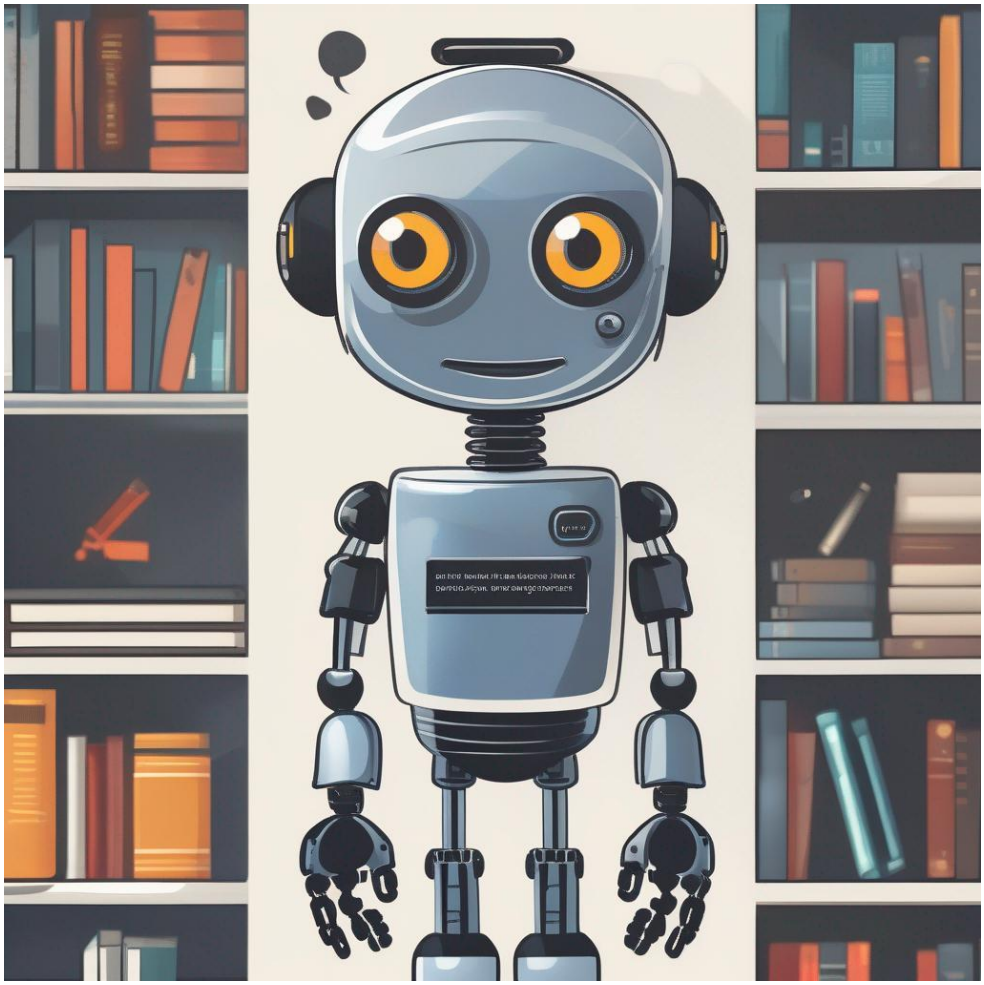
2024



# CUSTOM KNOWLEDGE BASE CHATBOT

## ARTIFICIAL INTELLIGENCE

B N SWAMINATHAN

D4 Insight Private Limited.
2nd Floor, East Wing, Khivraj Building Plot No.6, Electrical & Electronic, Landmark above Maruti Suzuki Arena, Industrial Estate, Perungudi, Chennai, Tamil Nadu 600096

## Table of Contents





D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

## EXECUTIVE SUMMARY

Internship is pleased to submit this proposal to SRM Institute of Science and Technology., The Custom Knowledge Base Chatbot project aims to develop and deploy an intelligent chatbot capable of answering questions and generating responses based on provided documents. Using Streamlit for the interface and NLP models from Hugging Face, the chatbot ensures accurate and efficient information retrieval, enhancing user experience across various domains

## INTRODUCTION

This project is focused on creating a sophisticated chatbot that uses pre-trained NLP models to understand and respond to user queries. It is built with Python and Streamlit, integrating multiple models to offer both question-answering and generative capabilities. The chatbot is designed for applications in customer service, education, and general information retrieval.

