# DESIGN AND IMPLEMENTATION OF A SPAM EMAIL DETECTION SYSTEM USING ARTIFICIAL INTELLIGENCE

**BY**

**SAMUEL TREASURE INYENE**

**PSC2003837**

**DEPARTMENT OF COMPUTER SCIENCE,**
**FACULTY OF PHYSICAL SCIENCES,**
**UNIVERSITY OF BENIN,**
**BENIN CITY,**
**EDO STATE, NIGERIA.**

**NOVEMBER 2025**

**DESIGN AND IMPLEMENTATION OF A SPAM EMAIL DETECTION SYSTEM USING ARTIFICIAL INTELLIGENCE**

**BY**

**SAMUEL TREASURE INYENE**
**PSC2003837**

**A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE**

**NOVEMBER 2025**

## CERTIFICATION

This is to certify that this project work was carried out by **SAMUEL TREASURE INYENE** with matriculation number, **PSC2003837** in partial fulfilment for the award of the degree of Bachelor of Science in computer Science, department of Computer Science, faculty of Physical Sciences, University of Benin, Benin city under my supervision.

| | |
|---|---|
| **DR. (MRS.) A.R. USIOBAIFO**<br>**Project Supervisor** | **DATE** |

## APPROVAL

This project work is hereby approved in partial fulfillment of the requirements for the award of Bachelor of Science (B.Sc) Degree in Computer Science from the University of Benin.

| | |
|---|---|
| **DR. (MRS.) A.R USIOBAIFO** | **DATE** |
| **Head of Department** | |

## DEDICATION

This work is dedicated to **GOD ALMIGHTY** for his overflowing grace, strength and power and for being present throughout my journey in **UNIBEN,** and to my wonderful parents for their support all through my journey.

## ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for His guidance, strength, and wisdom throughout this project.

I sincerely appreciate my project supervisor and Head of Department, Dr. Mrs. A.R. Usiobaifo, for her invaluable guidance, support, and encouragement.

I am deeply grateful to my parents, Mr. Ime and Mrs. Grace Samuel, for their love, prayers, and unwavering support.

My heartfelt thanks go to my mentor, Major Ali Abdul, for his immense support, advice, and motivation throughout this journey.

I also extend my gratitude to my friends Wemimo, Stella, Francisca, Michael, Raphael, and others, for their assistance and encouragement in one way or the other.

Finally, I thank everyone who directly or indirectly contributed to the successful completion of this project.

## ABSTRACT

The increasing use Jervers for continuous learning.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background Study

In today's digital environment, email remains a primary means of communication for individuals and organisations. Unfortunately, this widespread use of email has given rise to unwanted and malicious messages commonly known as spam.

Spam emails impose multiple costs: wasted time, clogged inboxes, increased storage and infrastructure burden, and crucially, exposure to phishing, malware distribution, and other security threats. According to a survey of spam-text detection, malicious and unsolicited messages are becoming ever more dynamic and sophisticated, employing obfuscated content, phishing links, and new attack patterns.

Historically, early spam detection methods relied on static, rule-based heuristics or simple keyword matching, blacklists/whitelists, and message-header heuristics. While valuable, these methods struggle to keep up with evolving spam tactics spammers adapt their content, alter links, craft social-engineering style messages, and use polymorphic text. For example, techniques like bulk email detection via fuzzy checksums (such as the Nilsimsa hash) were used to detect repeated identical message bodies, but even those have limitations given modern variants.

With the rise of artificial intelligence (AI) and machine learning (ML), more advanced spam detection systems have emerged. These use features derived from email content (text tokens, TF-IDF vectors, embeddings), metadata (sender, headers, links), and behavioural patterns (sender history, network features) to build classifiers that can discriminate between legitimate ("ham") and spam. A comprehensive survey of AI/ML methods for spam-email detection outlines how

various supervised learning models (Naïve Bayes, SVM, Random Forest) and recently deep-learning models (LSTM, GRU) have been applied in this domain.

The use of AI-driven systems shows strong promise: for example, hybrid deep-learning models combining LSTM and GRU have been shown to improve detection accuracy in recent studies.

Given that email volumes continue to grow, and adversaries continue to innovate, developing a robust AI-based spam detection system is both timely and necessary. Thus, this work investigates the design and implementation of a spam-email detection system using artificial intelligence, covering data collection/pre-processing, feature extraction, model selection, training, validation, and deployment considerations.

