CODE 4 GovTech
Building Digital Public Goods

Protean Plus Insight Chatbot

C4GT DMP - Proposal Template

Name	Aman Kumar Srivastava
Email ID	aman.apk01@gmail.com
Phone Number	7739704188
GitHub ID	https://github.com/Iconic-Aman
Discord ID	1181166175221198901
Current occupation *(Working Professionals - add current organization & years of exp)*	Working Professional - ML intern at Unified Mentor (2 months experience)
Education Details *(College Name - Degree Name and branch of engineering or other course/specialization)*	Indian Institute of Information Technology, Kottayam B.Tech (Computer Science and Engineering)
Technical skills with level *(Mention tech skills/languages known/UI-UX and level - Novice/Intermediate/Expert)*	Python - Expert Java- Intermediate Git/Github - Intermediate TensorFlow- Expert FastAPI/Flask - Novice Linux - Intermediate GCP/AWS - Novice MySQL - Intermediate

Summary:

This project aims to build a multilingual AI-powered chatbot that allows decision-makers to access key business insights. The chatbot will enable users to interact with business metrics in English and multiple Indian languages like Hindi and Bengali. By combining LLMs, vector databases and real-time dashboard, the chatbot will provide fast, contextual and accurate responses related to business KPIs.
My approach involves designing a scalable backend using python, REST APIs and open-source tools for LLM integration. This AI-powered chatbot will support natural language queries for core business metrics and dashboard will provide visualizations and alerting features.
Having previously built a chatbot using a RAG pipeline with Ollama(LLaMA 2), I have planned to apply similar methods in this project focusing on contextual retrieval. As a 2024 graduate passionate in machine learning, I aim to contribute to this project by developing modular components that align with enterprise needs. This open-source project will create a long-term impact within the Indian enterprise ecosystem.

Project Detail:

1. Project Overview:

Understanding of the project:

This project aims to develop a multilingual chatbot interface for Protean Plus, enabling business users to retrieve KPIs and insights using natural language. The chatbot will act as a bridge(mediator) between business related data and Indian people, allowing them to interact with enterprise data just by writing a simple query in their own language.

Key requirements include:

- Multilingual NLP Support: The chatbot must understand English and at least two Indian languages like Hindi and Bengali.
- Vector Database Integration: Utilize open-source vector stores like Milvus or Weaviate to retrieve relevant context based on user queries.
- Natural Language for Query: Map user queries to dynamically generate responses using a Retrieval-Augmented Generation pipeline.
- Data Source Connectivity: Fetch KPI data from various internal sources eg. finance sales via different APIs calls.
- Security and Access Control: Implement Role-Based Access Control to ensure data privacy .
- Insight Visualization Dashboard: Provide a React-based dashboard for users to view data charts.

Problems (if any):

Query Understanding:

- Handling variations in user phrasing and business context requires a strong NLP pipeline.

Multilingual NLP:

- Translating queries and ensuring semantic understanding across multiple Indian languages can be difficult without proper tokenization models.

Data Integration:

- Connecting to heterogeneous data sources may require different authentication.

Vector Store Selection:

- Due to different working of vector databases based on performance and cost might create problems to attach with the model.

UI Expectations:

- Lack of clarity in dashboard layout and chatbot might create problems for users and difficult to align frontend with backend capabilities.

Security Requirements:

- Implementing enterprise-grade RBAC and secure logging practises from scratch is a challenge for a fresher.

Solutions:

Query Understanding:

- Implement a RAG pipeline using LLaMA2 or any open source LLM with domain-specific embeddings and chunking strategies.

Multilingual NLP:

- Use pre-trained Indic models such as IndicNLP or MuRIL(specifically designed for Indian Languages) with translation related APIs.

Data Integration:

- Start training with dummy KPI datasets and JSON APIs first and then move on to the business related dataset

Vector Store Selection:

- Benchmark Weaviate and Milvus vector database on sampel enterprise data and choose the one with optimal latency and ease of integration with React dashboard.