## DETAILS ABOUT THE TRAINING

During my internship at D4 Insight under the mentorship of Pothi Raja, I had the opportunity to select and work on a project of my choice. This allowed me to gain hands-on experience and deepen my understanding of key concepts in DevOps and Azure Cloud.

1. **Mentorship**: Guided by Pothi Raja, I received valuable insights and feedback that enhanced my learning process and project outcomes.
2. **Project Selection and Execution**: I was given the autonomy to choose a project that aligned with my interests and career goals. This freedom enabled me to explore various aspects of project development and management
3. **Skills Acquired**:
   DevOps Basics: Learned the fundamentals of DevOps, including continuous integration, continuous deployment, and infrastructure as code (IaC).
   Azure Cloud: Gained practical knowledge of Azure services, cloud computing principles, and how to deploy and manage applications on the Azure platform.
4. **Practical Experience**: Implemented and managed a project from conception to deployment, applying the skills and knowledge acquired during the internship. Worked on setting up environments, integrating tools, and automating processes, which are critical aspects of DevOps.

## PROJECT PLAN SUMMARY



D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

## 1. Project Title: Custom Knowledge Base Chatbot

## 2. Objective:

- To develop an intelligent chatbot capable of answering questions and generating responses based on provided documents.
- To leverage NLP models from Hugging Face for accurate and efficient information retrieval.

## 3. Scope:

- Load and process documents from user-provided URLs and text files.
- Provide accurate answers and generate responses based on user queries using QA and generative models.
- Ensure seamless integration and user-friendly interaction through a web-based interface.

## 4. Tools and Technologies:

- **Python**: The primary programming language for the project.
- **Streamlit**: For building an interactive web interface.
- **Hugging Face Transformers**: For implementing pre-trained NLP models.
- **Sentence Transformers**: For embedding documents and queries.
- **FAISS**: For efficient similarity search and document indexing.
- **Requests**: For fetching documents from URLs.
- **LangChain**: For handling document processing and embeddings.
- **Cachetools**: For caching results to improve performance.

## 5. Project Phases:

### *Phase 1: Requirement Analysis*

- Define project goals and expected outcomes.
- Identify target functionalities and specify data requirements.

### *Phase 2: Environment Setup*

- Install Python and necessary libraries.
- Set up a virtual environment and version control.

### *Phase 3: Model Integration*

- Load and configure pre-trained NLP models for document embedding, question answering, and text generation.
- Integrate models using Hugging Face Transformers and Sentence Transformers.

### *Phase 4: Document Processing*

- Develop scripts to load documents from URLs and text files.
- Implement text splitting and embedding for efficient processing.
- Create document indexing using FAISS for similarity search.

### *Phase 5: Query Handling and Response Generation*

- Implement query processing to retrieve relevant documents.
- Develop functions for question answering and generative response modes.
- Integrate caching to optimize performance.

### *Phase 6: Streamlit Interface Development*

- Design a user-friendly interface for inputting URLs and queries.
- Display responses interactively through the Streamlit app.

### *Phase 7: Testing and Debugging*

- Perform unit and integration testing.
- Implement error handling and optimize performance.
- Debug and ensure robustness of the application.

### *Phase 8: Documentation and Presentation*

- Document the code, process steps, and setup instructions.
- Prepare a user guide for running the application and understanding the workflow.
- Create a final presentation summarizing the project outcomes.

## 6. Deliverables:

- Python scripts for loading documents, query processing, and generating responses.
- A fully functional Streamlit application for user interaction.
- Clean and structured documentation.
- Comprehensive user guide and final presentation.

## 7. Timeline:

- **Week 1**: Requirement analysis and environment setup.
- **Week 2**: Model integration and document processing.
- **Week 3**: Query handling, response generation, and interface development.
- **Week 4**: Testing, debugging, and documentation.
- **Week 5**: Final presentation preparation.



D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

## 8. Expected Outcomes:

- Efficient and accurate question answering and response generation based on user-provided documents.
- A user-friendly web interface for seamless interaction.
- High-quality, maintainable code with comprehensive documentation.
- Enhanced technical skills in NLP, web development, and cloud deployment.