### 1.2 Motivation

The motivation for this study arises from several intersecting factors:

1. **Escalating Volume and Sophistication of Spam:** As email users increase globally, spammers respond by increasing volume and by using more sophisticated content—phishing links, malicious attachments, impersonation, adaptive language. Traditional filters alone are insufficient. The literature indicates a growing need for adaptive, context-aware systems.
2. **Impact on Productivity and Security:** For organisations and individuals alike, spam wastes time, uses up storage and bandwidth, and represents a security risk (phishing, malware). An accurate spam detection system thus contributes directly to improving user experience and reducing risk.
3. **Technological Opportunity:** Advances in AI (especially in natural-language processing, embeddings, recurrent neural networks) offer capabilities not present in older rule-based

systems. As seen in recent works, models leveraging LSTM/GRU layers yield improved detection of spam in complex real-world datasets.

4. **Practical Need for Scalable Systems:** With large email volumes, solutions need to scale efficiently, handle real-time or near-real-time detection, and adapt to changing spam tactics. Research surveys highlight the necessity of intelligent, scalable systems.

Given these factors, the motivation of this study is to design and implement a spam email detection system that harnesses AI approaches, is effective and efficient in practical settings, and can adapt to evolving spam tactics.

### 1.3 Statement of the Problem

The persistence of spam emails continues to threaten the integrity and security of email communication. Despite the availability of basic filters in modern email platforms, users are still bombarded with large volumes of unsolicited emails daily. Spam emails often contain harmful links, phishing attempts, or deceptive advertisements that may lead to identity theft, financial loss, or data breaches.

Traditional filtering approaches such as keyword matching and manual blocking are no longer sufficient, as spammers constantly adapt to evade detection. These limitations result in:

1. Reduced productivity for email users due to time wasted sorting and deleting spam.
2. Increased vulnerability to cyberattacks through phishing and malicious links.
3. Inefficient email management for organizations and individuals.

Therefore, there is a clear need for a robust AI-based spam detection system that can intelligently classify emails and adapt to evolving spam techniques.

## 1.4 Aim and Objectives

### Aim

The aim of this study is to design and implement an intelligent spam email detection system using artificial intelligence (AI), capable of accurately distinguishing between spam and legitimate emails and reducing the number of unwanted emails users receive.

### Objectives

To achieve this aim, the study will pursue the following objectives:

1. 1 Design a spam detection model using AI-based techniques such as Naïve Bayes or Decision Tree algorithms, leveraging their proven performance in text-classification tasks.
2. Simulate how the proposed system can classify emails into spam and non-spam categories through model training, testing, and validation using an appropriate dataset.
3. Evaluate the effectiveness of the proposed system in reducing spam emails by analysing performance metrics such as accuracy, precision, recall, and F1-score.
4. Recommend improvements for future spam detection systems based on the evaluation results, with suggestions for enhanced adaptability, data handling, and AI model selection.

## 1.5 Scope of Study

This research focuses on the design and conceptual implementation of a spam email detection system using Artificial Intelligence techniques. The scope covers:

- analysis of spam and ham email datasets,
- application of AI algorithms such as Naïve Bayes and Decision Trees for classification, theoretical testing and evaluation of how the system would perform in detecting spam.

The study does not cover large-scale deployment or real-time integration into commercial email platforms, but rather provides a design framework and model simulation.

## 1.6 Methodology

This project employed an experimental and analytical approach, analyzing existing materials and research works related to spam email detection to determine the best techniques for the proposed system.

The system was developed using the Object-Oriented Analysis and Design (OOAD) methodology, which structures the system as interacting components such as the user interface, preprocessing unit, and classifier module.

Following this methodology, it was possible to design and implement a spam detection model using Naïve Bayes and Decision Tree algorithms, capable of classifying emails into spam and non-spam categories. This approach ensured a robust, efficient, and user-friendly system.

## 1.7 Significance of the Study

This study is significant because it addresses one of the most common challenges in digital communication. The proposed system will:

1. Enhance the security of email communication by minimizing exposure to phishing and malicious spam.
2. Improve productivity by reducing the time users spend managing spam emails.
3. Contribute to the academic field of Artificial Intelligence by demonstrating its practical application in spam detection.
4. Provide a framework that organizations and email service providers can adopt or extend to strengthen their spam filtering capabilities.

# CHAPTER TWO

# LITERATURE REVIEW

This chapter provides an overview of spam emails, their types, and detection methods. It also discusses the role of artificial intelligence (AI) in building intelligent spam filters, the common machine learning algorithms applied, evaluation metrics, and challenges faced in the development of effective spam detection systems. Finally, it reviews related works in this research area.

