The Plan: Python Script for Auto-Updating Rumble News Template

Yes, this is a solid plan! It's scalable, dynamic, and perfect for your setup. The Python script will scrape rumble.com/category/news for fresh live/24/7 streams (titles, channels, embed URLs), generate an updated HTML template, and commit/push to GitHub. You host the repo on GitHub Pages (free static hosting), and embed the latest page into your website via an `<iframe src="https://yourusername.github.io/repo/index.html">` —it auto-refreshes on load or with a timer. No server needed, and it runs on your PC (cron job or manual).

I'll give you:

1. **Python script** (`update_rumble.py`) – Scrapes, generates HTML, pushes to GitHub.
2. **Base HTML template** (`template.html`) – The page structure with placeholders.
3. **Setup instructions** – How to run on PC, GitHub, and embed.

1. Python Script (`update_rumble.py`)

Save this as `update_rumble.py` . It uses `requests` and `beautifulsoup4` (pip install them). It scrapes 7 live news streams, fills the template, and pushes to GitHub.

Python

Copy

```
import requests
from bs4 import BeautifulSoup
import os
import subprocess
from datetime import datetime

# GitHub repo details (replace with yours)
REPO_DIR = '/path/to/your/local/repo'  # e.g., '/Users/yourname/rumble-news'
REPO_NAME = 'rumble-news'  # Your GitHub repo name
GITHUB_TOKEN = 'your_github_pat_token'  # Personal Access Token for git push

# Base template file
TEMPLATE_FILE = 'template.html'
OUTPUT_FILE = 'index.html'

def scrape_rumble_live_news(num_streams=7):
    """Scrape rumble.com/category/news for live streams."""
```

```
    url = 'https://rumble.com/category/news'
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) Apple
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    streams = []
    # Find live video cards (adjust selectors based on Rumble's HTML structur
    videos = soup.find_all('div', class_='video-card-container')[:num_streams

    for video in videos:
        # Extract title
        title_elem = video.find('h3', class_='video-title')
        title = title_elem.text.strip() if title_elem else 'Unknown Title'

        # Extract channel
        chan_elem = video.find('a', class_='channel-name')
        chan = chan_elem.text.strip() if chan_elem else 'Unknown Channel'

        # Extract embed ID from link (e.g., /vID from href)
        link_elem = video.find('a', href=True)
        href = link_elem['href'] if link_elem else ''
        video_id = href.split('/v')[-1].split('/')[0] if '/v' in href else No

        # Filter for live (look for 'LIVE' in title or class)
        if 'live' in title.lower() or '24/7' in title.lower() or video_id:
            embed_url = f"https://rumble.com/embed/v{video_id}/?pub=4" if vid
            if embed_url:
                # Get thumbnail (Rumble pattern)
                thumb_url = f"https://sp.rmbl.ws/s8/2/v/{video_id[0]}/{video_
                streams.append({
                    'title': title,
                    'chan': chan,
                    'src': embed_url,
                    'thumb': thumb_url
                })
                if len(streams) == num_streams:
                    break

    return streams

def generate_html(streams):
    """Fill template with scraped data."""
```

```
    with open(TEMPLATE_FILE, 'r') as f:
        template = f.read()

    # Placeholder for grid content
    grid_html = ''
    for stream in streams:
        grid_html += f'''
    <div class="card" data-src="{stream['src']}">
      <div class="thumb">
        <img src="{stream['thumb']}" alt="">
        <div class="play-icon"><svg viewBox="0 0 24 24" fill="white"><path d=
      </div>
      <div class="info"><div class="live">● LIVE</div><div class="title">{str
    </div>
    '''

    html = template.replace('<!-- STREAMS_PLACEHOLDER -->', grid_html)
    # Update initial player to first stream
    if streams:
        html = html.replace('<!-- INITIAL_SRC -->', streams[0]['src'])

    with open(OUTPUT_FILE, 'w') as f:
        f.write(html)

    print(f"Generated {OUTPUT_FILE} with {len(streams)} streams at {datetime.

def git_push():
    """Commit and push to GitHub."""
    os.chdir(REPO_DIR)
    subprocess.run(['git', 'add', OUTPUT_FILE])
    subprocess.run(['git', 'commit', '-m', f'Update news streams {datetime.no
    subprocess.run(['git', 'push', f'https://{GITHUB_TOKEN}@github.com/yourus

if __name__ == '__main__':
    streams = scrape_rumble_live_news(7)
    if streams:
        generate_html(streams)
        git_push()
        print("Updated and pushed to GitHub!")
    else:
        print("No streams found – check scraper.")
```

2. Base HTML Template (`template.html`)

Save this as `template.html` in your repo. The script replaces `<!-- STREAMS_PLACEHOLDER -->` with scraped content and `<!-- INITIAL_SRC -->` with the first stream's URL.

