Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Software Requirement Specifications

AI-Powered-Email-Spam-and-Phishing-Detection-System

Submitted by

Rubab Malik

F22BINFT1M01171

Submitted to

Ma’am musarat karim

Meeting Details

Sr No	Details	Date	Supervisor Signature

Summary

This project focuses on building a real-time Email Spam Detection App using Logistic Regression and TF-IDF with a Streamlit interface. The system takes email text as input, preprocesses it, converts it into numerical features through TF-IDF, and predicts whether it is Spam or Not Spam using a trained machine learning model. The application is lightweight, user-friendly, and capable of providing instant results. It supports local as well as cloud deployment, making it practical for real-world use in filtering unwanted emails..

Table of Contents

1. Introduction

Email communication is one of the most widely used mediums for personal, academic, and professional interaction, but with this convenience comes a significant increase in unwanted and fraudulent emails, commonly known as spam. To address this growing problem, the Email Spam Detection App is designed to automatically classify incoming email text as ***Spam*** *or* ***Not Spam (Ham)*** *using Machine Learning techniques. This system utilizes* ***TF-IDF Vectorization*** *to convert text into numerical features and* ***Logistic Regression*** *for accurate classification. The application provides a simple and interactive interface developed in* ***Streamlit****, allowing users to input email text and receive instant predictions. With real-time processing, high accuracy, and ease of use, this application serves as an efficient solution for reducing spam email threats and improving digital communication safety.*

1.1. Purpose

This SRS document defines the requirements for the Email Spam Detection Application.
The purpose of this software is to classify email text as ***Spam*** *or* ***Not Spam (Ham)*** *using Machine Learning (Logistic Regression) along with TF-IDF Vectorization and provide a user-friendly interface through Streamlit.*

1.2. Scope

The system will:

- *Take email text input from the user.*
- *Preprocess and convert text into TF-IDF features.*
- *Predict whether the email is* ***Spam / Ham*** *using Logistic Regression.*
- *Display results instantly on the Streamlit interface.*
- *Support both local and cloud deployment.*

*The system does **not**:*

- *Analyze attachments or multimedia content.*
- *Detect phishing intent beyond spam text classification.*
- *Provide email sending/receiving features.*

1.3. User Characteristics

The primary users of the Email Spam Detection App are general email users who want to filter unwanted or suspicious emails efficiently. They may have varying levels of technical expertise, ranging from beginners to moderately experienced users, but no advanced programming or machine learning knowledge is required. Users are expected to have basic familiarity with typing or pasting email content into a web interface and interpreting simple messages such as "This email is SPAM" or "This email is NOT spam." The system is designed to be intuitive and user-friendly, ensuring that users of all ages and backgrounds can interact with the application without confusion. Additionally, users are assumed to have access to a device capable of running a web browser or Streamlit application.

1.4. Proposed Technologies

The proposed project will leverage the following technologies:

1. *Programming Language: Python 3.x*

2. *Machine Learning Algorithm: Logistic Regression (scikit-learn)*

3. *Text Feature Extraction: TF-IDF Vectorizer (scikit-learn)*

4. *Web Framework: Streamlit (for interactive GUI)*

5. *Model & Vectorizer Storage: joblib (for saving/loading pre-trained models)*

6. *Data Handling Libraries: pandas, numpy*

7. *Deployment Platforms: Local system or Streamlit Cloud*

2. Requirements

The Email Spam Detection App requires users to input email text, which is then preprocessed, converted into numerical features using TF-IDF, and classified by a pre-trained Logistic Regression model. The system must provide instant predictions, displaying whether the email is Spam or Not Spam. It should handle invalid or empty inputs gracefully and be accessible through a simple Streamlit web interface. The application depends on Python 3.x and relevant libraries such as scikit-learn, pandas, numpy, joblib, and Streamlit, and can be deployed either locally or on Streamlit Cloud for easy access.

2.1. Function Requirements

2.1.1. Email Text Input

- **Id:** FR001
- **Title:** Email Text Input
- **Description:** The system shall allow users to enter or paste email text into the application for spam analysis.

2.1.2. Spam Classification

- **Id:** FR002
- **Title:** Spam Classification
- **Description:** The system shall classify the provided email text as Spam or Not Spam using the trained Logistic Regression model.

2.1.3. Display Classification Results

- **Id:** FR003
- **Title:** Display Classification Results
- **Description:** The system shall display the final prediction result, clearly indicating whether the email is “SPAM” or “NOT SPAM,”.

2.1.4. Text Preprocessing

- **Id:** FR004
- **Title:** Text Preprocessing
- **Description:** The system shall preprocess the input email text by cleaning, tokenizing, and transforming it into numerical features using TF-IDF vectorization to ensure accurate classification.

