</ Multiple Choice Question Generator />

} /> [

Team

Mehak, Shivanshi, Ajinkya, Virti

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Introduction

What is it?

A tool that automatically generates multiple-choice questions (MCQs) from uploaded document.

Key Features:

- Upload documents (PDF, text file)
- Automatically create contextually relevant MCQs
- Provide interactive quiz interfaces

Objective:

Simplify the process of creating quizzes, saving time and effort for educators and learners.

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Problem Statement

Challenges Faced

- Manual MCQ creation is time-consuming, error-prone, and limits scalability.
- Relies heavily on manual efforts, causing inefficiency, bias, and errors.
- Increasing demand for automated MCQ generation in e-learning and assessments.

Solution Created

- Automated system for generating accurate, diverse, and high-quality MCQs.
- Reduces manual effort and enhances efficiency.
- Supports scalability for online education and assessments.

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Application Scope

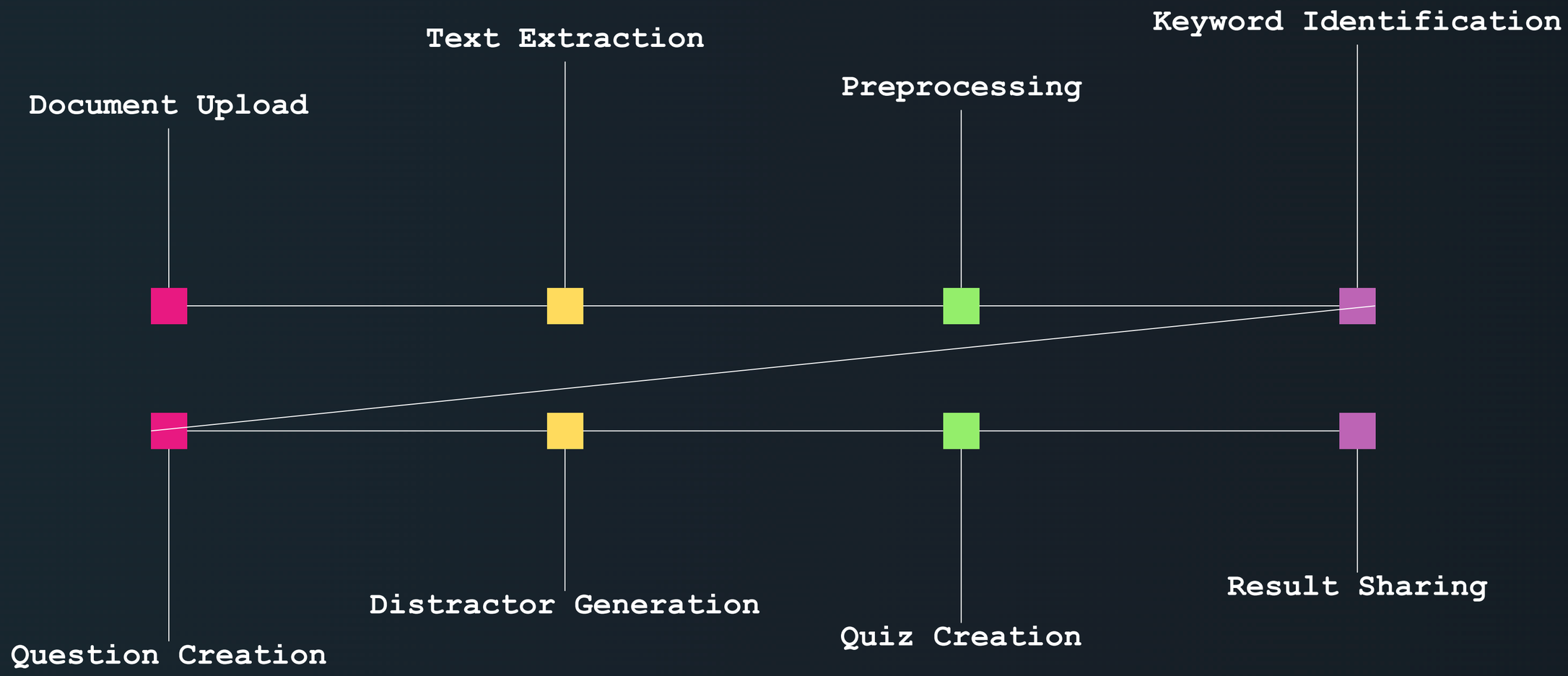
Who Can Use It? How Can They Use It?

- **Education:**
  Automates MCQ creation for schools, universities, and online courses.
- **Corporate Training:**
  Simplifies quiz generation for employee training and certifications.
- **E-Learning Platforms:**
  Provides adaptive and dynamic quizzes for personalized learning.
- **Content Creators:**
  Assists in creating practice tests and supplemental quizzes.
- **Research & Development:**
  Enables analysis of question quality and enhances AI tools.
- **Gamification:**
  Powers engaging educational games and trivia apps.

