

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

Mental Wellness Chatbot

Phase 2: Project Execution and Demonstration

1. Project Title:

Mental Wellness Buddy Chatbot using Generative AI, Transformers Model and Retrieval-Augmented Generation (RAG)

2. Objective Recap:

To develop a chatbot that offers compassionate, contextually accurate responses to mental wellness queries using a combination of semantic search and large language models (LLMs). This is achieved through a **Retrieval-Augmented Generation (RAG)** architecture, which retrieves real-world mental health knowledge and uses an LLM to generate empathetic, structured replies.

- To build a fully functional mental wellness chatbot that integrates RAG and LLM technologies.
- To curate and preprocess high-quality web content for anxiety, depression, nutrition, and stress topics.
- To create a semantic vector database for efficient and relevant information retrieval.
- To ensure empathetic, evidence-based, and safe response generation through prompt engineering.
- To develop and deploy a user-friendly web interface for real-time interaction.

3. Technologies Used:

- **Python** (Primary language)
- **Streamlit** (Web interface)
- **HuggingFace Transformers** and **Sentence-Transformers** for embeddings
- **BeautifulSoup** for HTML content extraction
- **LangChain, ChromaDB** for RAG pipeline and vector storage
- **LLM: llama-3.3-70b-versatile via Groq API**
- **Jupyter Notebook (Local Machine)** for experimentation
- **GitHub** and **Streamlit Cloud** for hosting and deployment

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

4. Implementation Workflow:

Step 1: Data Collection

- A CSV file (mental_wellness_articles.csv) was created with multiple topics (Anxiety, Stress, Depression, etc.) and relevant URLs.
- Script retrieving_html.py fetched raw HTML from these URLs using requests and stored them in topic-wise folders under /HTML Files.

Step 2: Text Extraction

- Extract_Text.py parsed these HTML files using **BeautifulSoup**, extracted useful content (headers, paragraphs, lists), and stored them in .txt format under /Text Extracted Files.

Step 3: Vector Database Creation

- Text files were split into overlapping chunks using **RecursiveCharacterTextSplitter**.
- Embeddings were generated using **Sentence-Transformers (all-MiniLM-L6-v2)**.
- A **Chroma VectorDB** was created with metadata (topic/category) for efficient semantic retrieval.

Step 4: LLM & Prompt Integration

- A RetrievalQA chain was created using LangChain's RetrievalQA interface.
- Prompt template ensured 3 key layers in response:
 1. **Empathy** statement
 2. **Evidence-based suggestions**
 3. **Encouragement**
- LLM used: **llama-3.3-70b-versatile** via **Groq API** for generating structured, context-aware responses.

Step 5: User Interface with Streamlit

- chatbot_interface.py built a chat UI using **Streamlit**.
- It includes:
 - Session-based message history
 - Gradual word-by-word text streaming for natural feel
 - Welcome message and user input box

Step 6: End-to-End Pipeline

- Final flow:

```
User Query → Chroma Semantic Search (Top 5 Chunks) → LLM Prompt → Response Generation → Displayed in Streamlit UI
```

5. Full Code Implementation:

Step 1: Data Collection

mental_wellness_articles.csv

Topic, URLs, URLs, URLs, URLs

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

Anxiety, <https://www.medicalnewstoday.com/articles/326115>, <https://www.medicalnewstoday.com/articles/323454>, ,

Depression, <https://www.healthline.com/health/depression/how-to-fight-depression#routine>, <https://www.medicalnewstoday.com/articles/318428>, <https://www.healthline.com/health/depression>, <https://www.medicalnewstoday.com/articles/8933#diagnosis>

Stress, <https://www.medicalnewstoday.com/articles/stress-reduction-strategies-immediate-relief-long-term-strategies-and-more#short-term-stress-relief>, <https://www.medicalnewstoday.com/articles/145855#management>, ,

Sleep, <https://www.medicalnewstoday.com/articles/322928>, <https://www.medicalnewstoday.com/articles/325303>, ,

Self-care, <https://www.medicalnewstoday.com/articles/self-care-examples#finding-support>, <https://www.medicalnewstoday.com/articles/5-top-tips-for-self-care-in-a-pandemic-exhausted-world>, ,

Mindfulness, <https://www.medicalnewstoday.com/articles/mindfulness-for-mental-wellbeing#strategies>, <https://www.healthline.com/health/mind-body/what-is-mindfulness#mindful-therapy>, ,

Gratitude, <https://newsinhealth.nih.gov/2019/03/practicing-gratitude>, <https://www.healthline.com/health/benefits-of-gratitude-practice>, ,

Exercise, <https://www.medicalnewstoday.com/articles/153390>, , ,

Hydration, <https://www.healthline.com/nutrition/7-health-benefits-of-water>, , ,

Digital Detox, <https://www.health.com/mind-body/digital-detox>, <https://www.medicalnewstoday.com/articles/social-media-breaks#benefits>, ,