## 2.1 Overview of Spam Email and Detection Systems

Spam emails, commonly known as junk or unsolicited messages, have become one of the most persistent challenges in electronic communication. They refer to unwanted emails sent in bulk, usually for commercial advertising, phishing, or spreading malware. According to Symantec's Internet Security Threat Report (2023), spam constitutes more than 45 % of all global email traffic. The growth of internet usage and the ease of automating bulk mailing systems have made spam detection a critical area of research in computer science and cybersecurity.

A spam email detection system is an intelligent software mechanism designed to automatically classify incoming emails into two categories – spam and non-spam (ham). The system typically analyzes the content, subject line, metadata, and embedded links of each email to determine its legitimacy (Al-Qurishi et al., 2021). These systems are essential because they help users save time, protect privacy, and prevent exposure to malicious links or phishing attempts. For example, a single spam message can carry fake investment offers or impersonation attempts to steal login credentials – a practice known as email spoofing.

Over time, spam detection techniques have evolved from simple keyword-based filters to sophisticated machine-learning-based classifiers and, more recently, AI-driven approaches**.** Early email clients like Yahoo Mail and Outlook relied on rule-based filters where messages were

flagged as spam if they contained suspicious words such as “lottery,” “urgent,” or “winner.” However, spammers quickly adapted by altering spellings (e.g., “w1nner” or “fr33 cash”), which rendered static rules ineffective. To overcome this limitation, modern systems now employ algorithms that learn patterns in data – for instance, the Naïve Bayes classifier or Decision Tree models (Sahin & Duman, 2020).

Furthermore, contemporary spam detection systems integrate natural language processing (NLP) and deep learning models such as convolutional neural networks (CNN) and recurrent neural networks (RNN) to better understand email semantics (Zhang et al., 2022). These AI-based models can analyze the relationships between words and contexts, making them capable of identifying subtle phishing or scam attempts that would bypass older filters.

A robust spam detection model typically goes through several stages:

**1 Data collection and preprocessing –** emails are gathered and cleaned to remove noise, stop words, or duplicate entries.

**2 Feature extractions –** the model identifies distinguishing features such as sender address, subject line, or frequency of suspicious keywords.

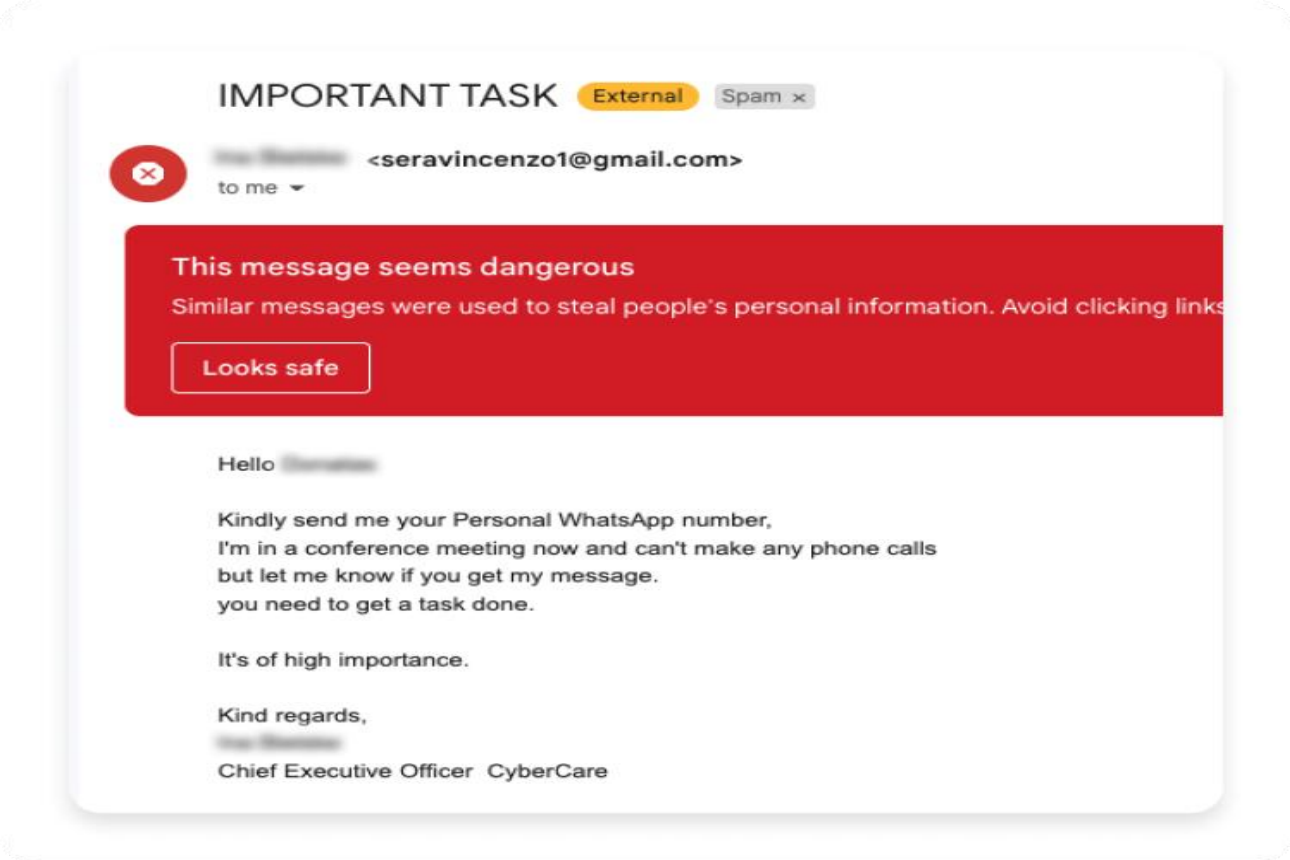
**3 Model training –** machine learning or AI algorithms are trained to recognize spam patterns from labeled datasets.