UI Expectations:

- Coordinate closely with Protean's design team for wireframes and test with simple Streamlit or Gradio UI before moving to React .

Security Requirements:

- Follow best practices such as using encrypted storage, JWT(JSON Web Token) via Djago's OAuth2 libraries.

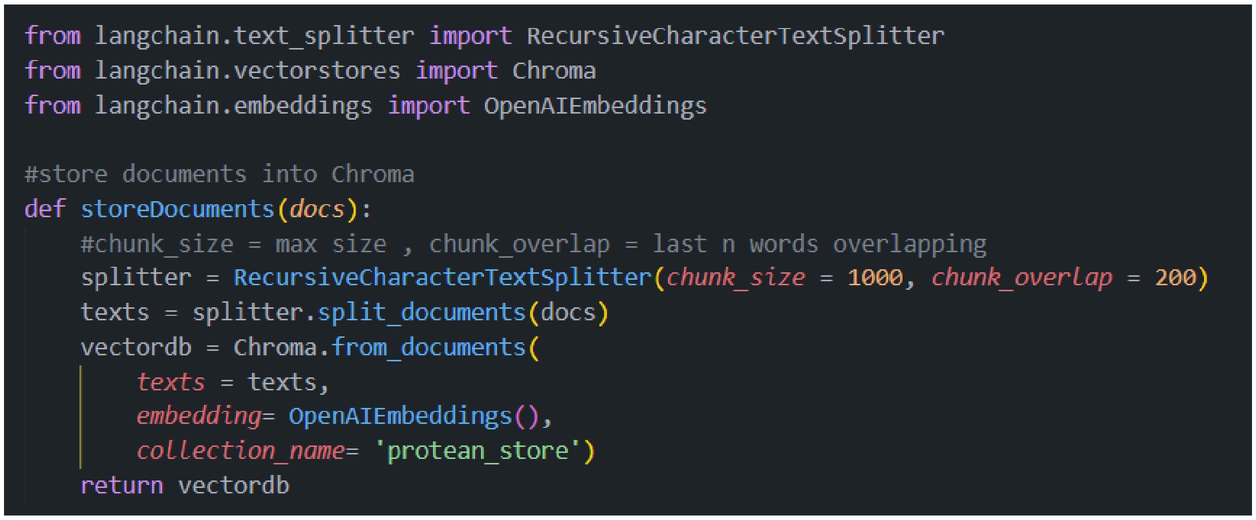
2. Macro Implementation Details with Timelines:

[Milestone 1:RAG Architecture Design & Core Implementation(Weeks 1-4)]

Deliverables:

- Define system architecture (LLM + Vector Store + API layer).
- Choose tech stack: FastAPI, LangChain, Ollama(Free)/OpenAI(paid), Chroma/Milvus.
- Implement document upload, text chunking and embedding.
- Set up retrieval + generation(RAG) response pipeline.