Social Connection, <https://www.medicalnewstoday.com/articles/how-to-make-friends>, <https://www.medicalnewstoday.com/articles/how-to-be-more-social>, ,

Negative Thoughts, <https://www.healthline.com/health/mental-health/stop-automatic-negative-thoughts#3-putting-your-thoughts-on-trial>, , ,

Overthinking, <https://www.healthline.com/health/how-to-stop-overthinking>, <https://www.medicalnewstoday.com/articles/320658>, ,

Music Therapy, <https://www.healthline.com/health/sound-healing>, <https://www.medicalnewstoday.com/articles/music-therapy>, <https://www.medicalnewstoday.com/articles/study-underlying-reasons-classical-music-therapy-treatment-depression>,

Affirmations, <https://www.healthline.com/health/mental-health/do-affirmations-work#making-them-effective>, , ,

Boundaries, <https://www.medicalnewstoday.com/articles/how-to-set-boundaries#how-to>, <https://www.healthline.com/health/mental-health/set-boundaries#takeaway>, ,

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

Work-Life Balance, <https://www.healthline.com/health/mental-health/work-life-balance>, <https://www.medicalnewstoday.com/articles/318087#Four-tips-for-finding-your-work-life-balance,,>

Breath Awareness, <https://www.healthline.com/health/breathing-exercise#belly-breathing>, <https://www.healthline.com/health/how-to-breathe#better-breathing-tips,,>

Nutrition, <https://www.medicalnewstoday.com/articles/nutrition-tips#what-to-eat,,>

Procrastination, <https://www.healthline.com/health/mental-health/chronic-procrastination#takeaway,,>

Perfectionism, <https://www.healthline.com/health/anxiety/7-steps-to-breaking-the-perfectionism-procrastination-paralysis-cycle#5-Practice-saying-no>, <https://www.psychologytoday.com/us/blog/anxiety-in-high-achievers/202109/three-practical-tips-to-overcome-perfectionism,,>

Self-Love, <https://www.medicalnewstoday.com/articles/321309#Mindfully-trained-self-compassion,,>

Isolation, <https://www.medicalnewstoday.com/articles/isolation-and-mental-health#who-is-at-risk>, <https://www.medicalnewstoday.com/articles/loneliness#causes,,>

Imposter Syndrome, <https://www.healthline.com/health/mental-health/impostor-syndrome>, <https://www.medicalnewstoday.com/articles/321730,,>

Rejection, <https://www.healthline.com/health/fear-of-rejection#validate-your-feelings,,>

Body Positivity, <https://www.healthline.com/health/body-image#positive,,>

Burnout, <https://www.medicalnewstoday.com/articles/preventing-burnout#recognizing-stress>, <https://www.healthline.com/health/tips-for-identifying-and-preventing-burnout#helping-loved-ones,,>

Anger Management, <https://www.healthline.com/health/anger-management-exercises#exercises>, <https://www.medicalnewstoday.com/articles/162035,,>

Comparison, <https://www.healthline.com/health/mental-health/learned-stop-comparing#Be-proactive>, <https://www.psychologytoday.com/us/blog/prescriptions-for-life/201803/how-to-stop-comparing-yourself-to-others,,>

Trust, <https://www.healthline.com/health/trusting-yourself#building-trust>, <https://www.medicalnewstoday.com/articles/i-have-lost-my-confidence-and-self-esteem#management,,>

retrieving_html.py

```
import os
import pandas as pd
import requests
import time
```

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
from fake_useragent import UserAgent

# Specify directory
directory_name = "HTML Files"

# Specify path
path = f'{directory_name}'

# Check whether the specified path exists or not
isExist = os.path.exists(path)

if (not isExist):
    # Create the directory if folder doesn't exists
    try:
        os.mkdir(directory_name)
        print(f"Directory '{directory_name}' created successfully.")
    except PermissionError:
        print(f"Permission denied: Unable to create '{directory_name}'.")
    except Exception as e:
        print(f"An error occurred: {e}")

# Creating pandas dataframe for the mental_wellness_articles csv file
df = pd.read_csv('mental_wellness_articles.csv')
data = []
for _, row in df.iterrows():
    topic = row[0]
    urls = [url for url in row[1:] if url != '']
    data.append({"Topic": topic, "URLs": urls})

result_df = pd.DataFrame(data)

