IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AutoQPGen – Automatic Question Paper Generator

An Institution-wide Assessment Creation and Management Platform.

[1]Darshan S, [2]Chethan Gowda M V, [3]Prof. Manzoor Ahmed

[1,2]Undergraduate Students, [3]Assistant Professor

Department of Artificial Intelligence and Machine Learning

Sri Krishna Institute of Technology, Bangalore, Karnataka, India

Abstract: This project aims to streamline the traditionally manual and time-consuming process of creating question papers. Designed with educational institutions in mind, the system provides an intuitive interface for instructors to upload a question bank and generate question papers efficiently. The tool incorporates various features, including customizable templates, support for setting course outcomes, mapping questions to difficulty levels, and adhering to predefined marking schemes. The development leverages modern web technologies, employing a user-friendly frontend with a robust backend.

The system integrates multiple question formats, intelligent question selection for fairness, and document exports in PDF and DOCX. By automating the repetitive aspects of question paper design, the project reduces errors, saves time, and promotes standardization.

Index Terms **- AI in Education, AI-Driven Document Processing, Assessment Automation, Automated Question Paper Generation, NLP for Question Banks, Secure Exam System, PDF and DOCX Export, Difficulty-Level Balancing, Fair Question Selection, Educational Technology, Role-Based Access Control, Automated Exam Paper Creation, Text Categorization, ML for Question Selection, Question Randomization Algorithm, Customizable Exam Templates**

I. INTRODUCTION

1.1 Overview

Traditional question paper generation is a manual, time-consuming, and error-prone task. Educators must ensure syllabus coverage, difficulty balancing, and avoid repetition while designing assessments. These challenges hinder efficiency, fairness, and standardization in assessments.

AutoQPGen is an automated solution that leverages **Natural Language Processing (NLP)** and **AI-based algorithms** to generate question papers dynamically. The system extracts and categorizes questions from uploaded PDFs, ensuring even difficulty distribution and mapping to course outcomes. With **randomization techniques** and **template-based formatting**, AutoQPGen reduces manual effort, improves accuracy, and aligns with institutional examination standards.

1.2 Objectives

- Automate question paper creation with minimal manual effort.
- Ensure balanced difficulty distribution.
- Ensure proper syllabus coverage based on inputs.
- Support multiple question formats (MCQs, Short and Long answer questions).
- Prevent question repetition across assessments.

- Ensure Fairness and Unbiased Selection.
- Allow customization in question selection based on institutional requirements.
- Ensure Secure Storage and Access.
- Provide standardized assessment and role-based access control.

1.3 Problem Statement

Traditional methods of question paper generation are often manual, time-consuming, and prone to errors. Educators face challenges in ensuring comprehensive coverage of the syllabus, maintaining appropriate difficulty levels, and preventing question repetition across different examinations. Additionally, the manual process can lead to biases and inconsistencies, affecting the fairness and standardization of assessments. These limitations hinder the ability to create question papers that are both diverse and aligned with learning objectives, resulting in uneven evaluation of students' knowledge. Furthermore, as academic institutions grow and adopt outcome-based education frameworks, the demand for scalable and flexible solutions has intensified. There is a pressing need for an automated system that can address these challenges by providing a streamlined, efficient, and reliable solution for question paper generation, ensuring fairness and adaptability to dynamic educational needs.

1.4 Motivation

The motivation behind developing the Automated Question Paper Generation Model stems from the desire to enhance the educational assessment process. By automating question paper creation, educators can save valuable time and resources, allowing them to focus more on teaching and student engagement. Furthermore, an automated system ensures a fair and unbiased selection of questions, promoting a standardized assessment environment. The integration of technology in education, especially in assessment methodologies, is crucial for adapting to the evolving educational landscape and meeting the demands of modern learning environments. Additionally, it addresses the need for scalability in institutions managing multiple courses and large student populations. By embracing such innovations, educational institutions can improve the overall quality and efficiency of their academic processes.

II. LITERATURE SURVEY

2.1 Literature Review

- **Number of Papers:** 18 recent papers from well-reputed journals world-wide.
- **Quality of Literature:** All the reviewed papers are from popular journals, globally recognized publishers only.
- **Recency:** We have reviewed quality papers ranging from 2008 up to present.
- **Relevancy:** Papers have been particularly searched and selected based on the filters of relevancy as the main priority. We have carefully reviewed our papers to make sure that all the sources that we are referring from are as relevant as possible to our topic.
- **Understanding:** We have understood that our idea is not new and many very relevant research have been conducted before. By reviewing these papers, we have got a better understanding about how to approach our idea especially about implementation part.

2.2 Literature Analysis and Understanding

We realized that there are a lot of work done in this field before us and here are some of the research papers we have used as foundation for building our system:

1. **Questionator - Automated Question Generation using Deep Learning (2020).**

Authors: Animesh Srivastava, Shantanu Shinde, Naeem Patel, Siddhesh Despande, Anuj Dalvi, Shweta Tripathi

This research presents Questionator, a deep learning-based system for automated question generation. It employs Natural Language Processing (NLP) and image captioning techniques to generate questions not only from text but also from visual inputs. The system is capable of generating distractors (incorrect answer choices) to enhance question variety, making it useful for automated assessments and educational tools.

2. **Automatic Multiple Choice Question Generation from Text (2020).**
Authors: Dhawaleswar Rao CH, Sujan Kumar Saha

This study reviews existing systems for automatic multiple-choice question (MCQ) generation from text. It establishes a generic workflow for question generation, covering text preprocessing, keyword extraction, distractor generation, and quality assessment. The paper also highlights limitations in existing systems, such as the difficulty of ensuring question relevance and difficulty-level balancing.

3. **Secure Automatic Question Paper with Reconfigurable Constraints (2021).**
Authors: Ragasudha and M. Saravanan

This research focuses on security and randomization in automated question paper generation. The system incorporates reconfigurable constraints, such as question difficulty, syllabus coverage, and topic weightage, ensuring fair distribution of questions. Additionally, it employs encryption techniques to prevent unauthorized access to question papers.

