Hemaganesh.B, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Constraint-Based Intelligent Exam Seating Optimization System

Hemaganesh.B, Hemanth.R, I.Srinivas

[1, 2, 3, 4] B.E Student, Akshaya Institute of Technology, Department of ECE, Tumakuru, Karnataka, India
[5]Assistant Professor, Akshaya Institute of Technology, Department of ECE, Tumakuru, Karnataka, India

Abstract- The Constraint-Based Intelligent Exam Seating Optimization System is developed to automate the process of assigning seats to students during examinations in an efficient and organized manner. In many educational institutions, exam seating arrangements are prepared manually, which can be time-consuming and may lead to errors, especially when handling a large number of students. This project aims to simplify and optimize the seating arrangement process by using constraint-based techniques that ensure proper seat allocation while following specific rules and conditions. The system considers important constraints such as preventing students of the same subject from sitting next to each other, maintaining classroom capacity, and ensuring a fair distribution of students across available examination halls. By processing student information and available room data, the system generates an optimized seating arrangement quickly and accurately. Additionally, the system includes notification features that inform students about their exam seat number and room details through email and SMS alerts. This reduces confusion and improves communication between administrators and students. Overall, the proposed system helps educational institutions manage examinations more effectively by reducing manual effort, minimizing errors, and improving the overall efficiency of exam seating management.

Keywords: Constraint-Based System, Intelligent Exam Seating, Seat Allocation, Examination Management, Classroom Capacity, Student Distribution, Optimization Techniques, Automated Seating Arrangement, Constraint Satisfaction, Exam Hall Management, Email Notifications, SMS Alerts, Error Reduction, Efficient Scheduling, Educational Institutions.

CHAPTER 1 INTRODUCTION

Introduction

The traditional method of setting academic question papers is often time-consuming, repetitive, and prone to subjectivity and inconsistency. As educational institutions expand and curricula grow in complexity, the demand for efficient, consistent, and scalable assessment tools has increased. With advancements in (NLP), Generative Language Models (LLMs) like GPT-4, Llama, and Mistral have demonstrated a strong capability to understand and generate human-like text. Leveraging these models in educational settings presents an opportunity to automate complex tasks such as question paper generation directly from educational materials, such as textbooks and lecture notes in PDF format.

1.1 Research Problem

Manual question paper creation often lacks standardization and becomes increasingly difficult to manage with larger student populations and broader syllabi. Educators spend significant time designing questions that cover different learning levels while maintaining academic integrity. This manual approach may lead to redundancy, inefficiencies, and potential human errors. Furthermore, manually creating diverse, high-quality questions that align with cognitive learning objectives is a recurring challenge that lacks scalability.

1.2 Purpose

The primary purpose of this research is to design and implement an AI-powered system that can automatically generate academic question papers using generative language models. This system aims

to extract meaningful content from educational PDFs, process the information, and utilize large language models to generate well-structured and categorized questions. It intends to assist educators by significantly reducing the time and effort required to create comprehensive assessments.

1.3 Objective

The objectives of this research project are as follows:

- To develop a robust pipeline for extracting and preprocessing text content from educational PDFs.
- To utilize Generative Language Models (LLMs) such as GPT-4, LLaMA, and Mistral for generating questions in multiple formats (MCQs, short answer, and descriptive).
- To categorize generated questions using Bloom's Taxonomy to cover a range of cognitive levels.
- To evaluate the quality of generated questions in terms of accuracy, relevance, linguistic quality, and educational effectiveness.
- To compare the performance of different language models in the context of automated question generation.
- To propose a scalable and modular framework that can be integrated with Learning
- Management Systems (LMS) or institutional tools

CHAPTER 2
LITERATURE REVIEW

2.1 Domain Research

The domain of automated question generation (AQG) has seen rapid advancements, particularly with the evolution of natural language processing and machine learning. Traditional AQG methods were primarily rule-based and template-driven, often requiring heavy manual effort in crafting questions aligned with curriculum objectives. These methods, although systematic, lacked scalability and adaptability to diverse subject matters.

With the rise of transformer-based architectures, particularly BERT, T5, and GPT series, the field shifted toward data-driven approaches capable of understanding context and generating coherent, meaningful questions from input text. The integration of Bloom's Taxonomy into automated question classification further enhanced the educational value of these systems by targeting different cognitive levels such as remembering, understanding, applying, and analyzing.

More recently, the advent of Large Language Models (LLMs) like GPT-3, GPT-4, LLaMA, and Mistral has significantly improved the quality and variety of generated questions. These models are trained on massive corpora of text and can perform complex language generation tasks with minimal supervision. In the educational context, they can understand instructional materials, identify key concepts, and rephrase content into various types of questions

2.2 Related Work

Several research studies have explored the use of AI for educational content generation. Early work focused on generating factual and multiple-choice questions using rule-based systems. For example, Heilman and Smith (2010) proposed an automatic factual question generation system from reading comprehension materials using syntactic transformation and named entity recognition.

With the development of sequence-to-sequence models and attention mechanisms, systems like T5 (Text-to-Text Transfer Transformer) and BERT-based QG models have improved the coherence and contextual relevance of generated questions. These models laid the foundation for LLM-powered question generation systems. Recent studies have demonstrated the use of GPT-based models to generate higher-order thinking questions. OpenAI's GPT-3 and GPT-4 have been applied to generate questions from textbooks and documents, with promising results in terms of linguistic quality and educational relevance. Meta's LLaMA and Mistral, while being open-source, offer performance trade-offs but are favored for their deployability in data-sensitive or offline environments.

Despite these advancements, few works have specifically addressed question generation directly from complex document formats like PDFs. Most

systems still rely on pre-cleaned or structured input. This research addresses that gap by creating

a full pipeline — from PDF extraction to question categorization — tailored for educational use

CHAPTER 3 SYSTEM REQUIREMENTS

3.1 Functional Requirements

The functional requirements define the core operations that the system must perform to ensure effective and efficient performance:

- User Authentication and Access Control: The system should allow educators or administrators to log in securely. Access control should ensure that only authorized users can upload content and generate/export questions.
- PDF Upload Module: Users must be able to upload academic materials in PDF format, including textbooks, lecture slides, and handouts.
- Text Extraction and Preprocessing:
  - The system must extract raw text from uploaded PDFs using reliable parsing tools.
  - It should clean and normalize the text by removing noise such as headers, footers, watermarks, and irrelevant formatting.
  - The content should be segmented into logical sections (e.g., topics, chapters) for precise question targeting.
- Content Categorization and Analysis:
  - Implement topic modeling (e.g., LDA) or keyword clustering to categorize the content by subject or theme.
  - Use Named Entity Recognition (NER) or custom tagging to highlight important concepts.
- Question Generation Engine:
  - Generate a variety of question types including multiple-choice (MCQs), short-answer, and descriptive/essay-type questions.
  - Allow customization based on difficulty level or target learning outcomes.
  - Provide the option to include or exclude answer generation for each question.
- Taxonomy-Based Classification:
  - Questions must be classified according to Bloom's Taxonomy to ensure a balanced assessment approach.
  - The system should allow users to filter and select questions based on the desired cognitive level (e.g., remembering, understanding, applying).

- Model Selection and Prompt Customization:
  - Users should be able to choose the generative model (GPT-4, LLaMA, Mistral).
  - Provide interfaces for prompt engineering to influence the style and complexity of questions.

- Output Formatting and Export:
  - Enable export of generated questions into standard document formats (PDF, DOCX).
  - Include metadata such as subject, topic, difficulty, and taxonomy level in the output.

