**AI Test Generation Bot Documentation**

***Project Overview:***

The AI Test Generation Bot is designed to process PDF files, extract relevant information, generate embeddings, and create multiple-choice questions based on the content. The project uses technologies such as TensorFlow Hub, PDFMiner, and ReportLab for PDF processing, embedding generation, and PDF export, respectively.

***Steps***

**Step 1: Setting Up the Environment**

Ensure that you have the necessary dependencies installed. Key dependencies include TensorFlow, TensorFlow Hub, PDFMiner, and ReportLab.

**Step 2: Importing Libraries**

*pip install tensorflow tensorflow-hub pdfminer.six reportlab*

**Step 3: Extracting Text from PDFs**

*import asyncio*

*from pdfminer.high\_level import extract\_text*

*import tensorflow\_hub as hub*

*from reportlab.pdfgen import canvas*

*import random*

**Step 4: Generating Embeddings**

*async def extract\_text\_from\_pdf(pdf\_path):*

*text = await asyncio.to\_thread(extract\_text, pdf\_path)*

*return text*

**Step 5: Generating Multiple-Choice Questions**

*def generate\_multiple\_choice\_question(embeddings, num\_choices=4, bias\_mitigation=True):*

*return question, choices*

**Step 6: User Interface**

*def user\_interface():*

*extracted\_texts = loop.run\_until\_complete(process\_pdfs(pdf\_paths))*

*for extracted\_text in extracted\_texts:*

*embeddings = generate\_embeddings(extracted\_text)*

*questions = []*

*for \_ in range(num\_questions):*

*question, choices = generate\_multiple\_choice\_question(embeddings, bias\_mitigation=apply\_bias\_mitigation)*

*questions.append((question, choices))*

*export\_to\_pdf\_multi(questions, difficulty\_level)*

*print(f"\nTest exported to 'generated\_test.pdf'.")*

**Step 7: Exporting Questions to PDF**

*def export\_to\_pdf\_multi(questions, difficulty\_level, pdf\_path=r"C:\Users\bardw\Desktop\PDF Processing\generated\_test.pdf"):*

*c = canvas.Canvas(pdf\_path)*

*c.save()*

**Step 8: Scalability and Architecture**

Consider architectural designs that can handle large volumes of PDF files efficiently. Optimize the code for scalability, ensuring it can process multiple files seamlessly.

**Step 9: Evaluation Criteria**

Evaluate the AI Test Generation Bot based on the following criteria:

Accuracy of PDF processing and information extraction.

Quality of generated embeddings, capturing key concepts and relationships.

Relevance, clarity, and grammatical correctness of generated test questions.

Diversity of question types and difficulty levels.

Usability and intuitiveness of the user interface.

Ability to handle different types of content through PDF.

**Step 10: Documentation**

Document the entire project, including setup instructions, code snippets, and explanations for each step. Include details on dependencies, libraries used, and the overall project structure.

***Conclusion***

The AI Test Generation Bot project provides a robust solution for generating tests from PDF content. The system extracts information, generates embeddings, and creates diverse and relevant multiple-choice questions. Scalability considerations and an evaluation framework ensure the project's effectiveness and usability.

Feel free to customize this documentation according to your specific project details and add any additional information that might be relevant for users or developers interacting with your AI Test Generation Bot.

**Summary Report**

Approach Taken

PDF Processing: Used PDFMiner to extract text from PDF files asynchronously.

Embedding Generation: Utilized TensorFlow Hub for generating embeddings using a pre-trained Universal Sentence Encoder.

Question Generation: Developed a function to create multiple-choice questions based on embeddings with an option for bias mitigation.

User Interface: Built a simple command-line interface for user interaction.

**Challenges Faced**

TensorFlow Compatibility: Faced compatibility issues with TensorFlow versions. Resolved by ensuring the appropriate version is installed.

PDF Text Extraction: Handling variations in PDF formats required adjusting the extraction process.

**Suggestions for Improvement**

Enhanced Bias Mitigation: Implement more sophisticated bias mitigation strategies for diverse and fair question generation.

User Interface Improvement: Develop a graphical user interface (GUI) for better usability.

Scalability: Optimize the code for scalability, allowing the application to efficiently process a large number of PDF files.