Configuration Manual

MSc Research Project
Cybersecurity

Shyam Natarajan
Student ID: X23140321

School of Computing
National College of Ireland

Supervisor: Liam Mcabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shyam Natarajan		
Student ID:	X23140321		
Programme:	MSc in Cybersecurity	**Year:**	2024
Module:	MSc Practicum		
Lecturer:	Liam Mcabe		
Submission Due Date:	12/12/2024		
Project Title:	A Standalone tool for Real Time phishing detection.		
Word Count:	**1020**	**Page Count:**	**09**

Signature: Shyam Natarajan

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shyam Natarajan
Student ID: x23140321

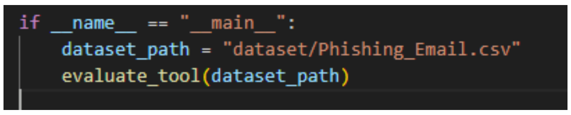
1 Introduction

The configuration manual contains important details about the list and versions of software and hardware used in this project. The research project is about creating a user friendly ai based real time phishing detection standalone tool which effectively identify phishing attempts and notify the user on the dashboard. It scans for URL attachments, suspicious html tags, media attachments, files, text with NLP module. The following manual contains the comprehensive manual about how to connect the scripts and deploy the tool which is used in this project.

2 Minimum hardware and software requirements.

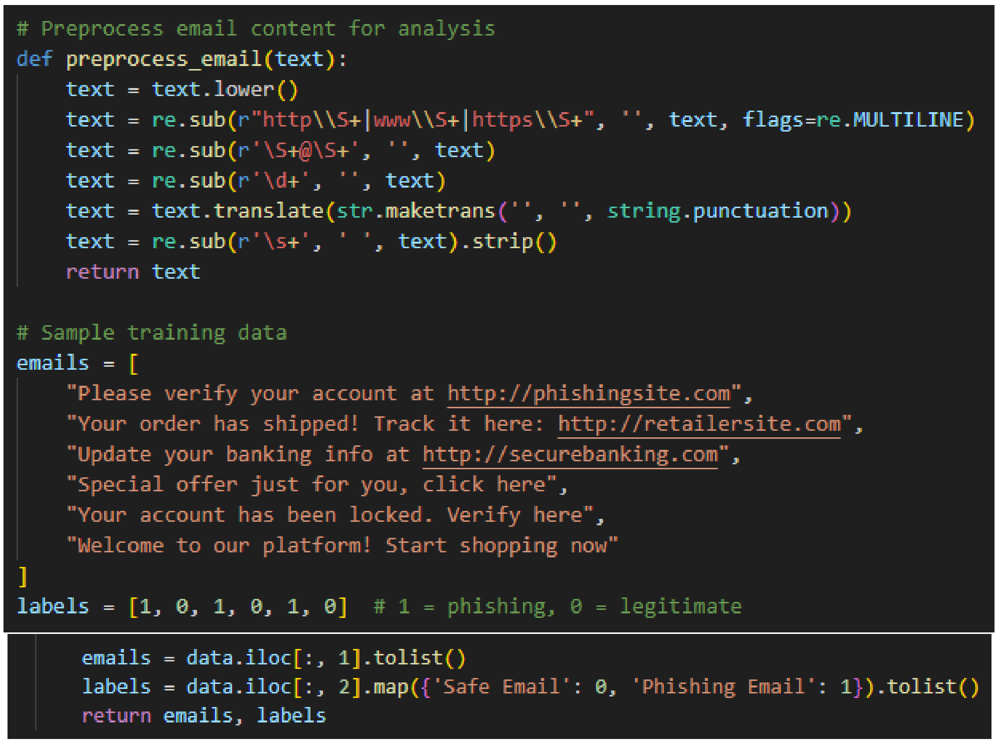
- Processor: 12th Gen Intel(R) Core (TM) i5-1235U, 1300 Mhz.
- Physical Memory (RAM) 4 GB DDR4, 3200 Mhz.
- Operating system used: Microsoft Windows 11 Home.
- System Type x64-based PC.
- Storage: 20 GB HDD/SSD.
- GPU: Intel(R) Iris(R) Xe Graphics 2gb (Shared GPU memory).
- Visual studio code: 1.95.3.
- Python: 3.12.5 [1].
- Microsoft edge 131.0.29.3.70.
- Microsoft Excel.

3 Python, html scripts in Visual Studio code IDE. Model training and importing and its uses.



```python
if __name__ == "__main__":
    dataset_path = "dataset/Phishing_Email.csv"
    evaluate_tool(dataset_path)
```

