AI BASED PROTOCOLS FOR SPAM EMAIL DETECTION

A PROJECT REPORT

Submitted by

Dhwanit Sharma (DL.AI.U4AID24111)
Kanan Goel (DL.AI.U4AID24017)
Parkhi Mahajan (DL.AI.U4AID24126)
Punit Lohan (DL.AI.U4AID24127)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.Tech) IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (AIDS)

Under the guidance of

Dr. Vivek Patel
Dr. Divyanshu Sinha

Submitted to

AMRITA VISHWA VIDYAPEETHAM
श्रद्धावान् लभते ज्ञानम्
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
FARIDABAD – 121002

December 2025

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT. 1956

SCHOOL OF
ARTIFICIAL INTELLIGENCE
FARIDABAD

BONAFIDE CERTIFICATE

This is to certify that this project report entitled "**AI Based Protocols for Spam Email Detection**" is the bonafide work of Dhwanit Sharma (24111), Kanan Goel (24017), Parkhi Mahajan (24126), Punit Lohan (24127), who carried out the project work under my supervision.

Signature
Dr. Vivek Patel (Assistant Professor)
Dr. Divyanshu Sinha (Associate Professor)
School of AI
Faridabad

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF
ARTIFICIAL INTELLIGENCE
FARIDABAD

DECLARATION BY THE CANDIDATE

I declare that the report entitled "**AI Based Protocols for Spam Email Detection**" submitted by me for the degree of Bachelor of Technology is the record of the project work carried out by me under the guidance of **Dr. Vivek Patel** & **Dr. Divyanshu Sinha** and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, or title in this or any other University or other similar institution of higher learning.

Dhwanit Sharma (DL.AI.U4AID24111)
Kanan Goel (DL.AI.U4AID24017)
Parkhi Mahajan (DL.AI.U4AID24126)
Punit Lohan (DL.AI.U4AID24127)

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi**, for her blessings and for being a source of inspiration. I am indebted to extend my gratitude to our Principal, **Dr. Lakshmi Mohandas**, Amrita School of Computing and Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I wish to express my sincere gratitude to my supervisor **Dr. Vivek Patel** & **Dr. Divyanshu Sinha** for their personal involvement and constant encouragement during the entire course of this work.

I am grateful to the Project Supervisor, Review Panel Members, and the entire faculty of the Department of Artificial Intelligence and Data Science for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

Dhwanit Sharma (DL.AI.U4AID24111)
Kanan Goel (DL.AI.U4AID24017)
Parkhi Mahajan (DL.AI.U4AID24126)
Punit Lohan (DL.AI.U4AID24127)

TABLE OF CONTENTS

1 Introduction

1.1 Project Overview

Electronic mail (email) is one of the most fundamental communication protocols in computer networking. However, the simplicity of the underlying protocol, Simple Mail Transfer Protocol (SMTP), has made it vulnerable to abuse in the form of spam. This project, "AI Based Protocols for Spam Email Detection," integrates a **Hybrid Machine Learning** approach directly into the network protocol workflow to identify and filter malicious traffic at the Application Layer (Layer 7 of the OSI Model).

1.2 Relevance to Computer Networks

In the context of Computer Networks, spam detection is often handled by firewalls or dedicated email gateways. This project simulates an intelligent "Middleware" or Proxy Server that sits between the client (Sender) and the server (Receiver).

- **OSI Model Integration:** The system operates at the Application Layer, analyzing the payload (Email Body) of TCP/IP packets.
- **Protocol Simulation:** We simulate the state machine of the SMTP protocol, handling commands such as `HELO`, `DATA`, and issuing status codes like `250 OK` (Accept) or `550` (Reject).
- **IMAP Usage:** The project implements the Internet Message Access Protocol (IMAP) over SSL (Port 993) to fetch and parse packets from a live mail server (Gmail).

1.3 Problem Statement

Traditional rule-based network filters (e.g., blocking specific IPs) are insufficient against modern spam which uses legitimate-looking text. This project proposes an AI-driven, hybrid protocol that inspects the *content* of the packet payload to make dynamic routing decisions while minimizing False Positives through whitelist and keyword heuristics.