2.1.5. User Interaction Interface

- **Id:** FR005
- **Title:** User Interaction Interface
- **Description:** The system shall provide a simple, intuitive, and user-friendly Streamlit interface allowing users to easily input email content, view predictions, and interact with the application in real time.

2.1.6. Model Training

- **Id:** FR006
- **Title:** Model Training
- **Description:** The system shall support training and retraining of the Logistic Regression model using labeled datasets of spam and ham emails to maintain high accuracy and robust performance.

2.2. Non Functional Requirements

2.2.1. Performance

The system shall provide real-time spam detection, ensuring predictions are generated within a few seconds of text submission.

2.2.2. Useability

The application shall offer a clean and intuitive Streamlit interface, enabling users to easily input email text and understand the classification results without technical expertise.

2.2.3. Reuseability

The system shall consistently produce accurate and stable results, handling various text formats and preventing system crashes or incorrect outputs.

2.2.4. Maintainability

The system shall be designed with modular and well-documented code, allowing easy updates to the model, preprocessing steps, or UI components in the future.

2.2.5. Security

The system shall ensure that user-provided email content is processed securely and not stored, logged, or shared, preserving user privacy.

2. Use Cases and Flow of Processes

The Email Spam Detection App allows users to input or paste email content into a simple Streamlit interface. Once the email is entered, the system first preprocesses the text by converting it to lowercase, removing special characters, numbers, and extra spaces, ensuring it is clean and ready for analysis. The cleaned text is then transformed into numerical feature vectors using the TF-IDF vectorizer, which captures the importance of each word relative to the dataset. These vectors are passed to a pre-trained Logistic Regression model, which predicts whether the email is Spam or Not Spam. Finally, the prediction result is displayed instantly on the app interface, showing either "This email is SPAM" or "This email is NOT spam." Users can then enter another email or exit the app, making the process simple, real-time, and interactive.

2.1. Input Email Text

ID	UC001
Name	Input Email Text
Requirement(s)	FR001,FR004,FR005
Actor(s)	User
Precondition	The user has access to the web application and has email text ready to input or paste.
Postcondition	The email text is successfully received and ready for preprocessing.
Basic Flow	1. The user navigates to the email input section of the Streamlit app. 2. The user types or pastes the email text into the provided text box. 3. The system validates the input (e.g., not empty). 4. The system accepts the text and prepares it for preprocessing.

2.2. Classify Email Scam

ID	UC002
Name	Classify Email Scam
Requirement(s)	FR001,FR002,FR003, FR004
Actor(s)	User
Precondition	Valid email text has been successfully provided by the user.
Postcondition	Spam classification results are generated by the machine learning model.
Basic Flow	1. The system preprocesses the input text (cleaning, tokenizing). 2. The system converts text into numerical features using TF-IDF. 3. The system loads the trained Logistic Regression model. 4. The system predicts whether the email is Spam or Not Spam (Ham). 5. The system computes the confidence score for the prediction (if implemented).

2.3. Display Classification Results

ID	UC003
Name	Display Classification Results
Requirement(s)	FR001,FR002,FR003, FR005
Actor(s)	User, System
Precondition	The email text has been successfully classified.
Postcondition	The user receives a clear spam or ham prediction with supporting information.
Basic Flow	1. The system displays the classification output (e.g., “SPAM” or “NOT SPAM”). 2. The system shows the probability/confidence score of the prediction. 3. The user reviews the result. 4. The user can choose to analyze another email.

2.4. Repeat Analysis

ID	UC004
Name	Repeat Analysis
Requirement(s)	FR001,FR002,FR003, FR005
Actor(s)	User
Precondition	*The user has already completed one spam detection operation.*
Postcondition	*The user can analyze multiple emails in the same session without restarting the app.*
Basic Flow	1. The user enters or pastes a new email text. 2. The system preprocesses the text and classifies it again. 3. The system displays updated results.

3. References

1. Venky73 ***"Spam Mails Dataset,"*** *Kaggle*, 2018. *https://www.kaggle.com/datasets/venky73/spam-mails-dataset*

2. H. Ahmed, ***"FYP Plant Disease Classification Using Deep Learning and Streamlit-Based Web Application,"*** *GitHub Repository*. *https://github.com/rubabmalik884u/AI-Powered-Email-Spam-and-Phishing-Detection-System*

3. TensorFlow, ***"Keras Guide,"*** *TensorFlow Documentation*. https://www.tensorflow.org/guide/keras

4. Streamlit, ***"Streamlit Documentation,"*** *Streamlit Web Tool*. https://docs.streamlit.io/

5. Google, ***"Colaboratory,"*** *Google Colab*. https://colab.google/