Discord Bot with Music Playback & Moderation Capabilities

Complete Development Guide for Building a Feature-Rich Bot Like LunaBot and Wick

Executive Summary

This comprehensive guide provides step-by-step instructions for creating a Discord bot that combines **music playback capabilities** from YouTube (similar to LunaBot) with **advanced moderation features** (similar to Wick Bot), along with an **interactive button-based interface**. The bot is built using Python 3.8+, discord.py 2.3+, yt-dlp for YouTube audio extraction, and FFmpeg for audio processing [1] [2] [3].

Given your background in AI development and desktop application creation, this bot leverages similar architectural patterns to your previous voice assistant and document processing projects, but adapted for the Discord platform [User Context].

Table of Contents

1. Core Features Overview

Music Playback Features (LunaBot-Inspired)

- **YouTube Integration**: Stream audio directly from YouTube URLs or search queries [1] [4] [5]
- **Queue Management**: Multi-song queue system with per-server isolation [6] [7]
- **Interactive Controls**: Play, pause, resume, skip, and stop buttons [4] [8] [9]
- **Rich Embeds**: Visual feedback with song information and playback status [1] [10]
- **Multi-Platform Support**: Works across multiple Discord servers simultaneously [11] [12]

Moderation Features (Wick-Inspired)

- **Command Suite**: Ban, kick, timeout/mute, and message purge commands [13] [14] [15]
- **Auto-Moderation**: Automatic profanity detection and content filtering [16] [17] [18]
- **Warning System**: Three-strike system with escalating punishments [19] [20]
- **Infractions Tracking**: Per-user violation logging [19]
- **Slash Commands**: Modern Discord command interface [21] [10] [22]

Interactive Interface

- **Persistent Buttons**: Control music playback through clickable UI elements [8] [9] [23]
- **Embedded Messages**: Rich, formatted messages with visual appeal [10] [24]
- **Real-time Updates**: Dynamic queue displays and status information [4]

2. Technical Architecture

System Components

The bot architecture consists of three primary systems working in concert:

Music System

- **yt-dlp**: YouTube audio extraction and streaming [25] [26] [27]
- **FFmpeg**: Audio encoding, decoding, and format conversion [1] [2] [28]
- **Queue Manager**: Per-guild song queue with loop functionality [6] [7]

Command Handler

- **Slash Commands**: Discord's native application command system [21] [10] [22]
- **Button Events**: Interactive UI component handlers [8] [9] [23]
- **Permission Checks**: Role-based access control for moderation [15] [19]

Moderation System

- **Profanity Filter**: Real-time message scanning using better-profanity library [16] [17] [18]
- **Infractions Database**: In-memory tracking of user violations [19] [18]
- **Auto-Punishment**: Automated timeout after threshold violations [29] [20]

Multi-Server Management

- **Guild-Specific Queues**: Dictionary-based queue isolation per server [6] [7] [12]
- **Event Loop**: Asynchronous handling of concurrent server events [11]

[chart:61]

3. Prerequisites & Installation

System Requirements

Python Version: 3.8 or higher required
Operating System: Windows, macOS, or Linux
Memory: Minimum 512MB RAM (1GB+ recommended for multiple servers)
Storage: 100MB for dependencies and cache

Installing FFmpeg

FFmpeg is critical for audio processing and must be installed separately from Python packages [1] [2].

Windows Installation:

1. Download FFmpeg from https://ffmpeg.org/download.html
2. Extract to `C:\ffmpeg`
3. Add `C:\ffmpeg\bin` to system PATH environment variable
4. Verify installation: Open Command Prompt and run `ffmpeg -version`

Linux (Ubuntu/Debian):

```
sudo apt update
sudo apt install ffmpeg
```

macOS:

```
brew install ffmpeg
```

Python Dependencies

Create a `requirements.txt` file with the following packages [1] [2] [30]:

```
discord.py&gt;=2.3.0
yt-dlp&gt;=2024.0.0
python-dotenv&gt;=1.0.0
better-profanity&gt;=0.7.0
PyNaCl&gt;=1.5.0
```

Install all dependencies:

```
pip install -r requirements.txt
```

Library Purposes:

- discord.py: Official Discord API wrapper for Python [1] [2]
- **yt-dlp**: YouTube audio extraction (successor to youtube-dl) [1] [25] [26]
- **python-dotenv**: Environment variable management for secure token storage [1]
- **better-profanity**: Fast profanity detection library [16] [17]
- **PyNaCl**: Voice support for Discord connections [1]

4. Bot Setup & Configuration

Creating the Discord Application

1. **Access Developer Portal**: Navigate to https://discord.com/developers/applications [1] [2]
2. **Create Application**: Click "New Application" and provide a bot name
3. **Add Bot User**: Go to "Bot" tab → Click "Add Bot" → Confirm creation
4. **Enable Intents**: Under "Privileged Gateway Intents", enable:
   - SERVER MEMBERS INTENT
   - MESSAGE CONTENT INTENT [2]
5. **Copy Token**: Click "Reset Token" → Copy and securely store the token [1] [31]

Setting Bot Permissions

Navigate to OAuth2 → URL Generator and select:

Scopes:

- `bot`
- `applications.commands`

Bot Permissions [1] [2]:

- Send Messages
- Manage Messages
- Connect (for voice)
- Speak (for voice)
- Use Slash Commands
- Kick Members
- Ban Members
- Moderate Members (for timeout)

Copy the generated URL and use it to invite the bot to your server.

Environment Configuration

Create a `.env` file in your project directory to store sensitive credentials securely [1] [31]:

```
BOT_TOKEN=your_bot_token_here
GUILD_ID=your_server_id_here
```

To obtain Guild ID:

1. Enable Developer Mode in Discord: Settings → Advanced → Developer Mode
2. Right-click your server icon → Copy ID

5. Music System Implementation

YouTube Audio Extraction

The music system uses yt-dlp to extract audio streams from YouTube without downloading full video files [1] [25] [26]:

```
import yt_dlp as youtube_dl
import discord
import asyncio

# Configure yt-dlp options
ytdl_format_options = {
    'format': 'bestaudio/best',
    'outtmpl': '%(extractor)s-%(id)s-%(title)s.%(ext)s',
    'restrictfilenames': True,
    'noplaylist': False,
    'nocheckcertificate': True,
    'ignoreerrors': False,
    'logtostderr': False,
    'quiet': True,
    'no_warnings': True,
    'default_search': 'auto',
    'source_address': '0.0.0.0'
}

ffmpeg_options = {
    'before_options': '-reconnect 1 -reconnect_streamed 1 -reconnect_delay_max 5',
    'options': '-vn'
}

ytdl = youtube_dl.YoutubeDL(ytdl_format_options)
```

The configuration prioritizes best audio quality while avoiding video streams (`-vn` flag), reducing bandwidth usage [1] [2].

Queue Management System

Each Discord server (guild) requires an isolated music queue to prevent cross-server interference [6] [7] [12]:

```
class MusicQueue:
    def __init__(self):
        self.queue = []
        self.current = None
        self.loop = False

    def add(self, song):
        self.queue.append(song)

    def next(self):
        if self.loop and self.current:
            return self.current
        if self.queue:
            self.current = self.queue.pop(0)
            return self.current
        return None

    def clear(self):
        self.queue.clear()
        self.current = None

# Bot-level queue storage
music_queues = {}  # Dictionary: guild_id -&gt; MusicQueue
```

This architecture ensures that playing music on Server A doesn't affect Server B's queue [11] [12].

Play Command Implementation

The `/play` command handles both YouTube URLs and search queries [1] [10]:

```
@bot.tree.command(name="play", description="Play a song from YouTube")
@app_commands.describe(query="The song name or YouTube URL")
async def play(interaction: discord.Interaction, query: str):
    await interaction.response.defer()

    # Verify user is in voice channel
    if not interaction.user.voice:
        await interaction.followup.send("You need to be in a voice channel!")
        return

    # Get or create queue for this guild
    guild_id = interaction.guild.id
    if guild_id not in bot.music_queues:
        bot.music_queues[guild_id] = MusicQueue()

    # Connect to voice channel
    voice_client = interaction.guild.voice_client
    if not voice_client:
        channel = interaction.user.voice.channel
        voice_client = await channel.connect()

    # Search YouTube if not a URL
    if not query.startswith('http'):
        query = f"ytsearch:{query}"

    player = await YTDLSource.from_url(query, loop=bot.loop)
    bot.music_queues[guild_id].add(player)

    # Create interactive control panel
    embed = discord.Embed(
        title="▶ Added to Queue",
        description=f"**{player.title}**",
        color=discord.Color.green()
    )

    view = MusicControlView(bot, guild_id)

    if not voice_client.is_playing():
        await play_next(interaction.guild)
        embed.title = "▶ Now Playing"

    await interaction.followup.send(embed=embed, view=view)
```