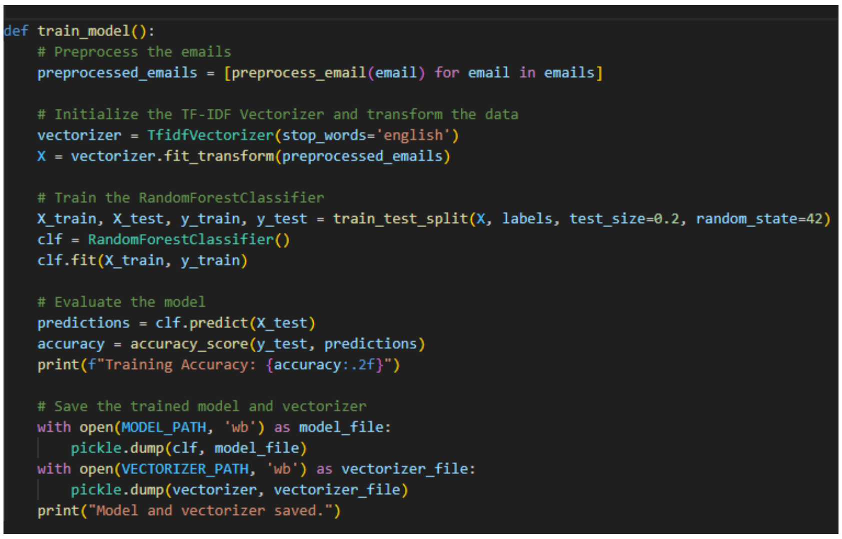
Figure 3.1 importing data set in phase 2 testing in the ifdof.py script.



```python
# Preprocess email content for analysis
def preprocess_email(text):
    text = text.lower()
    text = re.sub(r"http\\S+|www\\S+|https\\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'\S+@\S+', '', text)
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Sample training data
emails = [
    "Please verify your account at http://phishingsite.com",
    "Your order has shipped! Track it here: http://retailersite.com",
    "Update your banking info at http://securebanking.com",
    "Special offer just for you, click here",
    "Your account has been locked. Verify here",
    "Welcome to our platform! Start shopping now"
]
labels = [1, 0, 1, 0, 1, 0]  # 1 = phishing, 0 = legitimate
```

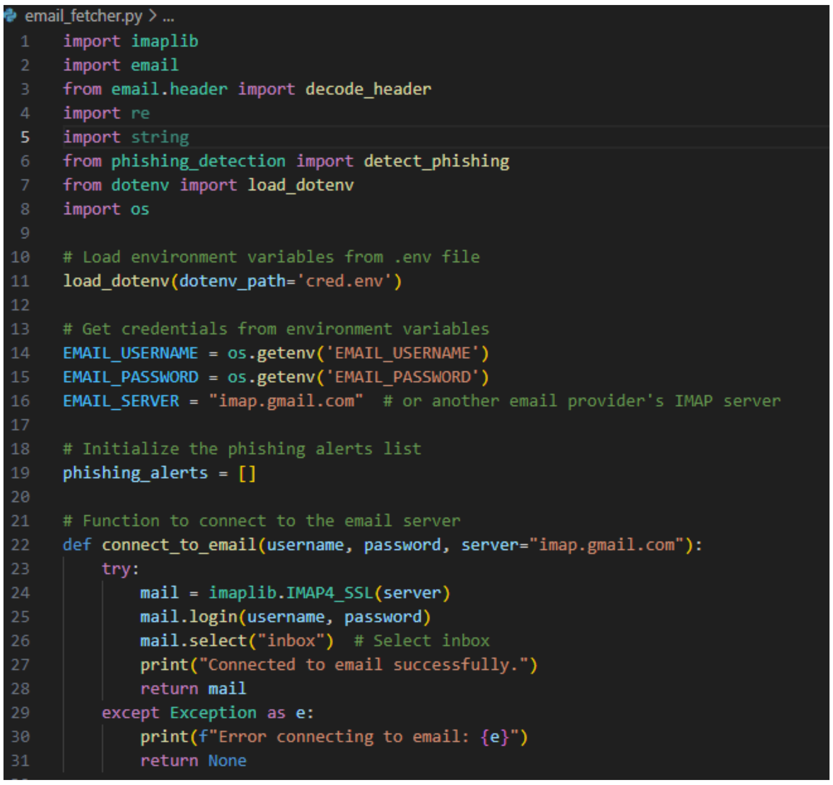
```python
    emails = data.iloc[:, 1].tolist()
    labels = data.iloc[:, 2].map({'Safe Email': 0, 'Phishing Email': 1}).tolist()
    return emails, labels
```

Figure 3.2 Label encoding with data preprocessing in phishing_detection.py and ifdof.py.



```python
def train_model():
    # Preprocess the emails
    preprocessed_emails = [preprocess_email(email) for email in emails]

    # Initialize the TF-IDF Vectorizer and transform the data
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(preprocessed_emails)

    # Train the RandomForestClassifier
    X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)
    clf = RandomForestClassifier()
    clf.fit(X_train, y_train)

    # Evaluate the model
    predictions = clf.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    print(f"Training Accuracy: {accuracy:.2f}")

    # Save the trained model and vectorizer
    with open(MODEL_PATH, 'wb') as model_file:
        pickle.dump(clf, model_file)
    with open(VECTORIZER_PATH, 'wb') as vectorizer_file:
        pickle.dump(vectorizer, vectorizer_file)
    print("Model and vectorizer saved.")
```

Figure 3.3 Training the Random Forest classifier model with 80:20 model training and testing data split.


email_fetcher.py > ...
```
import imaplib
import email
from email.header import decode_header
import re
import string
from phishing_detection import detect_phishing
from dotenv import load_dotenv
import os

# Load environment variables from .env file
load_dotenv(dotenv_path='cred.env')

# Get credentials from environment variables
EMAIL_USERNAME = os.getenv('EMAIL_USERNAME')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
EMAIL_SERVER = "imap.gmail.com"  # or another email provider's IMAP server

# Initialize the phishing alerts list
phishing_alerts = []

# Function to connect to the email server
def connect_to_email(username, password, server="imap.gmail.com"):
    try:
        mail = imaplib.IMAP4_SSL(server)
        mail.login(username, password)
        mail.select("inbox")  # Select inbox
        print("Connected to email successfully.")
        return mail
    except Exception as e:
        print(f"Error connecting to email: {e}")
        return None
```
Figure 3.4 in email_fetcher.py using environment variable cred.env stored in the project directory the credentials will be loaded and connected with the IMAP Gmail server.