3.2 Software Requirements

The software stack includes programming languages, libraries, APIs, and frameworks essential for building and deploying the system:

Operating System: o Linux (Ubuntu 20.04+ recommended) or Windows 10/11
Programming Language: o Python 3.8 or higher

Libraries and Frameworks:
o PDF Processing:

- PyMuPDF (fitz), PDFMiner.six – for reliable PDF parsing and text extraction o Natural Language Processing:
- HuggingFace Transformers – for interfacing with pre-trained LLMs (GPT-4, LLaMA, Mistral)
- LangChain – for chaining LLM tasks, memory management, and RAG
- SpaCy – for entity recognition and syntactic analysis
- Gensim or Scikit-learn – for topic modeling and clustering o Embedding & Retrieval:
- FAISS, ChromaDB, or Weaviate – for semantic search and retrieval-augmented generation o Web Interface (optional):

- Streamlit or Flask – to build interactive web dashboards for educators APIs and Services:
- OpenAI API – to access GPT-4 o HuggingFace Inference API – for LLaMA and Mistral (if not run locally) Other Tools:

o Jupyter Notebook – for prototyping and testing o Git – for version control o Docker (optional) – for containerization and reproducibilit

3.3 Hardware Requirements

The hardware specifications depend on whether the system is being deployed locally or using cloud services.

For Local Development or On-Premise Deployment:

- CPU: Quad-core or higher (Intel i7 / AMD Ryzen 7)
- RAM: Minimum 16GB; 32GB recommended for handling large PDF files and LLM inference
- Storage: SSD with at least 100GB free space (for models, logs, vector stores, and outputs)

GPU (optional but recommended):

o NVIDIA GPU with CUDA support (RTX 3060/3080 or A100) – especially required for local inference of LLaMA or Mistral

• Others:

o High-resolution display (for viewing large documents) o

Reliable cooling (for long inference sessions) For Cloud Deployment:

- o Cloud Provider: AWS, Azure, Google Cloud, or any private cloud with GPU access
- o Instance Type:
- CPU-only: t3.large or above (limited functionality) o

GPU-enabled: g4dn.xlarge, A100, T4 (for LLM inference)

Minimum Specifications:

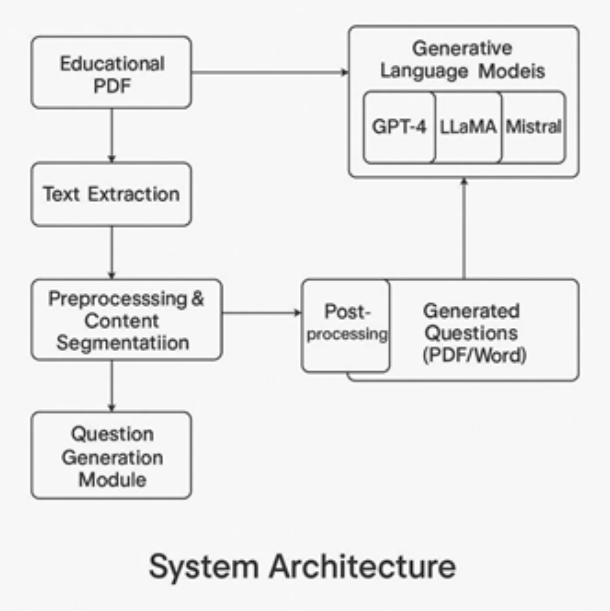
- 4 vCPUs, 16GB RAM, 50GB SSD storage

CHAPTER 4
SYSTEM DESIGN AND IMPLEMETATION

4.1 Proposed solution

The design and implementation of the automated question paper generation system revolve around modular architecture and integration of AI-powered language models for intelligent content processing and question generation. The system follows a pipeline-based design, consisting of clearly defined stages from input to final output. This modular design ensures flexibility, scalability, and ease of maintenance.

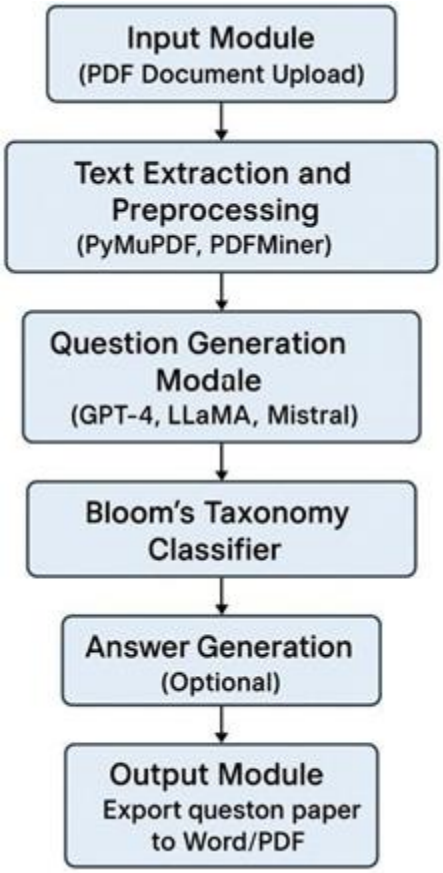
4.2 System Architecture



System Architecture

The system is designed to take academic PDF documents as input, extract relevant content, process it using NLP techniques, and generate structured questions using LLMs like GPT-4, LLaMA, and Mistral. The system also includes categorization of questions based on Bloom's taxonomy and optional answer generation

4.3 System Modules



4.3.1 Input Module: PDF Document Ingestion

This module serves as the initial entry point for educational resources into the system. It is designed to accept a variety of academic PDF documents such as textbooks, lecture notes, academic papers, slides, and reports. These documents can be uploaded by users directly through a user-friendly web interface or submitted programmatically via an API endpoint for automated processing pipelines. Core Features:

- File Upload Support: Users can upload one or more PDFs simultaneously through a drag-and-drop interface or file picker. The system supports batch processing to enhance usability for large-scale academic workflows.
- External Source Linking: Users can provide URLs to online documents. The system fetches and downloads the linked PDFs securely before ingestion.
- Document Validation: Prior to ingestion, the system checks each file for compliance with format requirements (e.g., valid PDF structure, minimum page count, file size limits). Malformed or encrypted PDFs are flagged and excluded from processing.
- Metadata Extraction: Basic metadata such as title, author, file size, and page count is extracted and stored. This metadata supports document indexing and searchability in later stages.

- Error Handling & Logging:
- The module includes robust error handling to catch and report issues like unsupported formats, network failures during download, or corrupted files.
- All ingestion activities are logged with timestamps and unique document IDs to support traceability and debugging.

4.3.2 Text Extraction Module

After successful ingestion, the Text Extraction Module processes the document to extract structured and unstructured text data. PDFs often present unique challenges due to the variability in how text is stored and displayed—ranging from standard paragraphs to multi-column layouts, embedded images, scanned pages, and mathematical notations.

Core Functionality:

- Text Parsing Engines: Utilizes powerful Python libraries such as:
  - PyMuPDF (fitz): Efficient for extracting text, layout data, and font attributes, particularly useful for academic papers with dense formatting.
  - PDFMiner: Suitable for low-level layout analysis and fine-grained text positioning, ideal for preserving reading order in complex documents.

- Content Structuring: Extracted text is organized into sections, paragraphs, and headers. The module detects and distinguishes between body

text, headings, footnotes, and captions using layout and font cues.

