# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

**Write a Python Script to Monitor an Application :** Create a Python script that sends periodic HTTP requests to your application and alerts you if it’s down.

Name : DHEENA S  
Department : MECHANICAL ENGINEERING



# Introduction

# Ensuring high availability and reliability of an application is crucial for maintaining a seamless user experience, as it ensures the application is consistently accessible and performs as expected without failures. This Proof of Concept (PoC) focuses on developing a lightweight Python script to monitor the health status of an application by sending periodic HTTP requests to its endpoint. By analyzing responses, such as status codes and response times, the script can detect downtime, measure performance, and proactively identify issues before they impact users. This approach provides a simple yet effective way to improve application reliability and deliver a smooth experience for users. Overview

### Python Script to Monitor an Application

# **1. **Set Up Monitoring**: Install Python and libraries like requests and smtplib. Set up an email account for alerts.**

# **2. **Send HTTP Requests**: Use requests to check the app’s URL regularly.**

# **3. **Check App Status**: If the response code is 200, the app is working. If not, it’s down.**

# **4. **Send Alerts**: If the app is down, use smtplib to email a notification.**

# **5. **Automate Checks**: Use a loop to check the app’s status every 60 seconds.**

# **6. **Log Errors**: Record any issues and handle errors to keep the script running smoothly.**Objectives

### Python Script to Monitor an Application

**1. Learn Web Monitoring: Understand how to check if a web app is working using HTTP requests.**

**2. Practice Python Scripting: Write scripts to interact with web services and manage errors.**

**3. Set Up Alerts: Create a script that detects app downtime and sends email notifications.**

**4. Understand Status Codes: Learn what HTTP codes like 200 (OK), 404 (Not Found), and 500 (Error) mean for app health.**

**5. Automate Emails: Use SMTP to send automatic alerts when problems occur.**

**6. Ensure Reliability: Build a simple, continuous monitoring system to keep your app running smoothly.**

# Importance Python Script to Monitor an Application

1. **Proactive Monitoring**: The script keeps an eye on your app’s health, catching issues before users notice.

**2. Get Instant Alerts: Email notifications let you fix problems quickly, reducing downtime.**

**3. Boost Reliability: Automated checks keep your app running smoothly for users.**

**4. Save Money: Fixing issues early avoids costly downtime and lost revenue.**

**5. Build Skills: Writing the script improves your skills in monitoring, error handling, and email alerts.**

**6. Grow with Your Needs: The script can be expanded to monitor more apps or fit into bigger systems for large-scale monitoring.**

# Step-by-Step Overview

#### ****Step 1: Install Python from Microsoft Store****

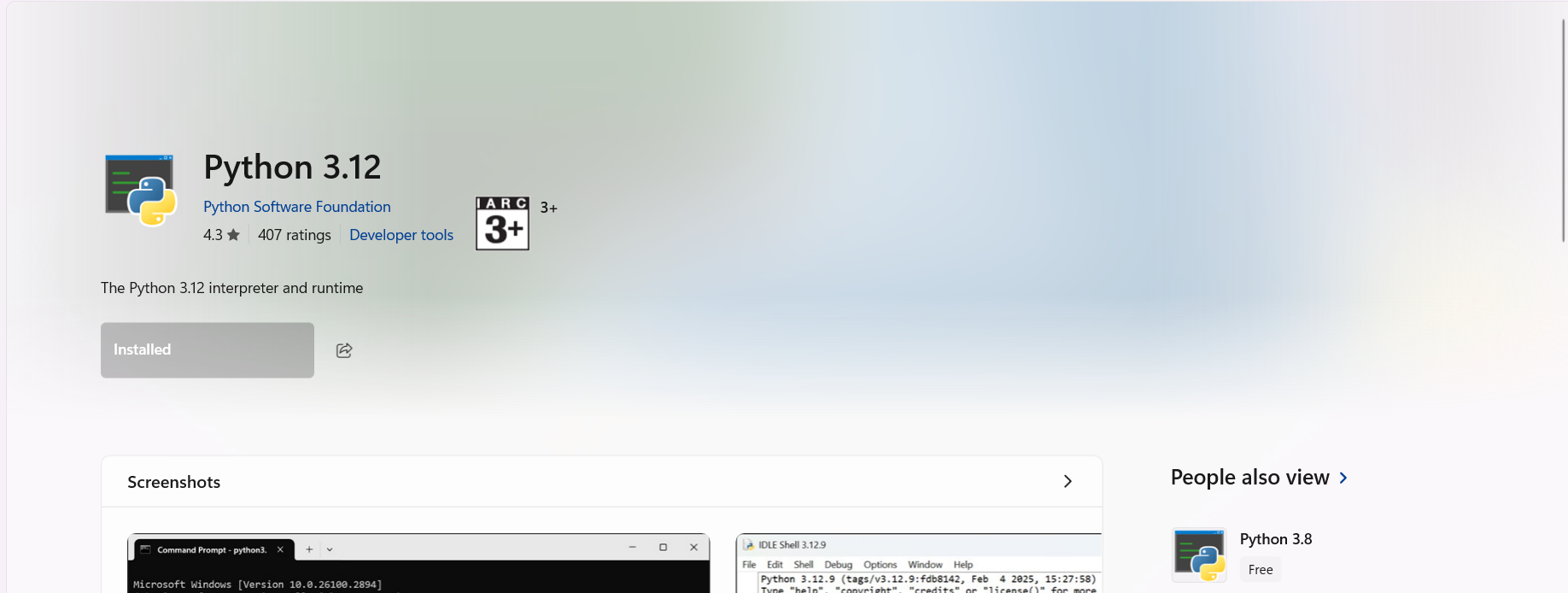
1. Open the Microsoft Store on your computer.