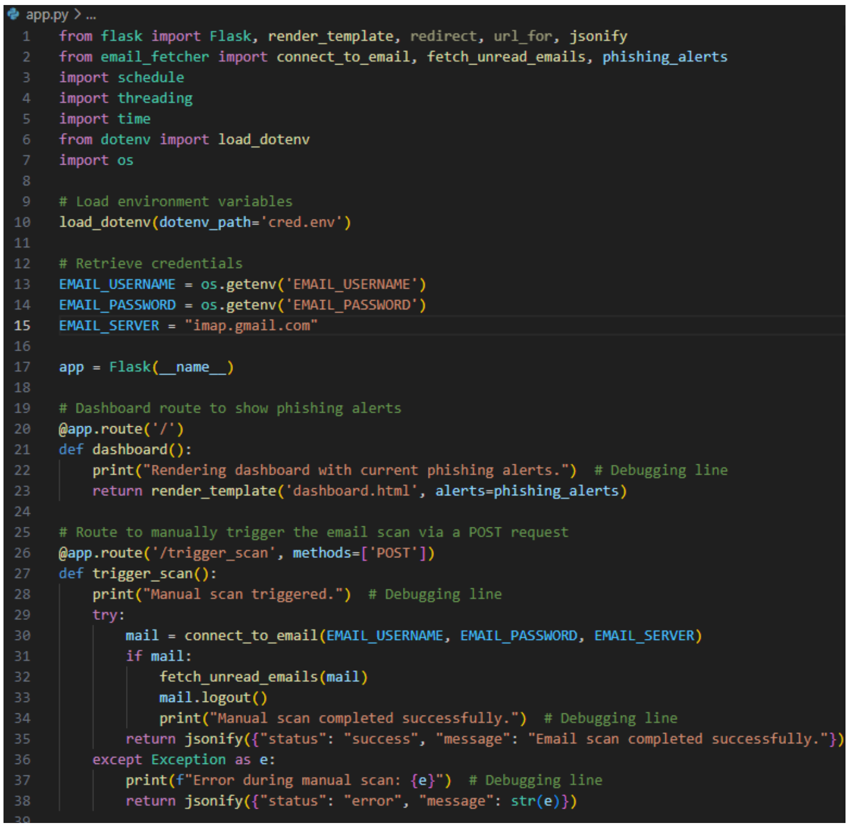
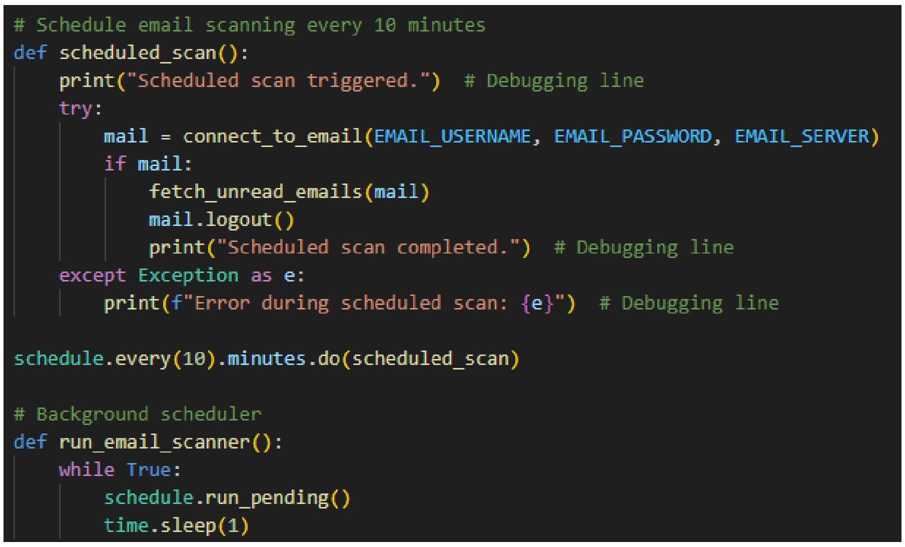

app.py > ...
```
from flask import Flask, render_template, redirect, url_for, jsonify
from email_fetcher import connect_to_email, fetch_unread_emails, phishing_alerts
import schedule
import threading
import time
from dotenv import load_dotenv
import os

# Load environment variables
load_dotenv(dotenv_path='cred.env')

# Retrieve credentials
EMAIL_USERNAME = os.getenv('EMAIL_USERNAME')
EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
EMAIL_SERVER = "imap.gmail.com"

app = Flask(__name__)

# Dashboard route to show phishing alerts
@app.route('/')
def dashboard():
    print("Rendering dashboard with current phishing alerts.")  # Debugging line
    return render_template('dashboard.html', alerts=phishing_alerts)

# Route to manually trigger the email scan via a POST request
@app.route('/trigger_scan', methods=['POST'])
def trigger_scan():
    print("Manual scan triggered.")  # Debugging line
    try:
        mail = connect_to_email(EMAIL_USERNAME, EMAIL_PASSWORD, EMAIL_SERVER)
        if mail:
            fetch_unread_emails(mail)
            mail.logout()
            print("Manual scan completed successfully.")  # Debugging line
        return jsonify({"status": "success", "message": "Email scan completed successfully."})
    except Exception as e:
        print(f"Error during manual scan: {e}")  # Debugging line
        return jsonify({"status": "error", "message": str(e)})
```