4. **Automated Exam Question Generator using Genetic Algorithm (2017).**
Authors: Tengku Nurulhuda, Tengku Abd Rahim, Zalilah Abd Aziz, Rose Hafsah Ab Rauf, and Noratikah Shamsudin

This study applies Genetic Algorithms (GA) to automate exam question generation. The system optimizes question selection by ensuring diverse difficulty levels and topic coverage while preventing question repetition. It also discusses the adaptability of AI-based techniques for generating customized question papers.

5. **Computational Intelligence Framework for Automatic Quiz Question Generation (2018).**
Authors: Akhil Killawala, Igor Khokhlov, Leon Reznik

This paper presents an AI-based framework for automatic quiz question generation. The system integrates semantic analysis, difficulty prediction, and question ranking to ensure high-quality question selection. It also introduces a feedback mechanism to evaluate the effectiveness of generated questions and improve future iterations.

2.3 Summary of Understanding

Based on the reviewed papers, several key gaps in the field of Automatic Question Generation (AQG) have been identified. Many systems demonstrate proficiency in generating specific question types, such as multiple-choice questions, but often fail to extend this capability to "Wh"-questions (e.g., what, why, when) or open-ended questions requiring detailed responses. Additionally, the domain specificity of most AQG systems poses a significant limitation, as their performance tends to decline when applied to new or unfamiliar domains without extensive retraining on domain-specific data.

Accuracy remains a challenge, with limitations in answer selection, question formation, and overall coherence impacting the quality and utility of generated questions. Rule-based approaches also face difficulties in handling complex sentence structures, restricting their applicability to diverse text formats.

Furthermore, issues related to readability and natural language fluency persist, with some questions sounding unnatural or grammatically flawed, which undermines their effectiveness in educational or assessment contexts. Concerns regarding security and bias in AQG systems further highlight the need for advancements to address potential manipulation and ensure equitable question generation across varied user demographics.

III. System Requirement Specification

3.1 Functional Requirements

- User Authentication and Authorization
- Uploadable Question Bank
- Question Analysis and Categorization
- Dynamic Question Shuffling
- Template-Based Formatting
- Export Options

- Preview and Edit Functionality
- Storage and Retrieval
- Report Generation
- Error Logging
- Secure Data Handling

3.2 Non-Functional Requirements

- **Performance:** The system must process uploaded question banks and generate question papers within a reasonable time frame (e.g., less than 10 seconds for standard-sized inputs).
- **Scalability:** The system should be scalable to handle multiple users and larger question banks without significant performance degradation.
- **Usability:** The user interface must be intuitive, user-friendly, and accessible, ensuring ease of navigation and operation for educators with varying technical expertise.
- **Reliability:** The system should ensure high reliability, with minimal downtime and robust handling of edge cases or erroneous inputs.
- **Security:** The system must implement secure authentication mechanisms and encryption protocols to safeguard user data and question banks.
- **Maintainability:** The software should be modular and well-documented to facilitate easy updates, debugging, and addition of new features.
- **Portability:** The system should be deployable on various platforms (e.g., Windows, Linux, or cloud environments) without requiring significant modifications.
- **Compatibility:** The system must support integration with common file formats such as PDF and DOCX for input and output.
- **Data Integrity:** The system must ensure the integrity of uploaded data, maintaining accuracy during analysis, processing, and storage.
- **Error Handling and Recovery:** The system should include robust error handling to manage unexpected failures and mechanisms to recover data or resume interrupted operations.
- **Scalable Storage Solutions:** The system must provide scalable storage to accommodate growing data volumes, such as question banks and generated papers.
- **Aesthetic and Visual Design:** The application's design should align with modern standards, ensuring visually appealing layouts and responsive elements.
- **Logging and Monitoring:** The system should maintain detailed logs for auditing and debugging purposes and enable monitoring of usage patterns.

3.3 Software Requirements

- **Operating System:** Windows 10/11, macOS, or Linux distributions (e.g., Ubuntu 20.04 or later).
- **Programming Language:** Python 3.8 or higher for backend logic and processing.
- **Web Framework:** Flask for developing the backend server and managing web application routing.
- **Frontend Technologies:** HTML5, CSS3, and JavaScript for creating the user interface, with optional integration of libraries like Bootstrap for responsiveness.
- **Database Management System:** SQLite for storing user data, question bank metadata, and generated question papers.
- **Template Rendering:** Jinja2 for rendering dynamic web pages and templates.
- **Document Processing Libraries:** docxtpl for generating DOCX documents and docx2pdf for generating PDFs.
- **Machine Learning and Natural Language Processing Libraries:** SpaCy for text analysis and question classification, with optional integration of AI tools for advanced functionalities.
- **PDF Processing Tools:** PDFPlumber for extracting and processing text from uploaded PDF question banks.
- **Version Control:** Git for version control, with the project repository hosted on GitHub for collaboration and tracking changes.
- **Integrated Development Environment (IDE):** Visual Studio Code, PyCharm, or any text editor/IDE supporting Python development.

- **Dependencies Management:** pip for managing required Python packages, with dependencies listed in a requirements.txt file.
- **Web Browser:** Latest versions of browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge for accessing the web application.
- **Virtual Environment:** venv or conda for isolating project dependencies.
- **Logging and Debugging Tools:** Python's logging module for error logging and debugging during development and deployment.