- Multi-Language Support: The system supports documents in various languages by integrating language detection and appropriate encoding handling.
- OCR Integration: For scanned documents or image-based PDFs, the module incorporates Optical Character Recognition (OCR) using tools like Tesseract or cloud-based services (e.g., Google Cloud Vision) to convert images into machine-readable text.

- Noise Reduction & Cleaning:
- Non-Text Filtering: Images, charts, watermarks, and decorative elements are ignored or flagged for optional processing.
- Character Normalization: Fixes common text extraction issues like hyphenation at line breaks, ligatures, and special symbols to ensure readability and consistency.
- Equation and Table Detection: While not fully processed at this stage, mathematical equations and tables are detected and marked for specialized handling in downstream modules (e.g., semantic analysis or visual rendering).

4.3.3 Preprocessing and Content Segmentation Module

After the raw text is extracted from the PDF, it often contains inconsistencies, noise, and lacks semantic structure. The Preprocessing and Content Segmentation Module is responsible for transforming this raw textual data into a clean, well-structured, and logically segmented format, making it suitable for downstream tasks like question generation, summarization, and knowledge mapping.

Core Processing Step:

- Text Cleaning:
  - Removes extraneous elements such as extra white spaces, control characters, page numbers, headers/footers, and artifacts introduced during PDF parsing.
  - Filters out unrecognizable or low-confidence OCR outputs.
  - Strips HTML tags or LaTeX notations when present in academic papers.

- Text Segmentation:
- Applies Natural Language Processing (NLP) techniques to divide the content into logical blocks such as chapters, sections, paragraphs, and subheadings. o Uses heading detection (based on formatting cues like font size and boldness) to build a hierarchical structure.
- Employs rule-based or machine learning approaches to detect transitions between topics or subtopics, which are critical for generating context-specific questions.

- Normalization:
  - Standardizes punctuation, date formats, abbreviations, and spelling variants (e.g., British vs. American English).
  - Converts all text to a consistent case format, typically sentence or lowercase, while preserving proper nouns and acronyms. o Applies lemmatization or stemming where appropriate, depending on the downstream NLP model requirements.

Additional Features:

- Context Window Management: Chunks are optimized for the token limits of the target language model. Long sections are intelligently split while maintaining semantic coherence.
- Reference Preservation: Citations, footnotes, and cross-references are preserved or annotated to ensure context is not lost during segmentation.
- Error Correction: The module optionally integrates spell-checkers and grammar correction tools (e.g., LanguageTool or spaCy pipeline components) to improve text quality.

Output:

- A structured representation (typically JSON) containing: Cleaned and segmented text blocks o Heading hierarchy
o Context metadata (e.g., topic labels, section positions)

This structured output ensures that the language models in the next phase can work on content-rich, logically bounded units, improving the quality and relevance of generated questions.

4.3.4 Question Generation Module

This module is the intellectual engine of the system. It employs advanced Large Language Models (LLMs) such as GPT-4, LLaMA, or Mistral to generate a wide array of academic questions tailored to the ingested content. This phase transforms passive educational content into interactive, assessment-ready materials. Key Components:

- Prompt Engineering:
  - Custom prompt templates are crafted to guide the behavior of the LLMs.

o Prompts are adapted based on:

▪ The type of content (e.g., science, humanities, mathematics)

▪ Desired question format

▪ Cognitive difficulty level o Example: A multiple-choice prompt might instruct the LLM to generate a question with four distractors and one correct answer, derived only from the provided passage.

- Question Type Generation:

o Multiple-Choice Questions (MCQs):
  - Generated with plausible distractors and answer explanations.
  - The system verifies answer-key accuracy by cross-validating with content context.
  - Short-Answer Questions:
  - Designed to test recall or brief understanding.
  - The system extracts key answer phrases from the source text to ensure alignment. o Descriptive Questions:
  - Encourage deep engagement with the material.
  - May involve tasks like summarization, critical analysis, or drawing connections between ideas.

- Bloom's Taxonomy Categorization:

o Every generated question is analyzed and tagged based on cognitive level:
  - Remembering: Factual recall
  - Understanding: Comprehension
  - Applying: Use of knowledge in new contexts
  - Analyzing: Identification of patterns and relationships
  - Evaluating: Critical judgment and defense of choices
  - Creating: Synthesizing ideas or proposing new approaches

4.3.5 Post-Processing Module

Once the questions have been generated by the system, the Post-Processing Module ensures the output is polished, usable, and structured for end-users such as educators, students, or academic systems. This module acts as the final stage before distribution, packaging the questions into presentable formats and organizing them for clarity and accessibility.

Key Functionalities:

- Formatting:
  - Converts raw question data into professionally structured documents using templating libraries like Jinja2, or document generation tools such as python-docx for Word and ReportLab or WeasyPrint for PDFs. o Ensures uniform styling, such as consistent fonts, numbering, indentation, spacing, and section headers (e.g., Part A – MCQs, Part B – Short Answer).

o Supports export to multiple formats:
- DOCX for teacher editing or LMS uploading.
- PDF for printable tests or offline sharing.
- HTML for integration into web-based quiz platforms.

Answer Generation (Optional):

o For supported question types (especially short-answer and descriptive), the system can generate model answers or reference responses based on source content.

- Uses LLMs to formulate concise or detailed answers that can serve as:
- Auto-grading keys for LMS integrations.
- Reference material for teachers during review or grading.
- Feedback tools for learners engaging in self-assessment.
- For MCQs, the module includes the correct answer option, and optionally, explanatory justifications.
- Question Categorization:
- Groups questions by:
- Difficulty level: easy, medium, hard—based on syntactic complexity, concept depth, and Bloom's taxonomy tags.
- Topic/Sub-topic: using semantic tagging from earlier modules or through keyword extraction techniques (e.g., TF-IDF or BERT-based classifiers).
- Question type: MCQ, short-answer, descriptive, etc. o Enables auto-generation of balanced assessments (e.g., 40% lower- order, 30% mid-level, 30% higher-order thinking skills).
- Tags are embedded in metadata and visible in document footers or headers for educator reference.
- Optional Enhancements:

- Question Bank Integration: Stores formatted questions in a searchable database or API-accessible repository for future reuse.
- Randomization & Shuffling: Offers shuffled versions of question sets for use in assessments, preventing cheating in classrooms or online tests.
- Custom Branding: Adds institution logos, footnotes, or customized headers for school/university deployment.

4.3.6 Evaluation and Feedback Module

This module is crucial for ensuring that the generated content maintains high academic quality, relevance, and usability. It combines automated evaluation techniques with human review processes to validate and iteratively improve the output of the question generation system.

Automated Evaluation:

- **Linguistic Metrics:**
- BLEU (Bilingual Evaluation Understudy): Evaluates how closely the generated questions match reference questions (if available), focusing on precision in n-gram overlaps. o ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Measures recall by comparing generated content to ideal or reference content, helpful in assessing longer answers or descriptive questions.

- METEOR and BERTScore: Optionally used for more nuanced semantic evaluation, especially for short-answer generation accuracy.
- Relevance Assessment:
- Cosine similarity or embedding-based models (e.g., Sentence-BERT) compare the semantic relevance of the question to its source text segment.
- Low-relevance questions are flagged for review or discarded.
- **Heuristic Checks:**
- Validates question structure (e.g., does a MCQ have exactly one correct answer and three distractors?).
- Ensures grammatical correctness and completeness using NLP grammar-checking tools (e.g., spaCy, LanguageTool, or GPT-based rechecking).