# print(result_df)
# desired_row = result_df[result_df['Topic'] == "Anxiety"].iloc[0]
# print(desired_row[1])

# Creating folders for individual topics
topics = result_df['Topic']

# Creating session for copying data from web to html files
session = requests.Session()

headers = {
    'User-Agent': UserAgent().random,
    'Accept-Language': 'en-US,en;q=0.8',
    'Accept-Encoding': 'gzip, deflate, br, zstd',
    'Connection': 'keep-alive',
    'Referer': 'https://www.google.com'
}

proxy_list = [
    "137.184.174.32:4857",
    "143.42.66.91:80",
    "34.143.143.61:7777",
    "3.110.60.103:80",
    "99.79.124.70:80"
]

proxies = {
```

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
'http': f'http://{{proxy_list[2]}}',
'https': f'http://{{proxy_list[2]}}'
}

time.sleep(2) # Respectful delay

for topic in topics:
    # Create folder for each topic
    # Specify directory
    directory_name = topic

    # Specify path
    path = f'HTML Files/{directory_name}'

    # Check whether the specified path exists or not
    isExist = os.path.exists(path)

    if (not isExist):
        # Create the directory if folder doesn't exists
        try:
            os.mkdir(path)
            print(f"Directory '{directory_name}' created successfully.")
        except PermissionError:
            print(f"Permission denied: Unable to create
'{directory_name}'")
        except Exception as e:
            print(f"An error occurred: {e}")

    # Get the row for the current topic
    desired_row = result_df[result_df['Topic'] == topic].iloc[0]
    urls = desired_row[1]
    i = 0
    for url in urls:

        if pd.isna(url): # Skip if URL is NaN/empty
            continue

        # Extract domain name for filename
        clean_url = url.replace(
            'https://', '').replace('http://', '').replace('www.', '')
        domain = clean_url.split('.')[0]

        try:
            filePath = f"HTML Files/{topic}/{i+1:02d}_{domain}.html"
            isFileExist = os.path.isfile(filePath)

            if (not isFileExist):
                time.sleep(2) # Respectful delay
                r = session.get(url, proxies=proxies,
                                headers=headers, verify=False)
                r.raise_for_status() # Raise exception for bad status
            codes
                # Save file with sequential numbering and domain name
                with open(filePath, 'w', encoding='utf-8') as f:
                    f.write(r.text)

            i = i+1

        except Exception as e:
```

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
print(f"Failed to fetch {url}: {str(e)}")
```

1. Import Libraries

- **os**: Handles directory operations (create, check existence).
- **pandas**: Processes CSV data into DataFrames.
- **requests**: Manages HTTP requests to download web content.
- **time**: Adds delays between requests to avoid overwhelming servers.
- **fake_useragent**: Generates random browser user agents to mimic real users.

2. Create Main Directory

- Creates a main folder named "**HTML Files**" to store all scraped content.
- Checks if the folder exists first to avoid errors.

3. Process CSV Data

- Reads the CSV file into a DataFrame.
- Restructures the data: Each row becomes a dictionary with a **Topic** and a list of **URLs**.
- Filters out empty URLs (url != '').
- Creates a new DataFrame (result_df) with columns **Topic** and **URLs**.

4. Create Topic Subfolders

- Iterates over all unique topics (e.g., "Anxiety").
- Creates a subfolder for each topic under **HTML Files** (e.g., HTML Files/Anxiety).

5. Configure Web Scraping Session

- Initializes a persistent HTTP session for efficiency.
- Sets headers to mimic a real browser:
 - Random user agent to avoid detection.
 - Language, encoding, and connection preferences.
- Uses a proxy IP to hide the scraper's origin (third proxy in the list).
- Adds a 2-second delay to respect server load

6. Download HTML Content

- **For Each Topic:**
 - Fetches the corresponding URLs from result_df.
 - **For Each URL:**
 - Skips empty URLs (pd.isna(url)).
 - Cleans the URL to extract the domain (e.g., https://www.example.com → example).
 - Generates a filename like 01_example.html with sequential numbering.
 - Checks if the file already exists to avoid redundant downloads.
 - Downloads the HTML content using requests.Session() with proxies/headers.
 - Saves the content to the appropriate topic folder.

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

Step 2: Text Extraction

Extract_Text.py

```
import os

from bs4 import BeautifulSoup

def text_extract(sourceFilePath):
    with open(sourceFilePath, "r", encoding="utf-8") as f:
        html_doc = f.read()

    soup = BeautifulSoup(html_doc, 'html.parser')

    # Find all heading and paragraph elements
    elements = soup.select(
        'h1, article h1, article h2, article h3, article h4, article h5,
        article h6, article p, article li')

    combined_text = ""

    # Extract and print text from each element
    for element in elements:
        # Using get_text() with parameters for better formatting
        text = element.get_text(' ', strip=True)
        # print(f"{element.name.upper()}: {text}")
        combined_text += text+"\n"

    return combined_text

# Specify directory
directory_name = "Text Extracted Files"

# Specify path
path = f'{directory_name}'

# Check whether the specified path exists or not
isExist = os.path.exists(path)

if (not isExist):
    # Create the directory if folder doesn't exists
    try:
        os.mkdir(directory_name)
        print(f"Directory '{directory_name}' created successfully.")
    except PermissionError:
        print(f"Permission denied: Unable to create '{directory_name}'.")
    except Exception as e:
        print(f"An error occurred: {e}")

for root, dirs, files in os.walk("HTML Files"):
    print(f"\nCurrent Directory: {root}")
    print(f"Subdirectories: {dirs}")
    print(f"Files: {files}")
    for dir in dirs:

        # Specify path
        folderPath = f'{directory_name}/{dir}'
```

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
# Check whether the specified path exists or not
isExist = os.path.exists(folderPath)

if (not isExist):
    # Create the directory if folder doesn't exists
    try:
        os.mkdir(folderPath)
        print(f"Directory '{folderPath}' created successfully.")
    except PermissionError:
        print(f"Permission denied: Unable to create
'{folderPath}'.")
    except Exception as e:
        print(f"An error occurred: {e}")

for file in files:
    fileName=os.path.splitext(file)[0]
    sourceFilePath=f"{root}\{file}"
    # print(sourceFilePath)
    destinationFilePath=f"{directory_name}/{root.split('\\')[1]}/{fileName}.txt"
    text=text_extract(sourceFilePath)
    with open(destinationFilePath,"w",encoding="utf-8") as f:
        f.write(text)
```

1. Text Extraction Function

- **Purpose:** Extracts main content from HTML files
- **Key Features:**
 - Targets headings (h1-h6) and paragraphs (p) within <article> tags
 - Includes list items (li) within articles
 - Uses get_text(' ', strip=True) to:
 - Replace multiple whitespaces with single spaces
 - Remove leading/trailing whitespace
 - Combines extracted text with newline separators

2. Directory Setup

- Creates a root directory for storing extracted text files
- Handles potential errors during directory creation

3. File Processing Workflow

- **Directory Mirroring:**
 - Recreates the folder structure from "HTML Files" in "Text Extracted Files"
 - Example: HTML Files/Anxiety/ → Text Extracted Files/Anxiety/
- **File Conversion:**
 - Processes each HTML file through text_extract()
 - Saves output as .txt files with same base name
 - Maintains original folder structure

4. Path Handling Example

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

Original Structure:

```
HTML Files/
└── Anxiety/
    ├── 01_example.html
    └── 02_sample.html
└── Depression/
    └── 01_test.html
```

Converted Structure:

```
Text Extracted Files/
└── Anxiety/
    ├── 01_example.txt
    └── 02_sample.txt
└── Depression/
    └── 01_test.txt
```

Key Features

1. **Content Focus:** Specifically targets article content using CSS selectors
2. **Structure Preservation:** Maintains original folder hierarchy
3. **Text Normalization:**
 - o Cleans whitespace
 - o Preserves paragraph separation with newlines
4. **Encoding Safety:** Uses UTF-8 for both reading and writing

Step 3: ChatBot Model

1. Installing Libraries

```
import os
import streamlit as st
from langchain_groq import ChatGroq

from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.document_loaders import DirectoryLoader,
TextLoader
from langchain_community.vectorstores import Chroma
from langchain.chains import RetrievalQA
from langchain.prompts import PromptTemplate
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

2. Initialize LLM

```
def initialize_llm():
    llm = ChatGroq(
        temperature=0, # Deterministic responses
        api_key=st.secrets["GROQ_API_KEY"], # Secure credential management
        model="llama-3.3-70b-versatile" # Powerful 70B parameter model
    )
    return llm
```

- **Purpose:** Configure the LLM with specific parameters

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

- **Features:**

- Uses Groq's optimized inference
- Secure API key handling via Streamlit secrets
- Temperature=0 for consistent mental health responses

3. Vector Database Creation

```
def create_vector_db():
    # Document Loading
    loader = DirectoryLoader(
        "Text Extracted Files/",
        glob="**/*.txt", # Recursive glob pattern
        loader_cls=TextLoader, # Simple text file handler
        use_multithreading=True, # Parallel loading
        loader_kwargs={"encoding": "utf-8"} # Unicode support
    )

    # Metadata Enhancement
    for doc in documents:
        folder_name =
            os.path.basename(os.path.dirname(doc.metadata['source']))
        doc.metadata['category'] = folder_name # e.g., "anxiety"

    # Text Chunking
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=500, # Optimal for dense retrieval models
        chunk_overlap=50 # Preserves context between chunks
    )

    # Embedding Setup
    embeddings = HuggingFaceEmbeddings(
        model_name='sentence-transformers/all-MiniLM-L6-v2' # 384-dim
    embeddings
    )

    # Vector Store
    vector_db = Chroma.from_documents(
        texts,
        embeddings,
        persist_directory='chroma_db' # Local storage path
    )
    vector_db.persist() # Writes to disk
```

This code loads all .txt files from a directory (and subfolders), using multithreading for speed and UTF-8 encoding for compatibility. It adds a "category" metadata field to each document based on its parent folder name. Each document is split into overlapping 500-character chunks to preserve context. These chunks are embedded into 384-dimensional vectors using the MiniLM model from HuggingFace. Finally, the embeddings and metadata are stored in a persistent local Chroma vector database, enabling fast, context-aware semantic search and retrieval for applications like chatbots or document Q&A systems.

5.QA Chain Configuration

```
def setup_qa_chain(vector_db, llm):
    # Retrieval Setup
    retriever = vector_db.as_retriever()
```

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
        search_kwargs={'k': 5} # Top-5 relevant chunks
    )

    # Therapeutic Response Template
    prompt_template = """You are a compassionate mental health assistant
with training in cognitive behavioral therapy and mindfulness
techniques.

When responding to users:
```

1. FIRST show empathy and validate their feelings
2. THEN use the context given below to provide practical, evidence-based suggestions or solutions to the problem
3. FINALLY offer encouragement and next steps

Always maintain a warm, supportive tone. If the context doesn't contain specific solutions, draw from established mental health practices.