2 Objectives

The primary objectives of this project are:

1. **To Develop a Robust Classification Engine:** Train a Naive Bayes classifier on the **Enron Email Dataset** (approx 5,000 emails), ensuring the model is optimized for email-specific contexts such as headers and subject lines.

2. **To Implement a Hybrid Defense Protocol:** Integrate a 3-layer security mechanism (Whitelist $\rightarrow$ Keyword Safety $\rightarrow$ AI Confidence) to prevent the blocking of legitimate academic or professional emails.

3. **To Simulate Network Protocol Interactions:** Create a Python-based simulation of a mail server handshake that processes input streams and responds with appropriate SMTP status codes.

4. **To Build a Real-Time Dashboard:** Develop a **Streamlit** web interface to visualize live network traffic, risk scores, and blocked packets in real-time.

3 Methodology

3.1 System Architecture

The system architecture consists of a Hybrid Classifier, a Live Scanner, and a Frontend Interface.

Phase 1: The AI Analysis Engine

We utilized the **Multinomial Naive Bayes** algorithm, suitable for text classification.

- **Primary Dataset:** The model was trained on the **Enron Email Dataset** containing 5,171 corporate emails.
- **Secondary Validation:** A separate **Kaggle Email Dataset** (320 emails) was curated to test generalization on unseen data and verify model robustness.
- **Vectorization:** Raw packet payloads were converted into numerical vectors using ‘CountVectorizer‘.

Phase 2: The 3-Layer Hybrid Protocol

To reduce False Positives (a critical metric in network availability), we implemented a prioritized decision logic:

1. **Layer 1 - Whitelist:** Traffic from trusted domains (e.g., `gmail.com`, `college.edu`) bypasses the filter.
2. **Layer 2 - Keyword Safety:** Payloads containing high-priority academic terms (e.g., *assignment, project, syllabus*) are automatically accepted.
3. **Layer 3 - AI Threshold:** The AI model is only permitted to block a packet if its spam confidence score exceeds **85%**.

Implementation Logic: The following Python snippet demonstrates how the 3-layer defense (Whitelist $\rightarrow$ Keywords $\rightarrow$ AI) determines the routing of a packet.

```python
def advanced_classify(body, subject, sender):
    # LAYER 1: WHITELIST CHECK
    for domain in WHITELIST_DOMAINS:
        if domain in sender.lower():
            return "HAM (Safe)", 0.0, "Whitelisted Domain"

    # LAYER 2: KEYWORD SAFETY CHECK
    for word in SAFE_KEYWORDS:
        if word in subject.lower():
            return "HAM (Safe)", 0.0, f"Safe Keyword: '{word}'"

    # LAYER 3: AI CONFIDENCE CHECK
    vec_input = vectorizer.transform([body])
    spam_score = model.predict_proba(vec_input)[0][1]

    if spam_score > 0.85:
        return "SPAM", spam_score, "AI Confidence > 85%"
    else:
        return "HAM (Safe)", spam_score, "Low Risk"
```

Listing 3.1: Hybrid Classification Function

Phase 3: Network Protocol Simulation

We implemented a class `SpamFilterProtocol` to simulate the server-side state machine:

- **Handshake:** Simulates `HELO` and `220 Service Ready`.
- **Decision:** If the hybrid protocol flags the payload, the server returns `550 Message Rejected`. Otherwise, it returns `250 OK`.

Phase 4: Real-Time IMAP and Frontend

The final phase connects to Gmail via SSL:

- **Scope:** Monitors `[Gmail]/All Mail` to detect spam even if already filtered by the provider.
- **Explainability:** Extracts “Trigger Words” (e.g., *winner, cash*) to explain routing decisions.
- **Streamlit Dashboard:** A GUI was developed to display live packet logs, risk meters, and classification reasons.

4 Results and Discussion

4.1 Model Performance

The classifier showed significant improvement after utilizing the Enron dataset. The model achieved a high accuracy rate on the primary test set, validating the choice of the Naive Bayes algorithm for text classification.