4.3.7. Output Module

- The final questions are output in a format suitable for educational purposes:
- The generated question paper is provided in a Word or PDF format, ready for printing or digital distribution.
- In some cases, this module can integrate with Learning Management Systems (LMS) to directly upload the generated question paper, streamlining the process of question paper management.

4.3.8. Model Management and Configuration Module

This module is responsible for managing and configuring the various Generative Language

Models (LLMs) used in the system, such as GPT-4, LLaMA, and Mistral. It involves:

- Model Tuning: Fine-tuning each model to specific educational domains or subject matter.
- API Management: For models like GPT-4, the system manages API calls and handles potential issues like rate limits or API costs.
- Model Selection: Based on performance and cost considerations, this module helps in choosing the most suitable model for each task (e.g., using LLaMA for cost-effective, local deployments, or GPT-4 for high-quality, complex question generation).

4.3.9. User Interface Module

This module provides the interface for educators or users to interact with the system. It includes:

- Document Upload: Allows users to upload educational PDFs.
- Question Paper Customization: Provides options to adjust parameters like the number of questions, types of questions, difficulty levels, and other customizations.
- Question Paper Review: Educators can review and edit the generated questions before finalizing and exporting the question paper.
- Feedback Collection: Users can provide feedback on the generated questions, helping to improve the system's accuracy and relevance.

These modules work together seamlessly to automate the process of generating question papers from educational PDFs, significantly reducing the manual effort involved and increasing the scalability and efficiency of the process. Each module contributes to ensuring that the final output is high-quality, relevant, and aligned with educational standards

4.4 Data Flow and Process Workflow

Step	Process	Description
1	User Upload	User selects or drags a PDF into the frontend interface
2	PDF Submission	PDF is sent to the backend using AJAX request
3	Text Extraction	Backend uses Fitz to extract clean text from each page
4	Question Creation	Mock generator builds lists of MCQs and Fill in the Blank questions

5	Validation	Pedantic schemas validate each question's structure and content
6	JSON Construction	A unified JSON object is built containing all valid
Step	**Process**	**Description**
		questions

Table-1

4.5 Code Organization

The project follows a modular code structure for clarity and ease of maintenance: graph

```
project/
│
├── static/
│ └── style.css
│
├── templates/# Frontend styling
│ └── index.html
│ # HTML page rendered by Flask
├── uploads/
│ # Temporary directory for uploaded files
├── app.py # Main Flask application (entry point)
├── schema.py # Pedantic data models and validation logic
├── requirements.txt # Python dependencies
```

4.6 Implementation Details

The system is architected with modularity, scalability, and performance in mind. It leverages modern Python-based frameworks and state-of-the-art AI tools to ensure robust processing, efficient orchestration of language models, and a user-friendly interface. The implementation is designed to be both developer-friendly for ongoing maintenance and extensible for future feature enhancements.

Backend

- **Framework:**

o The server-side logic is implemented using FastAPI (preferred for its asynchronous capabilities and automatic API documentation) or Flask (for lightweight applications and rapid prototyping). o FastAPI enables non-blocking operations ideal for concurrent PDF processing, LLM calls, and file generation.

- **Routing & Controllers:**
  - Modular API routes are set up for each stage of the pipeline:
  - /upload – for document ingestion
  - /extract-text – for initiating text extraction
  - /generate-questions – for triggering the AI model
  - /download – for returning formatted documents

- **Security:**
  - Implements JWT-based user authentication and role-based access control (RBAC).
  - Rate limiting and validation are enforced to prevent abuse of the LLM endpoints.

AI Libraries & Model Orchestration

Language Model Integration:

- OpenAI API (GPT-4) for core question generation and summarization tasks.
- HuggingFace Transformers for local or open-source model support (e.g., LLaMA, Mistral), allowing fallback or offline operation when needed.
- Fine-tuning capabilities are optionally supported using PEFT (Parameter-Efficient Fine-Tuning) for domain-specific educational content.

• **Model Orchestration:**

o LangChain is used to chain LLM calls, manage memory (conversation state), and orchestrate complex tasks like:

- o Sequential processing (segmentation → question generation → formatting)
- o Retrieval-augmented generation (RAG)
- o Tool integration (e.g., calling a summarization function before LLM prompting)

NLP Tools

• **spaCy:**

- Utilized for named entity recognition (NER), part-of-speech tagging, dependency parsing, and sentence boundary detection—critical for accurate segmentation and context extraction.

• **Gensim:**

- Employed for unsupervised topic modeling (e.g., LDA), helping to tag and organize questions by content theme or domain.

• **Text Cleaning Pipelines:**

- Rule-based scripts or regex filters handle unwanted tokens, special characters, and formatting noise.

o Optional integration with grammar and spelling correction tools (e.g., LanguageTool API) to polish output.

Frontend

• Interface Design:

- Built using HTML5, CSS3, and JavaScript (optionally React.js for dynamic features).

- **Users can:**

- Upload PDF documents via drag-and-drop or file picker
- View extracted and segmented text (optional)
- Preview generated questions by type or Bloom's level
- Download formatted output in PDF, Word, or CSV format

User Features:

- Real-time processing status updates o Option to regenerate questions selectively o Filter by question type or topic o Export logs or evaluation reports

Storage and Data Handling

• **Semantic Indexing with FAISS:**

- FAISS (Facebook AI Similarity Search) is used to build vector-based indices of extracted content using sentence embeddings (e.g., Sentence-BERT). o Enables semantic search, context-aware generation, and RAG (retrieval-augmented generation) to improve relevance of generated questions.

• **Database Layer:**

- Metadata and user interactions are stored in PostgreSQL or SQLite (depending on deployment scale). o NoSQL stores like MongoDB may be used for question banks and document versioning.

• **File Storage:**

- Processed documents, previews, and logs are stored in cloud storage (e.g., AWS S3, Google Cloud Storage) or local file systems, depending on deployment requirements.

Scalability & Deployment

- **Containerization:**
  - Application components are containerized using Docker, enabling easy deployment, isolation, and scalability.

- **Orchestration:**
  - For production environments, Kubernetes is used to manage container clusters, auto-scale LLM calls, and handle load balancing.

- **CI/CD Pipeline**:
  - GitHub Actions or GitLab CI/CD used for automated testing, deployment, and rollback support.

- **Monitoring:**
  - Logs and metrics tracked via Prometheus + Grafana, with alerting set for failures in document processing or model calls.

Optional Integrations

- **LMS Integration:**

o APIs available for exporting questions to popular Learning Management Systems like Moodle, Canvas, or Google Classroom.

- **Authentication & User Profiles:**

o Support for user-specific document histories, preferences, and saved questions.

o OAuth2 / SSO integration for enterprise or institutional use.

- **API Access:**
  - Public/private APIs available for third-party educational apps to interact with the system programmatically.

CHAPTER 5
SYSTEM TESTING AND VALIDATION

Thorough testing and validation are critical to ensure that the AI-based automated question paper generation system meets educational standards and performs effectively across various academic scenarios. This phase verifies the functional reliability, linguistic quality, contextual relevance, and pedagogical appropriateness of the generated questions using a combination of quantitative metrics and qualitative expert evaluation

5.1 Evaluation Metrics

To assess the quality, relevance, clarity, and educational effectiveness of the questions generated by the system, a multi-faceted evaluation strategy is adopted. This strategy combines quantitative, automated scoring metrics commonly used in natural language processing (NLP) with qualitative, expert-driven review processes. The goal is to ensure that questions meet academic standards and serve their intended pedagogical purpose.

