Project Roadmap: Hybrid AI-Driven Phishing Email Detection System

---

1. Executive Summary

This project aims to develop a locally hosted, privacy-centric application capable of detecting phishing attempts with high accuracy. The system utilizes a **Hybrid Architecture** that synergizes deterministic rule-based heuristics (for technical header analysis) with probabilistic Deep Learning models (for semantic content analysis). The final deliverable will be a functional Streamlit application that processes raw .eml files and provides an explainable risk assessment.

---

2. System Architecture

The solution is divided into two parallel analysis engines that feed into a unified decision logic:

- **Engine A (Cybersecurity Layer):** A Python-based heuristic engine analyzing email metadata (SPF, DKIM, DMARC) and structural indicators (URL domains, HTML density).
- **Engine B (Deep Learning Layer):** A fine-tuned DistilBERT transformer model analyzing the semantic context of the email body (urgency, manipulation tactics, tone).
- **Integration Layer:** A weighted scoring algorithm that synthesizes outputs from both engines to minimize false positives.

---

3. Implementation Phases

Phase 1: Data Acquisition

Objective: Secure a robust dataset to ensure statistical significance during model training.

- **1.1 Source Identification:** Utilize the "Phishing Email Dataset" from Kaggle or similar open-source repositories containing labeled Safe vs. Phishing examples.
- **1.2 Data Verification:** Validate that the dataset includes essential fields (Email Body Content, Label) and sufficient volume (Target: 5,000+ samples).
- **1.3 Storage Setup:** Establish a structured Google Drive repository to host the raw dataset for cloud-based access.

Phase 2: Data Preprocessing & Normalization Pipeline

Objective: Transform raw, noisy text into clean, structured data to prevent "Garbage-In, Garbage-Out" failures.

- **2.1 Data Cleaning (Sanitization):**
    - Removal of HTML tags, scripts, and non-textual elements to isolate semantic content.
    - Deduplication of repeated entries to prevent model overfitting.
    - Handling of null/empty values.
- **2.2 Class Balancing:** Apply undersampling techniques to the majority class to achieve a 50/50 distribution between "Phishing" and "Safe" emails, eliminating bias.
- **2.3 Data Splitting:** Partition the dataset into **Training (80%)** and **Testing (20%)** subsets to ensure scientifically valid performance evaluation.
- **2.4 Tokenization:** Convert text data into numerical vectors using the specific DistilBERT tokenizer to prepare for ingestion.

Phase 3: Model Development (Cloud-Based Training)

Objective: Leverage cloud resources to fine-tune a Large Language Model (LLM) without impacting local hardware resources.

- **3.1 Environment Setup:** Configure Google Colab with T4 GPU acceleration.
- **3.2 Model Selection:** Initialize a pre-trained **DistilBERT** model (optimized for speed and low memory footprint).
- **3.3 Fine-Tuning:** Train the model on the preprocessed training set for 2–3 epochs to specialize it in phishing detection.
- **3.4 Evaluation:** Generate performance metrics (Accuracy, Precision, Recall, F1-Score) using the Testing set.
- **3.5 Export:** Serialize (save) the trained model and tokenizer to a portable format (.pt or .h5) for local download.

Phase 4: Heuristic Engine Development (Local Logic)

Objective: Refactor existing scripts into a modular, reusable Python library.

- **4.1 Library Creation:** Develop utils.py to house all non-AI logic.
- **4.2 Parser Implementation:** Implement extract_email_content(file) to parse raw .eml files and separate Headers from Body.
- **4.3 Rule Implementation:** Develop run_heuristic_checks() to detect:
    - Authentication Failures (SPF/DKIM/DMARC status).

- Domain Mismatches (From Header vs. URL targets).
- High-Risk Keywords (Urgency/Financial requests).

Phase 5: Interface Development (Frontend)

Objective: Create a user-friendly dashboard for demonstration purposes.

- **5.1 Framework Setup:** Initialize a **Streamlit** application (app.py).
- **5.2 UI Design:** Implement a file uploader widget for .eml files and a secure text display area.
- **5.3 Privacy Controls:** Ensure processed email data remains local and is not stored permanently.

Phase 6: Hybrid Integration & Logic

Objective: Connect the frontend, heuristic engine, and AI model into a cohesive system.

- **6.1 Model Loading:** Implement caching logic to load the DistilBERT model efficiently on application startup.
- **6.2 Live Preprocessing:** Replicate the cleaning steps from Phase 2.1 within the live app to ensure the input data matches the model's expected format.
- **6.3 Scoring Algorithm:** Implement the weighted decision logic:
  - Final_Score = (Heuristic_Score * 0.4) + (AI_Confidence * 0.6)
- **6.4 Result Visualization:** Display a final "Risk Meter" (0-100%) alongside a breakdown of triggered rules.

Phase 7: Validation, Documentation & Deployment

Objective: Finalize the project for academic submission and professional showcasing.

- **7.1 Confusion Matrix:** Produce a visualization showing True Positives vs. False Positives.
- **7.2 Documentation:** Complete README.md with installation instructions, architectural diagrams, and credit to the dataset source.
- **7.3 Demo Asset:** Record a short walkthrough video demonstrating the system analyzing a real phishing email.

---

4. Tools & Technologies

- **Language:** Python 3.9+

- **Frontend:** Streamlit
- **AI/ML:** PyTorch, Transformers (Hugging Face), Scikit-Learn
- **Compute (Training):** Google Colab (T4 GPU)
- **Compute (Inference):** Local CPU (Laptop)