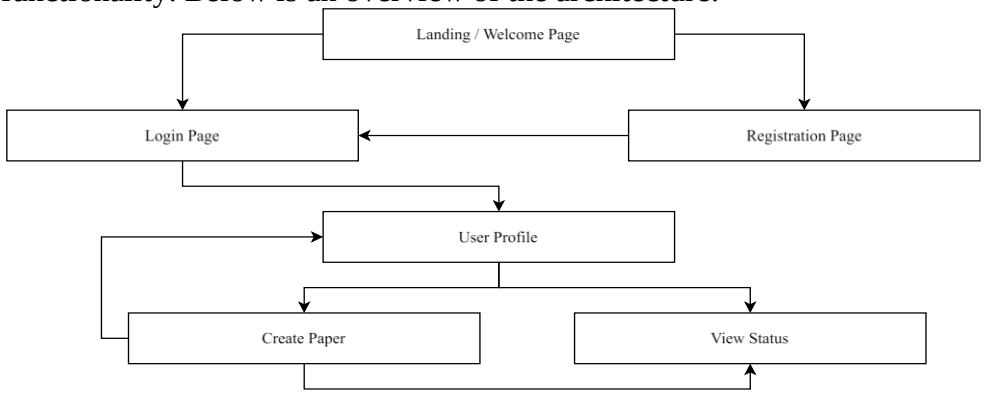
3.4 Hardware Requirements

- **Processor**
  - Minimum: Intel Core i5 (8th Generation) or AMD Ryzen 5 equivalent
  - Recommended: Intel Core i7 (10th Generation) or AMD Ryzen 7 equivalent for optimal performance.
- **RAM**
  - Minimum: 8 GB
  - Recommended: 16 GB for smoother multitasking and handling large datasets.
- **Storage**
  - Minimum: 256 GB SSD or HDD
  - Recommended: 512 GB SSD for faster read/write speeds and adequate space for storing project files and generated documents.
- **Display**
  - Resolution: 1366 x 768 or higher
  - Recommended: Full HD (1920 x 1080) or higher for a better development and user experience.
- **Graphics Card**
  - Not required for basic functionality
  - Recommended: Dedicated GPU (e.g., NVIDIA GTX 1050 or higher) for systems running additional graphical or AI tools.
- **Network**
  - Stable internet connection for accessing GitHub, downloading dependencies, and hosting the web application locally or on a cloud server.
- **Peripherals**
  - Standard keyboard and mouse
  - Optional: External monitor for enhanced productivity during development.
- **Server Requirements (if hosted)**
  - Minimum: 2 CPU cores, 4 GB RAM, and 50 GB storage for hosting the application on a cloud server.
  - Recommended: 4 CPU cores, 8 GB RAM, and 100 GB storage for handling multiple users simultaneously.

IV. System Design And Methodologies

4.1 System Architecture

The system architecture of the Automated Question Paper Generator follows a modular approach, ensuring that each component operates independently while contributing to the overall system functionality. Below is an overview of the architecture:

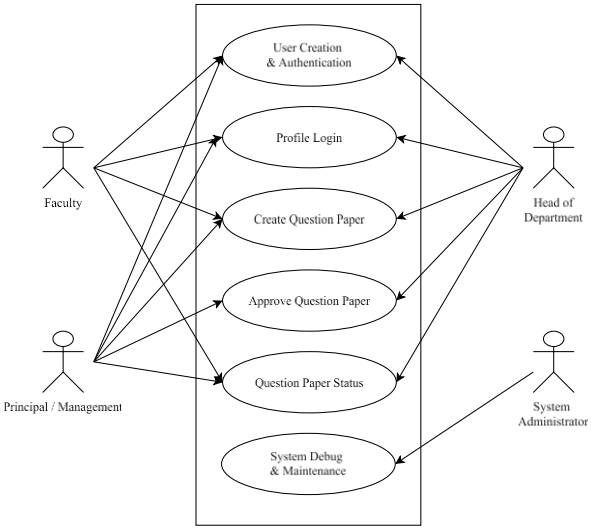


- **User Layer:**
  - **Educators** interact with the system through a web interface to upload question banks, customize question paper criteria, and download generated papers.
  - **Administrators** manage user roles, monitor system activity, and oversee question bank usage.
- **Application Layer:**
  - **Authentication Module:** Handles secure user login and registration, ensuring role-based access control.
  - **Question Bank Processing Module:** Processes uploaded question banks by extracting text, analysing content, and categorizing questions using natural language processing (NLP).
  - **Question Paper Generation Module:** Implements algorithms for random question selection, syllabus coverage validation, and template population.
  - **Document Export Module:** Generates question papers in DOCX and PDF formats using predefined templates.
  - **Error Handling and Logging Module:** Monitors system operations, logs errors, and provides user-friendly notifications.
- **Database Layer:**
  - **Database Management System (DBMS):** SQLite is used to handle user credentials, metadata, and generated documents. The database ensures secure and efficient data handling.
- **Frontend Layer:**
  - Developed using HTML, CSS, and JavaScript, with responsiveness achieved through Bootstrap.
  - Jinja2 templates dynamically render web pages based on user interactions and system processes.
- **Backend Layer:**
  - Flask is used as the web framework to handle routing, API integration, and backend processing.
  - Libraries such as PDFPlumber, SpaCy, docxtpl, and more are integrated to implement key functionalities.
- **Security and Access Control:**

  Security measures, including encryption and role-based access, are implemented to ensure data integrity and prevent unauthorized access to sensitive information.
- **Workflow Overview:**
  - Users upload question banks, which are processed to extract and categorize questions.
  - Customizable parameters are set for question paper generation.

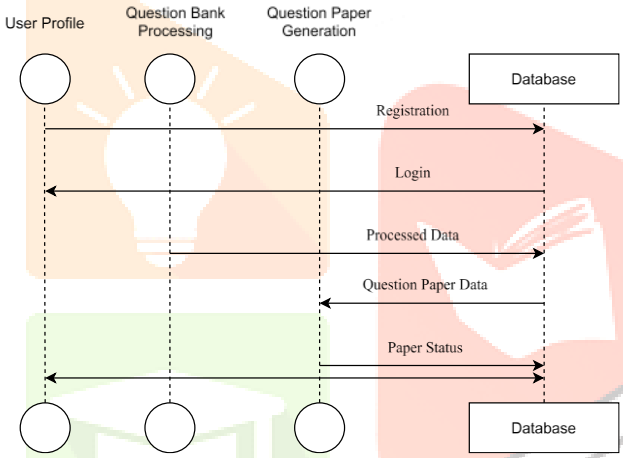
  The system dynamically generates the question paper and exports it in the desired format.