HTML

Copy

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Rumble Live News – Auto-Updated Streams</title>
<style>
  *,*::before,*::after{box-sizing:border-box}
  html,body{margin:0;padding:0;width:100%;height:100%;background:#000;overflo
  #player{position:fixed;top:0;left:0;width:100%;height:100%;border:none;z-in
  #fallback{position:absolute;top:50%;left:50%;transform:translate(-50%,-50%)
  #bar{position:fixed;bottom:0;left:0;width:100%;height:90px;background:linea
  #bar:hover{height:20vh;background:rgba(10,10,30,0.98);backdrop-filter:blur(
  #grid{position:absolute;bottom:10px;left:0;width:100%;padding:0 20px;displa
  #bar:hover #grid{opacity:1}
  .card{min-width:300px;background:rgba(255,255,255,0.08);border-radius:18px;
  .card:hover{transform:translateY(-10px);box-shadow:0 15px 30px rgba(255,45,
  .thumb{width:100%;height:130px;object-fit:cover;background:#222;position:re
  .thumb::after{content:'';position:absolute;inset:0;background:linear-gradie
  .card:hover .thumb::after{opacity:1}
  .play-icon{position:absolute;top:50%;left:50%;transform:translate(-50%,-50%
  .play-icon svg{width:24px;height:24px;margin-left:3px}
  .card:hover .play-icon{opacity:1;transform:translate(-50%,-50%) scale(1.1)}
  .info{padding:14px}
  .live{background:#ff2d55;padding:4px 10px;border-radius:20px;font-size:11px
  .title{font-size:16px;font-weight:700;line-height:1.2}
  .chan{font-size:13px;opacity:0.85;margin-top:4px}
  #grid::-webkit-scrollbar{height:6px}
  #grid::-webkit-scrollbar-thumb{background:rgba(255,255,255,0.3);border-radi
</style>
</head>
<body>

<!-- Main Player – initial src will be replaced by script -->
<iframe id="player" class="rumble" width="640" height="360" src="<!-- INITIAL
```

```
<div id="fallback">
  <h2>Stream temporarily unavailable</h2>
  <p>Please select another channel</p>
</div>

<div id="bar">
  <div id="grid">
    <!-- STREAMS_PLACEHOLDER -->
  </div>
</div>

<script>
// Click handler - recreates iframe
document.querySelectorAll('.card').forEach(card => {
  card.addEventListener('click', () => {
    const src = card.dataset.src;
    const container = document.body;
    const oldPlayer = document.getElementById('player');

    document.getElementById('fallback').style.display = 'none';

    const newPlayer = document.createElement('iframe');
    newPlayer.id = 'player';
    newPlayer.className = 'rumble';
    newPlayer.width = '640';
    newPlayer.height = '360';
    newPlayer.frameBorder = '0';
    newPlayer.allowFullscreen = true;
    newPlayer.src = src + '&autoplay=1&mute=1&rel=0&controls=1&t=' + Date.now

    container.replaceChild(newPlayer, oldPlayer);

    document.querySelectorAll('.card').forEach(c => c.style.boxShadow = '');
    card.style.boxShadow = '0 0 40px rgba(255,45,85,1)';
  });
});

// Error handling
document.addEventListener('DOMContentLoaded', () => {
  const checkPlayer = () => {
    const player = document.getElementById('player');
    if (player) {
```

```
      player.onerror = () => {
        document.getElementById('fallback').style.display = 'block';
        player.style.display = 'none';
      };
      player.onload = () => {
        document.getElementById('fallback').style.display = 'none';
      };
    }
  };
  checkPlayer();
  new MutationObserver(checkPlayer).observe(document.body, { childList: true,
});
</script>

</body>
</html>
```

3. Setup Instructions

1. **PC Setup**:
    - Install Python 3.12+.
    - `pip install requests beautifulsoup4`.
    - Clone your GitHub repo: `git clone https://github.com/yourusername/rumble-news.git`.
    - Put `update_rumble.py` and `template.html` in the repo folder.
    - Edit `REPO_DIR` and `GITHUB_TOKEN` in the script (get token at github.com/settings/tokens – select "repo" scope).
    - Run: `python update_rumble.py` – it scrapes, generates `index.html`, commits/pushes.
2. **GitHub Hosting**:
    - Enable GitHub Pages in repo settings > Pages > Source: "Deploy from a branch" > main > / (root).
    - Your page: https://yourusername.github.io/rumble-news/ – auto-updates on push.
3. **Embed in Your Website**:
    - Add to your site: `<iframe src="https://yourusername.github.io/rumble-news/" width="100%" height="100%" frameborder="0" allowfullscreen></iframe>`.
    - For auto-refresh: Add JS timer in your site: `setInterval(() => location.reload(), 3600000);` (every hour).
4. **Auto-Run on PC**:
    - Windows: Task Scheduler – run `python update_rumble.py` daily.
    - Mac/Linux: Cron job – `crontab -e`, add `0 8 * * * /usr/bin/python3 /path/to/update_rumble.py` (daily at 8AM).

This keeps your embeds fresh—no more 410s. Run the script manually first to test. What do you think—ready to roll?

↳ Add error handling to scraper

↳ Alternative: Use YouTube live streams

↳ Make code more robust