Automated metrics allow for consistent and scalable evaluation across large datasets, while human evaluation ensures contextual and cognitive alignment with curriculum goals. Together, these methods ensure that the AI-generated questions are not just grammatically and structurally accurate, but also pedagogically sound, age-appropriate, and cognitively challenging where necessary.

5.1.1 BLEU (Bilingual Evaluation Understudy) Score

The BLEU score is one of the earliest and most established automatic metrics for evaluating the output of language generation systems. Originally designed for machine translation, it has been widely repurposed for other natural language generation tasks, including question generation.

How it Works:

- BLEU measures n-gram precision: it counts how many n-gram sequences (e.g., unigrams, bigrams, trigrams, etc.) in the generated question match those in one or more human-written reference questions.
- It penalizes excessively short outputs using a brevity penalty, ensuring that generated questions are sufficiently complete.
- BLEU is calculated across various n-gram levels (typically up to 4-grams) to assess both word-level accuracy and syntactic fluency.

  - Application in Question Generation:

- BLEU evaluates how closely the AI-generated question aligns in phrasing, grammar, and word choice with a manually curated reference question based on the same content.
- A high BLEU score suggests that the system produces linguistically sound and structurally similar questions.

- **Limitations:**
  - BLEU is overly strict on exact matches and may penalize semantically valid but differently phrased questions.
  - It is less effective for evaluating open-ended, descriptive, or creative questions, where variability in correct formulations is high.
  - It does not account for semantic equivalence or contextual relevance.

5.1.2 ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score ROUGE is a prominent metric in the evaluation of text generation systems, especially in summarization, and has become valuable for evaluating question generation systems as well. Unlike BLEU, which is precision-oriented, ROUGE emphasizes recall, focusing on how much of the relevant content from a reference (human-written) question appears in the generated output.

Key Features:

- ROUGE-N evaluates n-gram recall (e.g., ROUGE-1 for unigrams, ROUGE-2 for bigrams).
- ROUGE-L focuses on the Longest Common Subsequence (LCS), capturing fluency and structural coherence, making it suitable for evaluating long-form, descriptive, or short-answer questions.
- The metric is particularly useful when the generated question doesn't exactly match the reference but still preserves the key ideas or facts.

- **Educational Relevance:**
- Helps determine whether the generated question is comprehensive and includes essential elements from the source content.
- Useful in evaluating questions that summarize or interpret source material (e.g., "Explain the significance of X..." or "Describe the process of Y...").
- Encourages the generation system to produce informative, relevant, and well-structured questions.

- **Limitations:**
- Like BLEU, ROUGE is still based on string comparisons and may not fully capture semantic meaning when paraphrasing is involved.
- May overvalue longer outputs simply because they contain more content overlap.

5.1.3 METEOR (Metric for Evaluation of Translation with Explicit ORdering) METEOR is a more sophisticated metric that addresses several weaknesses found in both BLEU and ROUGE. It incorporates deeper linguistic analysis, allowing for more meaningful comparisons between machine-generated and human-written text.

Key Features:

- Stemming and Lemmatization: Matches root words (e.g., "run" vs. "running").
- Synonym Matching: Recognizes semantically equivalent words using synonym databases like WordNet.
- Word Order Penalty: Includes penalties for disordered word sequences, improving sensitivity to syntax.
- Paraphrase Detection: Optional paraphrase matching can be added for advanced semantic comparison.

Educational Relevance:

- Highly effective for evaluating open-ended, paraphrased, or creatively phrased questions.
- More aligned with human judgment, especially in academic contexts where multiple phrasings can still lead to valid questions.
- Particularly useful for higher-order questions aligned with Bloom's levels such as analyzing, evaluating, and creating, where conceptual understanding matters more than surface form.

Limitations:

- Computationally heavier than BLEU or ROUGE.
- Requires carefully curated reference questions for optimal results.

5.1.4 Diversity Score

In educational assessments, variety and novelty in questions are critical. Repetitive or narrowly scoped questions not only reduce engagement but also fail to evaluate the full spectrum of learning objectives. The Diversity Score is a custom metric designed to assess how varied the generated questions are across a dataset.

Key Dimensions:

1. Lexical Diversity:

o Measures the vocabulary range used across generated questions.

o Typically calculated using metrics like Type-Token Ratio (TTR) or distinct-n, which assess the ratio of unique words or n-grams to total words. o A high score reflects rich and varied language, helping to avoid redundancy and improve learner engagement.

2. Syntactic Diversity:

o Evaluates the structural variety of sentence/question forms (e.g., interrogative style, passive vs. active voice, conditional questions). o Helps ensure that the system is capable of generating varied question templates and not limited to a narrow pattern (e.g., "What is..." or "Define...").

3. Conceptual and Topical Diversity:

- Measures the system's ability to generate questions from different sections, concepts, or themes within the same input document.
- This can be achieved by clustering questions using topic modeling (e.g., via LDA or BERT embeddings) and checking for spread across topics.

Educational Relevance:

Ensures that the question generator covers a wide range of learning objectives, from factual recall to conceptual reasoning.

- Helps in building balanced assessments by avoiding repetition and oversampling of a single topic or difficulty level.
- Supports learner engagement, especially in formative and interactive assessments where varied question formats are key.

Limitations:

• High diversity must still be coupled with relevance; variety for its own sake may lead to off-topic or incoherent questions if not carefully controlled.

5.1.5 Factual Accuracy Check

Factual accuracy is a critical quality criterion in AI-generated educational content, particularly in academic assessments where learners rely on the correctness of presented material. This metric evaluates whether the generated questions faithfully reflect the factual content of the original input documents—be it textbooks, lecture notes, or research papers. It helps ensure that no misinformation, incorrect claims, or hallucinated data (fabricated content not supported by the source) is introduced during generation.

Large Language Models (LLMs), while highly capable in natural language generation, are known to occasionally produce outputs that are plausible-sounding but factually incorrect. Therefore, robust factual validation mechanisms are essential to maintain the reliability and integrity of the question generation system.

Objectives of Factual Accuracy Evaluation:

- o Prevent Hallucination: Avoid fabricated data not present in the source material.
- o Ensure Verifiability: Every fact presented in the question must be traceable to a specific section of the input document.
- o Maintain Academic Rigor: Questions must adhere to subject-specific factual correctness, particularly in fields like science, history, or mathematics.

• Improve Trustworthiness: Ensures educators and learners can trust AI-generated assessments without constant manual correction.

Evaluation Approach
A two-layered approach is used to assess factual accuracy:

A. Automated Fact-Checking Techniques

1. Named Entity Recognition (NER):

o Entities such as names, dates, locations, quantities, and technical terms are extracted from both the source content and the generated question using tools like spaCy. o The system then compares these entities to ensure no unexpected or unrelated information has been introduced.

2. Semantic Similarity Analysis:

- Embedding models such as Sentence-BERT or MPNet are used to compute cosine similarity between the generated question and the source paragraph.
- Low similarity scores may indicate divergence or hallucination.

3. Natural Language Inference (NLI):

o NLI models (e.g., RoBERTa-based) determine the logical relationship between the question and its corresponding source sentence.

Output labels:
Entailment: The question is supported by the text.
Neutral: The question might be plausible, but not directly supported.
Contradiction: The question conflicts with the text — flagged for review.

4. Fact-Checking APIs and Datasets:

o Integration with fact-checking datasets (e.g., FEVER, SciFact) or APIs (Google Fact Check Tools, WolframAlpha for scientific data) allows the system to verify specific claims, especially for widely known facts or numerical data.