Figure 3.5 app.py script contains the following necessary function like app.rout to dashboard which direct the operation to main dashboard where it contains manual triggering the email scan and scheduled phishing detection once in every 10 mins.



```python
# Schedule email scanning every 10 minutes
def scheduled_scan():
    print("Scheduled scan triggered.")  # Debugging line
    try:
        mail = connect_to_email(EMAIL_USERNAME, EMAIL_PASSWORD, EMAIL_SERVER)
        if mail:
            fetch_unread_emails(mail)
            mail.logout()
            print("Scheduled scan completed.")  # Debugging line
    except Exception as e:
        print(f"Error during scheduled scan: {e}")  # Debugging line

schedule.every(10).minutes.do(scheduled_scan)

# Background scheduler
def run_email_scanner():
    while True:
        schedule.run_pending()
        time.sleep(1)
```

Figure 3.6 app.py script contains this scheduled scanning functionality for the phishing detection tool, which ensure the tool is triggered once in every 10 minutes which doesn’t leaves out any unready emails out which ensures absolute protection for the connected inbox.

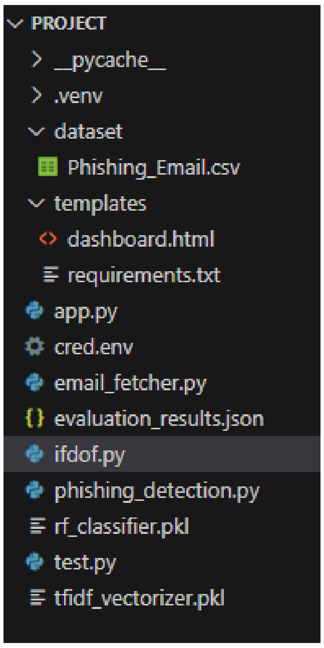
File name	Purpose
App.py (Flask web application setup)	This python flask script provides user interface with interactive feature to manually trigger the process and also contains the schedule function which automatically trigger the fetching and detection function for every 10 minutes which ensure periodic detection.
Email_fetcher.py (email extractor)	This python script provide the core purpose of connecting and fetching emails from Gmail server in a small matter of time. The IMAP server extracts email and pass it down to other scripts for efficient email detection and classification.
Phishing_detection.py (detection and classification script)	This script contains the core components of the whole project, it serves as base for machine learning model and classify the fetched emails using the Email_fetcher.py script. Further details of this script will be explained in the following sections of the report.
Ifdof.py (Evaluation script)	When connected with Imap gmail server the performance cannot be measured parallely using the Phishing_detection.py script so the same model is deployed in this separate script and which is trained and tested with large data set and acquire graphical representation of accuracy and different evaluation metrics, which supports the performance and

	reliability of the model under different circumstances.

Figure 3.7 list of important script involved in the project with its functionality.

4 Directory structure.

There are 4 python script and ".env" fil and csv file involved in this project. Dataset file must be stored in "dataset folder" and "dashboard.html" must be stored in "templates" folder and mail credentials must be stored in "cred.env" file. The structure of the project folder is shown in the below figure 8.1.



```
PROJECT
  > __pycache__
  > .venv
  v dataset
      Phishing_Email.csv
  v templates
      dashboard.html
      requirements.txt
  app.py
  cred.env
  email_fetcher.py
  evaluation_results.json
  ifdof.py
  phishing_detection.py
  rf_classifier.pkl
  test.py
  tfidf_vectorizer.pkl
```

Figure 4.1 Project folder structure.

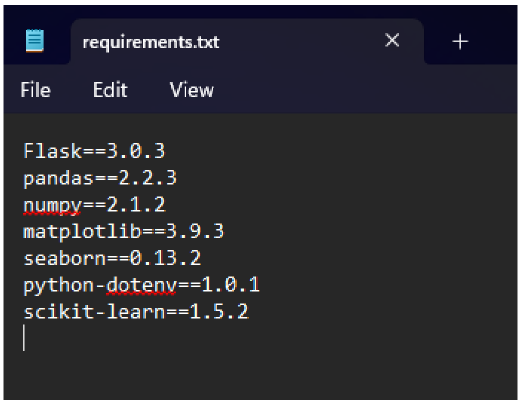
5 Libraries Used

The below mentioned python libraries were installed in the project directory with "pip install <package_name>" command. For that python needs to be installed and to be updated to the latest version.

- Flask: 3.0.3
- Pandas: 2.2.3
- Numpy: 2.1.2
- Matplotlib: 3.9.3
- Seaborn: 0.13.2
- Python-dotenv: 1.0.1
- Scikit-learn: 1.5.2
- Python-dotenv: 1.0.1
- Imaplib: For fetching unread emails and processing them for phishing detection.