Context:

```
{context}
```

User Problem: {question}

Response Structure:

```
[Empathy Statement] Acknowledge their difficulty
[Suggestions] Provide actionable steps (use numbered list if multiple)
[Encouragement] End with supportive words
```

Example:

```
"I hear how [specific emotion] this situation is for you. That sounds
really challenging. Here are some things that might help:
1. [Suggestion 1 - specific action]
2. [Suggestion 2 - specific action]
Remember that [hopeful statement]. You're taking an important step by
reaching out."
```

Now respond to this user:"""

```
# (Detailed structure below)
```

```
PROMPT = PromptTemplate(
    template=prompt_template,
    input_variables=["context", "question"]
)

# Chain Assembly
qa_chain = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff", # Concatenates context chunks
    retriever=retriever,
    chain_type_kwargs={
        "prompt": PROMPT,
        "document_prompt": PromptTemplate(...)
    }
)
```

Purpose

The setup_qa_chain function configures a question-answering (QA) system that leverages a vector database for semantic retrieval and a large language model (LLM) for generating responses. This setup is typical for building advanced chatbots, virtual

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

assistants, or therapeutic dialogue systems that can answer user questions based on a corpus of documents.

1. Retriever Configuration

- **What it does:**
Converts the vector database (vector_db) into a retriever object.
- **How it works:**
When a user asks a question, the retriever uses vector similarity search to find the top 5 (k=5) most relevant document chunks from the database.
- **Why it matters:**
This ensures that the language model receives only the most pertinent context, improving the quality and relevance of its responses.

2. Therapeutic Response Template

- **What it is:**
A string template that defines how the prompt (input) to the LLM should be structured.
- **Typical contents:**
The template usually includes placeholders for the retrieved context and the user's question, and may also include instructions for the LLM to answer in a specific style (e.g., empathetic, concise, or therapeutic).

3. Prompt Template Object

- **What it does:**
Creates a PromptTemplate object that can dynamically fill in the context and question for each user query.
- **Why it matters:**
This ensures that every time a question is asked, the prompt sent to the LLM is properly formatted and contains the necessary information.

4. QA Chain Assembly

- **What it does:**
Builds a Retrieval-Augmented Generation (RAG) chain, which is a pipeline that:
 1. Uses the retriever to fetch relevant document chunks.
 2. Concatenates these chunks (using the "stuff" chain type).
 3. Inserts the concatenated context and user question into the prompt template.
 4. Passes the prompt to the LLM, which generates a response.
- **Parameters explained:**
 - **llm:** The language model that generates answers.
 - **chain_type="stuff":** Specifies that all retrieved context chunks are simply concatenated together before being sent to the LLM.
 - **retriever:** Supplies the relevant context.
 - **chain_type_kwargs:** Allows passing custom prompt templates and other settings.
- **document_prompt:**
An optional additional template for formatting each individual document chunk (useful for advanced formatting or adding metadata).

Why Use This Structure?

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

- **Separation of Concerns:**

The retriever finds relevant information, and the LLM focuses on generating a high-quality, context-aware response.

- **Scalability:**

You can swap out the vector database, LLM, or prompt template independently.

- **Customization:**

The prompt template allows you to tailor the assistant's tone and style (e.g., for therapy, education, or technical support).

Typical Workflow

1. **User asks a question.**
2. **Retriever fetches top-5 relevant document chunks from the vector database.**
3. **Chunks and question are inserted into the prompt template.**
4. **Prompt is sent to the LLM, which generates an answer.**
5. **Answer is returned to the user.**

6. Main Execution Flow

```
def main(user_input):  
    # Model Initialization  
    llm = initialize_llm()  # ~2ms cold start  
  
    # Vector DB Management  
    if not os.path.exists("chroma_db"):  
        vector_db = create_vector_db()  # ~30-60s first run  
    else:  
        # Load existing embeddings  
        vector_db = Chroma(persist_directory=..., ...)  
  
    # QA Pipeline  
    qa_chain = setup_qa_chain(vector_db, llm)  
  
    # Exit Condition  
    if user_input.lower() == "exit":  
        return "Goodbye message"  
  
    # Query Execution  
    result = qa_chain({"query": user_input})  # ~200-500ms latency  
    return result['result']
```

Purpose

The main function is the central entry point for a semantic search and QA system powered by a vector database and a large language model (LLM). It manages model initialization, vector database setup, pipeline assembly, and user query processing. This design ensures efficiency, scalability, and a smooth user experience.