4.2 Use Case Diagram



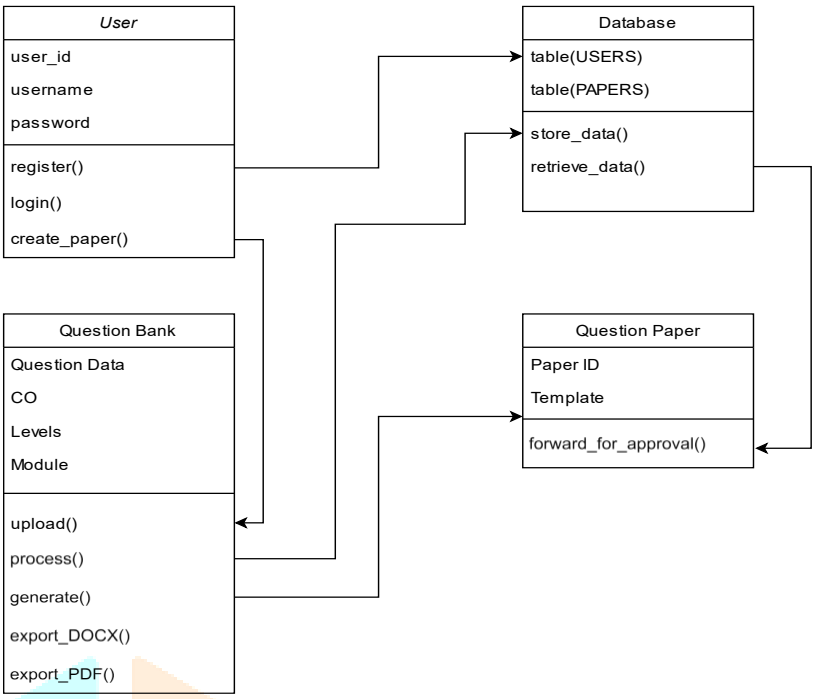
The following diagram illustrates the interaction between the users (educators and administrators) and the system. It highlights the system's core functionalities, including user authentication, question bank processing, dynamic question paper generation, and document export.

4.3 Sequence Diagram



The following sequence diagram represents the flow of interactions between the users (educators) and the core components of the AutoQPGen system. It demonstrates the step-by-step process of user authentication, question bank processing, question paper generation, and document export, ensuring a clear depiction of the system's operation.

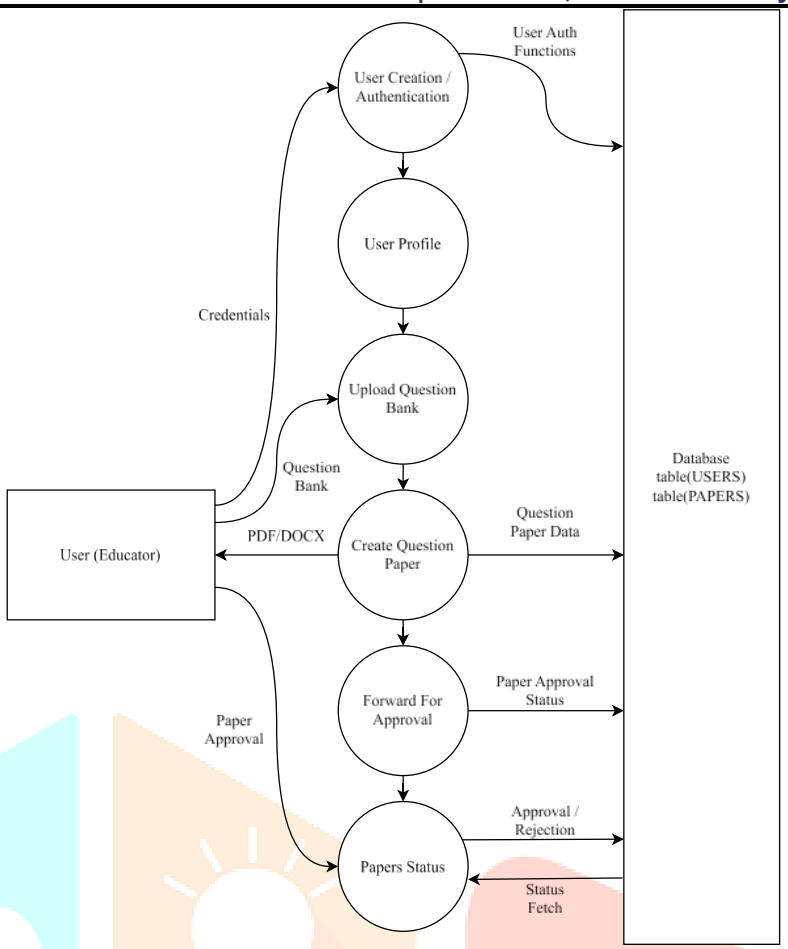
4.4 Class Diagram



The Class Diagram illustrates the static structure of the product system, detailing the system's key classes, their attributes, and methods. It also showcases the relationships between these classes, including associations and dependencies, providing a comprehensive view of the system's design.