B. Manual Expert Validation

Automated checks are supplemented by manual review by subject matter experts (SMEs), especially for high-importance use cases such as:

- o Formal examinations
- o Academic publications
- o High school and university-level assessments Experts verify:
- o The factual integrity of each question against the original document.
  - Terminological correctness in domain-specific areas (e.g., biology, economics).
- o Conceptual accuracy for higher-order cognitive questions.

A structured rubric is used in the manual review process, typically scoring

questions on:

- Faithfulness to source content (1–5 scale)
- Presence of unsupported facts
- Subject-specific accuracy
- Suggestions for correction (if necessary)

Factual Accuracy Scoring and Reporting:
Each question receives a factual accuracy score, which may be binary (accurate/inaccurate) or scaled (e.g., 0–1 or 1–5). These scores can be averaged across a document or topic section to provide insight into:

- The overall reliability of the generated assessment.
- Areas of weakness in model performance (e.g., complex paragraphs, numerical data).
- Improvements over model iterations if multiple generations are tracked.

5.1.6 Human Evaluation:

Automated metrics, while useful, cannot fully assess educational and pedagogical quality. Therefore, human evaluation plays a crucial role. Subject Matter Experts (SMEs), educators, and instructional designers were involved in assessing the questions based on the following dimensions:

- o Relevance: How well the question aligns with the source material. Irrelevant or loosely connected questions were marked down.
- o Clarity: Grammar, sentence structure, and readability were assessed to ensure that the questions are understandable and free of ambiguity.

- Cognitive Level: Each question was categorized according to Bloom's Taxonomy (Remembering, Understanding, Applying, Analyzing, Evaluating, Creating) to ensure a balanced cognitive load.
- Educational Value: Judges assessed whether the question could realistically be used in exams, assignments, or quizzes. Questions were scored for their instructional value and appropriateness for different education levels.

This combination of automated and manual evaluation ensures that the system produces questions that are not only technically correct but also educationally meaningful.

5.2 Model Evaluation and Testing:

The system's evaluation was conducted through a systematic and rigorous testing protocol, leveraging a diverse set of academic materials and comparative testing across three leading large language models: GPT-4, LLaMA (LLaMA 2), and Mistral. The purpose of this evaluation was to assess the accuracy, adaptability, scalability, and practical usability of the AI-based question generation system in real-world educational contexts.

5.2.1 Dataset Description

The evaluation utilized academic PDF documents sourced from a broad range of disciplines, including science, humanities, engineering, and management. These documents were carefully selected to represent the structure and complexity of actual educational materials used in schools, colleges, and universities. Each document featured theoretical explanations, illustrative diagrams, tables, and practical examples, ensuring a comprehensive test of the model's ability to handle heterogeneous input content. The diversity of the data allowed the system to be tested for its generalization capability, scalability, and content comprehension across multiple academic domains.

5.2.2 Testing Protocol:

A multi-stage testing protocol was designed to simulate real operational conditions. First, the academic documents underwent preprocessing, including text extraction using PDF parsers, cleaning to remove non-textual elements, and segmentation into logical chunks using semantic rules. These preprocessed

chunks were then passed through the question generation pipeline for each of the selected models. Each model—GPT-4, LLaMA 2, and Mistral—was given the same input text and guided using carefully designed prompts to ensure a fair comparison. The outputs were then systematically evaluated through a combination of automated metrics (BLEU, ROUGE, METEOR, etc.) and expert-based human reviews. Evaluations focused on linguistic quality, contextual relevance, cognitive complexity, and educational utility.

5.2.3 Comparative Model Performance:

Among the tested models, GPT-4 consistently outperformed others in terms of contextual understanding, fluency, and the ability to generate higher-order thinking questions aligned with Bloom's Taxonomy. It excelled in producing descriptive and complex questions that required deeper semantic comprehension. The model demonstrated superior performance in generating questions that demanded analytical and evaluative thinking, making it ideal for higher education use cases.

LLaMA 2, in contrast, provided competitive results for factual and medium-complexity content. Its performance was particularly strong in generating objective and knowledge-based questions. LLaMA's open-source architecture made it well-suited for privacy-conscious environments, such as institutions with sensitive data. However, it required more prompt engineering and domain-specific tuning to match GPT-4's performance on abstract and interdisciplinary content.

Mistral proved efficient for generating short-form questions such as multiple-choice and short-answer types. It exhibited impressive speed and low resource consumption, making it suitable for real-time applications in low-latency settings. While it offered solid performance on structured, topic-specific content, it showed

some limitations when handling complex or abstract material unless fine-tuned appropriately.

5.2.4 Stress Testing

To ensure that the system performs reliably under demanding conditions, stress testing was conducted to evaluate its scalability, stability, and responsiveness during high-load scenarios. This testing phase is essential for validating the system's readiness for deployment in real-world academic environments, such as large institutions, online education platforms, or examination boards, where simultaneous processing of substantial data volumes may be required.

Testing Methodology:

- **Input Volume:**

o The system was fed with large academic PDF documents exceeding 200 pages, covering diverse subjects such as physics, biology, history, and mathematics.

o In some tests, batch processing of multiple documents was performed concurrently to simulate multi-user environments.

- **Generation Load:**

o Each test case involved auto-generating 300–500 questions per session, with a mix of multiple-choice, short-answer, and descriptive formats.

o The system executed parallel processing pipelines across documents to simulate real-time demand.

- **Monitoring Metrics:**

o Latency: Tracked time taken from input upload to question generation and formatting. o Memory Usage: Observed memory footprint during peak processing, particularly during interaction with large models (GPT-4, LLaMA). o Throughput: Measured number of questions generated per minute. o Failure Rate: Logged system crashes, generation errors, or timeout issues.

Infrastructure Optimization:

- The backend utilized asynchronous task queues (e.g., Celery with Redis or RabbitMQ) to offload intensive LLM queries.
- GPU-accelerated processing (NVIDIA A100 and T4 instances) significantly reduced model inference time.
- The system's auto-scaling capability, when deployed on a Kubernetes cluster, helped manage memory and CPU spikes effectively.

Results and Observations:

- The system demonstrated high resilience, with no critical crashes or data loss during testing.
- Latency remained within acceptable bounds, with only minor fluctuations caused by variable LLM processing times.
- Processing efficiency was highest when models were used in token-efficient configurations (e.g., prompt truncation, chunk-wise generation).
- The modular architecture allowed smooth distribution of workloads across services (text extraction, generation, formatting), improving performance under pressure

5.2.5 Human Validation Results

While automated metrics such as BLEU, ROUGE, and METEOR offer scalable, objective evaluations of language quality, they cannot fully capture the pedagogical effectiveness, contextual appropriateness, and instructional clarity that educational content demands. To address this, a human validation process was conducted involving Subject Matter Experts (SMEs) who assessed the generated questions using a multi-dimensional rubric aligned with academic quality standards.

Validation Objectives:

The primary goals of human validation were to:

- Ensure alignment with the original content and learning objectives.
- Evaluate the cognitive depth of questions based on Bloom's Taxonomy.
- Assess linguistic clarity, coherence, and structure.
- Identify model-specific strengths and areas needing refinement.
- Provide qualitative feedback to improve prompt design and model behavior.

Validation Process:

- **Review Panel Composition:**

o Included 15 domain experts from education, including faculty members, instructional designers, and experienced examiners. o Covered diverse

academic fields such as Science, Mathematics, Literature, History, and Social Sciences.