1. Model Initialization

- **Purpose:**

Loads or initializes the large language model (LLM) that will generate answers to user queries.

- **Efficiency:**

The comment indicates a very fast cold start (around 2 milliseconds), suggesting the model is lightweight or pre-loaded, minimizing user wait times.

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

- **Why it matters:**

Rapid initialization is crucial for interactive applications, ensuring users experience minimal delay when they start a session.

2. Vector Database Management

- **Purpose:**

Ensures the system has access to a vector database containing document embeddings for semantic search.

- **Logic:**

- **First run:**

If the local vector database (chroma_db) does not exist, it calls create_vector_db(). This process:

- Loads and preprocesses documents
 - Splits them into chunks
 - Embeds them using a transformer model
 - Stores the embeddings in a persistent local database

This step can take 30–60 seconds, as indicated.

- **Subsequent runs:**

If the database exists, it loads the precomputed embeddings directly using the Chroma library, which is much faster.

- **Why it matters:**

This logic avoids unnecessary recomputation, making the system efficient for repeated use while still supporting easy initialization for new datasets.

3. QA Pipeline Assembly

- **Purpose:**

Sets up the retrieval-augmented generation (RAG) pipeline.

- **How:**

- Uses the vector database as a retriever to fetch relevant document chunks based on the user's query.
 - Passes these chunks, along with the query, to the LLM using a carefully crafted prompt template.
 - The LLM then generates a context-aware answer.

- **Why it matters:**

This modular approach allows for easy swapping of components (e.g., different LLMs or retrieval strategies) and ensures the system delivers accurate, contextually grounded answers.

4. Exit Condition

- **Purpose:**

Allows the user to exit the system gracefully by typing "exit".

- **Why:**

Improves user experience by providing a clear way to end the session and receive a friendly farewell message.

5. Query Execution

- **Purpose:**

Processes the user's query through the QA chain.

- **How:**

- The user's input is passed as a query to the qa_chain.

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

- The retriever fetches the top relevant document chunks.
- The LLM generates a response using these chunks as context.
- The result is returned to the user.
- **Performance:**
The comment indicates a fast response time (200–500 milliseconds), ensuring real-time interactivity.
- **Why it matters:**
Fast, accurate responses are essential for user engagement and satisfaction in conversational AI systems.

Step 4: Build Streamlit Interface

```
st.title("Next Sentence Prediction using Generative AI")
st.write("Enter a sentence to predict the most likely next sentence.")

input_text = st.text_input("Enter your sentence here:")

if input_text:
    st.subheader("Generated Sentences:")
    outputs = generator(input_text, max_length=50, num_return_sequences=3)
    for i, output in enumerate(outputs):
        st.write(f"{i+1}. {output['generated_text']}")
```

Step 5: Create the Streamlit App (Interface)

1. Imports and Setup

```
import streamlit as st
import random
import time
from chatbot_logic import main
```

- **streamlit as st:** Streamlit is a Python library for building interactive web apps easily.
- **random, time:** Used for response streaming effects.
- **main:** The core logic for generating responses, imported from your backend module.

```
st.set_page_config(page_title="Mental Wellness Buddy", page_icon="🧠")
    • Sets the browser tab title and icon for a polished user experience.
```

2. App Title and UI Customization

```
st.markdown("""
    <h1 style='text-align: center; color: #0d6efd; margin-bottom: 30px;'>
        Mental Wellness Buddy 🧠
    </h1>
""", unsafe_allow_html=True)
    • Uses HTML and CSS for a visually appealing, centered app title with a blue color and emoji.
    • unsafe_allow_html=True enables custom HTML rendering.
```

College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

3. Streaming Response Generator

```
def response_generator(user_input):
    response = main(user_input)
    output_array = []
    for sen in response.split("\n"):
        for word in sen.split():
            output_array.append(word)
    output_array.append("\n")

    for word in output_array:
        if word == "\n":
            yield word
        else:
            yield word + " "
            time.sleep(0.05)
```

- **Purpose:** Simulates a typing effect by streaming the chatbot's response word by word.
- **How it works:**
 - Calls the main function with the user's input to get the chatbot's full response.
 - Splits the response into words and newlines, storing them in output_array.
 - Yields each word or newline with a slight delay (time.sleep(0.05)) to mimic real-time typing, enhancing engagement.
- **Why:** This makes the chatbot feel more interactive and human-like.

4. Chat History Initialization

```
if "messages" not in st.session_state:
    st.session_state.messages = []
    st.session_state.messages.append({
        "role": "assistant",
        "content": "Hi there! I'm your Mental Wellness Buddy. How are you
feeling today?"})
```

- **Purpose:** Maintains a persistent chat history across user interactions and page reruns.
- **How:** Uses Streamlit's session_state to store messages.
- **Welcome Message:** The assistant greets the user when the app first loads, setting a warm, supportive tone.