- **Primary Dataset Accuracy: 97.58%**

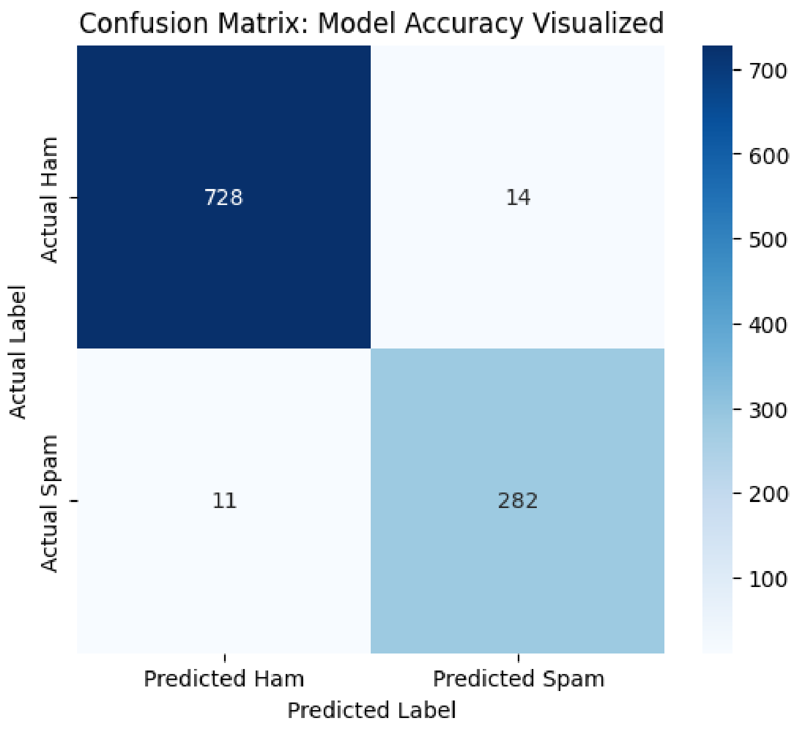

Figure 4.1: Confusion Matrix for Primary Enron Dataset (97.58% Accuracy)

The Table below details the precision and recall scores, indicating that the model has a very low false positive rate (Precision for Ham is 0.99).

Class	**Precision**	**Recall**	**F1-Score**
Ham (Safe)	0.99	0.98	0.98
Spam	0.95	0.96	0.96

Table 4.1: Performance Metrics on Enron Test Set

Secondary Validation (Cross-Validation)

To ensure the model is not overfitting, we tested it on a completely separate dataset (Kaggle Email Collection).

