This API provides functionality to manage and update the availability status on a website, post Instagram stories based on status changes, and retrieve the current status and last update time. The API is designed to handle status updates for "Nightline" platforms, allowing the team to easily update information on the website and on Instagram.

Features

- Change the status by making a simple GET request
  - Easy to embed into, e.g. moodle
- Automatically post an Instagram story based on the status
- Automatically remove the last Instagram story, if the status changes
  - Only works for stories uploaded through the API
- Dynamically display a message on your website, informing users about the status
- Reset to default at 00:00.

Setup

Make sure to read the "security notes"

Prerequisites

1. Enter the base folder `NightLight`
2. Rename `.env.example` to `.env`
3. Configure the variables in `.env`
   - Use a strong API-Key! E.g. 2048-bit with Mixed letters & Numbers
     - https://generate-random.org/api-key-generator
4. In the `NightLight/website` folder you find HTML which you can copy and paste to your website. It will fetch the status and display a little notification if the status equals to "canceled" or "english". Feel free to adjust the design or displayed statuses.
   - IMPORTANT: To make the code functioning, replace STATUS_API_URL(line 8) with the URL under which the API is callable. If the status can be requested on status.example.com, replace `STATUS_URL` with `https://status.example.com`.
5. Place the images you would like to post in your Instagram story, in the `NightLight/assets` folder. If you don't want to use the Instagram feature and have configured the required variables in .env accordingly, you do not have to place images in the assets folder.
   - For status update to "canceled", name the image "canceled.png"
   - For status update to "english", name the image "english.png"

Optional

Change the reset time

If you would like to reset the status before or after 00:00, update the cronjob. E.g. to run the reset at 01:00, do the following, based on the Method to run the API.

Docker Compose / Docker: Enter the file: `NightLight/Dockerfile` and change the time in the cron job command. Change: `RUN echo "0 1 * * * /bin/bash /app/reset_status.sh >> /var/log/cron.log 2>&1" >> /etc/cron.d/reset-status-cron`

to

`RUN echo "0 2 * * * /bin/bash /app/reset_status.sh >> /var/log/cron.log 2>&1" >> /etc/cron.d/reset-status-cron`

Manual: Insted of running `echo "0 0 * * * /bin/bash /app/reset_status.sh >> /var/log/cron.log 2>&1" > /etc/cron.d/reset-status-cron`

run:

`echo "0 1 * * * /bin/bash /app/reset_status.sh >> /var/log/cron.log 2>&1" > /etc/cron.d/reset-status-cron`

Methods to run the API

For every method, you should also configure a reverse proxy.

Docker Compose

This method requires docker compose to be installed.

1. Enter the base folder `NightLight`
2. In a terminal run `docker compose up`

Docker

This method requires docker to be installed

1. Enter the base folder `NightLight`
2. In a terminal run the following commands
   - `docker build -t status-api . --no-cache`
   - Make sure to insert the port you also entered in your .env file
     - `docker run --env-file .env -p THE_PORT_YOU_ENTERED_IN_DOT_ENV:5000 status-api`

Manual (test/dev deployment)

This method requires python and pip to be installed.

> [!CAUTION] Do not run the API like this on your sever! For manual server deployment see the section below

1. Enter the base folder `NightLight`
2. Install the required python modules by running: `pip install -r requirements.txt`
   - You can also use a virtual environment to install the requirements in there.
3. In a terminal run `python server.py`

Manual (production deployment)

Because the API is based on a flask application, we want to use WSGI to run the app. The Web server served by flask is perfect for development, but not as robust and hardened than others which are made for production.

Make sure to insert the host you also entered in your .env file.

1. Enter the base folder `NightLight`
2. Install the required python modules by running: `pip install -r requirements.txt`
   - You can also use a virtual environment to install the requirements in there.

3.
   ```
   gunicorn --log-level info --timeout 120 -w 3 -b
   THE_HOST_YOU_ENTERED_IN_DOT_ENV:5000 server:app
   ```
   - To keep Gunicorn running in the background and let you close the terminal, run:

     ```
     nohup gunicorn --workers 3 --bind THE_HOST_YOU_ENTERED_IN_DOT_ENV:8000
     server:app > gunicorn.log 2>&1 &
     ```
4. Now you can configure the reset cron job to reset the status to "default" every night. If you want to change the time, the reset is triggered, take a look at the "optional" section above. Replace PATH_TO_NIGHTLIGHT with the actual path to the NightLight folder
   1. Make the reset script executable: `chmod +x PATH_TO_NIGHTLIGHT/NightLight/reset_status.sh`
   2. Configure the cron job: `echo "0 1 * * * /bin/bash PATH_TO_NIGHTLIGHT/NightLight/reset_status.sh >> /var/log/cron.log 2>&1" > /etc/cron.d/reset-status-cron`
   3. Set correct permissions for the cron job configuration: `chmod 0644 /etc/cron.d/reset-status-cron`
   4. Add the cronjob config to crontab: `crontab /etc/cron.d/reset-status-cron`

Reverse Proxy

nginx

Sample nginx reverse proxy configuration (untested)

Make sure to replace the `THE_PORT_YOU_ENTERED_IN_DOT_ENV` variable in the last block, as well as `your_domain.com` with the actual domain, the status API is reachable under. E.g. status.example.com

```
# Redirect all HTTP traffic to HTTPS
server {
    listen 80;
    server_name your_domain.com;

    # Redirect HTTP to HTTPS
    return 301 https://$host$request_uri;
```

```
}

# Main HTTPS server block
server {
    listen 443 ssl;
    server_name your_domain.com;

    # SSL certificate and key paths (use the path to your SSL certificate)
    ssl_certificate /etc/letsencrypt/live/your_domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your_domain.com/privkey.pem;

    # Optional: Specify SSL settings (make sure to adapt or verify for your use)
    ssl_protocols TLSv1.2 TLSv1.3;  # Disable older protocols (SSLv3, TLSv1.0,
TLSv1.1)
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:...';
# Set secure ciphers
    ssl_prefer_server_ciphers on;

    # Optional: Enable HTTP/2 for better performance
    http2;

    # Optional: Enable SSL session caching (improves performance)
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 1d;
    ssl_session_tickets off;

    # Additional configurations (location block for reverse proxy to Gunicorn)
    location / {
        proxy_pass http://127.0.0.1:THE_PORT_YOU_ENTERED_IN_DOT_ENV;  # Point this
to your Flask app running with Gunicorn
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

apache

Sample apache reverse proxy configuration (tested)

Make sure to replace the `THE_PORT_YOU_ENTERED_IN_DOT_ENV` variable in the last block, as well as `your_domain.com` with the actual domain, the status API is reachable under. E.g. status.example.com

```
<IfModule mod_ssl.c>
    <VirtualHost *:80>
        ServerName your_domain.com

        Redirect permanent / https://your_domain.com/
    </VirtualHost>

    <VirtualHost *:443>
        ServerName your_domain.com

        <IfModule headers_module>
            RequestHeader set X-Forwarded-Proto "https"
           # RequestHeader set X-Real-IP %{REMOTE_ADDR}
           # RequestHeader append X-Forwarded-For %{REMOTE_ADDR}
        </IfModule>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        ProxyRequests     Off
        ProxyPreserveHost On
        ProxyVia On
        ProxyPass / http://127.0.0.1:THE_PORT_YOU_ENTERED_IN_DOT_ENV/
        ProxyPassReverse / http://127.0.0.1:THE_PORT_YOU_ENTERED_IN_DOT_ENV/

        Include /etc/letsencrypt/options-ssl-apache.conf
        SSLCertificateFile /etc/letsencrypt/live/your_domain.com/fullchain.>
        SSLCertificateKeyFile /etc/letsencrypt/live/your_domain.com/privkey>
    </VirtualHost>
</IfModule>
```

Security notes

This section will talk about security aspects and things to keep in mind when running the API. Read it carefully and make sure you understand everything before running the API.

API-KEY

Running a GET-API is probably uncommon, as other request methods provide a little more security regarding exposure of API-Keys. The NightLight API uses an API-Key as a URL-Parameter. This results in the Key being stored in the browser history of users, as well as logs etc. However, to keep things simple, especially for users who want to change the status, this might be the easiest solution. When embedding the Links e.g. on moodle, there is no option for e.g. POST requests. Users should have easy access to the API.

Risks:

- Calling the API over HTTP will expose the API key to everybody on the network route or in the same network
    - E.g. if you call the API over HTTP in a coffee shop, everybody would be able to sniff on you and see the API Key
- Strangers knowing the API-Key can do the following things that might cause harm
    - Change the status
    - DOS Attack by fast changing the status which overwhelms the server, causing it to crash
    - Get banned from Instagram.
        - When having Instagram story posts configured, the server will post a story on status changes. Because Instagram is suspicious, it detects behavior which might come from

automations. When too many actions are done in a short period of time, Instagram might lock you out or even ban your account.