Interactive Music Controls

Persistent button interface for playback control [8] [9] [23]:

```
class MusicControlView(discord.ui.View):
    def __init__(self, bot, guild_id):
        super().__init__(timeout=None)
        self.bot = bot
        self.guild_id = guild_id

    @discord.ui.button(label="⏸ Pause", style=discord.ButtonStyle.primary)
    async def pause_button(self, interaction: discord.Interaction, button: discord.ui.Butto
        voice_client = interaction.guild.voice_client
        if voice_client and voice_client.is_playing():
```

```
            voice_client.pause()
            await interaction.response.send_message("⏸ Paused!", ephemeral=True)
        else:
            await interaction.response.send_message("Nothing playing!", ephemeral=True)

    @discord.ui.button(label="▶ Resume", style=discord.ButtonStyle.success)
    async def resume_button(self, interaction: discord.Interaction, button: discord.ui.But
        voice_client = interaction.guild.voice_client
        if voice_client and voice_client.is_paused():
            voice_client.resume()
            await interaction.response.send_message("▶ Resumed!", ephemeral=True)

    @discord.ui.button(label="⏭ Skip", style=discord.ButtonStyle.secondary)
    async def skip_button(self, interaction: discord.Interaction, button: discord.ui.Butto
        voice_client = interaction.guild.voice_client
        if voice_client and voice_client.is_playing():
            voice_client.stop()
            await interaction.response.send_message("⏭ Skipped!", ephemeral=True)

    @discord.ui.button(label="⏹ Stop", style=discord.ButtonStyle.danger)
    async def stop_button(self, interaction: discord.Interaction, button: discord.ui.Butto
        voice_client = interaction.guild.voice_client
        if voice_client:
            if self.guild_id in self.bot.music_queues:
                self.bot.music_queues[self.guild_id].clear()
            voice_client.stop()
            await voice_client.disconnect()
            await interaction.response.send_message("⏹ Stopped!", ephemeral=True)
```

Buttons persist across bot restarts when using custom_id parameters [8] [23].

6. Moderation System Implementation

Automatic Profanity Filter

Real-time message scanning with automatic deletion and warning system [16] [17] [18]:

```
from better_profanity import profanity
from datetime import timedelta

profanity.load_censor_words()
infractions = {}  # user_id -&gt; warning_count

@bot.event
async def on_message(message):
    if message.author.bot:
        return

    # Check for profanity
    if profanity.contains_profanity(message.content):
        await message.delete()

        user_id = message.author.id
        infractions[user_id] = infractions.get(user_id, 0) + 1

        warning_msg = await message.channel.send(
            f"⚠ {message.author.mention} Warning {infractions[user_id]}/3"
        )
        await asyncio.sleep(5)
        await warning_msg.delete()

        # Auto-timeout after 3 warnings
```

```
        if infractions[user_id] &gt;= 3:
            await message.author.timeout(
                timedelta(minutes=10),
                reason="Multiple profanity violations"
            )
            await message.channel.send(
                f"{message.author.mention} timed out for 10 minutes"
            )
            infractions[user_id] = 0

    await bot.process_commands(message)
```

This implements a three-strike system similar to Wick Bot's heat-based moderation [14] [19] [20].

