MCQ Generator using AI Flask

NLP Project

Abdul Choudhari (A029)
Anil Vhatkar (A004)

Introduction

The project is a web application designed to automate the generation of multiple-choice questions (MCQs) from text documents using Google Generative AI. This tool is particularly beneficial for educators, students, and developers who require a quick and efficient way to create assessment materials based on existing content.

The application supports various file formats, including PDF, DOCX, and TXT, allowing users to upload their documents seamlessly. Upon uploading a file, users can specify the number of MCQs they wish to generate. The application processes the text extracted from the document and utilizes advanced AI capabilities to formulate clear and structured MCQs.

The core functionality of the application is implemented in a Python script using the Flask framework. This script manages file uploads, text extraction, and communication with the Google Generative AI API to generate questions. The generated MCQs are then presented to users in a user-friendly format, with options to download them as text or PDF files. The project consists of three main components:

1. **Backend Logic (app.py)**: This Python script handles all backend operations, including file handling, text extraction from documents, generating MCQs using AI, and serving the generated content back to the user.
2. **User Interface (index.html)**: The front-end HTML file provides an intuitive interface for users to upload their documents and specify their requirements for MCQ generation.
3. **Results Display (results.html)**: After processing the uploaded document, this HTML file displays the generated MCQs along with download options in both text and PDF formats.

By integrating these components, the application streamlines the process of creating educational assessments, making it easier for users to generate high-quality questions quickly and effectively.

OBJECTIVE

The primary objective of this project is to develop a web application that automates the generation of multiple-choice questions (MCQs) from text documents using advanced AI technology, specifically Google Generative AI. This tool aims to streamline the process of creating educational assessments, making it easier for educators, students, and developers to produce high-quality questions quickly and efficiently.

Key Objectives:

1. **Automate MCQ Generation:**
   Utilize AI to automatically generate MCQs from various text formats (PDF, DOCX, TXT), reducing the manual effort required in question creation.

2. **Support Multiple File Formats:**
   Allow users to upload documents in different formats, ensuring flexibility and accessibility for various users.

3. **User-Friendly Interface:**
   Provide an intuitive web interface that enables users to easily upload files, specify the number of questions they wish to generate, and download the results.

4. **Text Extraction and Processing:**
   Implement robust text extraction capabilities that accurately retrieve content from uploaded documents, ensuring that the generated questions are relevant and contextually appropriate.

5. **Downloadable Results:**
   Offer generated MCQs in both text and PDF formats for easy distribution and use in educational settings.

6. **Enhance Educational Resources:**
   Support educators by providing a quick way to create assessments, quizzes, or study materials based on existing content, thereby improving teaching efficiency and effectiveness.

7. **Leverage AI Technology:**
   Integrate Google Generative AI to enhance the quality of generated questions, ensuring they are well-structured and varied in difficulty.

   By achieving these objectives, the project not only simplifies the process of creating educational assessments but also enhances the overall learning experience by providing educators with valuable tools to engage students effectively.

API Key

An API key is a unique identifier used to authenticate requests made to an API (Application Programming Interface). In the context of this project, the API key is essential for accessing Google Generative AI, which generates multiple-choice questions (MCQs) from uploaded text documents.

How to Obtain an API Key

To use Google Generative AI, you need to obtain an API key by following these steps:

1. **Create a Google Cloud Project**:
    - Go to the Google Cloud Console.
    - Click on "Select a project" and then "New Project" to create a new project.
2. **Enable the Generative Language API**:
    - In your project dashboard, navigate to "APIs & Services" > "Library".
    - Search for "Generative Language API" and click on it.
    - Click "Enable" to activate the API for your project.
3. **Create Credentials:**
    - Go to "APIs & Services" > "Credentials".
    - Click on "+ Create Credentials" and select "API Key".
    - Your new API key will be generated. Copy this key for use in your application.
4. **Set Up Billing (if required):**
    - Depending on your usage, you may need to set up billing for your Google Cloud project. Follow the prompts in the console to link a payment method if necessary.
5. **Secure Your API Key:**
    - Store your API key securely and avoid exposing it in public repositories or shared codebases.

Working of the Project

The project is a web application that automates the generation of multiple-choice questions (MCQs) from text documents using Google Generative AI. Below is a detailed explanation of how the application works, focusing on the app.py file and the two HTML files (index.html and results.html).

