**README**

**1. Project's Title**

Automating Content Takedown Across Social Media Platforms.

**2. Project Description**

Our project aims to develop an advanced system for automating the process of content takedown across various social media platforms such as Facebook, Instagram, TikTok and many others. With the exponential growth of user-generated content, social media platforms face significant challenges in moderating and removing inappropriate or harmful content efficiently. Manual content moderation processes are often slow, resource-intensive, and prone to human error.

The application utilizes Python and Selenium to automate the process of content takedown across multiple social media platforms. Through Selenium, the application interacts with the web interfaces of the platforms, enabling it to monitor new content uploads in real-time. When content violating community guidelines or legal regulations is identified, the application initiates the takedown process automatically. Administrators can configure moderation policies and access detailed reports and analytics on content moderation activities. By leveraging Python and Selenium, the application streamlines content moderation, reduces manual effort, and ensures swift removal of harmful content, thereby enhancing user safety and platform compliance.

Python is renowned for its simplicity and readability, making it accessible to developers of all skill levels. Its straightforward syntax allows for rapid development and easy maintenance of automation scripts.

Selenium WebDriver, a collection of language-specific bindings, is the preferred choice for developing robust, browser-based regression automation suites and tests. It operates by driving browsers natively, mimicking user interactions, and can be executed either locally or on remote machines via the Selenium server, representing a significant advancement in browser automation technology.

The term "Selenium WebDriver" encompasses both the language bindings and the implementations of the browser-controlling code, often referred to simply as "WebDriver." Notably, Selenium WebDriver has achieved the status of a W3C Recommendation, indicating its widespread acceptance and standardization within the web development community.

As an API, Selenium WebDriver is intentionally crafted to offer a simpler and more concise programming interface compared to its predecessors. It adopts a compact, object-oriented design, facilitating ease of use and streamlined implementation of browser automation tasks.

One of the key strengths of Selenium WebDriver lies in its effectiveness in driving browsers, allowing for seamless navigation and interaction with web elements. This capability ensures the reliable execution of automated tests across diverse browser environments, contributing to enhanced testing efficiency and accuracy.

Challenges Encountered:

1. Captcha: Captcha mechanisms present a significant challenge for automated systems, as they are designed to distinguish between human users and bots. Overcoming Captcha requires sophisticated algorithms capable of recognizing and interpreting visual or textual puzzles accurately.

2. OTP (One-Time Password): OTP verification is commonly used as an additional security measure during authentication processes. Automating OTP entry necessitates methods for intercepting and processing OTPs received via various channels, such as SMS or email, in real-time.

3. File Upload: Automated handling of file uploads on web platforms poses challenges, particularly in identifying, selecting, and uploading files from local directories. This task requires robust automation techniques capable of mimicking human file selection and upload actions accurately.

4. Speech to Text: Converting speech input into text format, known as speech-to-text conversion, presents challenges due to variations in speech patterns, accents, and background noise. Accurate speech recognition algorithms are essential for effectively processing spoken content.

Features to be implemented:

1. AI for Information Filling: Integration of artificial intelligence (AI) enables the system to intelligently fill out forms and input fields by analyzing contextual information and predicting user inputs. AI algorithms can streamline the data entry process, enhancing efficiency and accuracy.

2. Captcha Solving AI: Advanced AI algorithms are employed to tackle Captcha challenges by analyzing Captcha images or text and generating accurate solutions. Machine learning models trained on diverse Captcha datasets can effectively recognize and decipher Captcha puzzles, enabling seamless automation of Captcha verification processes.

**3. Table of Contents (Optional)**

If your README is very long, you might want to add a table of contents to make it easy for users to navigate to different sections easily. It will make it easier for readers to move around the project with ease.

**4. How to Install and Run the Project**

Here's a step-by-step guide to installing Python on your local machine and setting up the specified dependencies:

Step 1: Install Python:

1. Navigate to the official Python website: (https://www.python.org/downloads/)

2. Download the latest version of Python for your operating system (Windows, macOS, or Linux).

3. Run the downloaded installer and follow the installation instructions.

4. Ensure that you select the option to add Python to your system's PATH during the installation process.

Step 2: Verify Python Installation:

1. Open a command prompt (Windows) or terminal (macOS/Linux).

2. Type the following command and press Enter:

python --version

3. If Python is installed correctly, you should see the version number displayed in the output.

Step 3: Install pipenv:

1. Open a command prompt or terminal.

2. Run the following command to install pipenv:

pip install pipenv

Step 4: Create a New Project Directory:

1. Create a new directory for your project and navigate into it using the command prompt or terminal.

Step 5: Set Up Virtual Environment:

1. Inside your project directory, run the following command to create a new virtual environment using pipenv:

pipenv shell

Step 6: Install Dependencies:

1. With the virtual environment activated, run the following commands to install the specified dependencies:

pipenv install selenium

pip install numpy

pip install opencv-python

pip install undetected\_chromedriver

pip install pandas

pip install bs4

pip install SpeechRecognition

pip install requests

pip install openpyxl

pip install pyautogui

pip install imaplib

pip install concurrent.futures

Step 7: Verify Installation:

1. To verify that all dependencies are installed correctly, you can create a Python script and import each module/package.

