**Project Title: Voice-Activated AI Chatbot**

**Objective:**

Create a voice-activated AI chatbot using Python, capable of responding to a range of commands, conducting online searches, fetching information from Wikipedia, interacting with system commands, and more. This project will demonstrate the integration of voice recognition, Natural Language Processing (NLP), and simple automation using Python.

---

**Instructions:**

1. **Setup**:
    - Use **Python 3.8 or above** and create a virtual environment for dependencies.
    - Install the required libraries using pip install:
        - speech_recognition (for speech-to-text conversion).
        - pyttsx3 (for text-to-speech functionality).
        - wikipedia-api (for Wikipedia integration).
        - webbrowser (for opening URLs).
        - datetime, time, os, ctypes, subprocess, etc., for system commands.
2. **Implement Core Functions**:
    - **Voice Commands**:
        - Use **speech_recognition** to capture and recognize voice input.
        - Create a **takeCommand()** function to handle and process user commands.
    - **Respond with Voice**:
        - Use pyttsx3 to implement a **speak()** function that makes the chatbot respond audibly.
    - **Greet the User**:
        - Create a **wishMe()** function that greets the user based on the time of day.
    - **Handle Commands**:
        - Respond to user queries such as "open Google," "search Wikipedia," "what is the time," etc.
        - Add specific command functions, e.g.,:
            - Open websites.
            - Fetch Wikipedia summaries.
            - Give system commands, like shutdown or restart.

        - Basic interaction responses (like greetings or small talk).
    - **Error Handling**:
        - Include error handling for unrecognized commands or issues in recognizing voice input.
    - **Custom Commands**:
        - Allow users to add simple custom commands, such as writing a note, setting reminders, or taking photos.
3. **Expand Functionalities (Optional)**:
    - Add system automation, like locking or shutting down the computer.
    - Create commands for personal notes or a to-do list.
    - Implement custom greetings and user-specific preferences.
4. **Testing and Optimization**:
    - Test the chatbot with various inputs and refine responses.
    - Optimize the response time by adjusting speech_recognition settings (e.g., pause_threshold).
5. **Submission**:
    - Submit the complete code in a .zip file.
    - Include a file with setup instructions, dependencies, and usage details.