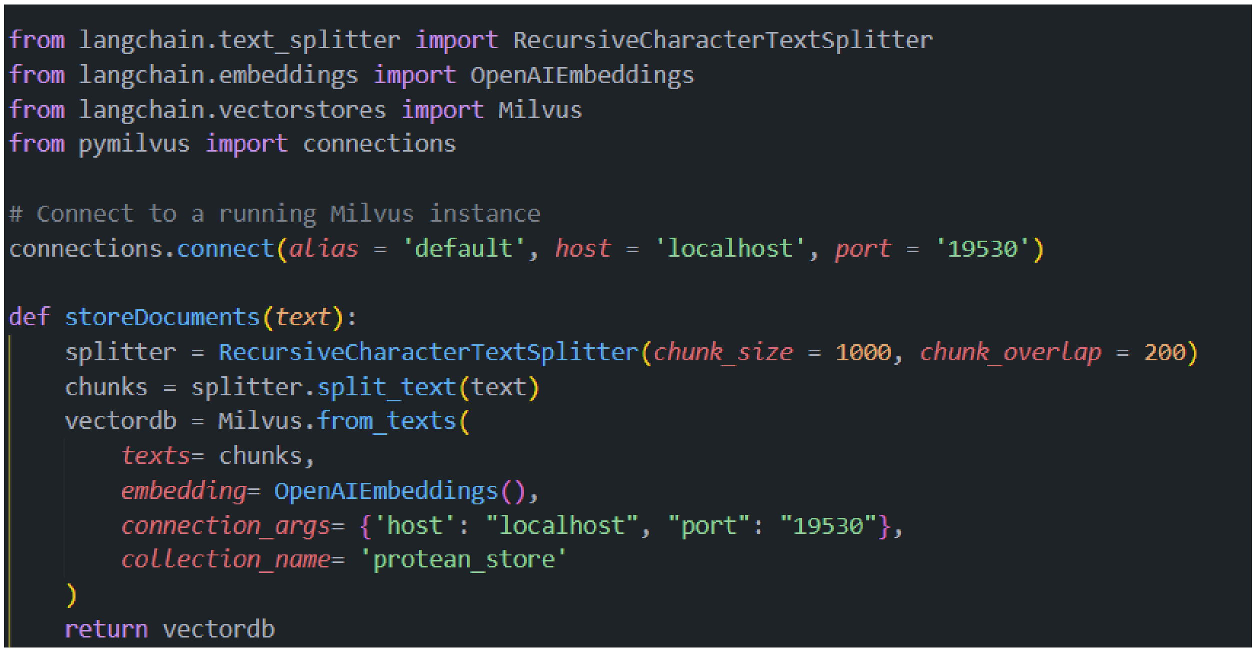
Option 1: Using Chroma for Vector Storage(Simple Local Setup)



```python
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings

#store documents into Chroma
def storeDocuments(docs):
    #chunk_size = max size , chunk_overlap = last n words overlapping
    splitter = RecursiveCharacterTextSplitter(chunk_size = 1000, chunk_overlap = 200)
    texts = splitter.split_documents(docs)
    vectordb = Chroma.from_documents(
        texts = texts,
        embedding= OpenAIEmbeddings(),
        collection_name= 'protean_store')
    return vectordb
```

