Lyra Discord Bot - Documentation & Setup Guide

This document provides instructions for setting up and running your Lyra Discord Bot, as well as a comprehensive list of commands and features.

Table of Contents

1. Overview

Lyra is a personal Discord AI assistant designed to run locally on your machine. She interacts with you in Direct Messages, leveraging Google's Generative AI to provide helpful responses, understand images, and remember past conversations. You can customize her personality and manage your data.

2. Prerequisites

Before you begin, ensure you have:

- **Python 3.8+:** Download and install from python.org.
- **A Discord Account:** To create and run a bot application.
- **A Google Cloud Project / Gemini API Access:** To obtain a Google API Key for the Generative AI model.

3. Setup Instructions

3.1. Obtain API Keys

1. **Discord Bot Token:**
    - Go to the Discord Developer Portal.
    - Log in and click "New Application."
    - Give your application a name (e.g., "Lyra Bot").
    - Navigate to the "Bot" tab on the left.
    - Click "Add Bot" then "Yes, do it!"
    - Under "Privileged Gateway Intents," enable **"Message Content Intent."** This is crucial for the bot to read message content.
    - Click "Reset Token" and copy the token. **Keep this token secret!**
2. **Google Gemini API Key:**
    - Go to the Google AI Studio or the Google Cloud Console.
    - Create a new API key. Ensure it has access to the Gemini API (generativelanguage.googleapis.com).
    - Copy the generated API key. **Keep this key secret!**

3.2. Project Setup

1. **Create a Project Folder:** Create a new folder on your computer for the bot (e.g., Lyra_Bot).
2. **Save the Python Code:** Save the provided Python code (e.g., Lyra.py or AIV.py) into this folder.
3. **Environment Variables:** You need to set your Discord Bot Token and Google API Key as environment variables.
    - **Recommended (for security and portability):**
        - On Windows (Command Prompt):
          setx DISCORD_BOT_TOKEN "YOUR_DISCORD_BOT_TOKEN_HERE"
          setx GOOGLE_API_KEY "YOUR_GOOGLE_API_KEY_HERE"

          (You might need to restart your terminal for these to take effect).
        - On Linux/macOS (Bash/Zsh):
          export DISCORD_BOT_TOKEN="YOUR_DISCORD_BOT_TOKEN_HERE"
          export GOOGLE_API_KEY="YOUR_GOOGLE_API_KEY_HERE"

          (Add these lines to your ~/.bashrc, ~/.zshrc, or similar for permanent setup).
    - **Alternative (less secure, but quick for testing):** You can directly hardcode them in the Python script at the top, but this is **not recommended** for production:
      # DISCORD_BOT_TOKEN = "YOUR_DISCORD_BOT_TOKEN_HERE"
      # GOOGLE_API_KEY = "YOUR_GOOGLE_API_KEY_HERE"

(Uncomment these lines and replace placeholders if you choose this method).

3.3. Create lyra_config.ini

In your Lyra_Bot project folder, create a file named lyra_config.ini with the following content:

```
[Paths]
UserData = ./user_memories
```

- UserData: This specifies where Lyra will store user-specific data (conversation history, personality settings, and images).
  - ./user_memories (default): This creates a folder named user_memories right next to your Lyra.py script or executable.
  - You can change this to an absolute path (e.g., C:\LyraData on Windows or /home/user/LyraData on Linux) if you prefer to store the data elsewhere. Make sure the specified directory exists or Lyra has permissions to create it.

3.4. Install Dependencies

1. **Open your terminal or command prompt.**
2. **Navigate to your project folder:**
   cd path/to/your/Lyra_Bot_folder
3. **Create a virtual environment (recommended):**
   python -m venv .venv
4. **Activate the virtual environment:**
   - Windows (Command Prompt): .\.venv\Scripts\activate.bat
   - Windows (PowerShell): .\.venv\Scripts\Activate.ps1
   - Linux/macOS: source ./.venv/bin/activate
5. **Install required Python libraries:**
   pip install discord.py google-generativeai Pillow configparser python-dotenv