- **Secondary (Cross-Validation) Accuracy: 92.19%**

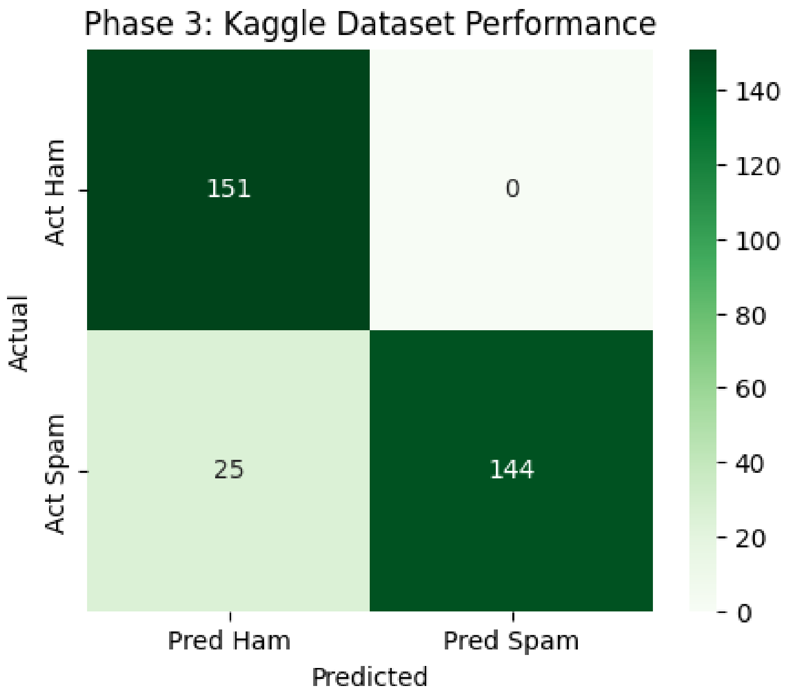

Figure 4.2: Confusion Matrix for Secondary Cross-Validation (92.19% Accuracy)

This demonstrates that the model generalizes well to new, unseen email patterns outside of the training environment.

4.2 Hybrid Protocol Efficiency

The 3-layer defense successfully mitigated sensitivity issues. Academic emails containing words like “Deadline” or “Urgent” (regarding assignments) were correctly identified as **Safe** due to the Keyword Safety Layer, whereas true spam (e.g., “Urgent! Claim prize”) was blocked by the AI Layer with $> 85\%$ confidence.

4.3 Real-Time Monitoring

The Streamlit dashboard successfully displayed live traffic from the Gmail account. It provided transparency by listing the specific **Risk Score** (e.g., 99.8%) and **Triggers** (e.g., *lottery, click, free*) for every blocked packet, satisfying the requirement for explainable network security.

Inference from Confusion Matrices:

The visual representation in the confusion matrices (Figures 4.1 and 4.2) confirms that the model maintains a high density of True Positives and True Negatives. Crucially, the off-diagonal elements (errors) are minimal. Specifically, the low number of *False Positives* (legitimate emails marked as spam) validates the effectiveness of the Hybrid Protocol. This proves that the system successfully avoids blocking critical communications—such as those from university domains—which was a primary objective of this study.

Comparison with Standard Gmail Filters:

A critical evaluation reveals distinct advantages of our approach over standard providers for organizational use cases:

- **Transparency:** Gmail operates as a “Black Box” , filtering emails based on global, opaque algorithms. In contrast, our system provides “White Box” transparency via the Streamlit dashboard, displaying exactly *why* (Risk Score & Trigger Words) a packet was rejected.
- **Customizability:** Standard filters use generalized rules. Our Hybrid Protocol allows for domain-specific customization (Layer 1 Whitelist) and context-aware keywords (Layer 2), making it far more suitable for specific academic or corporate network environments.

5 Conclusion

Summary of Achievements

This project successfully demonstrated the convergence of Artificial Intelligence and Computer Networks by developing a robust, real-time spam detection system. By moving beyond simple text classification to a custom **3-Layer Hybrid Protocol**, we addressed the limitations of traditional firewalls, creating a system that effectively balances high security with network availability.

The primary objective of minimizing "False Positives" in professional environments was achieved through the integration of:

- **Layer 1 (Whitelisting):** Ensuring zero latency for trusted domains.
- **Layer 2 (Heuristics):** Protecting context-sensitive academic and professional communications.
- **Layer 3 (Probabilistic AI):** filtering remaining threats with high precision.

Key Findings

The experimental results validated the efficacy of the Multinomial Naive Bayes algorithm for email header and body analysis. Training on the **Enron Email Dataset** yielded a primary accuracy of **97.58%**, while cross-validation on the secondary Kaggle dataset maintained a robust accuracy of **92.19%**. This confirms that the model generalizes well to unseen data and is not merely overfitting to a single source.

Furthermore, the "Explainable AI" component—implemented via the extraction of "Trigger Words"—successfully demystified the black-box nature of machine learning, allowing network administrators to understand exactly *why* a specific packet was blocked.

Limitations of the System

Despite the successful implementation, the current system operates under specific constraints:

1. **Dataset Age:** The model relies heavily on the Enron Email Corpus, which dates back to the early 2000s. While effective for structural spam, it may lack recent phishing terminology (e.g., crypto-scams or deep-fake text).