# ENVIRONMENTS

## Hardware Environment

- Development Machine: Requires a machine with at least 16 GB RAM, multi-core CPU, and a CUDA-enabled GPU (NVIDIA preferred) for model inference.
- Deployment Server: Should be a cloud server (e.g., AWS, Azure) with similar or better specifications to handle concurrent user queries efficiently.

## Software Environment

- Operating System: Ubuntu 20.04 LTS or Windows 10/11
- Programming Language: Python 3.8 or higher
- Libraries and Frameworks: Streamlit Hugging Face Transformers FAISS Sentence Transformers LangChain Cachetools

Virtual Environment: Python virtual environment for managing dependencies.

# PROJECT DESCRIPTIONS

The chatbot project aims to create an intuitive application that can process and respond to user queries based on provided documents. It leverages advanced NLP models for different functionalities, ensuring accurate and contextually relevant answers and responses.

# PROJECT DESIGN

## System Architecture

The project follows a modular design to ensure scalability and maintainability:
1. **Document Loading and Processing**: Handles loading documents from URLs and splitting them into chunks.
2. **Model Loading**: Pre-trained NLP models are loaded for various tasks.
3. **Embedding Computation**: Computes embeddings for document chunks.
4. **Indexing**: Uses FAISS to create an index for fast similarity search.

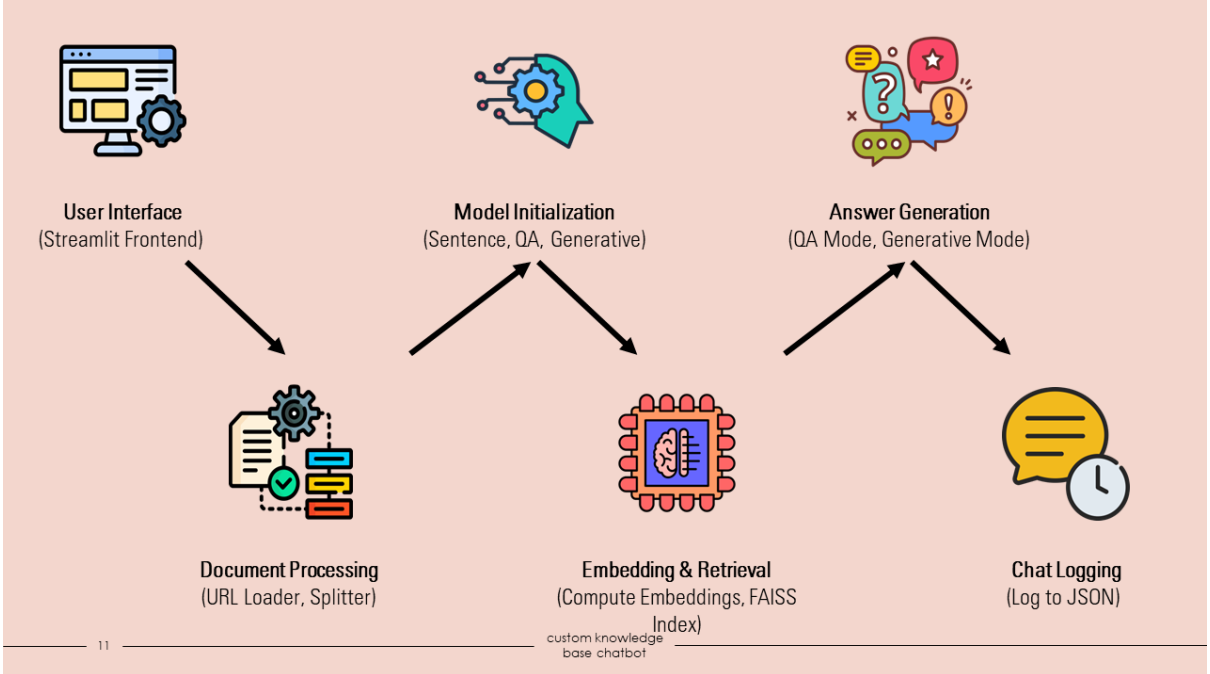
5. **Query Handling**: Processes queries to retrieve relevant documents and generate responses.
6. **Caching**: Implements caching for repeated queries to improve performance

### Data Flow

1. **User Input**: Users provide document URLs and queries through Streamlit.
2. **Document Processing**: Documents are loaded, split, and embedded.
3. **Query Processing**: Queries are embedded and relevant documents are retrieved.
4. **Response Generation**: Depending on the mode, responses are generated using QA or generative models.
5. **Output**: Responses are displayed in the Streamlit interface.

