Date: 29 May 2025

Prepared by: Swapnil Srivastava

Project Objective

Develop a basic software application capable of detecting phishing attempts in emails by analyzing their content, sender domains, URLs, and attachments.

Technology Stack

- Frontend: React (with Vite and Tailwind CSS for styling)
- Backend: Node.js with Express framework
- Email Parsing: Utilized mailparser library to extract email metadata and body content
- File Handling: multer middleware to handle .eml email file uploads
- Communication: Axios library for frontend-backend API interaction

Key Features Implemented

- Email Upload & Parsing:

  Users can upload raw email files in .eml format via a React-based interface. The backend receives and parses these files to extract sender information, URLs, text content, and attachments.

- Phishing Detection Logic:

  The backend scans emails for common phishing indicators including:

  - Presence of suspicious keywords such as "verify," "urgent," "click here"
  - Sender domain analysis for irregularities like hyphens or excessive subdomains which are common in spoofed addresses
  - Identification of dangerous attachment types (.exe, .zip, etc.) that often carry malware

- Result Feedback:

  After scanning, the system returns a clear verdict displayed on the frontend:

  - Safe Email if no suspicious signs are detected

  - Suspicious Email warning if phishing indicators are found

Current Limitations

- The system requires manual upload of .eml files; it does not automatically access or scan live email inboxes.

- No persistent storage or historical scan tracking has been implemented yet.

Gmail API Integration (Planned)

- To enhance automation, integration with the Gmail API will allow direct fetching and scanning of emails from users' Gmail inboxes.

- This will require implementing OAuth2 authentication for secure access.

- The current app architecture is designed to accommodate this future extension.

Next Steps & Recommendations

- Implement Gmail API integration to automate email retrieval and scanning.

- Add a database (e.g., MongoDB) to store scan results and enable reporting/history.

- Improve frontend UX with features like copy-paste email text scanning and detailed reports.

- Explore machine learning models for more advanced phishing detection beyond keyword matching.