2. Search for "Python" in the search bar and press Enter.

3. Select the latest version of Python (e.g., Python 3.x.x).

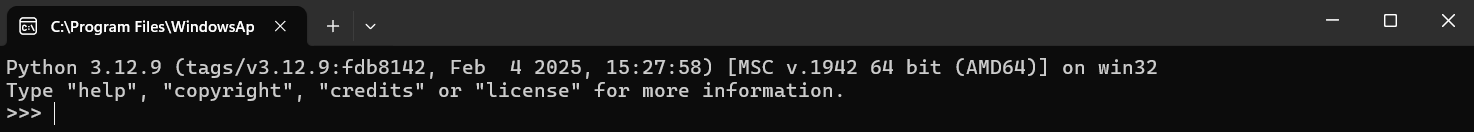
4. Click the Install button to install Python.

- This will automatically add Python to your system’s PATH environment variable.



#### ****Step 2: Verify Python Installation****

1. Open the **Command Prompt**:
2. Check the python version, which will show on the top.

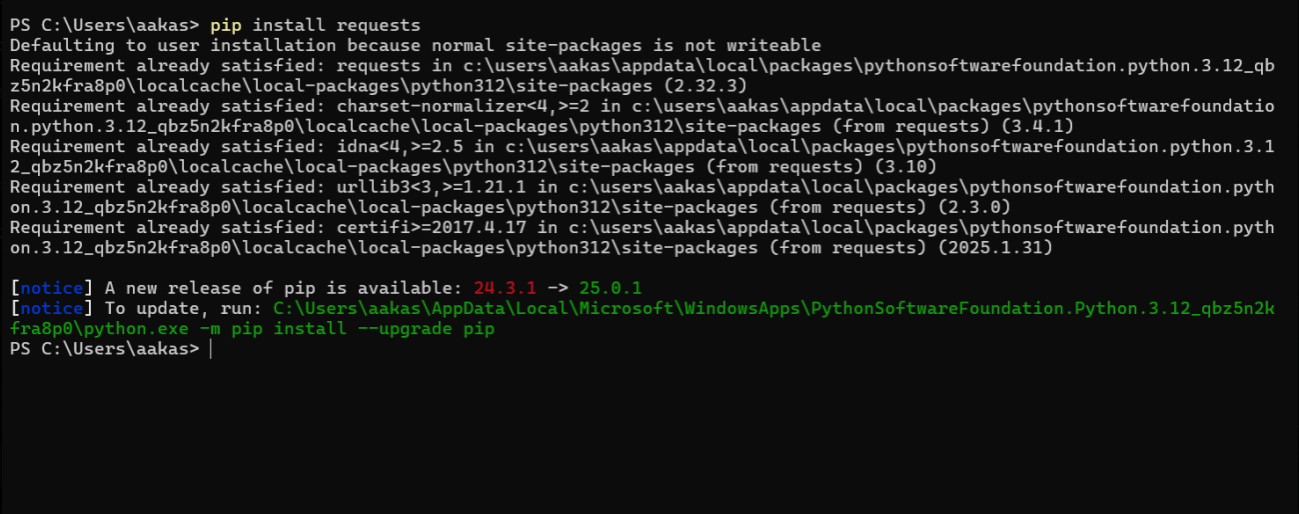


#### ****Step 3: Install Required Libraries (requests, smtplib)****

1. In **Command Prompt (CMD)**, type the following command to install the **requests** library:

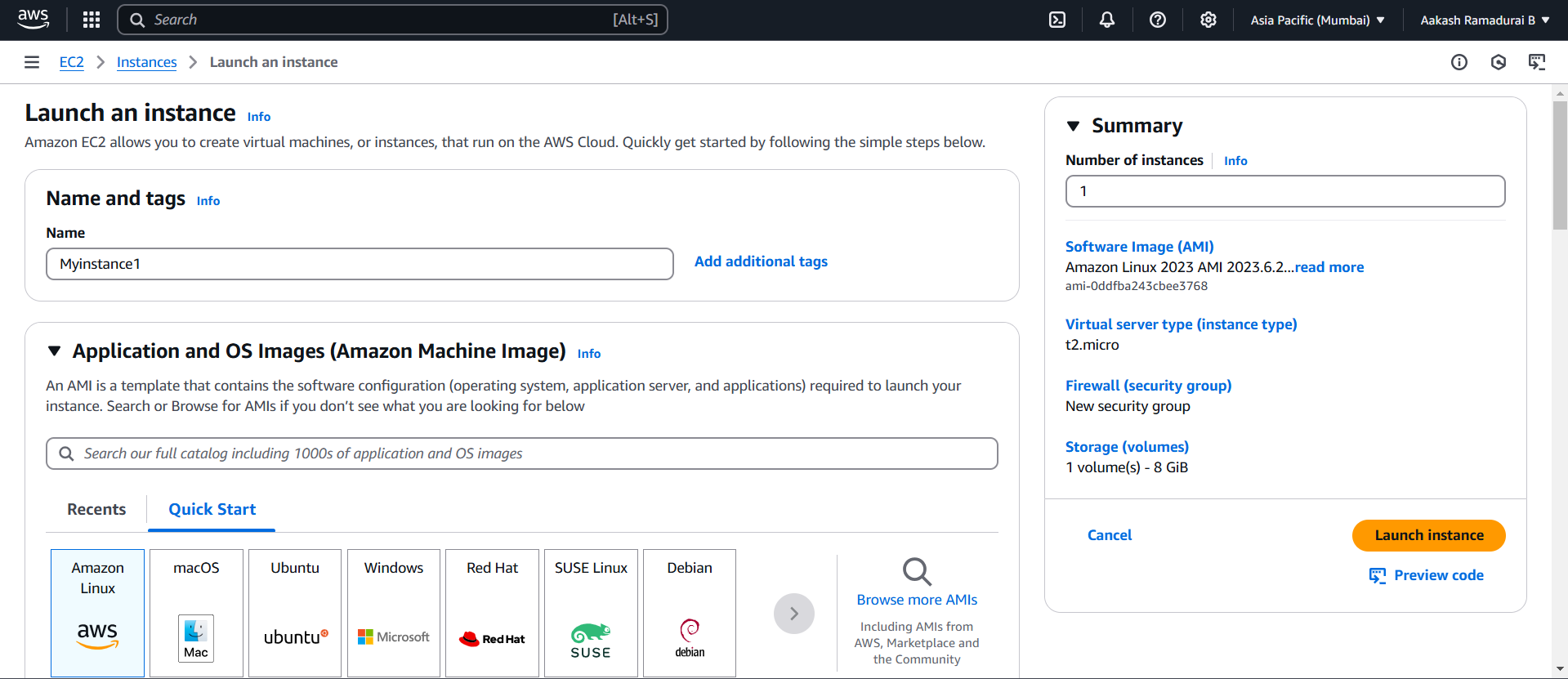
**pip install requests**

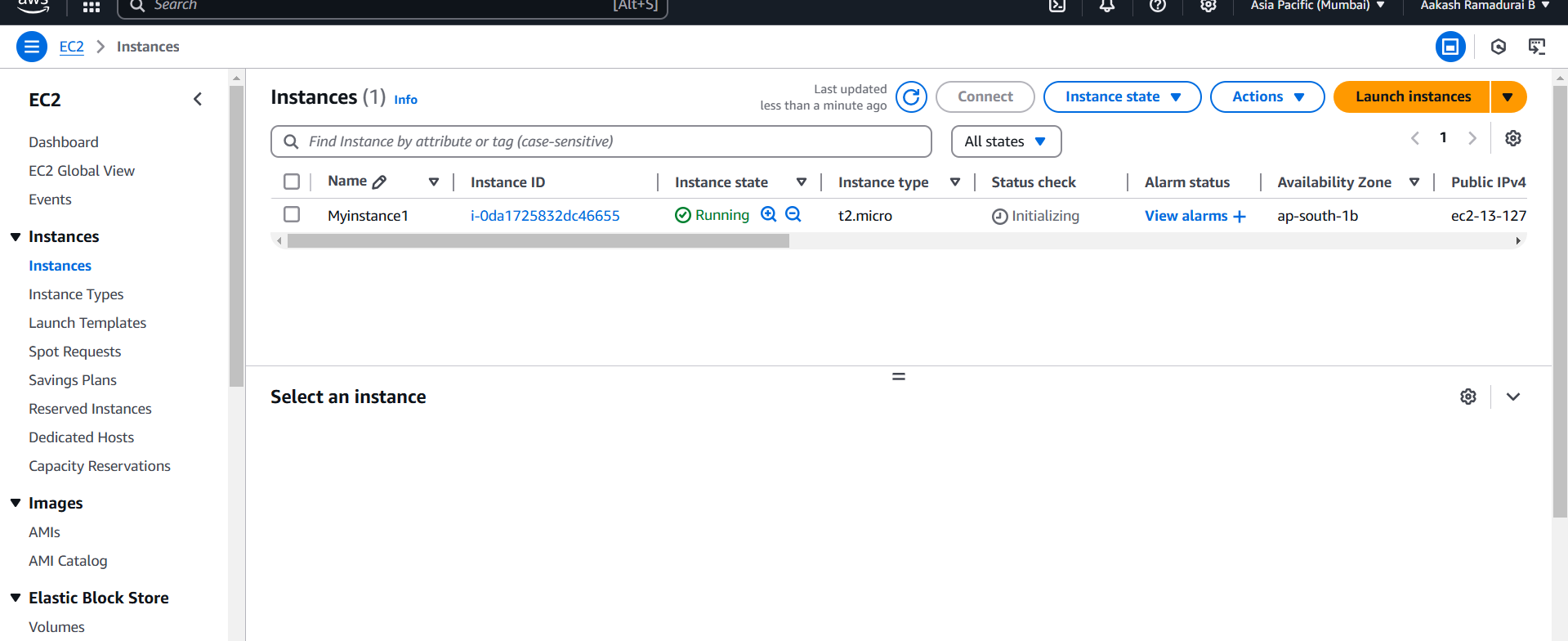
1. The **smtplib** library is included with Python by default, so no installation is needed for it.