Use a reuirement.txt file with all the above packages to install the dependies with pip install <package name > or use this command "python/pip install requirements.txt". or use this command,

pip install -r requirements.txt


requirements.txt

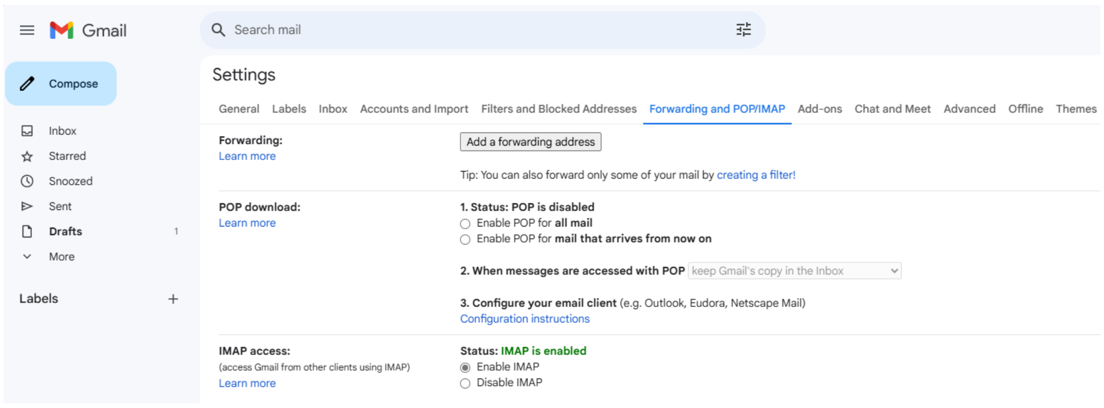
File Edit View

```
Flask==3.0.3
pandas==2.2.3
numpy==2.1.2
matplotlib==3.9.3
seaborn==0.13.2
python-dotenv==1.0.1
scikit-learn==1.5.2
```

Figure 5.1 Requirements.txt file's content.

6 Connecting the standalone tool with live Gmail server.

Step 1: On your Gmail messenger settings make sure to enable IMAP access to access from the VS Code IDE.



Settings

General Labels Inbox Accounts and Import Filters and Blocked Addresses Forwarding and POP/IMAP Add-ons Chat and Meet Advanced Offline Themes

Forwarding: Learn more — Add a forwarding address

Tip: You can also forward only some of your mail by creating a filter!

POP download: Learn more

1. Status: POP is disabled
- Enable POP for all mail
- Enable POP for mail that arrives from now on

2. When messages are accessed with POP: keep Gmail's copy in the Inbox

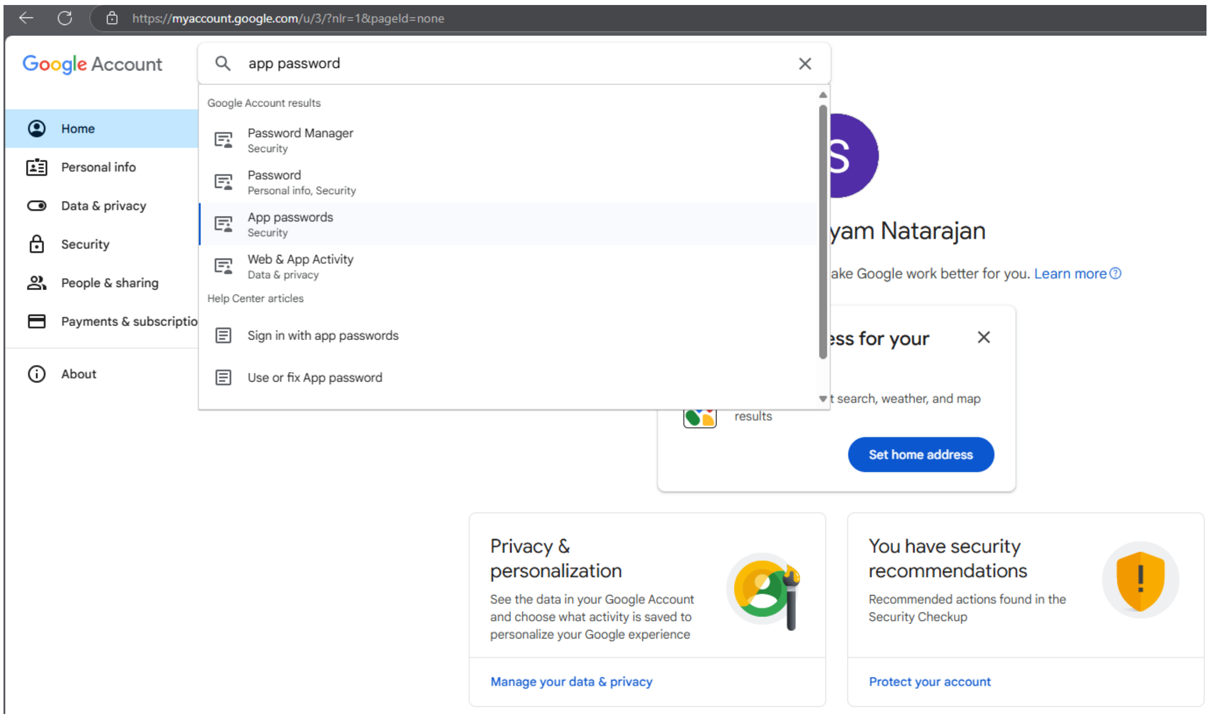
3. Configure your email client (e.g. Outlook, Eudora, Netscape Mail) Configuration instructions

IMAP access: (access Gmail from other clients using IMAP) Learn more

Status: IMAP is enabled
- [✓ Enable IMAP]
- Disable IMAP

Figure 5.1 Enabling IMAP access in Gmail settings.

