Source Code Explanation from GPT

This should hopefully be pretty informative regarding what is happening in the bot.py file if you are struggling with understanding what's going on.

Verbose Explanation of the Discord Bot Code

This bot is designed for **HackRPI**, specifically for managing announcements in a structured, role-restricted manner. It leverages **discord.py** with **modals, buttons, and scheduled tasks** to create and schedule announcements efficiently. Below is a **detailed breakdown** of each key component in the code:

1️⃣ Setup & Configuration

Environment Variables

```
from dotenv import load_dotenv
import os

load_dotenv()
APP_ID = os.getenv("APP_ID")
```

• **Why?**: Instead of hardcoding the bot token, the bot loads it securely from a .env file.

• **If the APP_ID is missing**, the bot **raises an error** to prevent unauthorized API calls.

Role & Channel IDs

```
adminRoll = "Admin HackRPI"
CHANNEL_ID = 1337117398322647070
ANNOUNCEMENTS_CHANNEL_ID = 1342590654592847903
```

• **adminRoll**: Only users with the "Admin HackRPI" role can use certain commands.

• **CHANNEL_ID**: General interaction channel (e.g., bot commands).

• **ANNOUNCEMENTS_CHANNEL_ID**: Dedicated channel where announcements are posted.

Bot Setup & Intents

```
intents = discord.Intents.default()
intents.messages = True
intents.guilds = True
intents.message_content = True

bot = commands.Bot(command_prefix="!", intents=intents)
```

• **Intents**: Grants the bot permission to read messages and interact in servers.

• **Commands Prefix**: ! (e.g., !announce_immediately).

2️⃣ on_ready() Event

```
@bot.event
async def on_ready():
    """Triggered when the bot successfully logs in."""
    await bot.wait_until_ready()  # Ensures bot is fully initialized before
execution
    print(f'Logged in as {bot.user} (ID: {bot.user.id})')
```

• This **prints a log message** when the bot is ready.

• It waits until the bot is **fully operational** before doing anything else.

Welcome Message

```
try:
    channel = await bot.fetch_channel(CHANNEL_ID)
    if isinstance(channel, discord.TextChannel):
        embed = discord.Embed(
            title="Hello, I am the HackRPI Discord Bot!",
            description="Use !help to see all commands.",
            color=discord.Color.blurple()
        )
        embed.set_footer(text="RCOS 2025")

        await channel.send(embed=embed)
except discord.NotFound:
    print(f"Error: Channel ID {CHANNEL_ID} not found.")
except discord.Forbidden:
    print("Error: Bot lacks permission to fetch/send messages in this
channel.")
except Exception as e:
    print(f"Unexpected error: {e}")
```

• **Fetches the channel** and sends a welcome message upon bot startup.

• **Handles errors**:

• **discord.NotFound**: Channel does not exist.

• **discord.Forbidden**: Bot lacks permission to send messages.

3️⃣ schedule_announcement() Command

Command Overview

• **Restricted to "Admin HackRPI" role**.

• Uses **a modal (form-based input)** instead of a plain text command.

• Allows **natural language parsing for date/time**.

Modal (User Input Form)

```
class ScheduleAnnouncement(ui.Modal, title="Schedule Announcement"):
    titleOfMessage = ui.TextInput(label="Title", placeholder="Enter your
title here...", required=True, style=discord.TextStyle.short)
    time = ui.TextInput(label="Date", placeholder="Input natural language:
'Tomorrow at 5PM...'", required=True, style=discord.TextStyle.short)
    message = ui.TextInput(label="Message", placeholder="Enter your message
here...", required=True, style=discord.TextStyle.long)
    links = ui.TextInput(label="Links", placeholder="Enter any links
here...", required=False)
    name = ui.TextInput(label="Name", placeholder="Enter your name
here...", required=True)
```

• Instead of typing everything in a single command (!schedule_announcement "Title" "Message"), users **fill out a form**.

• **dateparser.parse(self.time.value)** allows users to **input time in natural language** ("Tomorrow at 5PM").

Handling Submission & Scheduling

```
async def on_submit(self, interaction: discord.Interaction):
    parsedTime = dateparser.parse(self.time.value, settings={'TIMEZONE':
'US/Eastern'})

    if parsedTime is None:
        await interaction.response.send_message("Invalid date format.
Please use natural language like 'July 30 at 5:30PM'.", ephemeral=True)
        return

    now = datetime.now()
    delay = (parsedTime - now).total_seconds()

    if delay <= 0:
        await interaction.response.send_message("The specified time is in
the past. Please provide a future time.", ephemeral=True)
        return

    await interaction.response.send_message(f'Announcement scheduled for
{parsedTime}', ephemeral=True)

    async def send_announcement_after_delay():
        await asyncio.sleep(delay)
        channel = await
interaction.client.fetch_channel(ANNOUNCEMENTS_CHANNEL_ID)

        embed = discord.Embed(
            title=self.titleOfMessage.value,
            description=self.message.value,
            color=discord.Color.red(),
        )

        if self.links.value:
            embed.add_field(name="Links", value=self.links.value)

        embed.set_footer(text=f'Announced by {self.name.value}')
        await channel.send(embed=embed)

    asyncio.create_task(send_announcement_after_delay())
```

What This Does

1. **Parses user input date** ("Tomorrow at 5PM" → Datetime object).

2. **Validates the date**:

• Rejects **invalid formats**.

• Prevents **past dates** from being scheduled.

3. **Schedules the announcement** (asyncio.create_task() prevents blocking the bot).

4. **Sends confirmation to the user**.

4️⃣ announce_immediately() Command

Command Overview

• **Restricted to "Admin HackRPI" role**.

• Uses **a modal** instead of plain-text input.

• Posts an **embed announcement** immediately.

Modal (User Input Form)

```
class AnnounceImmediately(ui.Modal, title="Announce Now!"):
    titleOfAnnouncement = ui.TextInput(label="Title", placeholder="Enter
your title here...")
    message = ui.TextInput(label="Message", placeholder="Enter your message
here...", style=discord.TextStyle.long)
    links = ui.TextInput(label="Links", placeholder="Paste any links you
want in the message...", required=False)
    name = ui.TextInput(label="Name", placeholder="Enter your name
here...")
```

• **Title & Message**: Required inputs.

• **Links**: Optional field for extra resources.

Handling Submission & Sending Announcement

```
async def on_submit(self, interaction: discord.Interaction):
    channel = await bot.fetch_channel(ANNOUNCEMENTS_CHANNEL_ID)
    embed = discord.Embed(
        title=self.titleOfAnnouncement.value,
        description=self.message.value,
        color=discord.Color.red()
    )

    if self.links.value:
        embed.add_field(name="Links", value=self.links.value)

    embed.set_footer(text=f'Announced by {self.name.value}')
    await channel.send(embed=embed)

    await interaction.response.send_message("Announcement sent
successfully!", ephemeral=True)
```

What This Does

1. **Fetches the announcement channel**.
2. **Creates an embed** with the announcement details.
3. **Adds links if provided**.
4. **Sends the embed to the announcements channel**.
5. **Sends a confirmation to the admin**.

5️⃣ bot.run(APP_ID)

```
bot.run(APP_ID)
```

• Starts the bot with the **securely loaded token**.

🌟 **Summary of Features**

✅ **Role-restricted commands** ("Admin HackRPI")

✅ **Scheduled announcements** with natural date parsing (Tomorrow at 5PM)

✅ **Immediate announcements** with structured input

✅ **Interactive UI elements** (Modals & Buttons)

✅ **Error handling for invalid input**

This bot ensures **clear communication** and **better organization** of announcements in a **professional and structured** manner.