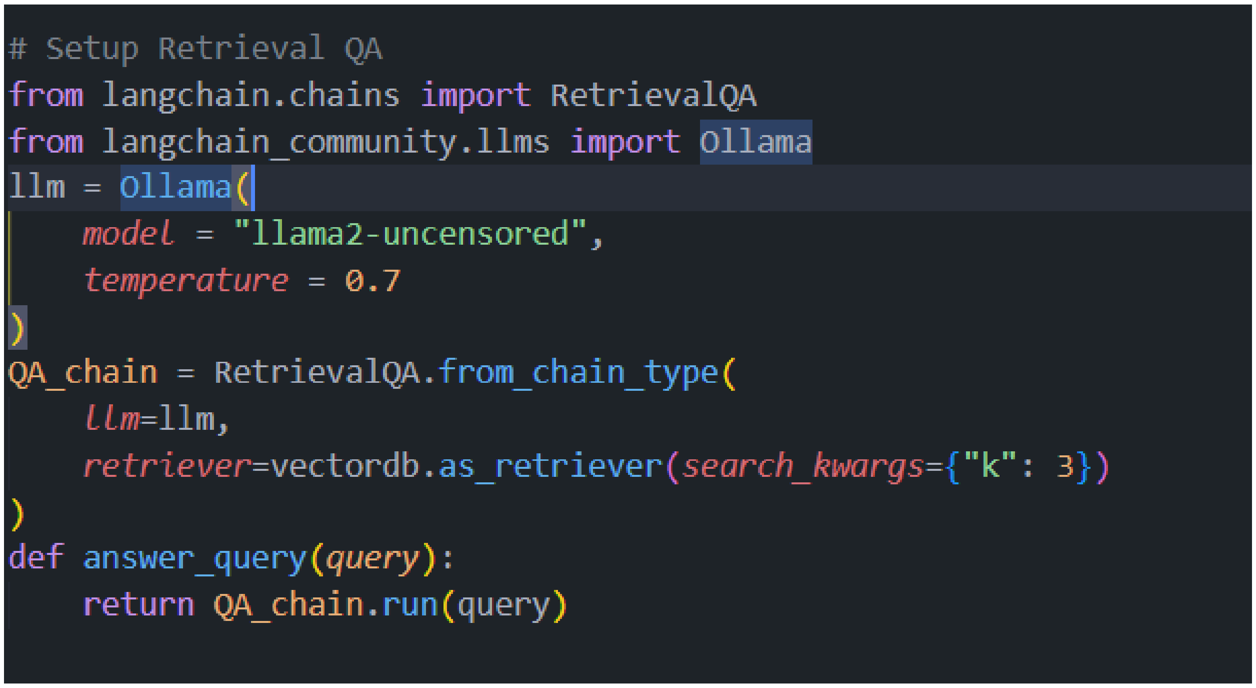
Option 2: Using Milvus for Vector Storage(Scalable & Production-Ready)



```python
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Milvus
from pymilvus import connections

# Connect to a running Milvus instance
connections.connect(alias = 'default', host = 'localhost', port = '19530')

def storeDocuments(text):
    splitter = RecursiveCharacterTextSplitter(chunk_size = 1000, chunk_overlap = 200)
    chunks = splitter.split_text(text)
    vectordb = Milvus.from_texts(
        texts= chunks,
        embedding= OpenAIEmbeddings(),
        connection_args= {'host': "localhost", "port": "19530"},
        collection_name= 'protean_store'
    )
    return vectordb
```



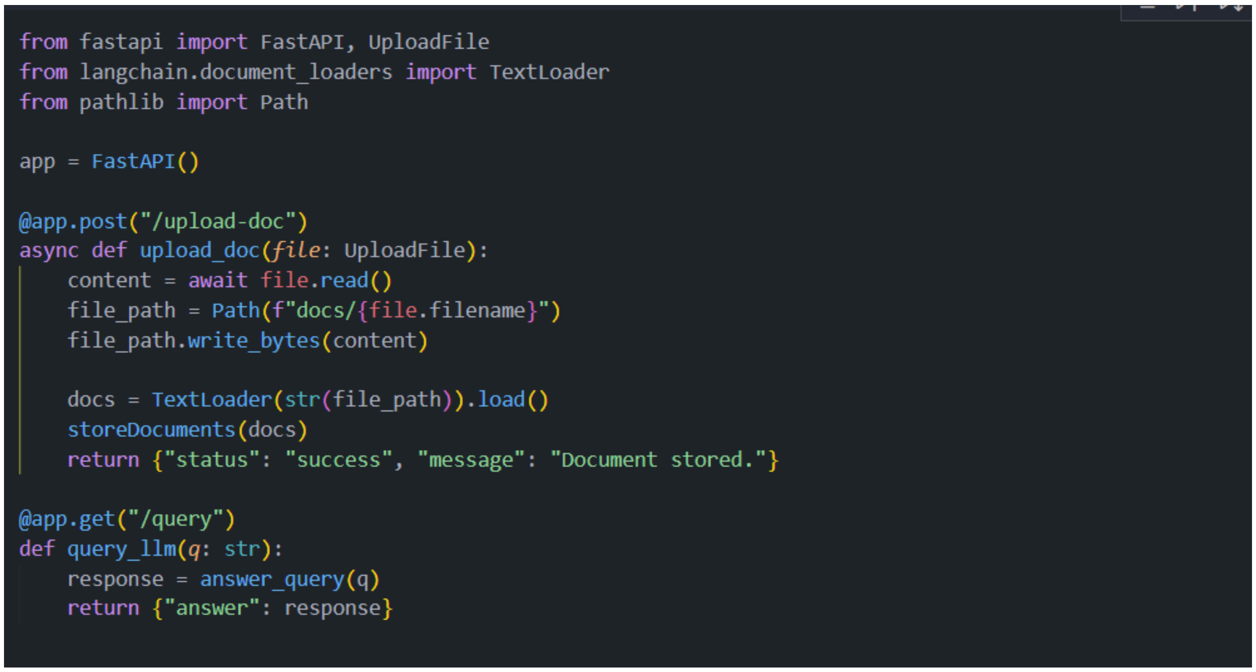
```python
# Setup Retrieval QA
from langchain.chains import RetrievalQA
from langchain_community.llms import Ollama
llm = Ollama(
    model = "llama2-uncensored",
    temperature = 0.7
)
QA_chain = RetrievalQA.from_chain_type(
    llm=llm,
    retriever=vectordb.as_retriever(search_kwargs={"k": 3})
)
def answer_query(query):
    return QA_chain.run(query)
```

