**A RADIO STREAMING SITE MONITOR AND NOTIFICATION PROGRAM USING PYWHATKIT**

JOSEPH ODUOR

KENNEDY MWANZA

KENYA BROADCASTING CORPORATION

22 September 2023

# OVERVIEW

This Python program monitors streaming websites for downtime and sends WhatsApp text notifications when a site is inaccessible. Additionally, it sends periodic alerts at 8 AM, noon, and 4 PM to confirm that the monitoring program is still running.

This documentation provides a detailed explanation of the program's functionality and usage.

# PREREQUISITES

Before running the program, make sure you have the following prerequisites installed:

* Python (3.x recommended)
* Required Python packages: **`requests`, `pywhatkit`, `selenium`**

You can install the required packages using pip:

python3 -m pip install requests pywhatkit selenium

# USAGE

## Setup Configuration:

Open the **`whatsapp.py`** file and configure the following parameters:

* **`radios`**: A list of streaming website names and URLs to monitor in the format **“radio\_name”: “https://streaminglink”**
* **`phone\_number`**: The WhatsApp phone number to which notifications may be sent.
* **`group\_id`**: The WhatsApp group ID to which notifications may be sent.

## Running the Program

Execute the **`whatsapp.py`** script to start monitoring streaming websites and sending notifications.

**python3 /Users/joseph/downloads/whatsapp.py**

WhatsApp web will prompt the user to scan a QR Code with the number to be used to send notifications.

## Scheduled Alerts

The program is set to send alerts at 8 AM, noon, and 4 PM to confirm it's running. You can adjust the schedule in the **`whatsapp.py`** script by modifying the **morning, noon,** and **evening** variables.

# PROGRAM FLOW

The program follows these main steps:

## Initialization

* Import required libraries.
* Initialize WhatsApp web on the host computer browser by having the number to be used to send notifications scan a QR code. This is only done on the first notification or when resetting WhatsApp web.

## Main Monitoring Loop:

* Continuously loop through the list of website URLs to monitor.
* Use the **`requests`** library to check if each website is accessible.
* If a website is down (returns a non-200 HTTP status code), send a WhatsApp text notification using WhatsApp web to the specified recipient, then wait 10 minutes before resuming to check the sites.

## Scheduled Alerts:

* The program schedules alerts to be sent at specific times (e.g., 8 AM, noon, and 4 PM) and the first time the program is run to confirm that it is running and operational.

## Error Handling:

* Handle exceptions gracefully, such as network errors or configuration issues.
* Log errors and exceptions for debugging.
* Automatically restart the program after a network issue has been resolved.

# CUSTOMIZATION

You can customize this program by:

* Modifying the list of streaming websites to monitor **(`radios`)** in the **`whatsapp.py`** file.
* Adjusting the alert schedule by modifying the **morning, noon,** and **evening** variables.
* Adjusting **`waiting\_time\_to\_send`** and **`waiting\_time\_to\_close`** both in seconds, to give enough time for the program to run to cater for slow internet speeds.
* Adjusting the time to wait before resuming to check the sites if one was previously found to be down, or the time to wait before restarting the program if the network connection is lost, by modifying the **`time.sleep()`** function.
* Extending the error handling and logging to meet your specific needs.
* Changing the **mode** to **“contact”** or **“group”** if you wish to send the notifications to an individual contact or group respectively.
* Changing the phone number and group ID link to which notifications are sent. The group should preferably be one where the contact used to set up WhatsApp web is an Admin.