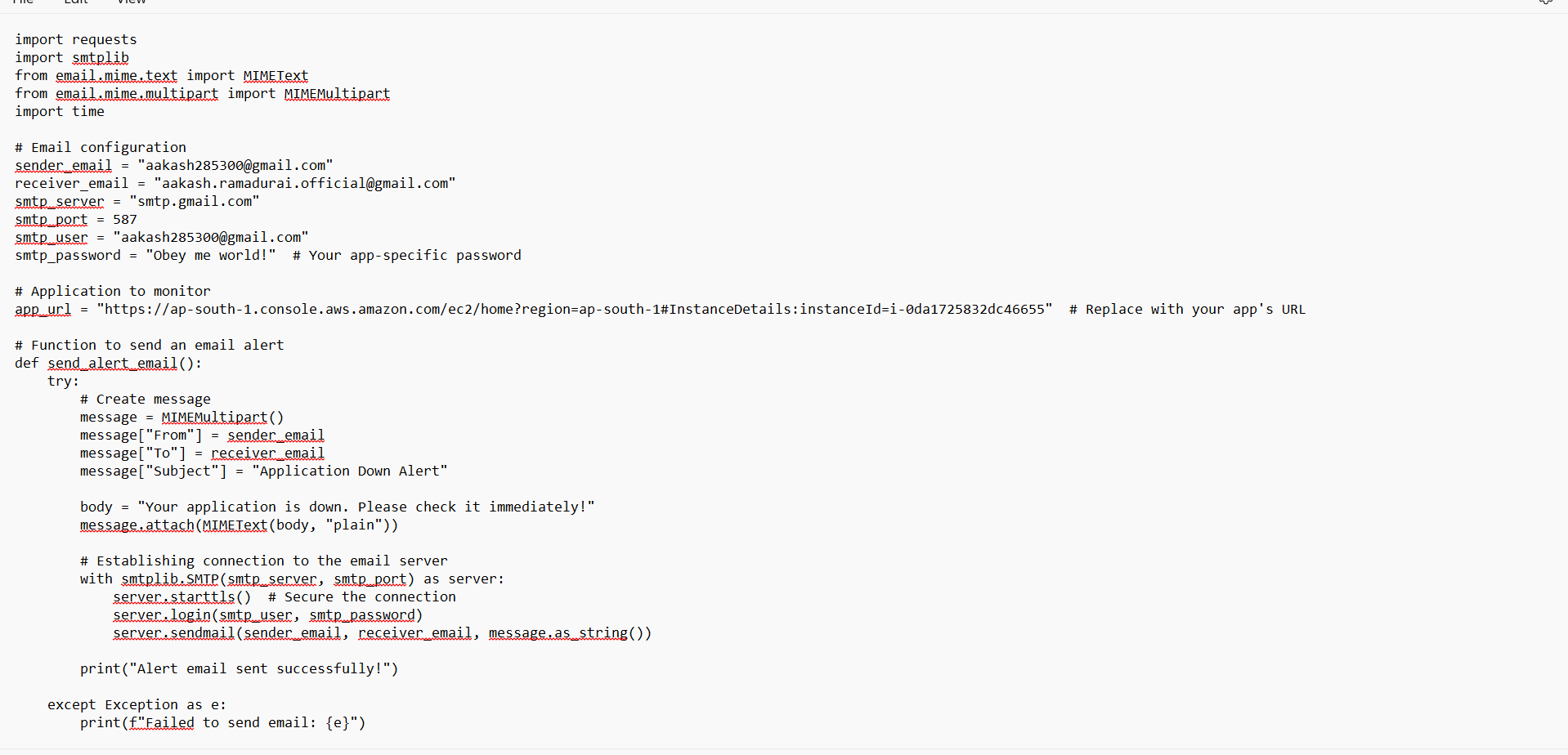


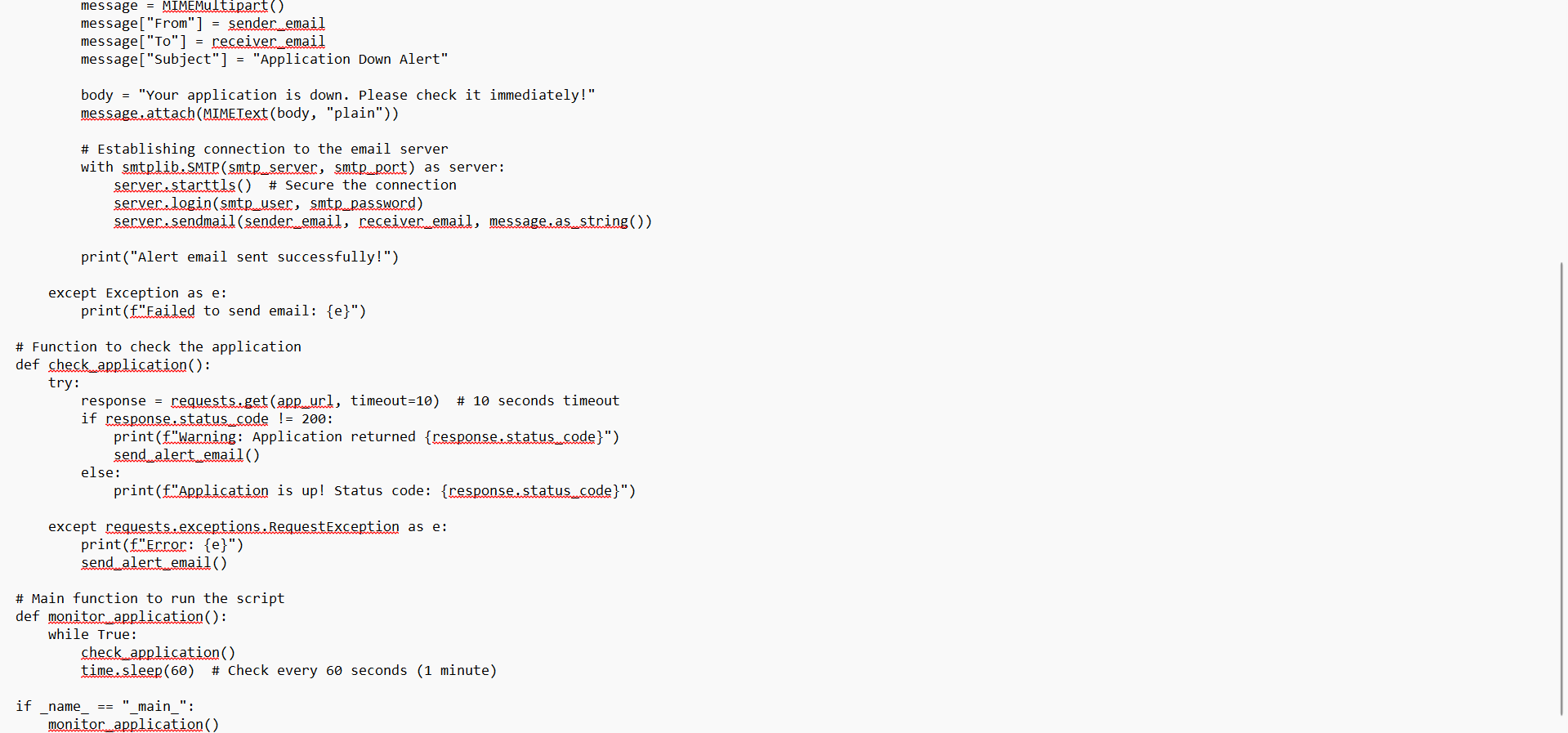
#### ****Step 4: Write the Python Script****