## SOLUTION OVERVIEW

The solution integrates Streamlit for the web interface with Hugging Face models for NLP tasks. It processes user-provided documents, indexes them, and retrieves relevant information to answer user queries. The chatbot operates in two modes: QA mode for precise answers and generative mode for detailed responses.




## PROJECT IMPLEMENTATION

### Step-by-Step Implementation

1. **Model Loading**: Load pre-trained models for embeddings, QA, and text generation.

2. **Document Loading and Processing**:

- Load documents from user-provided URLs.
- Split documents into smaller chunks.
- Compute embeddings for document chunks.

3. **Indexing**: Use FAISS to create an index from document embeddings for fast similarity search.

4. **Query Handling**:

- Embed user queries.
- Retrieve relevant document chunks using FAISS.
- Generate responses using QA or generative models.

5. **Caching**: Implement caching using cachetools to store responses for repeated queries.

6. **Streamlit Interface**: Develop a user-friendly interface for inputting URLs and queries and displaying responses.

## Code:

app.py

```
import streamlit as st
import json
from transformers import pipeline
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import UnstructuredURLLoader
from streamlit_chat import message

# Utility functions
from utils import load_sentence_model, load_qa_pipeline,
load_generative_pipeline, compute_embeddings, initialize_faiss_index,
retrieve_relevant_documents, generate_answer, log_chat

# Initialize models
sentence_model = load_sentence_model()
qa_pipeline = load_qa_pipeline()
generative_pipeline = load_generative_pipeline()

# Streamlit UI
st.title("Custom Knowledge Base Chatbot")

if 'document_embeddings' not in st.session_state:
    st.session_state['document_embeddings'] = None
```

```
    st.session_state['docs'] = None
    st.session_state['questions'] = []
    st.session_state['answers'] = []
    st.session_state['faiss_index'] = None

urls = st.text_input("Enter document URLs (comma-separated):")
if urls:
    try:
        url_list = [url.strip() for url in urls.split(',')]
        loader = UnstructuredURLLoader(urls=url_list)
        documents = loader.load()
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=200,
chunk_overlap=50)
        split_docs = text_splitter.split_documents(documents)
        docs = [doc.page_content for doc in split_docs]
        document_embeddings = compute_embeddings(docs, sentence_model)
        faiss_index = initialize_faiss_index(document_embeddings)
        st.session_state['docs'] = docs
        st.session_state['document_embeddings'] = document_embeddings
        st.session_state['faiss_index'] = faiss_index
        st.success("Documents loaded and processed.")
    except Exception as e:
        st.error(f"Error loading documents: {e}")

mode = st.radio("Choose mode:", ("QA", "Generative"))

question = st.text_input("Ask a question:")
if question:
    st.session_state['questions'].append(question)
    if st.session_state['faiss_index'] and st.session_state['docs']:
        relevant_docs = retrieve_relevant_documents(question,
st.session_state['faiss_index'], st.session_state['docs'], sentence_model,
top_k=2)
        if relevant_docs:
            answer = generate_answer(question, relevant_docs, mode,
qa_pipeline, generative_pipeline)
        else:
            answer = "No relevant documents found. Generating answer..."
            answer += generate_answer(question, [], "Generative", qa_pipeline,
generative_pipeline)
    else:
        # If no documents are loaded, use generative model directly
```



D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

```
        answer = generate_answer(question, [], mode, qa_pipeline,
generative_pipeline)

    st.session_state['answers'].append(answer)
    log_chat(question, answer)
    message(question)
    message(answer, is_user=True)

# Display recent chat history and allow expansion to view entire history
if st.session_state['questions']:
    with st.expander("Chat History"):
        for q, a in zip(st.session_state['questions'],
st.session_state['answers']):
            st.write(f"**Q:** {q}")
            st.write(f"**A:** {a}")
    if len(st.session_state['questions']) > 0:
        st.write("## Recent Chat")
        st.write(f"**Q:** {st.session_state['questions'][-1]}")
        st.write(f"**A:** {st.session_state['answers'][-1]}")
```