2. Open a text editor, create a new Python script (e.g., `verify\_dependencies.py`), and import the required modules/packages.

3. Run the script using Python to ensure that there are no import errors.

That's it! You've successfully installed Python on your local machine and set up the specified dependencies using pipenv for your project. You're now ready to start developing with these tools and libraries.

**5. How to Use the Project**





Setup:

Ensure you have Python installed along with the required libraries (selenium, pyautogui, openpyxl, and tiktok\_verification).

Make sure you have the GeckoDriver executable (for Firefox) in your system path or specify its location.

Input Data:

Replace "samira\_raad2000@outlook.com" with your email address.

Running the Script:

Execute the script, and it will open Firefox and navigate to the TikTok copyright report page.

The script will fill in the necessary information for the report, such as the problem selection, affected content, your details, and the description.

It will upload a screenshot named "ss.jpg" (adjust the name if needed) as evidence.

Finally, it will submit the report and close the browser.

Repeat:

The script will repeat this process five times (as per the loop for \_ in range(5):). Adjust the loop according to your needs.

**6. Include Credits**

Eddy Gergi imeddygergi@gmail.com

Emilio Dandan emilio.dandan@gmail.com

Mary Karam maryjanekaram10@gmail.com

Repository link: <https://github.com/theddygergi/fyp>

**7. Add a License**

For most README files, this is usually considered the last part. It lets other developers know what they can and cannot do with your project.

We have different types of licenses depending on the kind of project you are working on. Depending on the one you will choose it will determine the contributions your project gets.

The most common one is the GPL License which allows other to make modification to your code and use it for commercial purposes. If you need help choosing a license, use check out this link: <https://choosealicense.com/>

Up to this point what we have covered are the minimum requirements for a good README. But you might also want to consider adding the following sections to make it more eye catching and interactive.

**Additional README Sections**

**8. Badges**

Badges aren't necessary, but using them is a simple way of letting other developers know that you know what you're doing.

Having this section can also be helpful to help link to important tools and also show some simple stats about your project like the number of forks, contributors, open issues etc...

Below is a screenshot from one of a project that shows how you can make use of badges:



The good thing about this section is that it automatically updates it self.

Don't know where to get them from? Check out the badges hosted by [shields.io](https://shields.io/). They have a ton of badges to help you get started. You may not understand what they all represent now, but you will in time.

**9. How to Contribute to the Project**

This mostly will be useful if you are developing an open-source project that you will need other developers to contribute to. You will want to add guidelines to let them know how they can contribute to your project.

Also it is important to make sure that the licence you choose for an open-source projects is correct to avoid future conflicts. And adding contribution guidelines will play a big role.

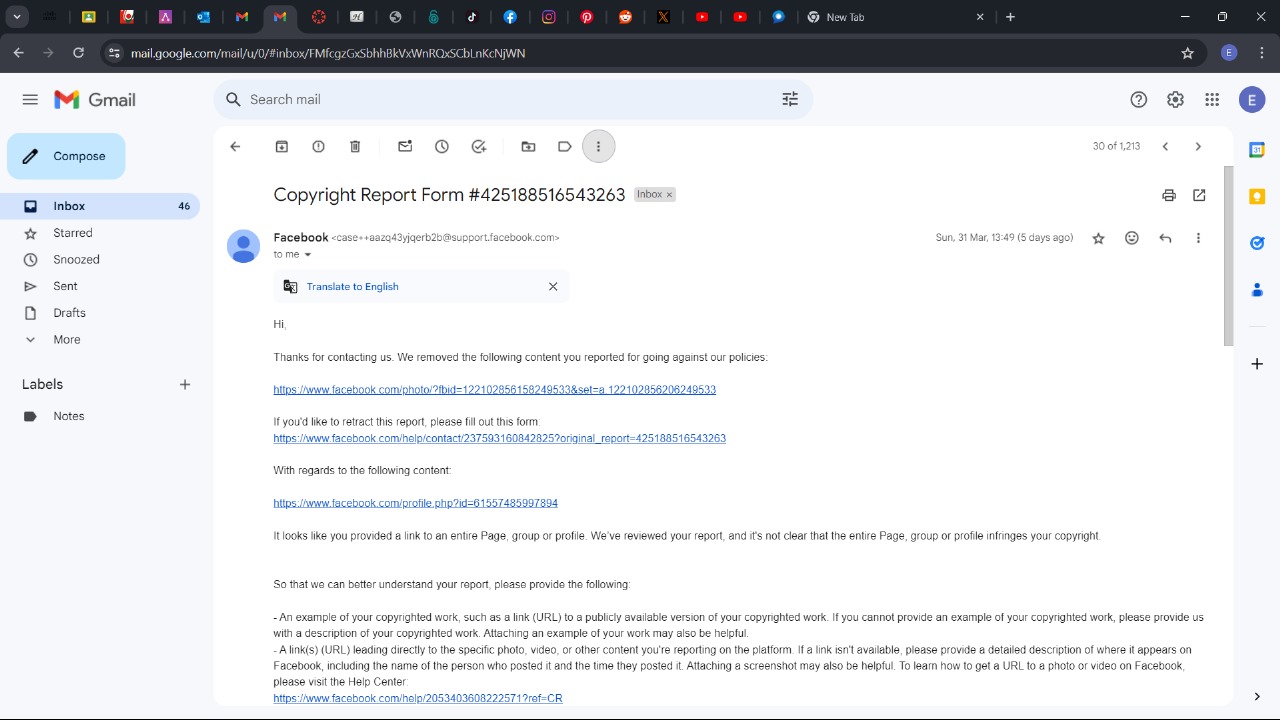
Some of the most common guidelines include the [Contributor Covenant](https://www.contributor-covenant.org/) and the [Contributing guide](https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors). Thes docs will give you the help you need when setting rules for your project.