1. Backend Logic (app.py)

The app.py file contains the core logic of the application, implemented using the Flask framework. Here's how it works:

- **Configuration and Setup**:
  - The application initializes Flask and sets up configurations for file uploads and allowed file types.
  - It also configures the Google API key required for accessing Google Generative AI.

```
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads/'
app.config['RESULTS_FOLDER'] = 'results/'
app.config['ALLOWED_EXTENSIONS'] = {'pdf', 'txt', 'docx'}
```

- **File Handling Functions**:
  - The function allowed_file(filename) checks if the uploaded file has an acceptable extension (PDF, TXT, DOCX).
  - The function extract_text_from_file(file_path) extracts text from the uploaded document based on its format. It uses pdfplumber for PDFs, python-docx for DOCX files, and standard file reading for TXT files.

```python
def extract_text_from_file(file_path):  1 usage
    ext = file_path.rsplit('.', 1)[1].lower()
    if ext == 'pdf':
        with pdfplumber.open(file_path) as pdf:
            return ''.join([page.extract_text() for page in pdf.pages])
    elif ext == 'docx':
        doc = docx.Document(file_path)
        return ' '.join([para.text for para in doc.paragraphs])
    elif ext == 'txt':
        with open(file_path, 'r') as file:
            return file.read()
    return None
```

- **MCQ Generation**:
    - Thefunction Question_mcqs_generator(input_text,num_questions) c onstructs a prompt for Google Generative AI to generate MCQs based on the extracted text. It specifies that each question should have four answer options and indicate the correct answer.

```python
def Question_mcqs_generator(input_text, num_questions):
    prompt = f"""
    You are an AI assistant helping the user generate multiple-choice questions (MCQs) based on the following text:
    '{input_text}'
    Please generate {num_questions} MCQs from the text. Each question should have:
    - A clear question
    - Four answer options (labeled A, B, C, D)
    - The correct answer clearly indicated

    Format:
    ## MCQ
    Question: [question]
    A) [option A]
    B) [option B]
    C) [option C]
    D) [option D]
    Correct Answer: [correct option]
    """

    response = model.generate_content(prompt).text.strip()
    return response
```

- **File Saving**:
    - The generated MCQs are saved in both TXT and PDF formats using save_mcqs_to_file(mcqs,filename) and create_pdf(mcqs, filename) functions.

```python
def save_mcqs_to_file(mcqs, filename):
    results_path = os.path.join(app.config['RESULTS_FOLDER'], filename)
    with open(results_path, 'w') as f:
        f.write(mcqs)
    return results_path
```

- **Flask Routes**:
    - The application defines several routes:
        - The root route (/) renders the index.html page.
        - The /generate route handles form submissions, processes uploaded files, generates MCQs, and renders the results page.
        - The /download/<filename> route allows users to download generated MCQs.

```python
@app.route('/')
def index():
    return render_template('index.html')


@app.route('/generate', methods=['POST'])
def generate_mcqs():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']
```

- **Application Initialization**:
    - At the end of the script, it checks if necessary directories exist and starts the Flask server.

```python
if __name__ == "__main__":
    if not os.path.exists(app.config['UPLOAD_FOLDER']):
        os.makedirs(app.config['UPLOAD_FOLDER'])

    if not os.path.exists(app.config['RESULTS_FOLDER']):
        os.makedirs(app.config['RESULTS_FOLDER'])

    app.run(debug=True)
```

1. **User Interface (index.html)**:
    - Provides a form for users to upload their documents and specify the number of questions they want to generate. It includes:
        - A file input for document uploads.
        - An input field for the number of MCQs desired.
        - A button to submit the form.
2. **Results Display (results.html)**:
    - Displays the generated MCQs after processing. It formats each question with its answer options and indicates the correct answer. Users can download the results in both text and PDF formats.

How It Works

- **User Interaction**:
    1. Users visit the homepage (index.html), upload a document, and specify how many MCQs they want.
    2. Upon submission, the application processes the file, extracts text, and generates MCQs using AI.
- **MCQ Generation:**
    1. The application constructs a prompt for Google Generative AI to create structured MCQs based on the provided text.
- **Results Presentation:**
    1. After generating the questions, users are redirected to results.html, where they can view their MCQs and download them in their preferred format.

CONCLUSION:

The project successfully demonstrates the integration of Google Generative AI with a Flask web application to automate the generation of multiple-choice questions (MCQs) from various text documents. By allowing users to upload files in PDF, DOCX, and TXT formats, the application streamlines the process of creating educational assessments. Users can specify the number of questions they wish to generate, and the AI efficiently produces well-structured MCQs that include clear questions and answer options. This functionality not only saves time for educators but also enhances the quality of learning materials available for students.

Overall, this project highlights the potential of leveraging artificial intelligence in educational settings. By simplifying the question-generation process, it empowers educators and students alike to focus more on learning and teaching rather than on manual content creation. The ability to download generated MCQs in both text and PDF formats further adds to the application's usability, making it a valuable tool for anyone involved in education or training.

SEE DEMO OF MCQ GENERATES VIDEO