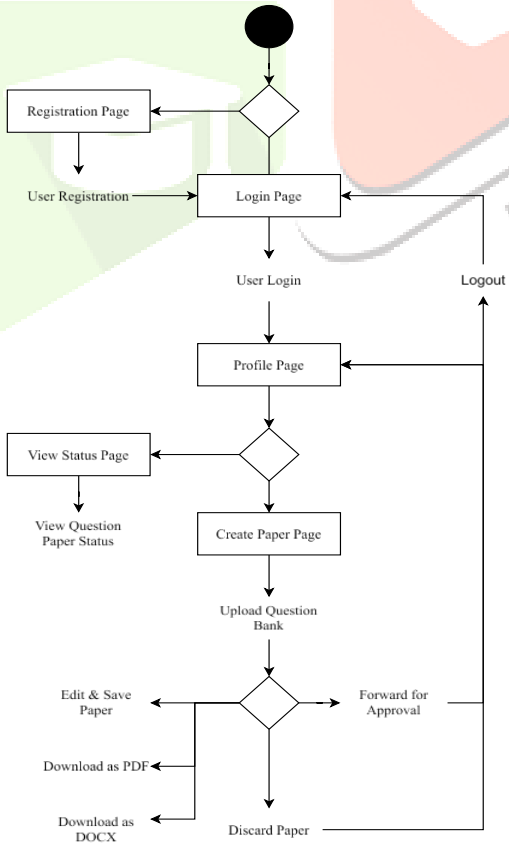
4.5 Data Flow Diagram

The Data Flow Diagram (DFD) depicts the movement of data within the AutoQPGen system, emphasizing interactions between processes, data stores, and external entities. User inputs, such as login credentials and uploaded question banks, flow into the system for authentication and processing. The processed data is categorized and stored in the database, which serves as a central repository for question bank metadata and generated question papers. Customization parameters provided by users guide the dynamic question paper generation process, and the finalized papers are exported as downloadable DOCX or PDF files. The flow ensures secure data handling and efficient processing across all modules.



4.6 Activity Diagram

The following Activity Diagram illustrates the step-by-step workflow of the system,



highlighting the sequence of activities from user authentication to question paper export.

V. IMPLEMENTATION

- **Process Methodology**

The implementation of the Automated Question Paper Generation Model followed a structured process methodology to ensure systematic development and deployment. The methodology adopted for this project emphasizes modular design and iterative refinement, ensuring each component functions independently while contributing to the overall system.

The system was deployed in a controlled environment, ensuring compatibility with various platforms. This methodology ensured that the Automated Question Paper Generation Model met all specified requirements, delivering a robust and scalable solution to streamline the question paper generation process.

- **Module Description**

1. **User Management Module**
   - Handles user authentication and authorization.
   - Provides secure login and registration functionalities with role-based access control (e.g., educator, administrator).
   - Stores user credentials and roles in the SQLite database.
2. **Question Bank Processing Module**
   - Facilitates the uploading of question banks in PDF format.
   - Extracts and processes text from PDFs using PDFPlumber for analysis.
   - Uses SpaCy's NLP capabilities to categorize questions based on course outcomes, difficulty levels, and modules.
3. **Question Paper Generation Module**
   - Dynamically generates question papers by selecting questions based on predefined criteria such as syllabus coverage, difficulty distribution, and course outcomes.
   - Incorporates randomization algorithms to prevent repetition and ensure unique question sets.
   - Supports customization options, allowing users to define question types and marks distribution.
4. **Document Export Module**
   - Generates question papers in both DOCX and PDF formats using docxtpl and docx2pdf.
   - Ensures adherence to predefined templates for consistent formatting.
   - Includes functionality for downloading and storing generated documents.
5. **Database Management Module**
   - Manages data storage for user information, uploaded question banks, and generated question papers.
   - Utilizes SQLite for efficient and lightweight database operations.
   - Ensures data integrity and secure access to stored records.
6. **Error Handling and Logging Module**
   - Captures and logs errors during system operations for debugging and maintenance purposes.
   - Provides real-time feedback to users in case of invalid inputs or processing failures.
7. **Web Interface Module**
   - Provides an intuitive and responsive interface using HTML, CSS, and JavaScript.
   - Implements dynamic rendering with Jinja2 templates for seamless user interaction.
   - Ensures accessibility across multiple devices and browsers.
8. **Administrative Dashboard Module**
   - Allows administrators to monitor system performance and manage user activities.
   - Provides insights into question bank usage, question paper generation statistics, and system health.

- **Algorithm**

Algorithm for Automated Question Paper Generation

1. **Input:**
   - Uploaded question bank in PDF format.
   - Parameters considered, including marks, Bloom's Taxonomy levels distribution, and modules.

2. **Preprocessing:**
   - Extract text from the uploaded PDF using PDFPlumber.
   - Clean and preprocess the text to remove unnecessary characters, numbers, or formatting issues.
   - Identify individual questions using regular expressions and NLP techniques.
3. **Question Categorization:**
   - Use SpaCy NLP to classify extracted questions into categories such as difficulty levels, course outcomes (COs), and modules.
   - Store the processed questions along with their metadata (e.g., CO, level, marks) in a structured format for further processing.
4. **Randomized Selection:**
   - Apply a randomization algorithm to select questions based on user-defined criteria:
     - Ensure balanced distribution of questions from both modules.
     - Avoid repetition of questions across sections or papers.
   - Validate the selected questions for syllabus coverage and module representation.
5. **Document Generation:**
   - Map the selected questions to predefined templates using the docxtpl library.
   - Populate placeholders in the template with questions, COs, marks, and additional metadata.
   - Generate a DOCX document and convert it to PDF format using docx2pdf when needed.
6. **Output:**
   - Provide the generated question paper for preview or forward for HOD approval.
   - Allow the user to download the question paper in the desired format (DOCX or PDF).
   - Save the document in the database for future retrieval.
7. **Error Handling and Logging:**
   - Log any errors encountered during the process, such as file format issues or invalid inputs.
   - Provide developer-friendly console event logs to identify errors and retry operations.