Step 2: Next on the google account settings set up separate application password and generate 16-digit hexa digit hash to by pass the 2 factor authentication in the Gmail account.



app password

Google Account results
- Password Manager — Security
- Password — Personal info, Security
- App passwords — Security
- Web & App Activity — Data & privacy

Help Center articles
- Sign in with app passwords
- Use or fix App password

yam Natarajan

ake Google work better for you. Learn more

ess for your

t search, weather, and map results

Set home address

Privacy & personalization

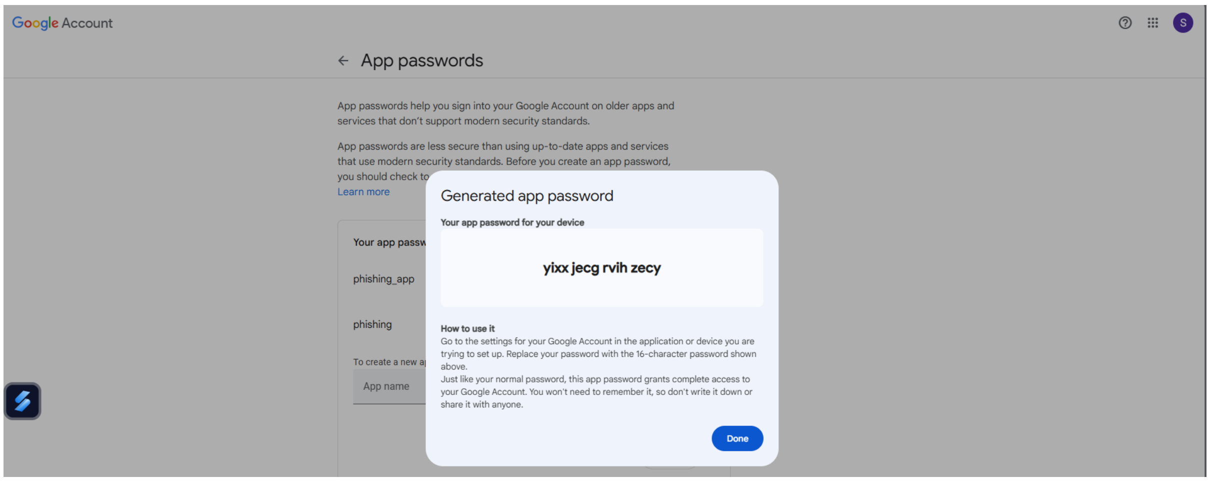
See the data in your Google Account and choose what activity is saved to personalize your Google experience

Manage your data & privacy

You have security recommendations

Recommended actions found in the Security Checkup

Protect your account



App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to

Generated app password

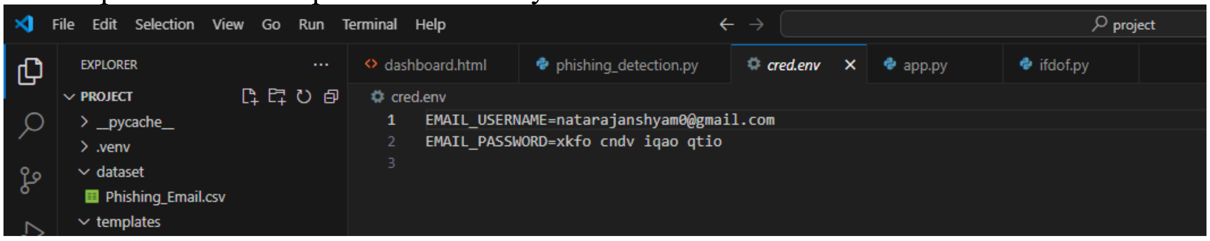
Your app password for your device

yixx jecg rvih zecy

How to use it
Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.
Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

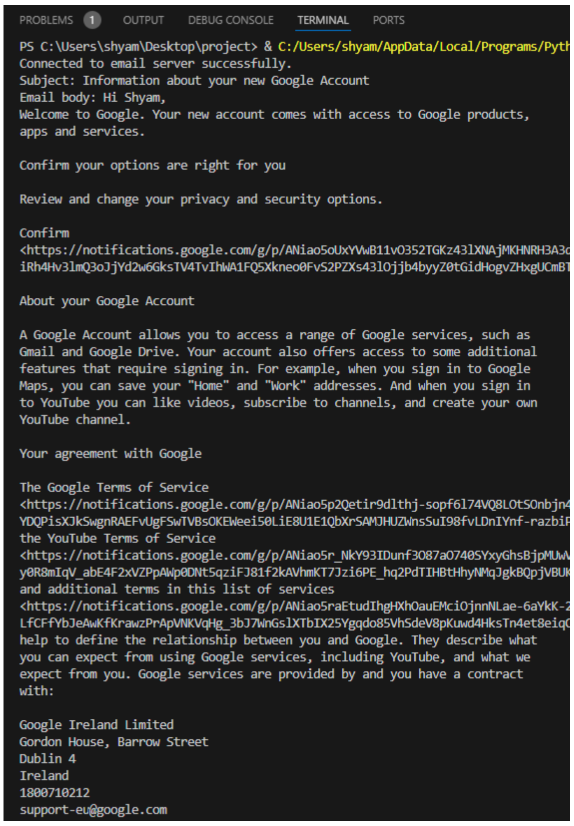
Figure 5.2 creating app password on Gmail account and generating the hash password.

Step 3: Create a .env file after importing python-dotenv library with the generated credentials. And import it in the script when necessary.



```
EMAIL_USERNAME=natarajanshyam0@gmail.com
EMAIL_PASSWORD=xkfo cndv iqao qtio
```

Figure 5.3 mail and app password used in the project.