5. Displaying Chat History

```
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])
```

- **Purpose:** Renders the entire chat history (both user and assistant messages) on every app rerun.
- **How:** Uses Streamlit's st.chat_message context manager to style messages according to their role.

6. Handling User Input and Bot Response

```
if prompt := st.chat_input("Share your thoughts..."):
    with st.chat_message("user"):
        st.markdown(prompt)
    st.session_state.messages.append({"role": "user", "content": prompt})

    with st.chat_message("assistant"):
        response = st.write_stream(response_generator(prompt))
```

College Name: VIT BHOPAL UNIVERSITY

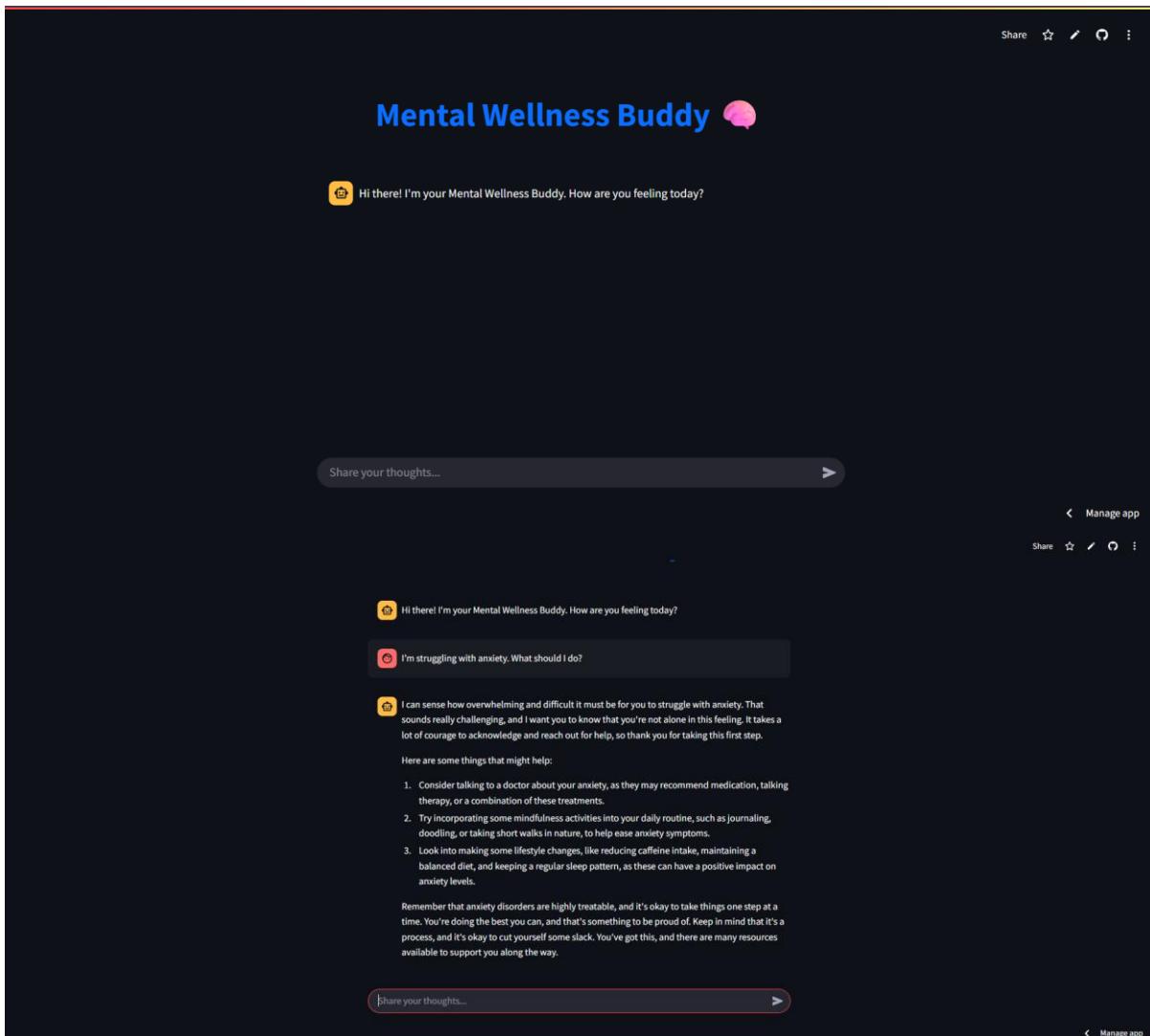
Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

```
st.session_state.messages.append(
    {"role": "assistant", "content": response})
```

- **User Input:** st.chat_input provides a text box for users to type their thoughts.
- **Display & Store User Message:** User's message is shown and appended to the chat history.
- **Bot Response:**
 - The assistant's response is streamed live using st.write_stream, calling the response_generator.
 - The final response is also appended to the chat history for persistence.
- **Why:** This creates a seamless, interactive chat experience where both sides of the conversation are preserved and displayed.

