🧠 AI-Powered Study Assistant

(Retrieval-Augmented Generation with Gemini 1.5 Flash)

🔍 Overview

The **AI-Powered Study Assistant** is a smart, interactive education platform that transforms **academic PDFs into study notes, quizzes, and personalized feedback** using the power of **Google Gemini 1.5 Flash**, OCR, and **Retrieval-Augmented Generation (RAG)**.

This project is crafted to assist **students, educators, and e-learning platforms** by automating tedious tasks like note-taking, question creation, and answer evaluation — directly from documents.

🧰 Features Overview

Feature	Description
📄 **Document Ingestion**	Upload any academic PDF (text or scanned)
🔎 **OCR Text Extraction**	Uses **Gemini** to extract text from scanned PDFs
📚 **Note Generator (RAG)**	Converts documents into structured, bullet-point notes using RAG
❓ **Auto Quiz Creator**	Generates **conceptual MCQs, True/False, Short Answer** questions
✍️ **Student Quiz Interface**	Interactive quiz UI for students to enter answers
🧠 **Feedback Generator**	Auto-grades answers and gives **contextual, constructive feedback**
💬 **Chat with PDF**	Chat-like interaction with document content using vector search
📤 **Download Notes & Feedback**	Export notes or feedback as **PDF or Word (DOCX)**

🎓 Designed For

📌 *"Built for students, optimized for tutors, and scalable for educational tech."*

- Self-learners and university students
- Educators needing summarized materials
- LMS platforms wanting AI integrations
- Tutors and training providers seeking auto-assessment tools

---

🚀 Project Highlights

- ✅ Seamless RAG workflow (retrieval + Gemini content generation)
- 📎 Supports scanned PDFs with vision-based OCR
- 📚 Educational formatting and prompt engineering tailored to document types
- 🧪 Dynamic evaluation logic with token-aware context limits
- 💾 Persistent vector search using Chroma or FAISS
- 📄 Export-ready study packs in PDF and DOCX formats

🧪 Demo Use Case

Imagine uploading your **case study, lecture slides**, or **research paper**. Within seconds:

✔️ Receive **high-quality notes**
✔️ Generate a **quiz with diverse formats**
✔️ Let students take the quiz
✔️ Get **AI-based scoring and feedback**
✔️ Export results instantly

🏗️ Project Architecture

```
📦 project_root/
|
├── main.py              → Streamlit app entry point
├── requirements.txt       → All dependencies
├── .env                 → API keys and environment configs
|
├── agents/              → Intelligent modules
|   ├── chat_agent.py        → Chat with document agent
|   ├── notes_agent.py        → RAG-based note generation
|   ├── qa_agent.py          → Quiz generation agent
|   ├── feedback_agent.py      → Feedback + evaluation logic
|   └── pdf_viewer_agent.py    → Embedded PDF viewer for Streamlit
|
├── utils/               → Supporting utilities
|   ├── pdf_utils.py         → OCR, text splitting, export (PDF/DOCX)
|   ├── vision_ocr.py        → OCR using Gemini Vision / Nanonets
|   ├── chat_utils.py        → Contextual document QA utilities
|   └── vector_utils.py      → Chroma / FAISS vector management
|
├── chroma_index/          → Persistent vector DB storage
└── Fonts/               → Unicode fonts (DejaVu) for PDF exports
```

💡 Core Technologies

Capability	Tools & Libraries
OCR	Gemini, PyMuPDF, pdf2image

Capability	Tools & Libraries
Language Model	Gemini 1.5 Flash via google.generativeai
Embeddings & Vectors	GoogleGenerativeAIEmbeddings, Chroma, FAISS
RAG Framework	LangChain - RetrievalQA, Chains
UI Framework	Streamlit for interactivity
File Export	FPDF, python-docx

📌 RAG Strategy Breakdown

RAG (Retrieval-Augmented Generation) enhances response quality by **grounding prompts in real data**.

📘 Here's how it's applied:

1. PDF is split into chunks using RecursiveCharacterTextSplitter
2. Each chunk is embedded using GoogleGenerativeAIEmbeddings
3. Stored in **Chroma/FAISS** vector DB
4. On user action, **semantic search** retrieves top k relevant chunks
5. Prompt is built with Gemini and **only valid chunk tokens**
6. Response (notes, quiz, feedback) is generated

🧑‍🏫 Document-Aware Prompting

Each document type uses **specialized prompt templates** to match context:

Doc Type	Prompt Customization
Study Topic	Headings, real-world applications, definitions
Case Study	Challenges, stakeholders, decisions, lessons
Research Paper	Methods, findings, limitations, implications
Lecture Notes	Topics, examples, summaries
Other	General thematic notes

💬 Chat with PDF

Users can interactively query content, ask follow-ups, and clarify sections of the document using RAG-powered semantic search with Gemini Flash.

🧾 Output Formats

- 📄 Study Notes (Streamlit + export to DOCX or PDF)
- ❓ Quiz in structured format (with student UI)
- 🧠 Auto Feedback in fixed rubric (score + suggestions)
- 🧾 Downloadable results (PDF/DOCX)

🛠️ Setup Instructions

🔧 1. Clone the repository

```
git clone https://github.com/SMJajoo/RAG-Based-Student-Tutor-App.git

cd RAG-Based-Student-Tutor-App
```

🧪 2. Create a virtual environment and install dependencies

```
python -m venv ed_venv

source ed_venv/bin/activate  # Windows: ed_venv\Scripts\activate

pip install -r requirements.txt
```

🔑 3. Set up .env file

```
GOOGLE_API_KEY=your_google_api_key
```

▶️ 4. Run the application

```
streamlit run main.py
```

📤 Deployment Options

Platform	Status
Streamlit Cloud	✅ Supported

Platform	Status
Render / Fly.io	✅ Compatible
Dockerized Deployment	🟡 (Optional setup)

📊 Limitations

- ⚠️ Gemini's token quota (free tier) may cause 429 errors under load
- 🧩 Quiz quality depends on document clarity and token context

---

🧩 Future Enhancements

- Token-aware chunk overflow handling
- Gemini Pro model support

🧑‍💻 Author & Maintainer

Sanskruti Jajoo
📧 LinkedIn Profile
💼 *AI for Education | RAG Systems | LangChain | Gemini 1.5*

📄 License

MIT License — Free for personal, academic, and open-source use. Attribution appreciated.