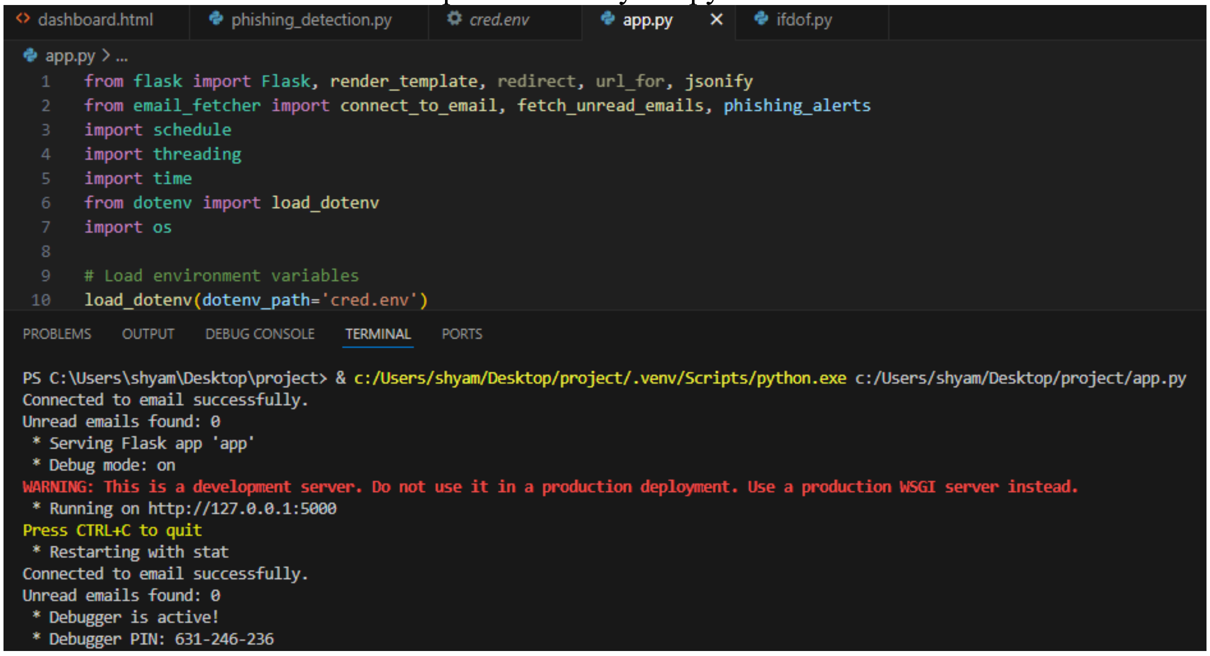
```
PS C:\Users\shyam\Desktop\project> & C:/Users/shyam/AppData/Local/Programs/Pyth
Connected to email server successfully.
Subject: Information about your new Google Account
Email body: Hi Shyam,
Welcome to Google. Your new account comes with access to Google products,
apps and services.

Confirm your options are right for you

Review and change your privacy and security options.

Confirm
<https://notifications.google.com/g/p/ANiao5oUxYVwB11v0352TGKz431XNAjMKHNRH3A3c
iRh4Hv3lmQ3oJjYd2w6GksTV4TvIhWA1FQ5Xkneo0FvS2PZXs43l0jjb4byyZ0tGidHogvZHxgUCmBT

About your Google Account

A Google Account allows you to access a range of Google services, such as
Gmail and Google Drive. Your account also offers access to some additional
features that require signing in. For example, when you sign in to Google
Maps, you can save your "Home" and "Work" addresses. And when you sign in
to YouTube you can like videos, subscribe to channels, and create your own
YouTube channel.

Your agreement with Google

The Google Terms of Service
<https://notifications.google.com/g/p/ANiao5p2Qetir9dlthj-sopf6l74VQ8LOtSOnbjn4
YDQPisXJkSwgnRAEFvUgFSwTVBsOKEWeei50LiE8U1E1QbXrSAMJHUZWnsSuI98fvLDnIYnf-razbiF
the YouTube Terms of Service
<https://notifications.google.com/g/p/ANiao5r_NkY93IDunf3O87aO740SYxyGhsBjpMUwV
y0R8mIqV_abE4F2xVZPpAWp0DNt5qziFJ81f2kAVhmKT7Jzi6PE_hq2PdTIHBtHhyNMqJgkBQpjVBUK
and additional terms in this list of services
<https://notifications.google.com/g/p/ANiao5raEtudIhgHXhOauEMciOjnnNLae-6aYkK-2
LfCFfYbJeAwKfKrawzPrApVNKVqHg_3bJ7WnGslXTbIX25Ygqdo85VhSdeV8pKuwd4HksTn4et8eiqC
help to define the relationship between you and Google. They describe what
you can expect from using Google services, including YouTube, and what we
expect from you. Google services are provided by and you have a contract
with:

Google Ireland Limited
Gordon House, Barrow Street
Dublin 4
Ireland
1800710212
support-eu@google.com
```

Figure 5.4 after successful connection from the vscode IDE with imap Gmail server.

7 Evaluation and testing

The evaluation and testing consist of 2 phases, phase 1 shows the results of the models working by connected with real time scanning results from Gmail server and testing its detection capability by conducting real time scanning against live messages that I have launched from separate mail account. [2] Phase 2 shows the results of the models from testing its performance against larger dataset with over 15000 columns of different emails.