- **Key Code**

Here are some of the important code snippets:

➢ **QScanEngine.py** (Extracting Question Bank data and Export to DB)

```
def extract_info_with_ner(text):
    info = {
        "subjectName": None,
        "subjectCode": None,
        "semester": None,
        "facultyName": None,
        "qBankContent": []
    }
    subjectName = re.search(r"Subject Name\s*:\s*(.*)", text)
    subjectCode = re.search(r"Subject Code\s*:\s*(\S+)", text)
    semester = re.search(r"SEM\s*:\s*(\S+)", text)
    facultyName = re.search(r"Faculty\s*:\s*(.*)", text)

    # Populate metadata if found
    if subjectName:
        info["subjectName"] = subjectName.group(1).strip()
    if subjectCode:
        info["subjectCode"] = subjectCode.group(1).strip()
```

```
    if semester:
        info["semester"] = semester.group(1).strip()
    if facultyName:
        info["facultyName"] = facultyName.group(1).strip()
    questions = re.findall(r"(\d+)[\.\)]\s*(.*?)\s+CO(\d+)\s+L(\d+)\s+(\d+)\s+(\d+)", text,
re.DOTALL | re.IGNORECASE)
    for q_no, question, co, level, marks, modnum in questions:
        cleaned_question = clean_question_text(question)
        # rephrasedQuestion = GenAIRephraser(cleaned_question) # Question rephrasal using
GenerativeAI model
        info["qBankContent"].append({ # This creates a LIST OF DICTIONARIES
            "question": cleaned_question,
            "co": co,
            "level": level,
            "modnum": modnum
        })
    return info
```

- **DocxDownloader.py** (Embedding Paper Data into DOCX and Download)

```
def docxSaver(paperData):
    questionMarkersList = ('id', 'userId', 'paperId', 'status', 'cieNumber', 'departmentName',
'semester', 'courseName', 'electiveChoice', 'date', 'timings', 'courseCode', 'maxMarks',
'mandatoryCount', paperData)

    paperDataDictionary = dict(zip(questionMarkersList, paperData))
    courseCode = paperDataDictionary['courseCode']
    uniqueNameFactor = paperDataDictionary['paperId']
        templatePath = './static/DocxTemplates/QuestionPaperTemplate_10_5.docx'
    questionPaperPreName = courseCode + "__" + uniqueNameFactor + ".docx"
    questionPaperPreOutputPath = './static/GeneratedDocx/' + questionPaperPreName
        doc = DocxTemplate(templatePath)
    context = paperDataDictionary
    doc.render(context)
    doc.save(questionPaperPreOutputPath)
```

- **QpprEmbedderEngine.py** (Template and File Downloads)

```
def QpprExport(paperData):
    questionMarkersList = ('id', 'userId', 'paperId', 'status', 'cieNumber', 'departmentName',
'semester', 'courseName', 'electiveChoice', 'date', 'timings', 'courseCode', 'maxMarks',
'mandatoryCount', 'q1a', 'co1a', 'lvl1a', 'marks1a', 'module1a', 'q1b', 'co1b', 'lvl1b',
'marks1b', 'module1b', 'q2a', 'co2a', 'lvl2a', 'marks2a', 'module2a', 'q2b', 'co2b', 'lvl2b',
'marks2b', 'module2b', 'q3a', 'co3a', 'lvl3a', 'marks3a', 'module3a', 'q3b', 'co3b', 'lvl3b',
'marks3b', 'module3b', 'q4a', 'co4a', 'lvl4a', 'marks4a', 'module4a', 'q4b', 'co4b', 'lvl4b',
'marks4b', 'module4b')

    paperDataDictionary = dict(zip(questionMarkersList, paperData))
    courseCode = paperDataDictionary['courseCode']
    uniqueNameFactor = paperDataDictionary['paperId']

    templatePath = './static/DocxTemplates/QuestionPaperTemplate_10_5.docx'
    questionPaperPreName = courseCode + "__" + uniqueNameFactor + ".docx"
    questionPaperPreOutputPath = './static/GeneratedDocx/' + questionPaperPreName
    questionPaperName = courseCode + "__" + uniqueNameFactor + ".pdf"
    questionPaperOutputPath = './static/GeneratedPapers/' + questionPaperName

    doc = DocxTemplate(templatePath)
    context = paperDataDictionary
    doc.render(context)
```

doc.save(questionPaperPreOutputPath)

VI. SYSTEM TESTING

- **Unit Testing**
  - **Question Bank Processing Module**:
    - Functions for PDF text extraction (extract_text_from_pdf) were tested with various question banks, including structured and unstructured formats, to ensure accurate data retrieval.
    - The SpaCy-based NLP classification was validated for correctly categorizing questions by course outcomes (COs), difficulty levels, and modules.
  - **Dynamic Question Paper Generation Module**:
    - The random question selection algorithm was tested to prevent repetition and ensure balanced difficulty level distribution.
    - Functions like docxSaver and QpprExport were tested for proper template mapping and document generation in both DOCX and PDF formats.

The **User Management Module** was another critical focus area. Secure authentication and registration functionalities were tested to handle scenarios such as invalid login attempts and duplicate registrations. Tests ensured that user data was accurately stored in the SQLite database, and unauthorized access to restricted pages like create_paper.html was prevented.

- **Error Handling and Logging Module**:
  - Robust error messages were validated for unsupported file formats, invalid inputs, or missing data.
  - Logging mechanisms were tested to confirm the capture of significant events and errors for debugging purposes.

Additionally, test cases simulated edge scenarios such as processing malformed question banks or handling incorrect user input to verify system resilience. The results of unit testing revealed that all modules worked seamlessly both individually and in coordination, laying the groundwork for integration and system-level testing. This comprehensive approach ensured that the system met functional and non-functional requirements effectively.

- **Integration Testing**

Testing Key Module Interactions:

- **Question Bank Processing and Question Paper Generation:**
  - The integration between the extract_text_from_pdf function and the question selection algorithm was tested to confirm that categorized questions from the uploaded question bank were accurately passed to the generation module.
  - The randomization and selection logic were validated to ensure alignment with user-defined parameters such as difficulty levels and syllabus coverage.
- **Document Generation and Template Mapping:**
  - Integration of docxSaver and QpprExport with the dynamic question paper generation logic was tested to ensure correct mapping of questions, course outcomes (COs), and marks to predefined templates.
  - Generated DOCX and PDF documents were checked for formatting consistency and data accuracy.
- **User Authentication and Secure Access:**
  - Interactions between the user management system and other modules were tested to confirm secure access control. Logged-in users could upload question banks, generate question papers, and download documents, while unauthorized users were restricted from these actions.
- **Error Handling Across Modules:**

Integration testing also focused on error propagation and recovery mechanisms. For instance:

- If an unsupported PDF file was uploaded, the error was accurately reported without interrupting the flow of the application.
- Errors logged during document generation or question randomization were seamlessly communicated to users, maintaining system stability.