Milestone 2:API Integration, Query Logging & Chat Interface (Weeks 4-7)

Deliverables:

- Implement a FastAPI layer to expose endpoints:
- Integrate query logging using NoSQL database(MongoDB or SQLite) for monitoring and debugging.
- Enable real-time query handling with proper input validation.
- Use a CLI tool(eg. Postman) to test API queries/endpoints.
- Ensure error handling and code structure.

FastAPI Backend with Upload & Query Endpoint



```python
from fastapi import FastAPI, UploadFile
from langchain.document_loaders import TextLoader
from pathlib import Path

app = FastAPI()

@app.post("/upload-doc")
async def upload_doc(file: UploadFile):
    content = await file.read()
    file_path = Path(f"docs/{file.filename}")
    file_path.write_bytes(content)

    docs = TextLoader(str(file_path)).load()
    storeDocuments(docs)
    return {"status": "success", "message": "Document stored."}

@app.get("/query")
def query_llm(q: str):
    response = answer_query(q)
    return {"answer": response}
```

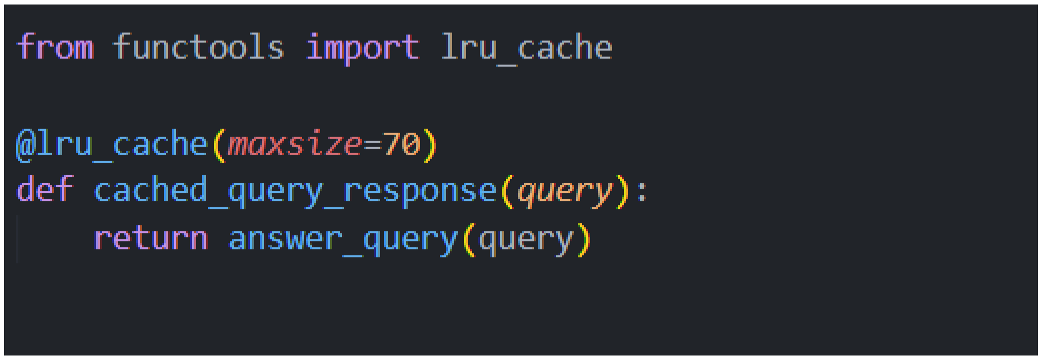
Milestone 3:Admin Dashboard and Deployment(Weeks 8-11)

Deliverables:

- Build an Admin Dashboard using Streamlit.
    - View uploaded documents.
    - Monitor query logs and chatbot performance.
- Add scalability features.
    - Create and store a cache of frequently asked questions using Python Dict.
- Perform end-to-end testing(unit + integration).
- Make this project Open-source.
    - Clear code structure

- Add a neat README.md file with setup instructions and contribution guidelines.