utils.py

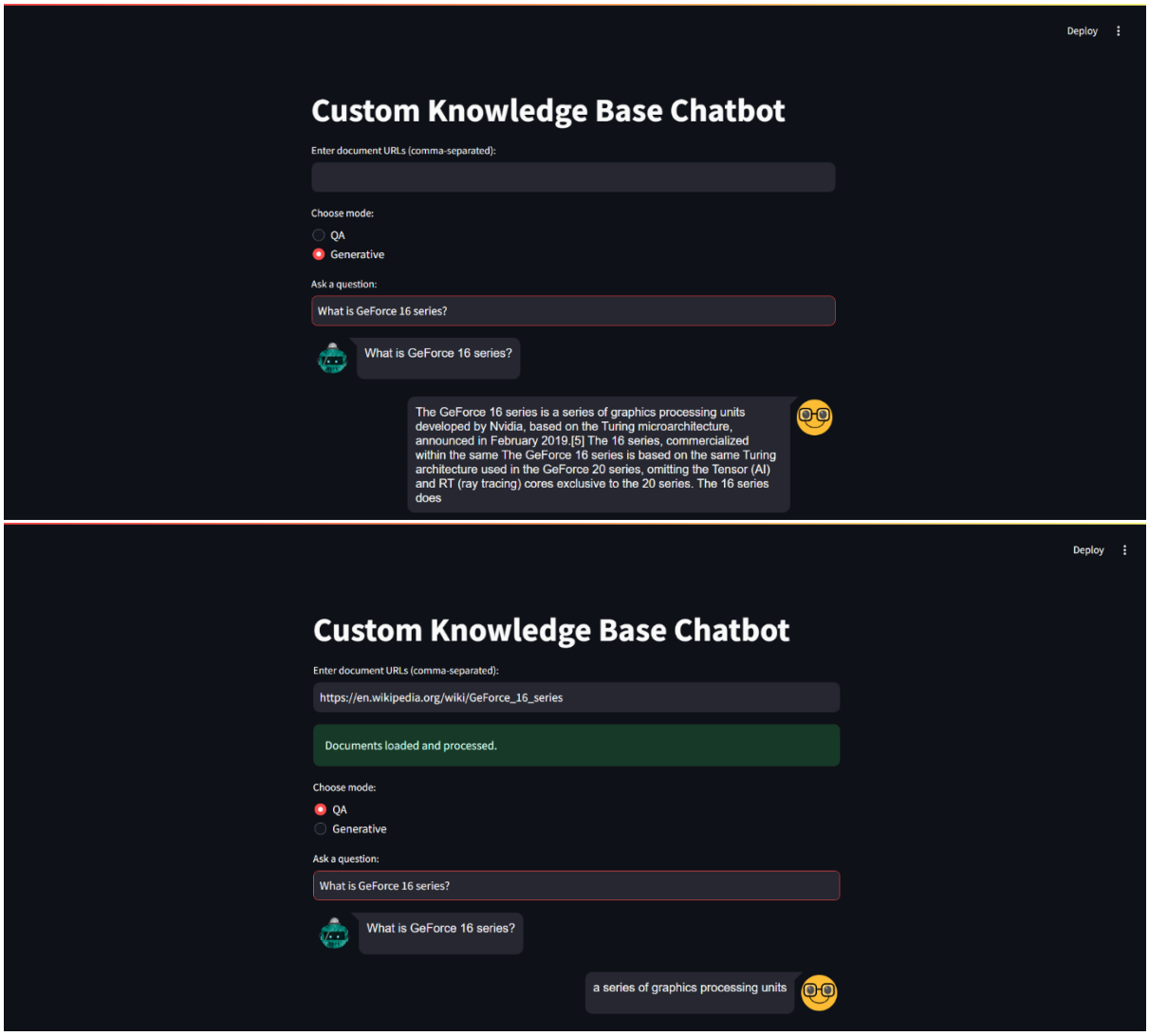
```
import os
import streamlit as st
import numpy as np
import faiss
from sentence_transformers import SentenceTransformer
from transformers import AutoTokenizer, AutoModelForQuestionAnswering,
AutoModelForCausalLM, pipeline
import json
import warnings

# Set API Key
os.environ['HF_API_KEY'] = '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'

# Initialize the sentence transformer model for embeddings
@st.cache_resource(show_spinner=False)
def load_sentence_model():
    return SentenceTransformer("paraphrase-MiniLM-L6-v2")  # Faster model

# Initialize the QA pipeline with a smaller, faster model
```

```
@st.cache_resource(show_spinner=False)
def load_qa_pipeline():
    model_name = "deepset/roberta-base-squad2"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForQuestionAnswering.from_pretrained(model_name)
    return pipeline("question-answering", model=model, tokenizer=tokenizer)

# Initialize the Generative pipeline for generating longer answers
@st.cache_resource(show_spinner=False)
def load_generative_pipeline():
    model_name = "EleutherAI/gpt-neo-2.7B"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForCausalLM.from_pretrained(model_name)
    return pipeline("text-generation", model=model, tokenizer=tokenizer)

# Function to compute document embeddings in batches
def compute_embeddings(docs, sentence_model, batch_size=16):
    embeddings = []
    for i in range(0, len(docs), batch_size):
        batch_docs = docs[i:i + batch_size]
        batch_embeddings = sentence_model.encode(batch_docs,
convert_to_tensor=True)
        embeddings.append(batch_embeddings)
    return np.vstack(embeddings)

# Function to initialize FAISS index
def initialize_faiss_index(embeddings):
    dimension = embeddings.shape[1]
    index = faiss.IndexFlatL2(dimension)
    index.add(embeddings)
    return index

# Function to retrieve relevant documents using FAISS
def retrieve_relevant_documents(question, faiss_index, documents,
sentence_model, top_k=3):
