Project Documentation for EdutainAI

1. Project Overview

Project Name: EdutainAI

Description:
EdutainAI is a Flask-based web application that allows users to upload documents (PDF, DOCX, TXT) and generate multiple-choice questions (MCQs) using AI. The application includes authentication and authorization using MongoDB, ensuring that only registered users can access the main functionalities. Additionally, it features 3D character models that provide an interactive learning experience.

2. Setup Instructions

Prerequisites:

- Python 3.7+
- MongoDB
- Pip (Python package installer)

Installation Steps:

1. **Clone the Repository:**
    - Clone the project repository to your local machine.
2. **Create a Virtual Environment:**
    - Set up a virtual environment for the project to manage dependencies.
3. **Install Dependencies:**
    - Install all required Python packages listed in requirements.txt.
4. **Set Up Environment Variables:**
    - Configure necessary environment variables including the Google API key.
5. **Run the Flask Application:**
    - Start the Flask development server.

3. Usage Guide

Login and Signup:

- **Login:** Users can log in through the /login page using their username and password.
- **Signup:** New users can create an account via the /signup page by providing a username and password.

Generating MCQs:

- **Upload Document:** Users can upload a document (PDF, DOCX, TXT) on the main page after logging in.

- **Specify Details:** Users need to specify the number of questions and select a character for the interactive experience.
- **Generate MCQs:** The AI generates MCQs based on the uploaded document.
- **Download Results:** Users can download the generated MCQs in TXT or PDF format.

4. Functionality Description

File Upload and Text Extraction:

- **Supported Formats:** The application supports PDF, DOCX, and TXT file formats.
- **Text Extraction:** Text is extracted from uploaded documents using specialized libraries for each format:
    - pdfplumber for PDFs
    - python-docx for DOCX files
    - Standard file reading for TXT files

MCQ Generation:

- **AI Integration:** Utilizes Google Generative AI to generate multiple-choice questions from the extracted text.
- **Prompt Configuration:** The AI is provided with a prompt to generate clear questions with four answer options, and the correct answer is indicated.

PDF and TXT File Generation:

- **Result Saving:** Generated MCQs are saved as TXT files.
- **PDF Conversion:** TXT files are converted to PDF format for easy distribution.

3D Character Models:

- **Interactive Experience:** Users can select from various 3D character models to enhance their learning experience.
- **Character Models:** The application includes characters such as Goku, Spider-Man, Halo Reach, Captain America, and Doremon.
- **Model Viewing:** Models are displayed using the model-viewer library, providing an interactive 3D experience.

5. API Integration Details

Google Generative AI:

- **API Key Configuration:** The AI service is configured using an API key set in the environment variables.
- **Content Generation:** Generates text-based content based on the provided prompt for MCQ creation.

6. Authentication and Authorization

User Authentication:

- **User Management:** User credentials are stored in MongoDB.
- **Password Hashing:** Passwords are securely hashed using werkzeug.security with pbkdf2:sha256.

Session Management:

- **Flask Sessions:** User sessions are managed using Flask sessions.
- **Protected Routes:** Routes that require authentication are protected, ensuring only logged-in users can access them.

7. Error Handling and Logging

Error Handling:

- **Custom Error Messages:** Provides custom error messages for invalid file formats, missing files, and other exceptions.
- **File Not Found Handling:** Ensures proper handling of file not found errors and other exceptions.

Logging:

- **Basic Logging:** Basic logging is implemented for debugging purposes. This can be extended as needed.

8. Directory Structure

Code

```
├── app.py
├── templates
|   ├── index.html
|   ├── login.html
|   ├── signup.html
|   ├── results.html
├── static
|   └── (static files if any)
├── uploads
|   └── (uploaded files)
├── results
|   └── (generated MCQ files)
├── models
```

| └── (3D models)

└── requirements.txt

9. Dependencies

Python Packages:

- **Flask:** Web framework for building the application.
- **pymongo:** Integration with MongoDB for user authentication.
- **pdfplumber:** PDF text extraction library.
- **python-docx:** DOCX text extraction library.
- **FPDF:** Library for generating PDF files.
- **Werkzeug:** Security and utility functions.
- **google.generativeai:** Integration with Google Generative AI for content generation.

10. Future Enhancements

Feature Enhancements:

- Add support for additional file formats.
- Improve AI prompts for generating better MCQs.

UI/UX Improvements:

- Enhance the frontend design for a better user experience.
- Provide real-time feedback to users.

Performance Optimization:

- Optimize text extraction and MCQ generation processes.
- Implement caching for frequently performed operations.