1. Create a EC2 Instance
2. Open any **text editor** (e.g., Notepad, VS Code).
3. Copy and paste the Python script to monitor your EC2 instance (from your PoC).
4. Change your\_email@example.com to your actual Gmail address (e.g., [your\_email@gmail.com](mailto:your_email@gmail.com)).
5. Set smtp\_user to your **Gmail** address as well.
6. Enter your **app-specific password** (not your Gmail password) for the smtp\_password field. If you don't have an app-specific password, you can create one in your Google Account settings (in the **Security** section under **App passwords**)
7. Also Change the app\_url to your Instance URL
8. Save the file with a **.py** extension, e.g., monitor\_app.py









#### ****Step 5: Run the Python Script****

1. In **Command Prompt (CMD)**, navigate to the folder where the Python script is saved using the cd command:

cd path\to\your\script\directory

1. Run the script with the following command:

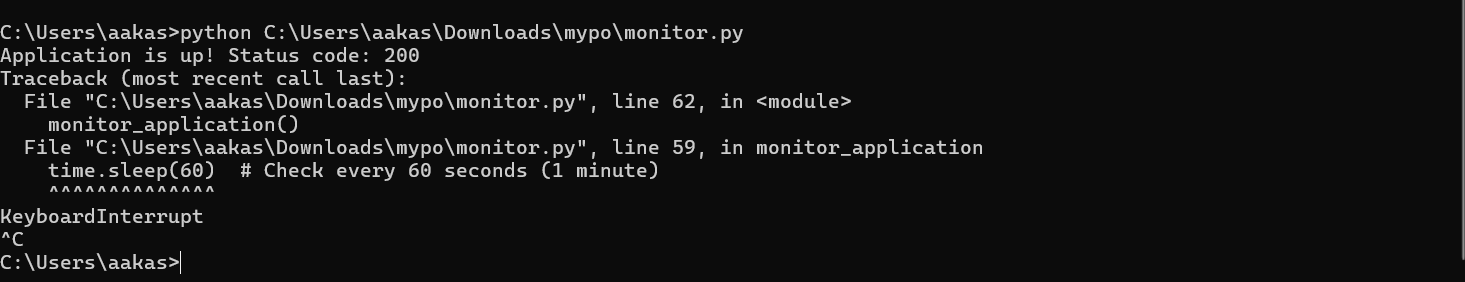
python monitor\_app.py

#### Screenshot 2025-02-26 103141

#### ****Step 6: Stop the Script****

To stop the script at any time, press **Ctrl + C** in the **Command Prompt** window.

.



# Outcome

- **Check App Health**: Regularly send HTTP requests to see if your app is working.

- **Send Alerts Automatically**: Email notifications are sent instantly if the app goes down, helping you act fast.

- **Handle Errors**: Detect and manage issues like network problems, timeouts, or failed requests.

- **Automate the Script**: Run the script every 60 seconds (or as needed) to keep checking the app’s status.

- **Boost Reliability**: Keep your app running smoothly by monitoring it in real-time and fixing issues quickly.

- **Set Up Email Alerts**: Use SMTP to notify admins or team members immediately if the app has problems.

THANK YOU!