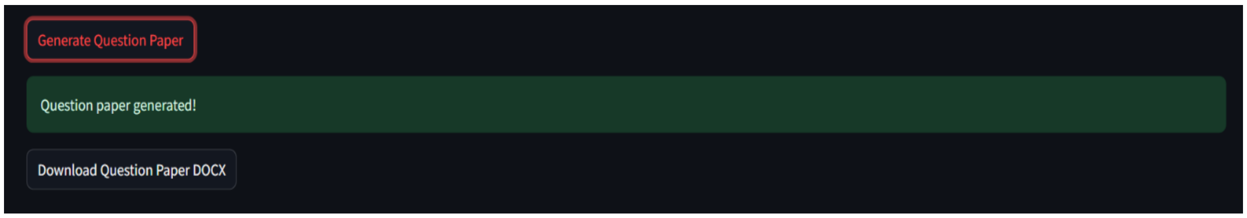
- Handles file uploads, content extraction, embedding generation, and API calls to the Groq-hosted LLM for question generation.
- Stores and retrieves semantic embeddings using the FAISS vector database.

Language Model Integration:

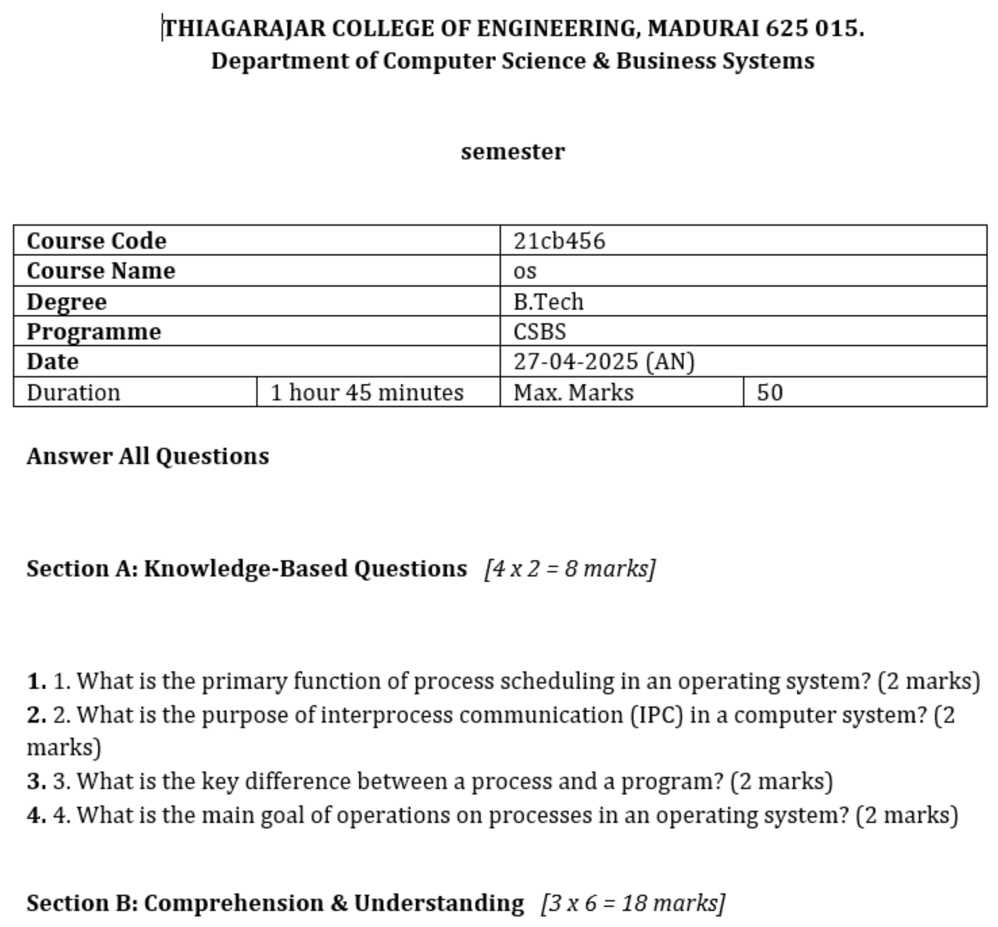
- The LangChain library manages prompt creation and response handling with the Groq API, ensuring Bloom's Taxonomy-aligned question generation.

Document Export Module:

- Assembles generated questions and exports the final output in DOCX format (using python-docx), allowing for easy editing and printing.
- These modules communicate through REST APIs and local file storage to provide a responsive user experience.



Question paper generated!



THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI 625 015.
Department of Computer Science & Business Systems

semester

Course Code		21cb456	
Course Name		os	
Degree		B.Tech	
Programme		CSBS	
Date		27-04-2025 (AN)	
Duration	1 hour 45 minutes	Max. Marks	50

Answer All Questions

Section A: Knowledge-Based Questions *[4 x 2 = 8 marks]*

1. 1. What is the primary function of process scheduling in an operating system? (2 marks)
2. 2. What is the purpose of interprocess communication (IPC) in a computer system? (2 marks)
3. 3. What is the key difference between a process and a program? (2 marks)
4. 4. What is the main goal of operations on processes in an operating system? (2 marks)

Section B: Comprehension & Understanding *[3 x 6 = 18 marks]*

Figure 5.3.1(g): Generated Question Paper

5.3.2 TESTING

To ensure the system's reliability and performance, the following testing steps were carried out:

Functional Testing:

- Verified file uploads, section detection, and user input handling.
- Confirmed accurate question generation across all cognitive levels (Remember, Understand, Apply).

Integration Testing:

- Ensured smooth workflow from syllabus upload to final document export.
- Validated that embedding creation, semantic retrieval, and question generation occur without errors.

Performance Testing:

- The system successfully generated question papers within 5 minutes for files up to 200 MB. Even large documents like 1000-page textbooks (~8 MB) were processed smoothly.

5.4 DEPLOYMENT STRATEGY

The system was deployed and tested in a local environment. The backend server was hosted locally using FastAPI, while the frontend was accessed through Streamlit running on the same machine. All modules, including text extraction, semantic embedding, question generation, and document export, were integrated and validated locally.Environment variables (.env) were used to securely store sensitive API keys, such as the Groq API key. The FAISS vector database was also maintained on the local disk for efficient retrieval operations. As cloud deployment was not required in the current phase, all testing and usage were successfully completed on a local machine.

6. RESULTS AND ANALYSIS

6.1 System Performance

The system was tested extensively with different syllabus documents, ranging from small course outlines to large textbooks. It consistently completed the entire question paper generation process from PDF parsing to final document export - within 5 minutes. The generated papers-maintained syllabus relevance, Bloom's Taxonomy cognitive balance (Remember, Understand, Apply), and grammatical accuracy.

6.2 User Feedback

- The system was reviewed by faculty members and fellow students using sample syllabus uploads.
- Users appreciated the simplicity of the interface, the fast generation time, and the editable DOCX output format.
- Faculty members specifically liked the categorization of questions based on Bloom's Taxonomy, which made it easier to design balanced assessments.
- Minor suggestions included adding more document templates and more fine-grained control over question complexity.
- Overall, the feedback indicated that the system is practical, reliable, and ready for real academic use with minor improvements planned.

6.3 Comparative Analysis

A comparison was made between traditional manual paper setting, existing automated tools, and the developed system.Manual paper setting typically took 2–3 hours and was prone to inconsistency. Existing tools offered faster generation but often lacked syllabus mapping and Bloom's cognitive level diversity. Our system effectively addresses these gaps, providing faster generation, syllabus alignment, and cognitive balance.

Feature	Manual Setting	Existing Tools	Our System
Time Taken	2–3 hours	20–30 minutes	3–5 minutes
Syllabus Mapping	Manual effort	Limited	Full syllabus understanding
Bloom's Taxonomy Coverage	Manual	Basic	Balanced (Remember, Understand, Apply)
Export Format	Word/PDF	Mostly PDF	Word (.docx), Editable
Customization	High (manual)	Low	High (easy and fast)

Table 6.1 Comparative Analysis of Question Paper Generation Methods

The results clearly demonstrate that our system offers a superior balance of speed, quality, and flexibility.

6.4 Future Enhancements

Based on system evaluation and user feedback, the following future improvements are proposed:

- Expanding question generation to include Analyze, Evaluate, and Create cognitive levels.
- Enhance question paper customization with advanced filters (difficulty, topic, question type).
- Enable question paper generation aligned with specific Course Outcomes (COs) for targeted assessment.
- Adding more export templates with customizable headers and institution branding.

These enhancements would strengthen the system’s scalability, usability, and impact in educational environments.

7.REFERENCES

[1] Vaswani et al., “Attention is All You Need”, NeurIPS, 2017.

[2] Touvron et al., “LLaMA 2: Open Foundation and Fine-Tuned Chat Models”, Meta AI, 2023.

[3] LangChain Documentation https://docs.langchain.com

[4] HuggingFace Embeddings https://huggingface.co