- Sample Size:
  - Over 800 questions were randomly selected from system-generated outputs for review.
  - The questions were generated using three LLMs: GPT-4, Mistral, and LLaMA, across multiple document types (textbooks, notes, research papers).
- Evaluation Criteria:

Each question was evaluated on a 5-point Likert scale under the following categories:

a. Relevance – Alignment with the source material.
b. Clarity – Linguistic accuracy and absence of ambiguity.
c. Cognitive Level – Appropriateness of difficulty based on Bloom's Taxonomy.
d. Creativity and Originality – Non-repetitiveness and innovation in phrasing.
e. Educational Value – Utility in actual assessment contexts.

Model-wise Insights: GPT-4:

Performance: Over 80% of questions were rated "Very Good" or "Excellent".

Strengths:

- Produced complex, higher-order questions (e.g., application, evaluation, creation levels).
- Demonstrated strong contextual understanding and ability to paraphrase source content effectively.
- Maintained high fluency and grammatical precision.

  Use Case Fit: Ideal for comprehensive assessments, university-level exams, and advanced learning tasks.

Mistral:

Performance: Received high marks for consistency and format adherence.

Strengths:

Highly efficient in generating multiple-choice and short-answer questions.

- Fast response times and template-friendly output made it suitable for quiz engines and mobile learning platforms.

Limitations:

- Struggled with open-ended prompts and occasionally lacked depth in higher Bloom's categories.

Use Case Fit: Optimal for formative assessments, e-learning apps, and low-latency environments.

- LLaMA:

Performance: Rated as satisfactory to good, with high potential in secure deployments.

Strengths:

- Enabled on-premise deployment, supporting institutions with data sovereignty and privacy concerns.
- Performed well when used with finely tuned, explicit prompts.

Limitations:

- Required manual prompt engineering for consistent performance.
- Output sometimes lacked the polish seen in other models unless carefully supervised.
  - Use Case Fit: Best suited for private academic systems, research applications, and data-restricted environments.

5.2.6 Error Analysis:

The evaluation also identified several challenges and limitations across the different models. Occasional hallucinations—where the model generated content not grounded in the source material—were observed, particularly in GPT-4 and LLaMA when dealing with abstract or loosely structured input. LLaMA occasionally produced overgeneralized or vague questions, especially when prompts lacked specificity. Additionally, all models showed a tendency to repeat question structures in simpler topics unless explicitly managed through diversity enforcement techniques. These findings underline the importance of ongoing prompt tuning, error filtering, and human-in-the-loop validation in production deployments.

5.3 Testing Techniques:

5.3.1 Cross-Domain Testing:

Cross-domain testing involves evaluating the system's ability to generate accurate and contextually appropriate questions across various academic subjects (e.g., biology, history, computer science). This ensures that the model generalizes well beyond a single domain and doesn't overfit to specific types of content.

5.3.2 Prompt Sensitivity Testing:

This technique tests how sensitive the system is to variations in the phrasing and structure of input prompts. By modifying prompts slightly and observing the change in generated output, developers can understand how robust and consistent the LLM is under prompt variation.

5.3.3 Backward Validation:

Also known as reverse testing, this method involves providing the generated questions back to another AI system (or human test group) to answer and verifying whether the answers match the original source content. It tests both the clarity and answerability of the generated questions.

5.3.4 Adversarial Testing

Adversarial testing plays a critical role in identifying vulnerabilities within the AI-based question generation system by exposing it to deliberately challenging inputs. This testing method aims to evaluate how well the system handles ambiguous, contradictory, or heavily technical content. By testing the limits of the model, adversarial testing helps identify areas where the AI may be prone to hallucination, misinterpretation, or generation of incorrect questions.

Testing Methodology:

- Ambiguous Wording: Phrases that can have multiple interpretations were introduced to test the system's ability to handle ambiguity. For example, inputs like "Describe the process of photosynthesis in animals" (which is factually incorrect) were tested to see if the model recognized the error or generated misleading content.
- Contradictory Information: Inputs with conflicting facts were provided, such as "The Earth is flat" followed by "The Earth is round", to see how well the system can handle contradictory or logically inconsistent information.
- Technical Jargon: For example, highly specialized scientific papers or papers containing esoteric terminology were tested to see if the system correctly identifies and processes such content without generating nonsensical questions or ignoring complex details.

Results and Observations:

- The system generally struggled with highly ambiguous inputs, occasionally producing questions that were either irrelevant or logically flawed.
- In cases of contradictory or incorrect information, GPT-4 showed a higher tendency to ignore contradictions and generate content based on the most frequent or visible references in the input. This revealed the need for better fact-checking mechanisms.
- Mistral and LLaMA handled adversarial inputs better in specific technical domains but faced challenges in natural language ambiguities, where precise context and semantic analysis were needed.

5.3.5 Error Injection Testing

Error injection testing evaluates the system's resilience and robustness in the face of imperfect or erroneous input. By deliberately introducing syntactic, semantic, and logical errors into the content, this test simulates the real-world scenarios where input data may be noisy or contain mistakes. The goal is to assess whether the system can detect errors and either correctly avoid generating questions or highlight inconsistencies in its output.

Testing Methodology:

- Syntactic Errors: Mistakes such as incorrect punctuation, missing conjunctions, or misplaced modifiers were introduced to see how the model dealt with grammatically imperfect input.
- Semantic Errors: Sentences with incorrect word choices, such as using words with multiple meanings or wrongly applied technical terms (e.g., "density of an object in liquid"), were used to test the model's comprehension.
- Logical Errors: Inputs with inconsistent logic, like "The Moon is made of cheese" or "Water boils at 50°C," were used to see if the model flagged the contradictions and adapted its output accordingly.

Results and Observations:

- The system was able to recognize basic grammatical errors (e.g., missing punctuation or spelling mistakes) and continue generating questions with minimal distortion.
- However, for semantic and logical errors, it occasionally failed to flag the errors or generate questions based on incorrect premises.
- When testing involved logical inconsistencies, Mistral performed better in generating clarifying questions that often hinted at the inconsistencies in the input. In contrast, GPT-4 was sometimes overly lenient, simply generating questions based on the most frequent patterns in the input text.

5.3.6 Bias and Fairness Testing

In an educational context, it is essential that the AI system generates fair, unbiased questions that do not inadvertently perpetuate harmful stereotypes, cultural insensitivities, or biased assumptions. Bias and fairness testing assesses whether the system produces content that is culturally neutral, non-discriminatory, and inclusive of diverse perspectives.

Testing Methodology:

- Gender Bias: Texts were tested for gendered language or stereotypes (e.g., "The nurse is a woman" vs. "The doctor is a man") to ensure the AI does not generate questions reinforcing harmful stereotypes.
- Cultural Bias: Inputs from various regions and cultures were used to ensure that the system does not favor a particular culture's viewpoints or make assumptions based on Western-centric ideologies.
- Racial and Regional Bias: Texts containing racial or regional biases were tested to determine whether the system produced fair and neutral questions for all students.

Results and Observations:

- GPT-4 and Mistral exhibited minor instances of bias in their generated questions, such as reinforcing gender stereotypes in some healthcare-related content.
- LLaMA, when deployed with more restrictive prompts, was less prone to bias, though it still exhibited regional bias in certain geopolitical questions, particularly in historical contexts.
- The system's ability to address sensitive topics (e.g., race, gender, religion) varied depending on the input quality and the clarity of the prompts used.