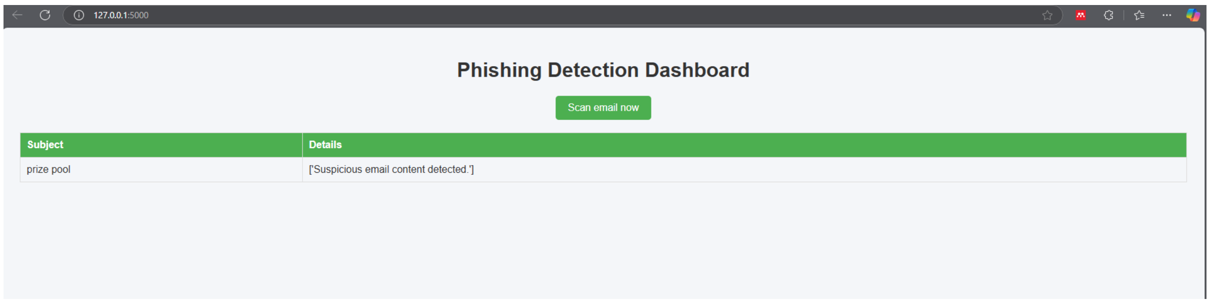
Phase1: The application is started by executing the flask script in the name "app.py" and connect with the web address and port created by the python server in the terminal.



```python
from flask import Flask, render_template, redirect, url_for, jsonify
from email_fetcher import connect_to_email, fetch_unread_emails, phishing_alerts
import schedule
import threading
import time
from dotenv import load_dotenv
import os

# Load environment variables
load_dotenv(dotenv_path='cred.env')
```

```
PS C:\Users\shyam\Desktop\project> & c:/Users/shyam/Desktop/project/.venv/Scripts/python.exe c:/Users/shyam/Desktop/project/app.py
Connected to email successfully.
Unread emails found: 0
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
Connected to email successfully.
Unread emails found: 0
 * Debugger is active!
 * Debugger PIN: 631-246-236
```

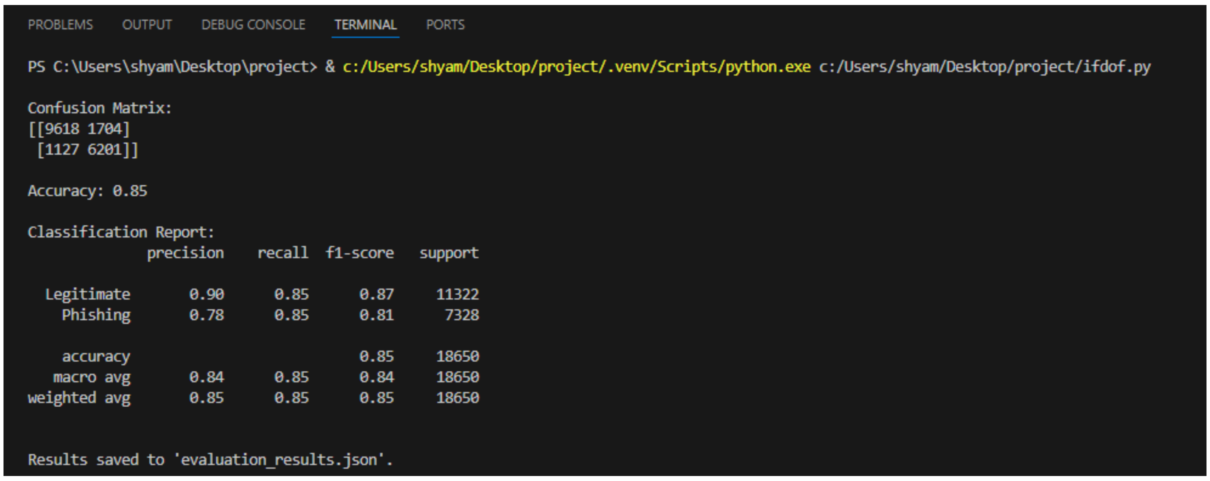
Figure 6.1 starting the application "app.py".



Subject	Details
prize pool	['Suspicious email content detected.']

Figure 6.2 Flask dashboard with detection triggering button and real time results and the unread emails will be scanned for phishing content is hosted on localhost: 127.0.0.1:5000

Phase 2: Stress/ performance test against a public data set from Kaggle to measure its accuracy and performance [3].



```
PS C:\Users\shyam\Desktop\project> & c:/Users/shyam/Desktop/project/.venv/Scripts/python.exe c:/Users/shyam/Desktop/project/ifdof.py

Confusion Matrix:
[[9618 1704]
 [1127 6201]]

Accuracy: 0.85

Classification Report:
              precision    recall  f1-score   support

  Legitimate       0.90      0.85      0.87     11322
    Phishing       0.78      0.85      0.81      7328

    accuracy                           0.85     18650
   macro avg       0.84      0.85      0.84     18650
weighted avg       0.85      0.85      0.85     18650


Results saved to 'evaluation_results.json'.
```

Figure 6.3 model’s training and testing accuracy

References

[1] “Download Python | Python.org.” Accessed: Dec. 04, 2024. [Online]. Available: https://www.python.org/downloads/

[2] S. Asiri, Y. Xiao, S. Alzahrani, and T. Li, “PhishingRTDS: A real-time detection system for phishing attacks using a Deep Learning model,” *Comput Secur*, vol. 141, p. 103843, Jun. 2024, doi: 10.1016/J.COSE.2024.103843.

[3] “Phishing Email Detection.” Accessed: Dec. 04, 2024. [Online]. Available: https://www.kaggle.com/datasets/subhajournal/phishingemails?resource=download