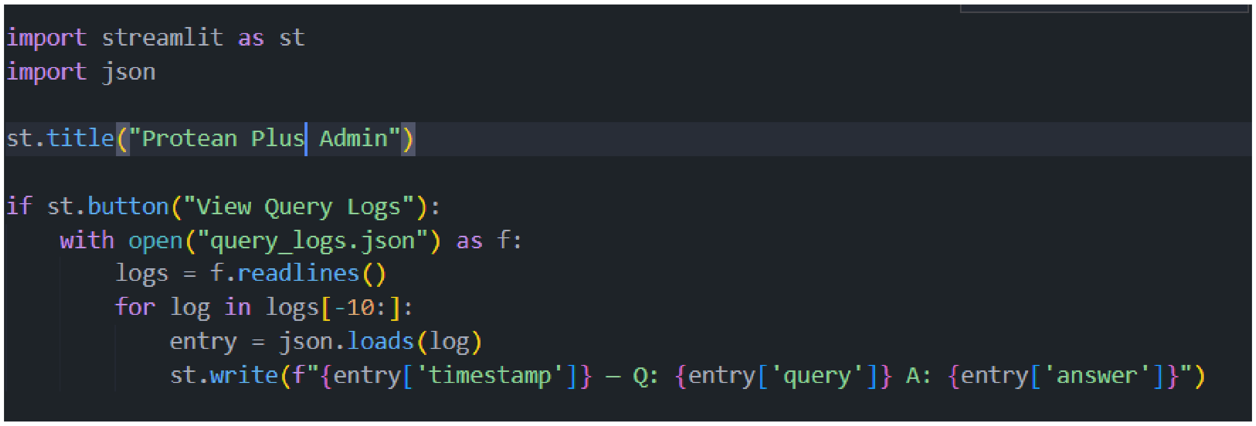
Caching Frequently asked questions(FAQs):



```python
from functools import lru_cache

@lru_cache(maxsize=70)
def cached_query_response(query):
    return answer_query(query)
```

Basic Streamlit Admin UI:



```python
import streamlit as st
import json

st.title("Protean Plus Admin")

if st.button("View Query Logs"):
    with open("query_logs.json") as f:
        logs = f.readlines()
        for log in logs[-10:]:
            entry = json.loads(log)
            st.write(f"{entry['timestamp']} - Q: {entry['query']} A: {entry['answer']}")
```

Availability

Number of hours available to dedicate to this project per week	30 Hours
Do you have any other engagements that will require your time? (projects/internships)	Full time job

Preferred Working hours: I will be mostly available in the evenings(IST) and more flexible during weekends.

Personal Information

About Me:

I'm Aman Kumar Srivastava, a passionate and self-motivated Machine Learning Engineer with hands-on Experience in building and deploying NLP pipelines, generative AI systems and RAG architectures. My github showcases end-to-end projects ranging from basic machine learning algo to creating end-to-end RAG pipelines and GAN-based image generation.

What is your motivation to apply for this project? Answer briefly in 5-10 lines:

As a fresher deeply passionate about AI and backend systems, this open-source project aligns perfectly with both my technical interests and learning goals. Over the past few years, I've independently built various types of end-to-end machine learning and NLP projects, including a

Retrieval-Augmented Generation(RAG) based chatbot, which showcases my ability to work with LLMs and APIs integration.
Contributing to this ONDC integration project excites me a lot because it combines backend development with real-world datasets and different kinds of APIs integration with open-source collaboration. I'm eager to apply my skills to this meaningful initiative that can impact India's digital commerce ecosystem as well as create a big impact in my life as an open-source contributor.

Previous experience/open source projects (Optional):

Project Name	Project Description	Links (if any)
RAG agent using Langchain	In this project I have built a RAG agent using Langchain. The idea is to connect a language model with a file so that it can give more accurate results rather than simple results. **Techstack: Python, FastAPI, langchain, FAISS, Ollama,**	**RAG-agent-using-LANGCHAIN**