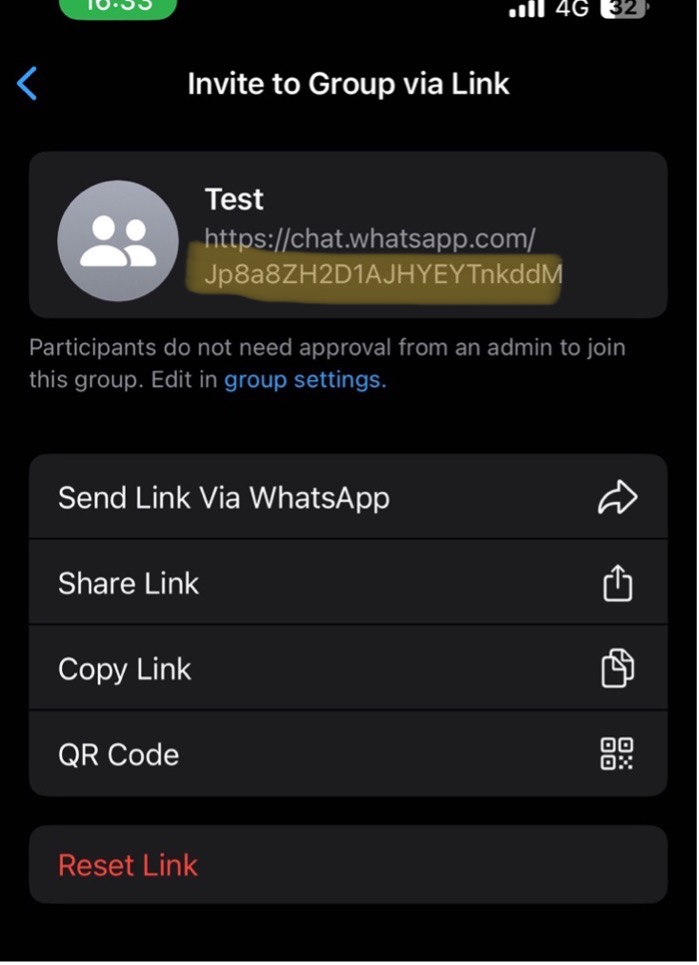


Figure : Highlighted characters represent the group ID to be used in the program.

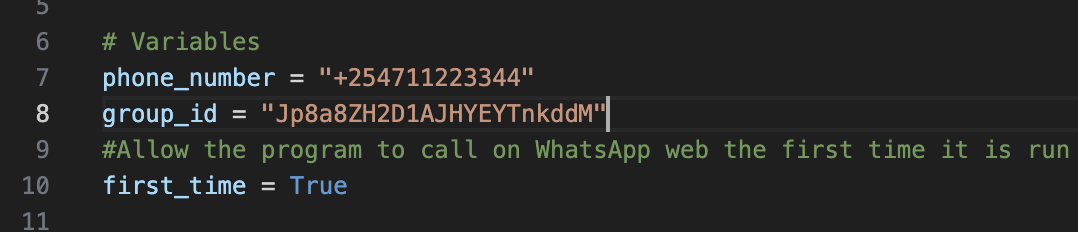


Figure : Group ID pasted in the correct placement on the script.

# CHALLENGES AND FIXES

* The initial plan was to use **crontab** to perpetually run the program in an infinite loop. **Crontab** however lacks terminal access for automated keyboard inputs hence an infinite loop within the source code was chosen instead.
* Designing the program to send a message immediately after being done checking the streamers proved unstable due to unreliable internet speeds that resulted in timestamp errors. It was therefore modified to send a message around two minutes after being done checking the streaming sites.
* Sending a message after each check where there was nothing wrong with the streamers proved unnecessary as it would spam inboxes. The program was therefore designed to only alert when at least one streaming site was down, with alerts at specific times of the day to confirm it is still up and running.
* If a site was discovered to be down, the program would spam inboxes with a message every minute. It was therefore adjusted to wait 10 minutes after a site was discovered to be down to give ample time for a fix and not spam inboxes with hundreds of messages in case no one was on site.
* A loss in internet connectivity caused the first prototype to terminate immediately. The script was therefore modified to log network connectivity errors and automatically restart once network connectivity is restored.
* Hiccups in network connectivity were discovered to cause the streamers to briefly lose connection before quickly coming back online. The monitor was modified to double check if the streamers were offline for a period of 10 seconds before sending a notification.

# CONCLUSION

This Python program monitors radio streaming websites for downtime and sends WhatsApp text notifications when issues are detected.

By following the provided instructions and customizing the program as needed, you can ensure the reliable operation of your streaming site monitoring system.