Campus Buddy Chatbot (RAG Lite) - Hackathon Solution

1. Problem Statement

Build a chatbot that answers college FAQs (departments, clubs, library rules, canteen timings, bus routes) using a small dataset (PDF/CSV). Must-have features: upload documents → ask questions → show the source snippet used.

2. Solution Overview (RAG Lite Architecture)

- Document Upload (PDF/CSV)
- Text Extraction and Chunking
- Embedding Generation
- Vector Storage (FAISS / ChromaDB)
- User Query Embedding
- Similarity Search (Top-K retrieval)
- LLM Answer Generation
- Display Answer + Source Snippet

3. Suggested Tech Stack

- Backend: Python (FastAPI / Flask)
- Frontend: Streamlit or simple HTML + JS
- Embeddings: OpenAI / HuggingFace
- Vector DB: FAISS or ChromaDB
- PDF Parsing: PyPDF2 or pdfplumber
- CSV Handling: Pandas

4. Implementation Steps

- Step 1: Upload and parse PDF/CSV documents.
- Step 2: Split text into chunks (300–500 words).

- Step 3: Generate embeddings for each chunk.
- Step 4: Store embeddings in a vector database.
- Step 5: Convert user question into embedding.
- Step 6: Retrieve top 3 most relevant chunks.
- Step 7: Send retrieved chunks + question to LLM.
- Step 8: Display generated answer with source snippet.

5. Sample RAG Lite Code (Python - Simplified)

```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings
from langchain.chains import RetrievalQA
from langchain.chat_models import ChatOpenAI

# Load document
loader = PyPDFLoader("college_info.pdf")
documents = loader.load()

# Split text
splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=50)
docs = splitter.split_documents(documents)

# Create embeddings & vector store
embeddings = OpenAIEmbeddings()
vectorstore = FAISS.from_documents(docs, embeddings)

# Create QA chain
qa = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(),
    retriever=vectorstore.as_retriever(search_kwargs={"k": 3})
)

# Ask question
query = "What are the library timings?"
result = qa.run(query)
print(result)
```

6. Evaluation Criteria & Bonus Ideas

- Accuracy of answers
- Proper source snippet display
- Clean UI/UX
- Fast retrieval time
- Bonus: Multi-document support
- Bonus: Role-based access (student/admin)
- Bonus: Deploy on Render/Streamlit Cloud