- **Data Storage and Retrieval Testing:**
  The interaction between the database and application logic was tested to ensure:
  - Proper storage and retrieval of question bank metadata, user credentials, and generated question papers.
  - No data loss or corruption occurred during module interactions.

By simulating real-world scenarios, integration testing verified that all modules, including user management, question bank processing, dynamic generation, and document export, operated cohesively. This ensured the reliability and robustness of the Automated Question Paper Generation Model as a unified system, ready for deployment and usage in academic settings.

- **System Testing**
  - **Testing Functional Requirements:**
    - **User Authentication and Role-Based Access:** Secure login and registration functionalities were verified for different roles, ensuring that unauthorized users could not access restricted pages like create_paper.html.
    - **Question Bank Processing:** The system accurately extracted and categorized questions from uploaded PDFs, handling different formats and structures seamlessly.
    - **Dynamic Question Paper Generation:** The paper generation process was tested for customization options, such as difficulty distribution, syllabus coverage, and question type selection.
    - **Document Export:** Generated question papers were validated in both DOCX and PDF formats for formatting accuracy and adherence to predefined templates.
  - **Testing Non-Functional Requirements:**
    - **Performance:** The system was tested under varying workloads, including large question banks and multiple simultaneous users, to ensure acceptable response times and no degradation in performance.
    - **Scalability:** The system's ability to handle increased data volumes and user load was validated through simulated high-traffic scenarios.
    - **Usability:** Feedback from educators during testing confirmed the intuitive design and ease of use of the interface.
  - **Error Handling and Recovery:**
    System testing included scenarios with invalid or corrupted question banks, missing inputs, and interrupted operations. The system consistently logged errors, provided clear notifications, and recovered gracefully without affecting the overall functionality.
  - **Cross-Browser and Platform Testing:**
    The system was tested on various web browsers (e.g., Chrome, Firefox, Edge) and platforms (e.g., Windows, macOS, Linux) to ensure compatibility and consistent performance.
  - **Security and Data Integrity:**
    - Authentication mechanisms were rigorously tested to prevent unauthorized access.
    - Data integrity checks ensured that uploaded question banks and generated question papers were stored and retrieved accurately without corruption.
  - **Final Validation:**
    System testing confirmed that the Automated Question Paper Generation Model delivered a robust, reliable, and user-friendly solution that met all specified requirements. The testing phase ensured that the system was ready for deployment, addressing both user expectations and operational efficiency.

- **Acceptance Testing**
  - **User-Centric Validation:**
    - **Functional Features:**
      - Educators were tasked with uploading question banks, generating question papers, and exporting them in DOCX and PDF formats.
      - The system's ability to categorize questions, apply difficulty-level distribution, and adhere to course outcomes (COs) was evaluated.
      - Role-based access control was verified by allowing users with different permissions to perform specific tasks, such as creating papers or managing user accounts.

- **Customization and Flexibility:**
  - Users tested the customization options available during question paper generation, such as selecting modules, question types, and marks distribution.
  - Feedback was collected to ensure that the system provided sufficient flexibility for different academic needs.

➢ **Performance and Reliability Validation:**
- The system was tested with large question banks to ensure stable performance and quick response times.
- Generated question papers were reviewed for accuracy, ensuring no repetition of questions and proper formatting.

➢ **Usability and Interface Testing:**
- Educators of varying technical expertise tested the system for usability, confirming that the interface was intuitive and user-friendly.
- The dynamic rendering of templates and the clarity of error messages were validated during testing sessions.

➢ **Stakeholder Sign-Off:**
- Feedback from educators and administrators indicated that the system met their requirements and expectations.
- Issues identified during acceptance testing were resolved promptly, ensuring that all stakeholders approved the system for deployment.

➢ **Real-World Scenario Testing:**

Simulated end-to-end workflows, including user registration, question bank uploads, paper generation, and document downloads, were conducted. This ensured that the system functioned seamlessly under conditions mirroring actual use.

- **Final Approval:**

Acceptance testing concluded successfully, with faculties and guide verifying that the Automated Question Paper Generation Model fulfilled all specified functional and non-functional requirements. The system was deemed ready for institutional use, meeting the goal of streamlining and automating the question paper generation process efficiently and reliably.

VII. RESULTS AND DISCUSSION

- **Results**

The Automated Question Paper Generation Model successfully achieves its primary objective of streamlining the process of question paper creation by automating critical tasks and minimizing manual effort. The following are the key results obtained from the implementation of the system:

➢ **Efficient Question Paper Generation:**
- The system enabled educators to generate question papers dynamically within seconds, adhering to predefined criteria such as syllabus coverage, difficulty levels, and course outcomes.
- Generated question papers were accurately formatted according to institutional templates, ensuring consistency and professionalism.

➢ **Accurate Text Processing:**
- The question bank processing module successfully extracted questions from various PDF formats, handling both structured and semi-structured files.
- Questions were categorized effectively based on course outcomes, difficulty levels, and modules using natural language processing (NLP) techniques.

➢ **Customizability and Flexibility:**
- The system allowed users to customize question papers by selecting specific topics, modules, and question types, providing educators with significant control over the content.
- Randomization algorithms ensured unique question paper versions without repetition, meeting the needs of diverse academic requirements.