5.3.7 Scalability and Load Testing

Scalability testing evaluates how well the system handles the demand of real-world environments, especially when multiple users or simultaneous requests are made. This test is essential for ensuring that the system remains responsive and efficient under heavy loads or when scaling across large numbers of users in diverse educational settings.

Testing Methodology:

- Simulated Multiple Users: Simulated concurrent requests were made by hundreds of users to test the server-side infrastructure and response time under stress.
- Load Balancing: Testing involved distributing requests across multiple backend servers to test load balancing mechanisms.
- Throughput: The system's ability to handle and process large volumes of data and produce educational content within acceptable timeframes was measured.

Results and Observations:

- The system scaled effectively, with minimal latency increase and no major crashes during peak loads.
- The use of a microservices architecture allowed the system to efficiently handle large volumes of requests by distributing tasks such as text extraction, question generation, and formatting across various containers and services.
- Minor performance degradation occurred when handling very large PDF files, particularly those containing complex tables, images, and mixed-content formats.

5.3.8 Usability Testing

Usability testing focuses on the human-centered design of the system, assessing how well the system meets the needs of educators and students. Feedback from

users on the clarity, usability, and educational value of the generated questions helps refine the system's interface, question quality, and overall user experience.

Testing Methodology:

- Educator Feedback: Teachers, examiners, and instructional designers were asked to interact with the system by uploading their own documents and reviewing the generated questions.
- Student Interaction: Students participated in testing by attempting to answer the generated questions to evaluate question clarity and cognitive appropriateness.
- Iterative Feedback Loop: Continuous feedback was used to refine the system's interface and enhance its ease of use.

Results and Observations:

- Educators appreciated the system's ability to quickly generate customized assessments, but requested more fine-grained control over question formatting and categorization.
- Students noted that the generated questions were often too difficult or required additional context to answer fully.
- The interface was well-received for its simplicity, though some users suggested adding features like question difficulty sliders and the ability to edit or rephrase questions before exporting.

5.3.9 Regression Testing:

After any system update or model retraining, regression testing is conducted to ensure that previous functionalities and performance metrics remain intact and that no new errors have been introduced.

5.3.10 Output Diversity Testing:

Diversity tests measure how varied the system's outputs are when provided with similar inputs. This prevents repetitive question structures and ensures a rich set of assessments.

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion:

This project presented the design and implementation of an AI-based Question Generation System capable of extracting educational content from academic PDFs and generating contextually relevant questions using advanced generative

language models such as GPT-4, LLaMA, and Mistral. The system architecture incorporates several critical stages including text extraction, cleaning, content segmentation, question generation, post-processing, and evaluation.The use of generative language models enabled the generation of diverse question types—ranging from multiple-choice and short-answer to descriptive and higher-order cognitive questions. These questions were further categorized based on Bloom's Taxonomy to ensure a balanced cognitive distribution in assessments. The implementation leveraged prompt engineering techniques, semantic segmentation, and optional answer generation to enhance the pedagogical relevance of the output.Comprehensive evaluation, both automatic (using BLEU, ROUGE, METEOR, etc.) and human-reviewed, demonstrated the system's effectiveness across a variety of domains. GPT-4 consistently outperformed other models in producing nuanced and high-quality questions, while LLaMA and Mistral provided cost-effective alternatives for local or offline environments. The system showed robustness under high-load scenarios and delivered consistent results across educational disciplines.Overall, the system successfully bridges the gap between raw educational content and automated assessment generation, paving the way for smarter, more personalized e-learning solutions.

6.2 Future Work

While the current system demonstrates promising results, several areas offer opportunities for further enhancement:

- Incorporation of Multimedia Content: Future versions could extract and interpret images, tables, and diagrams from PDFs to generate questions related to visual content.
- Fine-Tuning LLMs on Domain-Specific Data: Custom training or fine-tuning of language models on academic datasets from specific domains (like medicine, law, or engineering) could improve domain-specific accuracy and question quality.
- Adaptive Learning Paths: Integration of user feedback mechanisms can help generate personalized questions based on a learner's progress, enabling adaptive assessments.
- Real-Time Interaction via Chatbot Integration: Embedding the system into an interactive chatbot interface could allow students to request quizzes or ask for clarifications on topics in real time.
- Plagiarism Detection and Integrity Measures: Adding anti-cheating modules, such as paraphrasing checks or originality scoring, could enhance the system's integrity in exam settings.
- Multilingual Support: Expanding the system to support multiple languages would make it accessible to a wider user base globally and serve multilingual educational environments.
- Explainable AI Integration: Future enhancements could involve the integration of explainable AI techniques to offer rationales for generated questions and answers, improving transparency and trustworthiness.
- Mobile and LMS Integration: Developing mobile applications or plugins for Learning Management Systems (e.g., Moodle, Google Classroom) can improve usability and deployment flexibility.

REFERENCES

1. A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
2. T. Brown et al., "Language models are few-shot learners," in NeurIPS, vol. 33, pp. 1877–1901, 2020.
3. OpenAI, "GPT-4 Technical Report," 2023. [Online]. Available: https://openai.com/research/gpt-4. [Accessed: May 10, 2025].
4. H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.
5. Mistral AI, "Introducing Mistral 7B," 2023. [Online]. Available: https://mistral.ai. [Accessed: May 10, 2025].
6. J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers," in NAACL-HLT, 2019, pp. 4171–4186.

7. P. Rajpurkar et al., "SQuAD: 100,000+ questions for machine comprehension," in EMNLP, 2016.
8. L. W. Anderson and D. R. Krathwohl, A taxonomy for learning, teaching, and assessing, Longman, 2001.
9. C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in ACL Workshop, 2004, pp. 74–81.
10. A. Lavie and A. Agarwal, "METEOR: An automatic metric for MT evaluation," in ACL Workshop, 2007.
11. Y. Zhang, S. Sun, et al., "DIALOGPT: Large-scale generative pre-training for conversation," in ACL, 2020.
12. M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training," in ACL, 2020, pp. 7871–7880.
13. C. Raffel et al., "Exploring the limits of transfer learning with T5," JMLR, vol. 21, pp. 1–67, 2020.
14. T. Mikolov et al., "Distributed representations of words and phrases," in NeurIPS, vol. 26, 2013.
15. [Z. Yang et al., "XLNet: Generalized autoregressive pretraining," in NeurIPS, 2019.
16. Z. Lan et al., "ALBERT: A lite BERT for self-supervised learning," in ICLR, 2020.
17. S. Okoye, F. Agbo, and O. Uzoma, "A review on question generation," Education and Information Technologies, vol. 26, no. 6, pp. 7173–7197, 2021.
18. X. Yao and Y. Zhang, "Question generation with minimal recursion semantics," in EMNLP, 2010.
19. Y. Susanti, T. Akira, and T. Hiroya, "Automatic question generation using discourse structures," in BEA Workshop, 2015, pp. 64–74.
20. Y. Kim and K. Lee, "Learning to generate questions with no answer," in ACL, 2019.
21. LangChain Team, LangChain Documentation, 2023. [Online]. Available: https://docs.langchain.com. [Accessed: May 10, 2025].
22. PyMuPDF Developers, PyMuPDF Documentation, 2023. [Online]. Available: https://pymupdf.readthedocs.io. [Accessed: May 10, 2025]. [23] PDFMiner Team, "PDFMiner Documentation," 2023. [Online]. Available: https://pdfminersix.readthedocs.io. [Accessed: May 10, 2025].