Moderation Commands

Ban Command [15] [19] [29]:

```
@bot.tree.command(name="ban", description="Ban a member")
@app_commands.describe(member="Member to ban", reason="Reason")
@app_commands.checks.has_permissions(ban_members=True)
async def ban(interaction: discord.Interaction, member: discord.Member,
              reason: Optional[str] = "No reason provided"):
    try:
        await member.ban(reason=reason)
        embed = discord.Embed(
            title="🔨 Member Banned",
            description=f"**{member}** banned\\n**Reason:** {reason}",
            color=discord.Color.red()
        )
        await interaction.response.send_message(embed=embed)
    except Exception as e:
        await interaction.response.send_message(
            f"Failed: {str(e)}", ephemeral=True
        )
```

Timeout Command [15] [19] [20]:

```
@bot.tree.command(name="timeout", description="Timeout a member")
@app_commands.describe(member="Member to timeout", duration="Minutes")
@app_commands.checks.has_permissions(moderate_members=True)
async def timeout(interaction: discord.Interaction, member: discord.Member,
                  duration: int, reason: Optional[str] = "No reason"):
    await member.timeout(timedelta(minutes=duration), reason=reason)
    embed = discord.Embed(
        title="⏱ Member Timed Out",
        description=f"**{member}** timed out for {duration} minutes",
        color=discord.Color.yellow()
    )
    await interaction.response.send_message(embed=embed)
```

Message Purge [15] [19] [29]:

```
@bot.tree.command(name="purge", description="Delete messages")
@app_commands.describe(amount="Number to delete (1-100)")
@app_commands.checks.has_permissions(manage_messages=True)
async def purge(interaction: discord.Interaction, amount: int):
    if amount &lt; 1 or amount &gt; 100:
        await interaction.response.send_message(
            "Amount must be 1-100!", ephemeral=True
        )
```

```
        return

    await interaction.response.defer(ephemeral=True)
    deleted = await interaction.channel.purge(limit=amount)
    await interaction.followup.send(
        f"Deleted {len(deleted)} messages!", ephemeral=True
    )
```

7. Interactive UI Components

Button Styling Options

Discord.py offers four button style options [8] [9]:

- **Primary** (Blue): Main actions like pause/play
- **Success** (Green): Positive actions like resume
- **Secondary** (Gray): Neutral actions like skip
- **Danger** (Red): Destructive actions like stop/delete

Embed Messages

Rich formatted messages enhance user experience [10] [24]:

```
embed = discord.Embed(
    title="▶ Now Playing",
    description=f"**{song.title}**",
    color=discord.Color.blue()
)
embed.add_field(name="Duration", value="3:45", inline=True)
embed.add_field(name="Requested by", value=interaction.user.mention, inline=True)
embed.set_thumbnail(url=thumbnail_url)
embed.set_footer(text=f"Queue position: 1/{queue_length}")

await interaction.response.send_message(embed=embed, view=view)
```

8. Multi-Server Management

Guild-Specific State Management

The bot must maintain separate state for each Discord server to prevent interference [11] [7] [12]:

```
# Global state dictionaries
music_queues = {}      # guild_id -&gt; MusicQueue
infractions = {}       # user_id -&gt; warning_count

@bot.event
async def on_guild_join(guild):
    # Initialize state for new server
    music_queues[guild.id] = MusicQueue()
    print(f"Joined new guild: {guild.name}")

@bot.event
async def on_guild_remove(guild):
    # Cleanup when removed from server
    if guild.id in music_queues:
```

```
        del music_queues[guild.id]
    print(f"Removed from guild: {guild.name}")
```

Event Handling Across Servers

Discord.py automatically queues and processes events from multiple servers sequentially using Python's asyncio event loop [11] [12]. No manual threading or multiprocessing is required.

9. Security Best Practices

Token Protection

Never expose your bot token in source code or public repositories [31] [32] [33]. Always use environment variables:

```
import os
from dotenv import load_dotenv

load_dotenv()
token = os.getenv('BOT_TOKEN')

if not token:
    print("ERROR: BOT_TOKEN not found in environment")
    exit(1)

bot.run(token)
```

Add `.env` to your `.gitignore` file to prevent accidental commits [31].

Input Validation

Prevent command injection and abuse through strict input validation [31] [32]:

```
class InputValidator:
    @staticmethod
    def sanitize_string(input_str, max_length=1000):
        if not isinstance(input_str, str):
            raise ValueError("Invalid input type")

        # Remove dangerous characters
        sanitized = input_str.trim()[:max_length]
        sanitized = sanitized.replace('&lt;', '').replace('&gt;', '')
        sanitized = sanitized.replace('@everyone', '').replace('@here', '')
        return sanitized

    @staticmethod
    def validate_duration(minutes):
        if not isinstance(minutes, int):
            return False
        return 1 &lt;= minutes &lt;= 10080  # Max 1 week
```