- **Secure and Reliable Operation:**
  - Role-based access control and secure data handling ensured that the system was safe from unauthorized access and data breaches.
  - Robust error handling mechanisms provided clear feedback for invalid inputs or file formats, ensuring a smooth user experience.
- **Seamless Document Export:**
  - Question papers were exported successfully in both DOCX and PDF formats, with proper integration of metadata such as course codes, department names, and examination details.
  - The generated documents were compatible with commonly used word processors, ensuring ease of distribution and printing.
- **Scalability and Performance:**
  - The system demonstrated excellent scalability, efficiently processing large question banks and managing multiple simultaneous users without performance degradation.

These results highlight the system's effectiveness in addressing the challenges of traditional question paper generation and its potential for widespread adoption in educational institutions.

- **Summary**

The results of the Automated Question Paper Generation Model were presented, demonstrating the system's ability to automate and optimize the process of question paper creation. The system successfully met its objectives by efficiently extracting and categorizing questions, generating papers based on predefined criteria, and exporting them in multiple formats. Through rigorous testing and validation, the system proved to be reliable, scalable, and user-friendly, addressing the challenges associated with traditional methods.

Snapshots provided in the preceding section showcased the core functionalities, including question bank uploads, question categorization, dynamic paper generation, and document export. These visual representations highlight the seamless integration of various modules and the intuitive design of the user interface, ensuring an enhanced experience for educators.

In conclusion, the Automated Question Paper Generation Model offers a robust and efficient solution for academic institutions, significantly reducing the time and effort involved in question paper preparation. The results validate the system's potential to transform the assessment process, aligning with the goals of modern, technology-driven education.

VIII. CONCLUSION AND ACKNOWLEDGEMENT

The **Automated Question Paper Generation Model (AutoQPGen)** was designed to overcome the inefficiencies of traditional question paper creation by leveraging **Natural Language Processing (NLP) and automation**. By integrating **advanced algorithms for question classification, randomization, and syllabus coverage**, the system ensures fairness, accuracy, and efficiency in the assessment process. This project demonstrates how **technology can modernize education** by reducing manual effort and minimizing human errors while maintaining institutional standards. With a user-friendly interface and structured workflow, AutoQPGen enables educators to focus more on student engagement and curriculum delivery rather than administrative tasks.

A key feature of this system is its ability to **support multiple question formats**, including **MCQs, Wh-questions, and essay-type questions**, providing flexibility for different evaluation methods. The **randomization algorithm prevents question repetition**, ensuring uniqueness across exams. Additionally, the system supports **secure role-based access and document export in DOCX and PDF formats**, aligning with common academic practices. Through **comprehensive testing and iterative refinements**, the project successfully addressed challenges such as handling **unstructured question banks and compatibility with various document templates**, resulting in a robust and adaptable solution.

We would like to express our **sincere gratitude** to **Sri Krishna Institute of Technology** for providing the resources and support necessary for this project. We extend our appreciation to **Prof. Manzoor Ahmed** for his valuable guidance and expertise throughout the development process. We are also thankful to our faculty members and peers for their feedback, which played a crucial role in refining the system. Looking ahead, future enhancements such as **AI-based question generation, multilingual support, and advanced analytics for syllabus mapping** will further expand the capabilities of AutoQPGen, making it an essential tool for academic institutions worldwide.

REFERENCES

[1] OpenAI, "Applications of Natural Language Processing in Automated Systems," [Online]. Available: https://openai.com/

[2] "Questionator - Automated Question Generation using Deep Learning" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/9077721

[3] "Automatic Multiple Choice Question Generation from Text" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/8585151

[4] "Android based exam paper generator (Android based E-PAGE)" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/8398926

[5] "Computational Intelligence Framework for Automatic Quiz Question Generation" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/8491624

[6] "Design of Question Answering System with Automated Question Generation" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/4624170

[7] "Automatic Fill-the-blank Question Generator for Student Self-assessment" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/7344291

[8] "Automatic Generation of Question Paper from User Entered Specifications using a Semantically Tagged Question Repository" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/7814813/

[9] "Automatic Question Generation and Evaluation" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/10029559/

[10] "Automated Question Paper Generation Model" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/10580391/

[11] "Secure Automatic Question Paper with Reconfigurable Constraints" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/9419410/

[12] "Automated Exam Question Generator using Genetic Algorithm" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/8409231/

[13] "A Review on Automated Examination Question Paper Template Generator" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/9068704/

[14] "A Systematic Review of Automatic Question Generation for Educational Purposes" [ResearchGate]: https://www.researchgate.net/publication/337433098_A_Systematic_Review_of_Automatic_Question_Generation_for_Educational_Purposes

[15] "Automated Question Generation Tool for Structured Data" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/7275833

[16] "Automatic question-answer pairs generation and question similarity mechanism in question answering system" [ResearchGate]: https://www.researchgate.net/publication/350707864_Automatic_question-answer_pairs_generation_and_question_similarity_mechanism_in_question_answering_system

[17] "User-defined Difficulty Levels for Automated Question Generation" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/7814689

[18] "Thematic Question Generation over Knowledge Bases" [IEEE XPLORE]: https://ieeexplore.ieee.org/document/8609569/

[19] "Automatic Opinion Question Generation" [ACL Anthology]: https://aclanthology.org/W18-6518

[20] SpaCy Documentation, "Natural Language Processing with SpaCy," [Online]. Available: https://spacy.io/

[21] PDFPlumber Documentation, "Plumb a PDF for detailed structured analysis." [Online]. Available: https://github.com/jsvine/pdfplumber

[22] docxtpl Documentation, "Using docxtpl for Template-based Document Generation," [Online]. Available: https://docxtpl.readthedocs.io/

[23] Flask Documentation, "Flask Web Framework," [Online]. Available: https://flask.palletsprojects.com/

[24] SQLite Documentation, "SQLite: A C Library that Implements an SQL Database Engine," [Online]. Available: https://sqlite.org/

[25] "Challenges in Automated Question Generation: A Survey," [ResearchGate]. Available:https://www.researchgate.net/publication/342742873_Challenges_in_target_selection_for_automatic_question_generation