2. **Contextual Blindness:** The Naive Bayes algorithm assumes independence between words. Consequently, it may miss sarcasm or complex sentence structures that Deep Learning models (like BERT or GPT) could detect.

3. **Keyword Rigidity:** The "Keyword Safety Layer", while useful for reducing false positives, is static. A sophisticated attacker could potentially bypass this layer by intentionally stuffing spam emails with safe academic terms (e.g., "Project" or "Meeting").

4. **Protocol Dependency:** The Real-Time Scanner is dependent on the stability of the IMAP protocol and the host's internet connection. High latency or connection timeouts can temporarily disrupt the live monitoring feed.

Practical Viability and Future Scope

The development of the live **Streamlit Dashboard** and the successful integration with the **IMAP** protocol proved that this system can operate in a real-world environment, scanning live traffic rather than just static datasets.

Future Enhancements: While the current system is highly effective, future iterations could include:

1. **Deep Learning Models:** Implementing LSTMs or BERT to better understand semantic nuances in phishing attacks.

2. **Image Analysis:** Integrating OCR (Optical Character Recognition) to detect spam hidden inside image attachments.

3. **Deployment:** Hosting the application on a dedicated cloud server (e.g., AWS or Azure) to act as a permanent proxy for an organization's email traffic.

In conclusion, this project highlights the practical viability of AI-driven "Middleware" in modern network security architectures, providing a scalable and intelligent defense against the evolving landscape of digital threats.

Bibliography

[1] Kurose, J. F., & Ross, K. W. (2021). *Computer Networking: A Top-Down Approach* (8th ed.). Pearson.

[2] Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6), e01802.

[3] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

[4] Streamlit Inc. (2024). *Streamlit Documentation*. Retrieved from https://docs.streamlit.io

Source Code

SETUP and AI MODEL TRAINING

```python
import pandas as pd
import requests
import io
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# 1. Load & Process Data
url = "https://raw.githubusercontent.com/HasanBurney/Spam-Detection/main/spam_ham_dataset.csv"
df = pd.read_csv(io.StringIO(requests.get(url).content.decode('utf-8')))
df = df[['text', 'label_num']].rename(columns={'text': 'message', 'label_num': 'label'})
df['label'] = df['label'].map({0: 'ham', 1: 'spam'})

# 2. Vectorize & Split
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['message'])
X_train, X_test, y_train, y_test = train_test_split(X, df['label'], test_size=0.2, random_state=42)

# 3. Train & Validate
model = MultinomialNB().fit(X_train, y_train)
print(f"Model Accuracy: {accuracy_score(y_test, model.predict(X_test)) * 100:.2f}%")
```

PERFORMANCE METRICS

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix

# 4. Evaluate & Visualize
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, target_names=['Ham', 'Spam']))

plt.figure(figsize=(6, 5))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix'); plt.show()
```

CROSS-VALIDATION

```python
# 5. Cross-Validation on Secondary Dataset (Kaggle)
df3 = pd.read_csv("email_spam_dataset.csv").dropna(subset=['email_text'])
df3['label'] = df3['label'].str.lower().str.strip() # Standardize labels

# Vectorize & Predict
X_new = vectorizer.transform(df3['email_text'])
y_pred_new = model.predict(X_new)

# Output Results
print(f"Validation Accuracy: {accuracy_score(df3['label'], y_pred_new) * 100:.2f}%")
print(classification_report(df3['label'], y_pred_new))

# Visualize
plt.figure(figsize=(5, 4))
sns.heatmap(confusion_matrix(df3['label'], y_pred_new), annot=True, fmt='d', cmap='Greens')
plt.title('Phase 3 Performance'); plt.show()
```

REAL-TIME SCANNER

```python
import imaplib
import email
import time
from email.header import decode_header