Rate Limiting

Prevent spam and abuse with per-user rate limits [31] [32]:

```
from datetime import datetime

rate_limits = {}  # user_id -&gt; [timestamp, timestamp, ...]
```

```
def is_rate_limited(user_id, max_requests=5, window_seconds=60):
    now = datetime.now()
    if user_id not in rate_limits:
        rate_limits[user_id] = []

    # Remove expired timestamps
    rate_limits[user_id] = [
        ts for ts in rate_limits[user_id]
        if (now - ts).total_seconds() &lt; window_seconds
    ]

    if len(rate_limits[user_id]) &gt;= max_requests:
        return True

    rate_limits[user_id].append(now)
    return False
```

Permission Hierarchy

Ensure bot role is positioned correctly in server role hierarchy [34] [35]:

1. Bot role must be **higher** than any role it needs to moderate
2. Protected roles (admin/mod) should be **higher** than bot role to prevent abuse
3. Check role positions before executing moderation commands

10. Deployment & Hosting

Local Testing

Run the bot locally for development:

```
python discord_bot_complete.py
```

Keep terminal open to maintain bot connection [1] [2].

Free Hosting Options (2025)

Render (Recommended) [36] [37] [38]:

- Free tier: 750 hours/month
- Automatic HTTPS
- GitHub integration
- **Limitation**: Goes to sleep after 15 minutes of inactivity
- **Workaround**: Use uptime monitoring service

Oracle Cloud Free Tier [39]:

- Always-free VM instances
- True 24/7 hosting
- Requires manual server setup
- More technical but reliable

Railway [39]:

- Free tier with monthly limits

- Easy deployment
- Can exhaust limits quickly with music bots

Keeping Bot Online 24/7

Use **UptimeRobot** or similar services to ping your bot every 5 minutes, preventing sleep mode [36] [37] [38]:

1. Create health check endpoint in bot:

```
from flask import Flask
import threading

app = Flask('')

@app.route('/')
def home():
    return "Bot is alive!", 200

def run():
    app.run(host='0.0.0.0', port=8080)

def keep_alive():
    t = threading.Thread(target=run)
    t.start()

# Call before bot.run()
keep_alive()
bot.run(token)
```

2. Register bot URL with UptimeRobot to ping every 5 minutes [36] [37] [38]

Production Considerations

For serious production use, consider paid hosting ($5-10/month) for:

- Reliable 24/7 uptime
- Better performance for music streaming
- No cold starts or sleep timers
- Dedicated support [40] [39]

11. Troubleshooting & Maintenance

Common Issues

Slash Commands Not Appearing [21] [41]:

- Wait 1-2 hours for global command sync
- Verify bot has `applications.commands` scope
- Try guild-specific commands first (faster sync)

Music Not Playing [2] [42] [43]:

- Verify FFmpeg is installed and in PATH
- Check bot has "Connect" and "Speak" permissions
- Ensure user is in voice channel before playing

Moderation Commands Failing [29] [34]:

- Check bot role is above target role in hierarchy
- Verify bot has required permissions (Ban/Kick/Moderate Members)
- Enable 2FA if server requires it for moderation actions

Bot Going Offline [36] [40] [39]:

- Free hosting services have limitations
- Implement keep-alive system or upgrade to paid hosting
- Check for unhandled exceptions crashing the bot

Logging and Monitoring

Implement comprehensive logging for debugging:

```
import logging

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('bot.log'),
        logging.StreamHandler()
    ]
)

logger = logging.getLogger('discord_bot')

@bot.event
async def on_error(event, *args, **kwargs):
    logger.error(f"Error in {event}", exc_info=True)
```

Regular Maintenance Tasks

Weekly:

- Review moderation logs
- Check bot uptime and performance
- Update profanity filter word list if needed

Monthly:

- Update dependencies: `pip install --upgrade -r requirements.txt`
- Review and rotate bot token if suspicious activity detected
- Audit bot permissions across all servers

Quarterly:

- Full security audit of bot code
- Review Discord.py changelog for breaking changes
- Test disaster recovery procedures

Conclusion

This Discord bot combines the best features of LunaBot's music capabilities and Wick Bot's moderation tools into a single, cohesive application. The architecture supports multiple servers simultaneously, uses modern slash commands and interactive buttons, and implements robust security practices.

