**Gemini Discord Bot**

Gemini Discord Bot is an advanced, multimodal Discord bot leveraging Google Generative AI capabilities. Besides text responses, this bot can read images, listen to audio files, watch videos, and interpret text files. This makes Gemini one of the most versatile AI assistants available for Discord.

**Features**

* **Multimodal Capabilities:** Interacts with images, videos, audio files, and text files.
* **Image Generation:** Generates images based on user prompts using multiple models like SD-XL, Playground, and more.
* **Story Generation:** Generates stories based on user prompts with a desired length.
* **Poem Generation:** Generates poems based on user prompts to give the theme of the poem.
* **Notifications:** Generates notifications that allows the user to keep track of events they wish to keep tabs on.
* **Alarms:** Generates alarm and messages based on user input with editing features such as viewing and deleting.
* **Speech Generation:** Converts text to speech in multiple languages.
* **Music Generation:** Generates music based on textual prompts.
* **Custom Personalities:** Supports creating custom personalities for users and servers.
* **Advanced Interaction:** Supports advanced user and server settings, including chat history management, response formatting, QNA, and help manual.

**Getting Started**

**Prerequisites**

* Node.js (v14 or later)
* Discord Bot account & token ([create a bot on Discord Developer Portal](https://discord.com/developers/applications))
* Google AI Studio & API key ([create a project on Google AI Studio](https://aistudio.google.com/apikey))

**Installation**

1. Clone the repository:

git clone https://github.com/keatkean/Gemini-Discord-Bot

cd Gemini-Discord-Bot

1. Install dependencies:

npm install @google/generative-ai axios cheerio discord.js dotenv eventsource fs sharp office-text-extractor youtube-transcript node-os-utils ws mathjs

1. Set up environment variables:

Edit the .env file in the root directory and add your Discord bot token and Google API key:

DISCORD\_BOT\_TOKEN=your\_discord\_bot\_token

GOOGLE\_API\_KEY=your\_google\_api\_key

CLIENT\_ID=your\_discord\_bot\_client\_ID

1. Setting up Discord Developer Portal

Refer to Discord Bot Set Up Instructions below

1. Configure the bot:

Edit the config.json file as per your preferences.

1. Start the bot:

npm start

**Discord Bot Set Up (Discord Developer Portal)**

* 1. Create New Application under Applications (Login to your Discord Account)

Name it “PEM PAL”

* 1. Finding Discord Token and Client ID:

Discord Token: Bot > Token > Reset Token

After resetting the token, save it in the .env file. You shouldn’t need to reset the token again after this step.

Client ID: OAuth2 > Client Information > Client ID

Save the Client ID in the .env file.

* 1. Setting up the Bot

Under Bot > Privileged Gateway Intents

Toggle “Presence Intent”, “Server Members Intent” and “Message Content Intent” from OFF to ON

* 1. Installing the Bot into your server

Go to Discord, create a NEW server in Discord

Return to the Discord Developer Portal, Under Installation > Install Link

**!!!IMPORTANT!!**

Using the Discord Provided Link, copy it and add this behind:

**“&scope=bot%20applications.commands&permissions=2147614728”**

This line allows the bot to use slash commands and join servers as an Application along with the permission integers that is needed to run the Bot

Discord doesn’t add it by default and without it, the bot will not run in your Discord Server.

Your Install Link should look like this:

https://discord.com/oauth2/authorize?client\_id=YOUR\_BOT\_ID**&scope=bot%20applications.commands&permissions=2147614728**

Put this into your browser and select the Discord Server you just created.

* 1. Return to Installation Instructions

**Commands**

**AI Generation**

**/imagine [prompt] [model] [resolution]: Generate an image using a selected model and resolution.**

**/speech [language] [prompt]: Generate speech from text in the specified language.**

**/music [prompt]: Generate music based on a prompt.**

**/write\_story [prompt] [length]: Generate a story based on the prompt with a choice of length.**

**/write\_poem [theme]: Generate a poem based on the theme given.**

**Chat Management**

**/respond\_to\_all: Enable the bot to respond to all messages in the channel.**

**/clear\_memory: Clear the conversation history.**

**/settings: Open user settings.**

**/server\_settings: Open server settings.**

**Admin Commands**

**/blacklist [user]: Blacklist a user from using certain interactions.**

**/whitelist [user]: Remove a user from the blacklist.**

**/status: Display bot's CPU and RAM usage in detail.**

**Assistance Commands**

**/questions: Generates a list of frequently asked questions.**

**/help: Generates the manual for PEM PAL.**

**Functions Commands**

**/alarm [date time message]: Generates an alarm for the user.**

**/view\_alarms: Allow users to view their alarms.**

**/delete\_alarm [number]: Allow users to delete their alarms.**

**/notify [message] [time]: Bot sets a notification for an event.**

**📢 Announcement Management**

**/announce [time] [title] [message] [date] [links] [ping\_role]: Schedule an announcement at a specific time and date with optional links and role mentions.**

