**Scam Slayer Roadmap (SANS AI Cybersecurity Hackathon)**

**📄 Project Overview**

**Goal:** Develop an AI-powered phishing email detection system that is lightweight, high-performing, and explainable. **Hardware Considerations:** Optimized for an Intel Core i5-1235U (8GB RAM) with no GPU acceleration. **Timeline:** 1 month (structured into 5 phases)

**🟢 Phase 1: Data Understanding & Preprocessing (Day 1-3)**

**📊 Data Exploration & Cleaning**

* Load datasets (merged\_phishing\_dataset.csv, phishing\_email.csv, CEAS\_08.csv).
* Perform **Exploratory Data Analysis (EDA)** – visualize word clouds, text length distribution, missing values.
* Identify and handle **imbalanced data** (phishing vs. non-phishing email ratios).

**💡 Text Processing**

* Remove **HTML tags**, special characters, extra spaces.
* Convert text to **lowercase** for uniformity.
* Remove **stopwords, punctuation** using **NLTK**.
* Tokenize text and apply **lemmatization**.

**⚙️ Feature Engineering**

* Extract **word count, character count, punctuation count, digits count, uppercase words count**.
* Identify **common words** in phishing vs. legitimate emails.
* Extract **email metadata** (sender domain, presence of links, number of attachments).

**🟡 Phase 2: Model Selection & Baseline (Day 4-7)**

**🎯 Choose a Lightweight ML Model**

* Train a **baseline** model using:
  + **Logistic Regression**
  + **Random Forest**
  + **LightGBM**
* Use **TF-IDF** and **CountVectorizer** for text representation.

**🔢 Train & Evaluate Model**

* Split data into **train (80%) / test (20%)**.
* Train model and measure **accuracy, precision, recall, F1-score**.
* **Baseline goal:** >80% precision & recall for phishing detection.

**🟣 Phase 3: Improve Performance (Day 8-14)**

**👩‍💻 Experiment with Different Models**

* Compare **Random Forest, XGBoost, LightGBM**.
* Try **DistilBERT embeddings** (lightweight transformer for text representation).

**⚙️ Feature Selection & Optimization**

* Use **SHAP** to identify the most important features.
* Perform **hyperparameter tuning** using GridSearchCV / Optuna.

**🔄 Optimize Model for Speed**

* Apply **model compression** (quantization, pruning).
* Reduce **memory footprint** for better performance on local machine.

**🔴 Phase 4: Explainability & Final Refinements (Day 15-20)**

**🔍 Add Explainability Features**

* Use **LIME / SHAP** to explain why an email is classified as phishing.
* Generate **visuals** (heatmaps, word importance, email metadata analysis).

**✉️ Improve Metadata Handling**

* Include **email header analysis** (blacklisted domains, SPF/DKIM validation).
* Extract **URL links** for phishing pattern detection.

**🌐 Deploy a Lightweight API (Optional)**

* Convert model into a **Flask/FastAPI** service for email classification.

**🏆 Phase 5: Final Testing & Presentation (Day 21-30)**

**🎉 Run on New Unseen Data**

* Test on real-world phishing emails (e.g., phishing email datasets online).
* Evaluate **false positives & false negatives**.

**🌟 Prepare Presentation & Report**

* Create a **clear, concise report** highlighting:
  + **Problem Statement & Solution**
  + **How Scam Slayer Works**
  + **Key Features & Model Performance**
  + **Why It’s Better Than Competitors**
* **Prepare a demo** showcasing real phishing email detection.

**🔗 Final Outcome**

💪 **Fast & Lightweight** – Works well on your machine.  
📊 **High Accuracy** – Optimized model with strong recall for phishing detection.  
🕵️ **Explainable** – Uses SHAP/LIME for transparency.  
💎 **Innovative** – Includes email metadata + text analysis + heuristics.