Key Takeaways

1. **Modular Design**: Separate music, moderation, and command systems allow independent updates
2. **Per-Server Isolation**: Guild-specific queues and state prevent cross-contamination
3. **Security First**: Token protection, input validation, and rate limiting are essential
4. **User Experience**: Interactive buttons and rich embeds create intuitive interfaces
5. **Scalability**: Asynchronous architecture handles multiple servers efficiently

Next Steps

Immediate:

- Set up Discord application and invite bot to test server
- Install dependencies and configure environment variables
- Test basic functionality with play and moderation commands

Short-term:

- Deploy to cloud hosting platform
- Implement logging and monitoring
- Add custom features specific to your community needs

Long-term:

- Consider paid hosting for production reliability
- Expand moderation features (anti-spam, anti-raid)
- Add analytics and usage statistics
- Implement database for persistent infractions tracking

Additional Resources

- Discord.py Documentation: https://discordpy.readthedocs.io/
- Discord Developer Portal: https://discord.com/developers/applications
- yt-dlp GitHub: https://github.com/yt-dlp/yt-dlp
- Discord Community Guidelines: https://discord.com/guidelines

Project Repository: Complete source code, installation scripts, and example configurations are available in the accompanying files: `discord_bot_complete.py`, `requirements.txt`, `.env.example`, and `SETUP_INSTRUCTIONS.md`.

For questions or issues, refer to the Discord.py community or create an issue in the project repository.

[44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80]

⁂

1. https://blog.stackademic.com/how-to-create-a-music-bot-using-discord-py-and-slash-commands-e3c0a0f92e53

2. https://www.askpython.com/python/examples/make-discord-bot-play-youtube-audio
3. https://www.youtube.com/watch?v=U5CUkxUh2CQ
4. https://www.toolify.ai/ai-news/enhance-your-discord-server-with-luna-bot-the-ultimate-music-experience-822165
5. https://lunabot.vc/about-us/
6. https://github.com/LaurenceRawlings/queue-bot
7. https://www.reddit.com/r/Discord_Bots/comments/p9fig1/different_song_queues_for_different_servers/
8. https://guide.pycord.dev/interactions/ui-components/buttons
9. https://www.youtube.com/watch?v=RCPPqPdlvE8
10. https://guide.pycord.dev/interactions/application-commands/slash-commands
11. https://stackoverflow.com/questions/57550170/how-does-a-discord-bot-handle-events-from-multiple-servers
12. https://community.latenode.com/t/how-does-a-discord-bot-work-across-different-servers-simultaneously/35523
13. https://docs.wickbot.com/intro/what-is-wick/
14. https://docs.wickbot.com/setup/
15. https://help.mee6.xyz/support/solutions/articles/101000385385-how-to-ban-kick-or-mute-with-mee6
16. https://pypi.org/project/profanity-filter/
17. https://github.com/areebbeigh/profanityfilter
18. https://stackoverflow.com/questions/66047514/in-discord-py-how-do-i-make-a-profanity-filter
19. https://moonlightbot.gitbook.io/docs/get-started/moderation-tutorial
20. https://docs.dyno.gg/en/modules/moderation
21. https://stackoverflow.com/questions/75415695/discord-py-slash-command
22. https://www.pythondiscord.com/pages/guides/python-guides/app-commands/
23. https://stackoverflow.com/questions/79653645/how-to-make-a-confirmation-prompt-with-buttons-with-a-discord-bot-in-discord-py
24. https://message.style
25. https://community.latenode.com/t/streaming-youtube-videos-in-a-discord-bot-with-python/31874
26. https://stackoverflow.com/questions/74262376/yt-dlp-how-do-i-extract-the-audio-file-python-discord-py
27. https://www.rapidseedbox.com/blog/yt-dlp-complete-guide
28. https://seasalt.ai/en/blog/65-how-to-download-audio-from-youtube/
29. https://www.youtube.com/watch?v=7xnrAnltcKU
30. https://www.youtube.com/watch?v=QnvvEGfgXHI
31. https://friendify.net/blog/discord-bot-security-best-practices-2025.html
32. https://botpress.com/blog/chatbot-security
33. https://itssc.rpi.edu/hc/en-us/articles/32018134944013-Discord-Best-Practices-Guidelines-on-how-to-keep-Discord-safe-and-secure
34. https://www.reddit.com/r/discordapp/comments/tjwyms/my_mods_cant_kickbantimeout_users_despite_having/
35. https://support.discord.com/hc/en-us/articles/4413305239191-Time-Out-FAQ
36. https://www.youtube.com/watch?v=kBdDmCPcbfs
37. https://infosecwriteups.com/running-discord-bots-24-7-for-free-with-replit-and-uptime-robot-43caebb0cb60
38. https://dev.to/cwkhan/how-to-keep-your-render-or-replit-projects-online-247-with-hostingaifordiscordxyz-4ddi
39. https://community.latenode.com/t/free-24-7-discord-bot-hosting-any-working-solutions-left/27101
40. https://www.reddit.com/r/Discord_Bots/comments/1m04jfy/running_a_discord_bot_247_for_free_seems/
41. https://www.reddit.com/r/Discord_Bots/comments/1hxirj3/i_need_help_with_discordpy_slash_commands_in_cogs/
42. https://community.latenode.com/t/discord-bot-audio-streaming-issues-with-yt-dlp-need-help/12403
43. https://www.reddit.com/r/youtubedl/comments/17legst/ytdlp_discord_bot_integration_stops_playing_music/
44. https://docs.wickbot.com/intro/features/
45. https://www.sololearn.com/en/Discuss/3314207/how-to-make-a-music-bot-for-discord-using-python-in-2025-please-dont-ban-me-im-just-a-student