**/listannouncements: View all scheduled announcements, including their date, time, and the role being pinged.**

**/removeannouncement [index]: Remove a scheduled announcement by specifying its index.**

**📊 Poll Management**

**/poll [question] [options] [anonymous]: Create a poll with multiple options with choice to hide voters choices.**

**/viewpolls: View all active polls along with their details.**

**/clearpolls: Clear all active polls from memory and storage.**

**/endpoll [poll\_id]: Manually end a specific poll before its duration expires.**

**🔞 NSFW Word Filtering & Logging System**

**/viewnsfwlogs: Displays a list of all NSFW word violations in a private embed message. Only users with Administrator permissions can use this.**

**/clearnsfwlogs: Permanently clears all NSFW logs from memory and storage. Also admin-only.**

**Configuration**

**config.json**

This file contains various settings for the bot:

* **defaultResponseFormat**: The default response format for the bot. (embedded or normal)
* **hexColour**: The default hex color for embeds.
* **defaultImgModel**: The default image generation model.
* **defaultUrlReading**: Default URL reading preference (ON or OFF).
* **workInDMs**: Whether the bot should work in DMs (true or false).
* **shouldDisplayPersonalityButtons**: Whether to display personality buttons in settings (true or false).
* **SEND\_RETRY\_ERRORS\_TO\_DISCORD**: Whether to send retry errors to Discord channel (true or false).
* **activities**: An array of activities for the bot to display.
* **defaultPersonality**: The default personality instructions for the bot.
* **defaultServerSettings**: The default settings for the server.

**NSFW Word Filtering (nsfwWords.json)**

* Contains an array of words that should be filtered from prompts.

**Notes/Tips for Future Batches.**

1. Use Visual Studio Code, download Github by KnisterPeter under Extensions.

Helps with cloning the GitHub repository and allows you to commit etc.

1. All functions and commands work as listed in the manual (/help)
2. For Image and Music generation, give it 1 to 2 minutes to generate your prompt
3. Take note of NSFW filter (Might need testing/fixing again)
4. If the bot doesn’t work, try deleting node\_modules and reinstalling dependencies.
5. Console log may require tidying up (notify function)
6. This line may appear on the console log, it is fine.

“ (node:23612) Warning: Supplying "ephemeral" for interaction response options is deprecated. Utilize flags instead.

(Use `node --trace-warnings ...` to show where the warning was created) “

**How to Install Docker on Your PC**

Follow the instructions below based on your operating system to install and verify Docker.

**🔵 For Windows**

**Step 1: Download Docker Desktop**

* Visit the official Docker download page:  
  👉 [Download Docker for Windows](https://www.docker.com/products/docker-desktop/)
* Click **"Download Docker Desktop for Windows"**.

**Step 2: Install Docker**

* Run the downloaded Docker Desktop Installer.exe.
* Follow the installation wizard:
  + Select **“Use WSL 2”** if prompted (recommended for better performance).
  + Click **Next** and then **Install**.
* Restart your PC if prompted.

**Step 3: Verify Installation**

* Open **Command Prompt** or **PowerShell** and run:

docker --version

✅ If Docker is installed successfully, it will return something like:

Docker version 24.x.x, build xxxxxxx

**Step 4: Run a Test Container**

* Open Docker Desktop and wait for it to initialize.
* Make sure the **Docker Engine is running**.
* Open a terminal and run:

docker run hello-world

✅ If everything is working, Docker will display a success message.

**🍎 For macOS**

**Step 1: Download Docker Desktop**

* Go to:  
  👉 [Download Docker for Mac](https://www.docker.com/products/docker-desktop/)
* Click **“Download Docker Desktop for Mac”**.

**Step 2: Install Docker**

* Open the downloaded .dmg file.
* Drag and drop Docker into the **Applications** folder.
* Launch Docker and follow the on-screen setup.

**Step 3: Verify Installation**

* Open **Terminal** and run:

docker --version

* Test Docker by running:

docker run hello-world

✅ Docker is working if it shows a confirmation message.