# Configuration
CHECK_INTERVAL = 2
SPAM_THRESHOLD = 0.85
WHITELIST = ["gmail.com", "linkedin.com", "google.com"]
KEYWORDS = ["interview", "project", "meeting"]

def advanced_classify(body, subject, sender):
    # 1. Whitelist
    if any(d in sender.lower() for d in WHITELIST): return "HAM", 0.0, "Trusted"
    # 2. Keywords
    if any(k in subject.lower() or k in body.lower() for k in KEYWORDS): return "HAM", 0.0, "Safe Keyword"
    # 3. AI Model
    score = model.predict_proba(vectorizer.transform([body]))[0][1]
    return ("SPAM", score, "AI High Confidence") if score > SPAM_THRESHOLD else ("HAM", score, "Low Risk")

def start_engine(user, password):
    mail = imaplib.IMAP4_SSL("imap.gmail.com")
    mail.login(user, password)
    mail.select('"[Gmail]/All Mail"')

    last_id = int(mail.search(None, "ALL")[1][0].split()[-1])
    print(">>> Monitoring Live Traffic...")

    while True:
        try:
            # Check for new emails
            current_id = int(mail.search(None, "ALL")[1][0].split()[-1])
            if current_id > last_id:
                # Fetch & Parse
                msg = email.message_from_bytes(mail.fetch(str(current_id).encode(), "(RFC822)")[1][0][1])
                body = msg.get_payload(decode=True).decode(errors="ignore") if not msg.is_multipart() else ""

                # Classify
                label, score, reason = advanced_classify(body, msg["Subject"], msg["From"])
                print(f"[{current_id}] {label}: {msg['Subject'][:30]}... (Risk: {score:.2f}) | {reason}")

                last_id = current_id
            time.sleep(CHECK_INTERVAL)
        except KeyboardInterrupt: break
```

STREAMLIT

```python
import streamlit as st
import pandas as pd
import requests
import io
import imaplib
import email
import html
from pyngrok import ngrok
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

# 1. Backend: Train Model on Startup
@st.cache_resource
def load_model():
    df = pd.read_csv(io.StringIO(requests.get("https://raw.githubusercontent.com/HasanBurney/Spam-Detection/main/
      spam_ham_dataset.csv").content.decode("utf-8")))
    vectorizer = CountVectorizer(stop_words="english")
    X = vectorizer.fit_transform(df["text"])
    model = MultinomialNB().fit(X, df["label_num"])
    return model, vectorizer

model, vectorizer = load_model()

# 2. Hybrid Classification Logic
def advanced_classify(body, subject, sender):
    if any(d in sender.lower() for d in ["gmail.com", "college.edu"]): return "HAM", 0.0, "Trusted"
    if any(k in subject.lower() for k in ["project", "meeting"]): return "HAM", 0.0, "Safe Keyword"

    score = model.predict_proba(vectorizer.transform([body]))[0][1]
    return ("SPAM", score, "AI High Confidence") if score > 0.85 else ("HAM", score, "Low Risk")

# 3. Frontend UI
st.set_page_config(page_title="AI Firewall", layout="wide")
tab1, tab2 = st.tabs(["Dashboard", "Live Scanner"])

with tab1:
    st.title("Network Firewall Dashboard")
    st.text_area("Test Email Content", key="input")
    if st.button("Analyze"):
        label, score, reason = advanced_classify(st.session_state.input, "", "")
        st.error(f"BLOCKED: {reason}") if label == "SPAM" else st.success(f"ALLOWED: {reason}")

with tab2:
    if st.button("Scan Gmail"):
        mail = imaplib.IMAP4_SSL("imap.gmail.com")
        mail.login(st.text_input("User"), st.text_input("Pass", type="password"))
        mail.select('"[Gmail]/All Mail"')

        # Fetch last 5 emails
        for e_id in mail.search(None, "ALL")[1][0].split()[-5:]:
            msg = email.message_from_bytes(mail.fetch(e_id, "(RFC822)")[1][0][1])
            body = msg.get_payload(decode=True).decode() if not msg.is_multipart() else ""
            label, score, reason = advanced_classify(body, msg["Subject"], msg["From"])

            st.markdown(f"**{label}**: {html.escape(msg['Subject'])} (Risk: {score:.2f})")

# 4. Launch Tunnel
# ngrok.set_auth_token("YOUR_TOKEN")
# print(ngrok.connect(8501).public_url)
```