    question_embedding = sentence_model.encode(question,
convert_to_tensor=True).reshape(1, -1)
    distances, indices = faiss_index.search(question_embedding, top_k)
    return [documents[idx] for idx in indices[0]]

# Function to generate an answer based on the mode
def generate_answer(question, relevant_docs, mode, qa_pipeline,
generative_pipeline):
```

```
    if mode == "QA" and relevant_docs:
        context = " ".join(relevant_docs)
        answer = qa_pipeline(question=question, context=context)["answer"]
    elif mode == "QA":
        answer = "I'm unable to find relevant information from the provided
documents. Please provide more context or rephrase your question."
    else:
        if relevant_docs:
            context = " ".join(relevant_docs)
            prompt = (
                f"Given the context below, provide a detailed and accurate
response to the question.\n\n"
                f"Context: {context}\n\n"
                f"Question: {question}\n\n"
                f"Answer:"
            )
        else:
            prompt = f"Provide a detailed and accurate response to the
following question.\n\nQuestion: {question}\n\nAnswer:"

        response = generative_pipeline(
            prompt,
            max_length=200,
            num_return_sequences=1,
            temperature=0.7,  # Adjust temperature for more focused answers
            top_p=0.9,        # Adjust top-p for more coherent answers
            pad_token_id=50256,
            truncation=True
        )[0]["generated_text"]

        # Filter incomplete or irrelevant responses
        answer = response.split("Answer:")[-1].strip()
        if not answer or len(answer) < 10:
            answer = "I'm unable to generate a relevant answer at the moment.
Please try asking in a different way."

    return answer

# Function to log chat history
def log_chat(question, answer):
    chat_history = {"questions": st.session_state['questions'], "answers":
st.session_state['answers']}
    with open("chat_history.json", "w") as f:
```

```
        json.dump(chat_history, f)
```