46. https://lunabot.vc
47. https://wickbot.com
48. https://www.youtube.com/watch?v=YD_N6Ffoojw
49. https://lunabot.vc/features/
50. https://top.gg/bot/536991182035746816
51. https://www.youtube.com/watch?v=CHbN_gB30Tw
52. https://www.topmediai.com/ai-tips/discord-music-bots/
53. https://www.youtube.com/watch?v=26Sj5hJFqUs
54. https://discord.com/developers/docs/components/reference
55. https://discordpy.readthedocs.io/en/stable/ext/commands/commands.html
56. https://gist.github.com/lykn/bac99b06d45ff8eed34c2220d86b6bf4
57. https://github.com/Rapptz/discord.py/discussions/8424
58. https://community.latenode.com/t/how-to-create-discord-bot-embeds-with-interactive-pagination-buttons/30487
59. https://skywork.ai/skypage/en/Lunabot-Deep-Dive-An-AI-Efficiency-Tool-You-Can't-Miss/1976571230034128896
60. https://www.reddit.com/r/AutoModerator/comments/ls9u0q/how_can_i_get_automod_to_filter_swear_words/
61. https://www.youtube.com/watch?v=Refyih_dRt0
62. https://top.gg/bot/679018301543677959
63. https://support.discord.com/hc/en-us/articles/4421269296535-AutoMod-FAQ
64. https://discord.com/discovery/applications/1282926786006614088
65. https://discord.com/safety/auto-moderation-in-discord
66. https://support.discord.com/hc/en-us/articles/24269903094167-How-To-Enable-Discord-Language-Filter
67. https://blog.stackademic.com/how-to-create-a-music-bot-using-discord-py-and-slash-commands-e3c0a0f92e53
68. https://blog.communityone.io/best-discord-bots/
69. https://www.youtube.com/watch?v=4fUXA5bimIA
70. https://www.youtube.com/watch?v=GBwNQImv9Ek
71. https://www.inmotionhosting.com/blog/discord-security-guide/
72. https://dev.to/fizal619/so-you-want-to-make-a-discord-bot-4f0n
73. https://securitybot.gg
74. https://github.com/DevSpen/24-7_hosting_replit
75. https://discord.com/guidelines
76. https://cybrancee.com/blog/how-to-make-a-simple-discord-bot-ultimate-2025-guide/
77. https://tokenminds.co/blog/crypto-marketing/best-discord-bots-for-business
78. https://www.youtube.com/watch?v=QBHr7Odc6Ok
79. https://www.newline.co/@kchan/creating-a-discord-bot-with-replit-agent-and-discordjs--9154e3f4
80. https://www.youtube.com/watch?v=sLWMPiJdLao