

# Form API Integration

- [Potential Options](#)

# Potential Options

## Option 1: Google Forms + Google Sheets + Discord Webhook + Apps Script

A Google Form would be created for the weekly progress report. It would automatically save responses in a linked Google Sheet. Then, Apps Script would be used to send a message using a Discord Webhook.

```
function postToDiscordWeekly() {  
  const webhookUrl = 'https://discord.com/api/webhooks/your-webhook-id';  
  const formLink = 'https://docs.google.com/forms/d/your-form-id/viewform';  
  
  const payload = {  
    content: `📅**Weekly Progress Report Time!**\nPlease fill out the form: ${formLink}`  
  };  
  
  const options = {  
    method: 'post',  
    contentType: 'application/json',  
    payload: JSON.stringify(payload)  
  };  
  
  UrlFetchApp.fetch(webhookUrl, options);  
}
```

- Embeds could be used to make the Discord message prettier
- This could also be done through the ANNI Bot
  - Could just be a simple modification of the already existing weekly form reminder message

### Pros:

- Free and easy
- No hosting needed
- Integrated workflow with Sheets
- Webhooks are native to discord

### Cons:

- Google Apps Script has execution limits
  - 90 minutes/day for free users
  - 100 URL fetch calls/day
  - Shouldn't be a big deal, CCS is not a huge company

## Option 2 - Discord Bot with Interactive Forms

Utilizes a relatively new Discord feature - Modals are pop-up forms that allow users to provide formatted inputs through submissions. The Discord bot would handle everything - including sending out the form weekly and collecting a user's information through the Modal.

```
import discord
from discord.ext import commands, tasks
from apscheduler.schedulers.asyncio import AsyncIOScheduler

intents = discord.Intents.default()
bot = commands.Bot(command_prefix="!", intents=intents)

CHANNEL_ID = 123456789012345678 # replace with your target channel ID

# --- Modal Definition ---
class WeeklyReportModal(discord.ui.Modal, title="Weekly Progress Report"):
    accomplishments = discord.ui.TextInput(label="What did you accomplish?",
style=discord.TextStyle.paragraph, required=True)
    blockers = discord.ui.TextInput(label="Any blockers or challenges?",
style=discord.TextStyle.paragraph, required=False)

    async def on_submit(self, interaction: discord.Interaction):
        # Print or store the results here
        print(f"From {interaction.user.display_name}:")
        print(f"Accomplishments: {self.accomplishments.value}")
        print(f"Blockers: {self.blockers.value}")
        await interaction.response.send_message("👍 Thanks for submitting your report!",
ephemeral=True)

# --- Button View ---
class ReportView(discord.ui.View):
    @discord.ui.button(label="Fill Out Weekly Report", style=discord.ButtonStyle.primary)
    async def report_button(self, interaction: discord.Interaction, button:
discord.ui.Button):
```

```

        await interaction.response.send_modal(WeeklyReportModal())

# --- Scheduled Task ---
@tasks.loop(hours=168) # once every 7 days
async def send_weekly_prompt():
    channel = bot.get_channel(CHANNEL_ID)
    if channel:
        await channel.send("📅 It's time for the weekly progress report!", view=ReportView())

# --- On Ready ---
@bot.event
async def on_ready():
    print(f"Logged in as {bot.user}")
    send_weekly_prompt.start()

# --- Start Bot ---
bot.run("YOUR_BOT_TOKEN")

```

## Pros:

- Fully native to Discord
- Better UX
- Secure
- Highly customizable

## Cons:

- Code heavy
- Needs hosting
  - Shouldn't be an issue, as the Discord bot is already being hosted
- Needs a manual response storage
  - Could be a file, Google Sheet, or a Database