Project Title

SmartQuizzer – Adaptive AI-Based Quiz Generator

🎯 Skills Takeaway From This Project

- Python Scripting
- Natural Language Processing (NLP)
- Large Language Models (LLMs)
- Prompt Engineering
- Data Preprocessing
- Streamlit Web Application
- MongoDB / Local JSON Storage
- Machine Learning (Difficulty Classification)
- Data Visualization (Matplotlib, Plotly)
- API Integration (OpenAI / HuggingFace)

🌍 Domain

- Education Technology
- E-Learning & Assessment Systems
- Artificial Intelligence & NLP
- Adaptive Learning Platforms

📝 Problem Statement

This project aims to build an AI-powered adaptive quiz generator capable of analyzing study materials and automatically generating relevant quiz questions. Using NLP and LLMs, the system creates questions, assigns difficulty levels, adapts to the user's performance in real-time, and provides insightful analytics through a Streamlit application.

The key objectives are to:

1. Use an LLM (GPT / Llama / T5) to understand and extract key concepts from uploaded text or PDF.
2. Automatically generate quiz questions (MCQ, True/False, Short Answer) along with answers and distractors.
3. Implement difficulty classification and develop an adaptive engine that adjusts question difficulty based on user performance.
4. Build a Streamlit web application where users can upload documents, take adaptive quizzes, and view detailed analytics.
5. Store generated questions and user attempts in MongoDB or JSON storage for reusability.
6. Visualize performance metrics such as accuracy, difficulty progression, weak topics, and response-time analysis.
7. Provide personalized learning recommendations using the analytics dashboard.

📂 Data / Input Description:

User-Provided Input

- PDF files
- Text files
- Typed or pasted notes
- Study materials or chapters
- Online articles (optional integration)

Example Data Structure (LLM Output):

```
{
  "question": "What does AI simulate?",
  "answer": "Human intelligence",
  "distractors": ["Machine behavior", "Animal instincts", "Natural processes"],
  "difficulty": "easy",
  "topic": "Artificial Intelligence"
```

}

🛠️ Approach

1️⃣Document Upload & Text Extraction

- Use Streamlit to allow users to upload PDFs or text files.
- Extract clean text using pdfplumber or direct text extraction.
- Preprocess text (tokenization, cleaning) for LLM readiness.

2️⃣LLM Content Understanding

- Use LLM prompts to extract:
  - Key concepts
  - Important sentences
  - Definitions
  - Topic summaries
- Store these as seed points for Question Generation.

3️⃣AI-Based Question Generation

- Use GPT/Llama/T5 models to generate:
  - MCQs
  - Short answer questions
  - True/False
  - Fill-in-the-blank
- Generate distractors using LLM reasoning.
- Ensure output is valid JSON for easy processing.

4️⃣Difficulty Classification

- Each question is tagged as:
    - **Easy**
    - **Medium**
    - **Hard**

Using:

- LLM classification prompt
  OR
- A simple ML classifier based on text features

5️⃣ Adaptive Engine Implementation

- Start with Medium difficulty questions.
- If the user answers correctly → Increase difficulty.
- If the user answers incorrectly → Decrease difficulty.
- Maintain session history to personalize the quiz path.

6️⃣ Interactive Streamlit Web Application

Develop an interactive interface with:

- Document upload
- Question preview
- Quiz with options
- Timer, score, and progress bar
- Next question selection using adaptive logic
- Real-time feedback

7️⃣ Performance Analytics & Visualization

Visualize:

- Total score
- Accuracy percentage
- Topic-wise performance
- Difficulty progression over time
- Response time per question
- Recommendations (e.g., “Revise Machine Learning Fundamentals”)

Using tools:

- Plotly
- Matplotlib

8️⃣ Optional: Database Integration

- Store generated questions in MongoDB Atlas or JSON file.
- Store user attempts and analytics for personalization.

9️⃣ Dashboard Creation (Optional Enhancement)

Use Power BI or Tableau to create:

- Topic-level performance charts
- Difficulty distributions
- User progress reports
- Drill-down dashboards for deeper insights

🎓 Learning Outcomes

1. NLP & LLM Integration

Learn how to use GPT/Llama/T5 for:

- Concept extraction
- Question generation

- Difficulty classification

2. Prompt Engineering

Design structured prompts for QG, distractors, and topic mappings.

3. Python Data Processing

Work with:

- Pandas
- Regex
- Text preprocessing pipelines

4. Streamlit App Development

Build interactive UI:

- File upload
- Dynamic quiz
- Session state
- Visual analytics

5. Adaptive Learning Systems

Understand:

- Difficulty modeling
- Reinforcement-like logic
- Personalized question sequencing

6. Data Visualization

Create clear and interactive visualizations using Plotly/PowerBI.

7. Database Handling

Connect to MongoDB Atlas for storing user data and generated content.

8. Application Deployment

Deploy Streamlit app to:

- Streamlit Cloud

- AWS EC2
- HuggingFace Spaces

9. End-to-End Project Development

Gain skills in modular coding, architecture planning, GitHub version control.

10. Communication & Presentation

Explain AI-driven quiz systems and present insights effectively.

🧮 Project Evaluation Metrics

Your project will be evaluated on:

✔ Code Quality & Modularization

Functions, clean architecture, reusable components.

✔ Maintainability

Readable, documented, follows PEP-8.

✔ Portability

Runs consistently on different systems (Windows/Linux/Mac).

✔ GitHub Repository

Must be public with:

- Code
- README
- Project Workflow
- Screenshots

✔ Documentation

README must include:

- Project description
- Workflow
- Setup instructions

- Usage Guide

✔ **Working Demo Video**

Upload to LinkedIn (Mandatory).

✔ **Analysis Quality**

Depth of insights and meaningful visuals.

✔ **UI & User Experience**

Streamlit app should be simple, responsive, and interactive.