## OUTPUT:




## SECURITY

1. **Data Security**: Secure handling of user inputs and document data to prevent data leakage.
2. **Secure Model Inference**: Perform model inference locally or on secure servers.

3. **Regular Updates:** Apply regular updates and patches to maintain security.
4. **Session/Cache Management**: Used cachetools throughout the project.

## SETUP AND DEPLOYMENT

### Setup

1.**Clone the Repository**:

git clone <repository_url> cd <repository_name>

2.**Create and Activate Virtual Environment**:

python -m venv venv source venv/bin/activate # On Windows, use `venv\Scripts\activate`

3.**Install Dependencies**:

pip install -r requirements.txt

### Deployment

- Run the Application Locally: streamlit run app-new.py
- Deploy on Cloud Server: Choose a cloud provider (AWS, Azure, GCP).
- Set up a virtual machine with the required specifications.
- Transfer the application files to the server.
- Set up the environment and install dependencies.
- Run the Streamlit application on the server.
- Configure a reverse proxy (e.g., Nginx) for handling requests to the Streamlit application.

## PROJECT WORK PLAN

Proposed approach is based on our previous experience and standard implementation methodologies. The following is the table of the project phases, their description and the related deliverables.


| Phase | Description | Deliverables |
|---|---|---|

| | | |
|---|---|---|
| **Blueprinting / Design** | – Conduct requirements workshops<br>– Accurately capture requirements<br>– Create Wireframes and designs<br>– Share Hardware & deployment architecture<br>– Propose Installation and configuration of hardware/software<br>– Self-Review & Peer to Peer review of all deliverables<br>– Sign-off from customer on Functional Specifications, Technical specifications, Wireframes, Design and Project Plan | – Functional Specifications<br>– Technical Specifications<br>– Web-Services Specifications<br>– Wireframes<br>– UI/UX Designs<br>– Project Plan |
| **Build** | – Prepare Unit test cases<br>– Development<br>– Unit Testing<br>– Code Review<br>– Defect Fixing<br>– Build verification testing | – Moving source code to test environment |
| **QA & Testing** | – Test plan<br>– Prepare System test cases<br>– System Testing<br>– Test Execution & Defect logging<br>– Defect fixing<br>– Regression Testing<br>– User Acceptance Testing support<br>– QA Sign-off | – Test documents<br>– Test execution reports<br>– Sign-off matrix<br>– Completely developed source code ready to be built for Go-Live<br>– Sign-off from System testing to move to UAT |
| **Deploy** | – Training to Customers/Users<br>– Move binaries to production servers<br>– Transition to Customer | – UAT Sign-off to move to production<br>– Application binaries in production<br>– Project Sign-Off |

## TASKS & TIMELINE



D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

| Tasks & Activities | Week 01 | Week 02 | Week 03 | Week 04 |
|---|---|---|---|---|
| Requirements Gathering & Understanding | **BA + Dev** | | | |
| Analysis, High- & Low-Level Design | **Dev** | | | |
| Coding/Unit Testing | **Dev** | | | |
| Functional Testing and Defect Reporting | **QA** | | | |
| Defect Fixing & Retesting | | | **Dev** | |
| Configuration Management | | | | |
| Project Management & Reporting | **Delivery Manager** | | | |
| Documentation & Sign off | | | | **Dev** |

## ROLES & RESPONSIBILITIES

| Role | Responsibilities | Count | Utilization |
|---|---|---|---|
| Developer | • Conduct requirements workshops.<br>• Accurately capture requirements<br>• Create Wireframes and designs<br>• Share Hardware & deployment architecture design<br>• Prepare Unit test cases<br>• Development<br>• Unit Testing<br>• Code Review<br>• Defect Fixing<br>• Build verification testing | 1 | 100% |

## COMMUNICATION PLAN

Email, Call, Chat, WhatsApp, Recurring meetings, active PM tools. At D4Insight, we realize communication is key to success, therefore we are open to adopt any communication preference that is comfortable to development team/stakeholders etc.,



D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096

| Review Meeting | Mode | Monitoring Frequency | Minimum Participants in Communication | Supporting Tools and Deliverables (if any) |
|---|---|---|---|---|
| Progress / Status Review | Conference Call / Meeting / Skype Chat | Daily | Working Committee | Daily / Weekly Report |
| Biweekly Plan/Forecast Meeting | Conference Call / Meeting | Bi-weekly | Working Committee | Work Request forecast plan |
| Program Update Meeting | Conference Call / Meeting | Weekly | Program Governance Committee | Program Monthly dashboard |

## CONCLUSION

The Custom Knowledge Base Chatbot project demonstrates the integration of advanced NLP models into a user-friendly chatbot application. The modular design, robust security measures, and structured deployment plan ensure a scalable and maintainable solution suitable for various use cases, including customer support and educational tools.

## REFERENCES


- Streamlit Documentation
- Hugging Face Transformers Documentation
- FAISS Documentation
- Sentence Transformers Documentation
- LangChain Documentation




D4 Insight Private Limited.
RMZ Millenia Business Park 2, Phase 2, 8th Floor, Campus 5 , No.143, Dr. M.G.R Road, Perungudi , Kanthanchavadi, Chennai - 600096