**4 Classification and evaluation –** the trained model predicts whether new incoming emails are spam or legitimate, and its accuracy is assessed using evaluation metrics like precision, recall, and F1 score (Alegbeleye et al., 2023).

An effective spam detection system is not only measured by accuracy but also by its ability to adapt to new spam strategies. Spammers frequently modify their tactics by embedding messages within images, compressing text, or using multilingual phrases to evade filters. Hence, ongoing

research focuses on improving adaptability through reinforcement learning and hybrid models that combine multiple algorithms for enhanced performance (Nizamani & Memon, 2023).

In summary, spam detection systems play an indispensable role in maintaining email security and user trust. The integration of artificial intelligence techniques such as Naïve Bayes and Decision Trees represents a significant advancement toward automated, scalable, and intelligent solutions capable of mitigating the ever-evolving threat of spam emails.




***Source:*** Violeta Lyskoit 2025

**Figure 2.1: Example of Spam Email Message**

### 2.2 Types of Spam Emails

Spam emails manifest in several forms, each serving a different malicious or deceptive purpose. Understanding these types is essential to building a system capable of identifying diverse spam

patterns accurately. Generally, spam emails can be classified into the following categories based on their intent and content.

### 2.2.1 Phishing Emails

Phishing is one of the most dangerous forms of spam because it targets users' personal or financial information. Such emails impersonate legitimate organizations like banks, social-media platforms, or government agencies—to trick recipients into revealing confidential data such as passwords, credit-card details, or account numbers. For example, a fake message from "PayPal Support" might warn that an account has been suspended and ask the recipient to click a link to "verify details." Once the user clicks, their credentials are stolen (Salahdine & Kaabouch, 2019). Phishing emails often contain urgent language, mismatched URLs, or forged logos to appear authentic.

### 2.2.2 Advertising and Marketing Spam

This is the most common category of spam. It involves unsolicited promotional messages about products or services, often from unknown or fake companies. While some may be legitimate marketing attempts, most violate privacy by targeting large lists of addresses obtained through scraping or data leaks. Typical examples include emails promising "quick-weight-loss products," "miracle health supplements," or "investment opportunities" (Alegbeleye et al., 2023). Though seemingly harmless, such spam clogs inboxes, wastes bandwidth, and can contain embedded malware links disguised as product advertisements.

### 2.2.3 Malware and Trojan Spam

These spam emails aim to infect devices with malicious software. They often arrive with attachments labeled as invoices, receipts, or delivery notes that, when opened, execute hidden

code. For instance, the “Emotet” and “Locky” malware campaigns spread primarily through infected email attachments disguised as Microsoft Word or PDF files (Alam et al., 2020). Once activated, these programs can steal data, encrypt files for ransom, or hijack the victim’s computer to send more spam.

### 2.2.4 Scam and Fraudulent Emails

Scam emails are designed to manipulate recipients emotionally or financially. A popular variant is the Nigerian Prince or advance-fee fraud, where victims are promised large sums of money in exchange for an upfront payment. Other examples include fake charity appeals during disasters and lottery notifications claiming the recipient has won a prize (Olowolagba et al., 2021). These scams exploit human trust and desperation, making them harder to combat using keyword filters alone.

### 2.2.5 Adult Content Spam

Adult spam contains pornographic materials, dating site promotions, or links to explicit content. Such messages are particularly harmful to minors and workplace environments. They often bypass filters by embedding obscene words within images or using character substitutions (e.g., “s3x,” “adul7”), thereby evading keyword-based systems (Sahin & Duman, 2020).