**10. Include Tests**

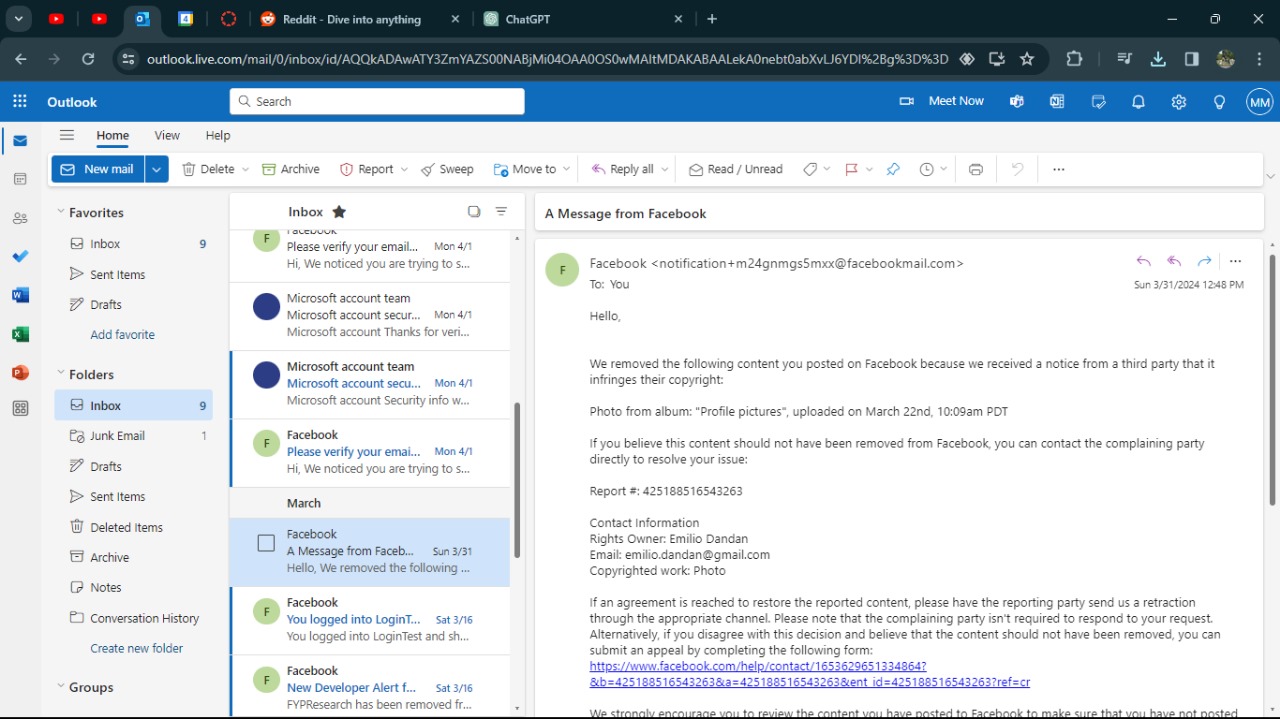
To validate the functionality of our code for reporting copyright violations on Facebook, we created a simulated scenario. We set up a fake account and uploaded a picture sourced from a genuine Facebook profile. Subsequently, we executed the Facebook report script.

Following the execution of the script, we obtained confirmation of successful report submissions. This confirms the effectiveness of our code in automating the process of reporting copyright infringements on Facebook.

Original account’s email:



The fake account’s email:



**Extra points**

Here are a few extra points to note when you're writing your README:

* Keep it up-to-date - It is a good practise to make sure your file is always up-to-date. In case there are changes make sure to update the file where necessary.
* Pick a language - We all come from different zones and we all speak different languages. But this does not mean you need to translate your code into vernacular. Writing your README in English will work since English is a globally accepted language. You might want to use a translator tool here if your target audience isn't familiar with English.