3.5. Run the Bot

1. Ensure your virtual environment is active (from step 3.4).
2. Run the Python script:
   python Lyra.py

3. You should see output in your terminal indicating that Lyra has logged in.

3.6. Create an Executable (Optional)

If you want to run Lyra as a standalone executable (e.g., .exe on Windows) without needing Python installed everywhere:

1. Ensure your virtual environment is active (from step 3.4).
2. Install PyInstaller:
   pip install pyinstaller

3. Generate the executable:
   pyinstaller --onefile --hidden-import="PIL._ImagePlugin" Lyra.py

   (Replace Lyra.py with AIV.py if that's your filename).
   - The --hidden-import="PIL._ImagePlugin" flag is often necessary for Pillow (PIL) to work correctly when bundled with PyInstaller for image processing.
4. The executable will be in the dist folder within your project directory.
5. **Important:** Remember to place the lyra_config.ini file in the *same directory* as the executable if you create one, as the bot looks for it relative to its own location.

4. Lyra Commands

Lyra primarily responds in Direct Messages (DMs). All commands start with !.

4.1. General Commands

- !help: Displays a list of all available commands for Lyra.

4.2. Personality Customization (Traits)

These commands allow you to set specific traits for Lyra's responses.

- !set_persona <trait> <value>: Sets a specific personality trait for Lyra for your conversations.
  - **Usage:** !set_persona <trait> <value>
  - **Example:** !set_persona tone humorous
  - **Available Traits and Values (case-insensitive):**
    - **tone**: friendly, formal, humorous, serious, empathetic
    - **verbosity**: concise, standard, detailed
    - **creativity**: low, standard, high
    - **humor**: none, low, moderate, high
    - **formality**: informal, standard, formal
- !show_persona: Shows your current custom personality settings for Lyra (trait-based).

- !reset_persona: Resets all your custom trait-based personality settings for Lyra to her defaults.

4.3. Advanced Personality Customization (Custom Prompt)

These commands allow for more free-form personality guidance.

- !set_custom_persona <your custom prompt here>: Sets a free-form custom personality prompt for Lyra. This will be added to Lyra's instructions for your conversations.
  - **Usage:** !set_custom_persona Always respond as a pirate with a treasure map in hand.
- !show_custom_persona: Shows your current free-form custom personality prompt for Lyra. It will indicate if you're using a custom prompt or the default one.
- !reset_custom_persona: Resets your free-form custom personality prompt for Lyra to the default built-in prompt.

4.4. Data Management Commands

These commands allow you to manage the data Lyra stores about your interactions.

- !delete_all_data: Deletes all your user data (conversation history, personality settings, custom prompt, and saved images). **Requires confirmation.**
- !download_my_data: Generates a .zip file of all your Lyra data and sends it to you.
- !delete_my_images: Deletes only your saved images. **Requires confirmation.**
- !delete_my_messages: Deletes only your conversation text history. **Requires confirmation.**

5. Data Storage & Privacy

Please refer to the PRIVACY_POLICY.md file for detailed information on how Lyra handles your data, including what is collected, how it's used, where it's stored, and your rights.

6. Troubleshooting

- **Bot not coming online:**
  - Check your DISCORD_BOT_TOKEN and GOOGLE_API_KEY environment variables. Ensure they are correct and properly set.
  - Verify "Message Content Intent" is enabled in your Discord Developer Portal bot settings.
  - Check your terminal for any error messages during bot startup.
- **Bot not responding in DMs:**
  - Ensure the bot is online.

    - Verify "Message Content Intent" is enabled.
    - Check the console running the bot for any errors after sending a message.
- **Commands not working or throwing errors:**
    - Ensure you are using the correct command prefix (!).
    - Refer to !help and !help_persona for correct command usage and arguments.
    - Check the console for CommandInvokeError or NameError messages, which provide clues.
- **Image processing issues:**
    - Ensure the Pillow library is correctly installed (pip install Pillow).
    - Check the user_memories/images/<your_user_id>/ directory for permission issues or if images are being saved correctly.
    - Large image files might cause timeouts or processing issues with the API.