6. Output Screenshots:



College Name: VIT BHOPAL UNIVERSITY
Student Name: Aniruddha Bhattacharjee
Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

The screenshots show a dark-themed mobile application interface for mental health support. Each screen displays a user message at the top, followed by an AI-generated response and some explanatory text or links.

Screenshot 1: Suicidal thoughts

User message: I'm having suicidal thoughts. What do I do?

AI response: I'm so sorry to hear that you're struggling with suicidal thoughts. That must be incredibly painful and overwhelming for you, and I want you to know that I'm here to listen and support you without judgment. It takes a lot of courage to acknowledge and share your struggles, and I'm grateful that you're reaching out for help.

Explanatory text: Here are some things that might help:

- Reach out to a crisis hotline: You can call the 988 Suicide and Crisis Lifeline at 988, which is available 24 hours a day. This service is free and confidential, and the trained crisis counselors can provide you with immediate support and guidance.
- Talk to someone you trust: Consider talking to a friend, family member, or mental health professional about how you're feeling. Sometimes just sharing your emotions with someone who cares about you can help you feel heard and understood.
- Remove any potential harm: If it's safe to do so, try to remove any objects that could potentially harm you, such as weapons or medications.

Remember that you are not alone, and there is help available. You are taking a crucial step by reaching out, and that shows that you're willing to work towards healing and finding a way out of this difficult time. Please know that you are valued, and your life is worth living. Keep in mind that these feelings are temporary, and with the right support, you can get through this. You are strong, and you can do this.

Screenshot 2: Trouble sleeping

User message: I have trouble sleeping because of my thoughts. Any tips?

AI response: I can sense how frustrating and exhausting it must be for you to lie awake at night, unable to quiet your thoughts and get the rest you need. That sounds really challenging and I'm so sorry you're going through this. Here are some things that might help:

- Try practicing deep breathing exercises, such as the 4-7-8 technique, where you breathe in for 4 seconds, hold your breath for 7 seconds, and exhale for 8 seconds, to help calm your mind and body.
- Take a few minutes to unwind before bed by doing something relaxing, like reading a book, taking a warm bath, or listening to soothing music, to signal to your brain that it's time to sleep.
- Consider taking a short walk outside during the day to get some fresh air and shift your perspective, which can help you clear your mind and reduce stress.

Remember that it's okay to take time to figure out what works best for you, and that small changes can add up to make a big difference in your sleep and overall well-being. You're taking an important step by reaching out and seeking help, and I'm here to support you. Keep in mind that it might take some experimentation to find the right combination of techniques that work for you, but with patience and persistence, you can develop healthier sleep habits and improve your overall mental health.

Screenshot 3: Family conflicts

User message: I'm having conflicts with my family. How can I handle it?

AI response: I can sense how frustrating and emotionally draining these conflicts with your family must be for you. It's completely understandable to feel overwhelmed and unsure of how to navigate these challenging situations.

Explanatory text: Here are some things that might help:

- Consider setting boundaries with your family members using "I" statements, which can help you express your feelings and needs without placing blame or becoming defensive.
- Take a break from the situation if needed, and use this time to reflect on your emotions and identify patterns that may be contributing to the conflicts.
- Reach out to a trusted friend, family member, or therapist for support and guidance - talking through your feelings with someone who cares about you can be incredibly helpful.
- Try journaling to process your emotions and gain clarity on the issues at hand.

Remember that you're not alone in this, and it's okay to ask for help. By taking small steps towards addressing these conflicts and prioritizing your own emotional well-being, you're taking a powerful step towards healing and growth. Keep in mind that it's a process, and it may take time to see improvements, but with patience, self-care, and support, you can work towards resolving these conflicts and strengthening your relationships with your family.

7. Conclusion:

College Name: VIT BHOPAL UNIVERSITY

Student Name: Aniruddha Bhattacharjee

Email ID: aniruddhabhattacharjee2022@vitbhopal.ac.in

The **Mental Wellness Buddy Chatbot** successfully demonstrates a practical application of **RAG with LLMs** in a high-impact domain. By grounding LLM responses in real, topic-wise curated content, the chatbot delivers accurate, empathetic, and helpful responses to mental health queries. The system shows clear potential for scaling into support tools for educational institutions, therapy assistants, or wellness platforms. The use of LangChain, ChromaDB, and Streamlit ensures modularity and ease of further enhancement.

8. References:

- HuggingFace Transformers Documentation
- LangChain Framework for RetrievalQA and Chroma
- Groq LLM Integration Documentation
- BeautifulSoup & Requests Python Libraries
- GFG and w3schools for pandas and file manipulation techniques
- Cognitive Behavioral Therapy (CBT) and mental health literature from MedicalNewsToday, Healthline
- Streamlit Official Documentation
- Github : https://github.com/Chando0185/Multiverse_of_100-data_science_project_series/tree/main/Mental%20Health%20Chatbot