PROJECT REPPORT

Problem statement: Develop a simple virtual assistant that can perform tasks like setting reminders, answering simple questions, and providing weather information by integrating with an API.

Code:

```
#main.py

from reminder import set_reminder

from weather import get_weather

from database import setup_db, log_to_db

def calculate(expression):

    try:

        result = eval(expression, {"__builtins__": {}})

        return f"The result is: {result}"

    except Exception as e:

        return f"Error in calculation: {e}"

def handle_input(user_input):

    user_input = user_input.lower()

    if user_input.startswith("remind"):

        task = user_input[7:]

        response = set_reminder(task)

    elif "weather" in user_input:

        response = get_weather()

    elif user_input.startswith("calculate"):

        expression = user_input[10:]

        response = calculate(expression)

    elif "hello" in user_input:

        response = "Hello! I'm your assistant."

    else:

        response = "Sorry, I didn't understand that."

    log_to_db(user_input, response)

    return response
```

PROJECT REPPORT

```
if __name__ == "__main__":
    setup_db()
    print("🧠 Virtual Assistant is active. Type 'exit' to quit.")
    while True:
        command = input(">> ")
        if command.lower() in ['exit', 'quit']:
            print("Goodbye!")
            break
        print(handle_input(command))
#reminder.py
import time
from threading import Timer
def notify(task):
    print(f"\n🔔 Reminder: {task}")
def set_reminder(task, delay=10):
    Timer(delay, notify, args=[task]).start()
    return f"Reminder set for '{task}' in {delay} seconds."
#weather.py
import requests
API_KEY = "6d38398c01bcc2b4a86185e348ff5b12"
CITY = "Hyderabad"
def get_weather():
    url = f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}&units=metric"
    try:
        response = requests.get(url).json()
        temp = response["main"]["temp"]
        desc = response["weather"][0]["description"]
        return f"🌤️ Weather in {CITY}: {temp}°C, {desc}"
```

```
    except:
        return "Failed to fetch weather data."
#database.py
import sqlite3
import pandas as pd
import os
def setup_db():
    if not os.path.exists("assistant.db"):
        with sqlite3.connect("assistant.db") as conn:
            conn.execute('''CREATE TABLE history (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        command TEXT,
                        response TEXT,
                        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP)''')
def log_to_db(command, response):
    with sqlite3.connect("assistant.db") as conn:
            conn.execute("INSERT INTO history (command, response) VALUES (?, ?)", (command, response))
        conn.commit()
def export_history():
    with sqlite3.connect("assistant.db") as conn:
        df = pd.read_sql_query("SELECT * FROM history", conn)
        df.to_csv("history.csv", index=False)
        return df
# gui.py
import tkinter as tk
from main import handle_input
from database import export_history, setup_db
def run_gui():
    setup_db()
```

PROJECT REPPORT

```
    root = tk.Tk()
    root.title("🤖 Virtual Assistant")
    root.geometry("400x500")
    root.configure(bg="lightblue")
    history_text = tk.Text(root, height=20, width=48)
    history_text.pack(pady=10)
    input_field = tk.Entry(root, width=40)
    input_field.pack(pady=5)
    def on_submit():
        user_input = input_field.get()
        response = handle_input(user_input)
        history_text.insert(tk.END, f"You: {user_input}\nAssistant: {response}\n\n")
        input_field.delete(0, tk.END)
    submit_btn = tk.Button(root, text="Submit", command=on_submit)
    submit_btn.pack(pady=5)
    def export():
        export_history()
        history_text.insert(tk.END, "✅ History exported to history.csv\n\n")
    export_btn = tk.Button(root, text="Export History", command=export)
    export_btn.pack(pady=5)
    root.mainloop()
if __name__ == "__main__":
    run_gui()
```

Run:

- Console: python main.py
- GUI: python gui.py

API Key:

API Key is from OpenWeatherMap API key in weather.py

PROJECT REPPORT

Tchnologies used:

1. **Python** – Core programming language for building the assistant.
2. **VS Code** – IDE used for writing and running the project.
3. **input()/print()** – Handles user interaction in the console.
4. **eval()** – Evaluates user-entered math expressions.
5. **requests** – Fetches real-time weather data from APIs.
6. **sqlite3** – Stores query and response history in a local database.
7. **pandas** –Generates tabular reports from logged data.
8. **logging** – Records chatbot activity to files.
9. **Tkinter** –Builds a simple GUI interface.
10. **OpenWeatherMap API** – Provides current weather information.
11. **requirements.txt** – Lists project dependencies for installation.

OUTPUT:

🧠 Virtual Assistant is active. Type 'exit' to quit.

>> hello

Hello! I'm your assistant.

>> remind drink water

Reminder set for 'drink water' in 10 seconds.

(After 10 Seconds):

🔔 Reminder: drink water

>> weather

🌤️ Weather in Hyderabad: 32°C, Clear sky.

>> calculate 10 + 5 * 2

The result is: 20

>> calculate (100 - 30) / 7

The result is: 10.0

>> exit

Goodbye