}])

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Workflow of the MCQ Generator

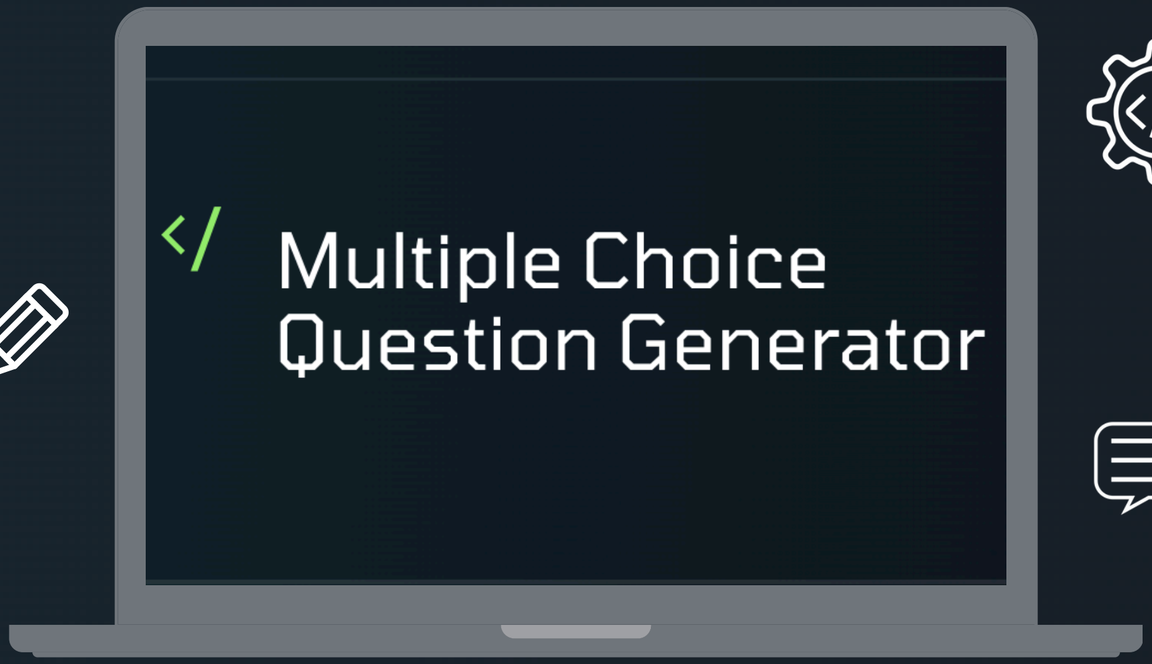


1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Project Structure

Frontend

- HTML
- CSS
- JavaScript
- Bootstrap
- Flask



Backend

- Python
- Flask

NLP Tools

- PyPDF2
- spaCy

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</>

Demo

http://127.0.0.1:8000

} /> [

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Implementation

```
mcqs = []
while len(mcqs) < num_questions and len(sentences) > 0:
    sentence = sentences.pop(random.randint(0, len(sentences) - 1))  # Randomly select a sentence
    sent_doc = nlp(sentence)

    # Identify key nouns, verbs, and adjectives to replace for creating the blank
    nouns = [token.text for token in sent_doc if token.pos_ == "NOUN"]
    verbs = [token.text for token in sent_doc if token.pos_ == "VERB"]
    adjectives = [token.text for token in sent_doc if token.pos_ == "ADJ"]
```

Dependancy Parsing

Part Of Speech (POS) Tagging

```
try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    raise Exception("Please run 'python -m spacy download en_core_web_sm' to install the spaCy model.")

def generate_mcqs(text, num_questions=5):
    if not text:
        return []

    doc = nlp(text)
    sentences = [sent.text for sent in doc.sents]
```

Sentence Tokenization

```
question_stem = sentence.replace(subject, "_____")

# Gather possible distractors (nouns, verbs, adjectives from the sentence)
distractors = list(set(nouns + verbs + adjectives) - {subject})
```

Rule-Based Word Replacement

Distractor Generation (Word Filtering)

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Future Work

Category	Planned Improvement
Question Quality	- Use advanced NLP models for better keyword extraction and distractor generation
Input Options	- Enable image and scanned document support with OCR integration
User Features	- Integrating Named Entity Recognition (NER) Models for domain specific questions & adaptive dificulty levels
Platform Integration	- Integrate with EdTech platforms like Moodle or Google Classroom
Scalability	- Deploy the application on cloud platforms for large-scale use

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</ Thank You />

} /> [

Any Questions?

1011 011 01 1011001 10 11011 011 01 110110 110111 1101