Software Requirements Specification (SRS)

Project Name: SenseFlow: Phishing & Social Engineering Detector **Version:** 1.0 **Status:** Draft

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for **SenseFlow**, a security application designed to detect "clean" phishing attacks (Business Email Compromise) that lack malicious payloads (links/attachments) but contain malicious intent.

1.2 Scope

SenseFlow is a web-based system that utilizes **Natural Language Processing (NLP)** and **Transformer models (BERT)** to analyze the semantic intent of text inputs. It integrates **Explainable AI (LIME)** to provide users with visual feedback on *why* a specific message was flagged. The system is designed to replace or augment traditional signature-based antivirus filters.

1.3 Definitions & Acronyms

- **BEC:** Business Email Compromise (Fraud that uses legitimate email accounts).
- **NLP:** Natural Language Processing.
- **BERT:** Bidirectional Encoder Representations from Transformers (The AI Model).
- **LIME:** Local Interpretable Model-agnostic Explanations.
- **PII:** Personally Identifiable Information.

2. Overall Description

2.1 Product Perspective

SenseFlow operates as a standalone web application (Prototype phase) with plans for browser extension integration. It interacts with:

- **User Interface:** Built with Streamlit.
- **Inference Engine:** PyTorch backend hosting the Fine-Tuned BERT model.
- **Data Layer:** Pandas/NumPy for preprocessing text inputs.

2.2 User Characteristics

- **End Users:** General employees or students who need to verify suspicious emails. Requires no technical knowledge; relies on visual cues (Red/Green highlights).
- **Security Analysts:** Technical users who review the "Risk Score" and LIME explanation features to identify new threat patterns.

3. System Features (Functional Requirements)

3.1 Semantic Intent Detection

- **Description:** The system shall accept text input (email body) and classify it as "Safe" or "Phishing/Social Engineering."
- **Functional Requirement 1:** The system must utilize a **BERT-based Transformer model** to analyze context rather than just keyword matching.
- **Functional Requirement 2:** The system must specifically detect psycholinguistic triggers: **Urgency**, **Authority Bias**, and **Scarcity**.
- **Functional Requirement 3:** The system must output a probabilistic **Risk Score** (0-100%).

3.2 Explainability Module (XAI)

- **Description:** The system shall provide transparency regarding its decision-making process.
- **Functional Requirement 1:** The system must integrate **LIME** to generate local explanations for the prediction.
- **Functional Requirement 2:** The system shall generate a **Visual Heatmap** highlighting words that contributed most to the "Phishing" classification in RED.

3.3 Data Preprocessing Pipeline

- **Description:** The system must sanitize inputs before analysis.
- **Functional Requirement 1:** The pipeline must remove HTML tags, special characters, and normalize text casing.
- **Functional Requirement 2:** The pipeline must mask PII (Personally Identifiable Information) to ensure user privacy during analysis.

4. Non-Functional Requirements

4.1 Performance Requirements

- **Latency:** The inference time (scan speed) must be under **1.5 seconds** per email to ensure a smooth user experience.
- **Accuracy:** The model must achieve a target accuracy of **>92%** on unseen "Spear Phishing" datasets.

4.2 Software Quality Attributes

- **Reliability:** The system must handle edge cases (empty strings, extremely long text) without crashing.
- **Portability:** The system must be deployable on local machines via Docker or Python virtual environments (VS Code/Streamlit support).

5. Interface Requirements

5.1 User Interfaces

- **Dashboard:** A Streamlit-based web page containing:
  - A text input box for pasting email content.
  - A "Scan" button.
  - A results panel showing the Risk Score (e.g., "85% Dangerous").
  - A visualization panel showing the LIME heatmap.

5.2 Software Interfaces

- **Libraries:** The system shall interface with **Hugging Face Transformers** for model weights and **PyTorch** for tensor computation.