**🐧 For Linux (Ubuntu/Debian-based)**

**Step 1: Install Required Packages**

sudo apt update

sudo apt install -y ca-certificates curl gnupg

**Step 2: Add Docker’s Official GPG Key**

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/keyrings/docker.asc > /dev/null

sudo chmod a+r /etc/apt/keyrings/docker.asc

**Step 3: Set Up Docker Repository**

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update

**Step 4: Install Docker**

sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

**Step 5: Verify Installation**

docker --version

Test Docker:

sudo docker run hello-world

✅ Docker is installed correctly if you see a success message.

**How To Deploy Bot Via Docker:**  
  
**1. Create a Dockerfile**

Inside your project root directory, create a file named Dockerfile and add the following:

# Use Node.js base image

FROM node:18-alpine

# Set the working directory inside the container

WORKDIR /app

# Copy package.json and package-lock.json first (to leverage caching)

COPY package\*.json ./

# Install dependencies

RUN npm install

# Copy the rest of the project files

COPY . .

# Copy the environment file (example.env) into the container

COPY example.env /app/.env

# Expose the port (if your bot uses a web server, otherwise omit this line)

EXPOSE 3000

# Clear old bot instances and start a new one

CMD ["sh", "-c", "docker stop pem-pal-disc-bot || true && docker rm pem-pal-disc-bot || true && node index.js"]

**2. Create a .dockerignore File**

To prevent unnecessary files from being copied into the container, create a .dockerignore file:

node\_modules

npm-debug.log

.DS\_Store

.env

.git

**3. Build the Docker Image**

Open a terminal in the project directory and run:

docker build -t gemini-discord-bot .

**4. Run the Docker Container**

Run the container with:

docker run -d --name pem-pal-disc-bot --env-file example.env -v ${PWD}\config:/app/config gemini-discord-bot

**Note:** Ensure that example.env contains your necessary environment variables like the **Discord bot token**.

**5. Check Running Container**

Verify if the container is running:

docker ps

If you need to check the logs:

docker logs pem-pal-disc-bot

**6. Stop and Remove the Container (if needed)**

To stop the container:

docker stop pem-pal-disc-bot

To remove the container:

docker rm pem-pal-disc-bot

**Next Steps**

* If you want to **automate** restarts, use:

docker run -d --restart unless-stopped --env-file example.env gemini-discord-bot

* If you need to make updates, **rebuild** and **restart** the container:

docker build -t gemini-discord-bot .

docker stop pem-pal-disc-bot

docker rm pem-pal-disc-bot

docker run -d --name pem-pal-disc-bot --env-file example.env -v ${PWD}\config:/app/config gemini-discord-bot

**(optional) 🚀 Step-by-Step: Deploy on Railway.app ($5.00 trial)**

**Gmail account used:  
Email:** [**pempaldiscordbot@gmail.com**](mailto:pempaldiscordbot@gmail.com)

**Password: $PemPalDiscordBot47**

**Step 0 (optional depending on your pc settings):**

**Enable Script Execution in PowerShell If disabled.**

**🛠 Step-by-Step**

1. **Open PowerShell as Administrator**:
   * Press **Windows Key**, type powershell.
   * Right-click **"Windows PowerShell"** → **Run as Administrator**.
2. **Allow Script Execution Temporarily** (safer): Run this in the admin PowerShell:

Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

This **only changes the policy for the current session**, so it's safe.

1. **Try running your Railway command again**:

railway init

**✅ Alternatively: Allow Scripts Permanently (if you're okay with it)**

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

This allows scripts you create or download to run (if they are signed). Restart your terminal after this.

**🧼 Want to reset later?**

You can always revert with:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Restricted

**Now onto the actual railway steps.**

**1️⃣ Create an Account & Install Railway CLI**

* Go to [**Railway.app**](https://railway.app)
* Sign up with **GitHub or Google (In this case use google sign in with the above gmail credentials)**.
* Install the Railway CLI:

npm install -g @railway/cli

**2️⃣ Initialize Your Project**

* Open your project folder in **VS Code**.
* Run:

railway init

* + Follow the prompts to **create a new project**.

**3️⃣ Deploy Your Discord Bot**

* Run:

railway up

* + This will **deploy your bot** and start it on Railway.

**4️⃣ Set Up Environment Variables**

* Go to your **Railway Dashboard** → Select Your Project
* Click **"Variables"** and add:

DISCORD\_TOKEN=your-discord-bot-token

**5️⃣ Keep the Bot Running**

* Railway **keeps containers running**.
* If it **stops** (free tier limits), you can restart it using:

railway up