### 2.2.6 Stock and Investment Spam

These emails attempt to manipulate stock prices by spreading false or exaggerated information. For instance, a spam message might claim that a small company’s stock will rise significantly, prompting recipients to invest and artificially inflate prices. Once the price increases, the spammer sells their shares at a profit, leaving others at a loss (Nizamani & Memon, 2023).

### 2.2.7 Fake Job and Employment Spam

Such emails falsely advertise job offers to harvest personal data or extort fees from applicants. A common example includes messages inviting recipients to “work from home” with unrealistic pay, then demanding payment for “registration” or “training materials.” These scams have risen dramatically since 2020 due to the growth of remote-work trends (Zhang et al., 2022). In summary, spam emails come in various forms from phishing and advertising to malware-infested messages, all designed to exploit human psychology or system vulnerabilities. Classifying these types accurately helps improve model training and enhances the effectiveness of spam detection systems.

### 2.3 Techniques Used in Spam Detection

Various techniques have been developed for spam detection, ranging from manually crafted rules to advanced artificial intelligence models. The methods can broadly be categorized into rule-based, machine learning-based, deep learning-based, and hybrid systems.

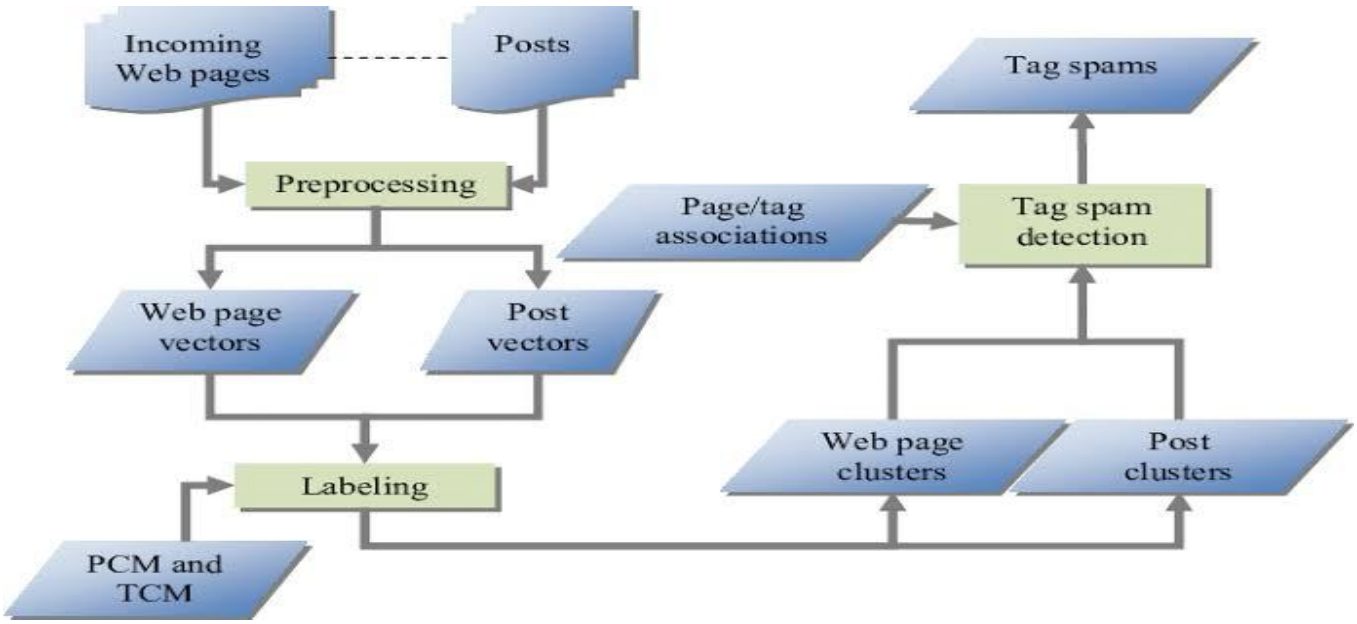



Figure 2.2 presents a general workflow of the spam detection process.

***Source:*** Uploaded by Chun-Hung Lee *2021*

**Figure 2.2: Workflow of Spam Detection Process**

### 2.3.1 Rule-Based Filtering

Rule-based filtering was one of the earliest approaches to spam detection. It uses manually defined patterns or keywords to identify spam messages. For instance, emails containing words like “lottery,” “free money,” or “urgent update” may automatically be classified as spam.

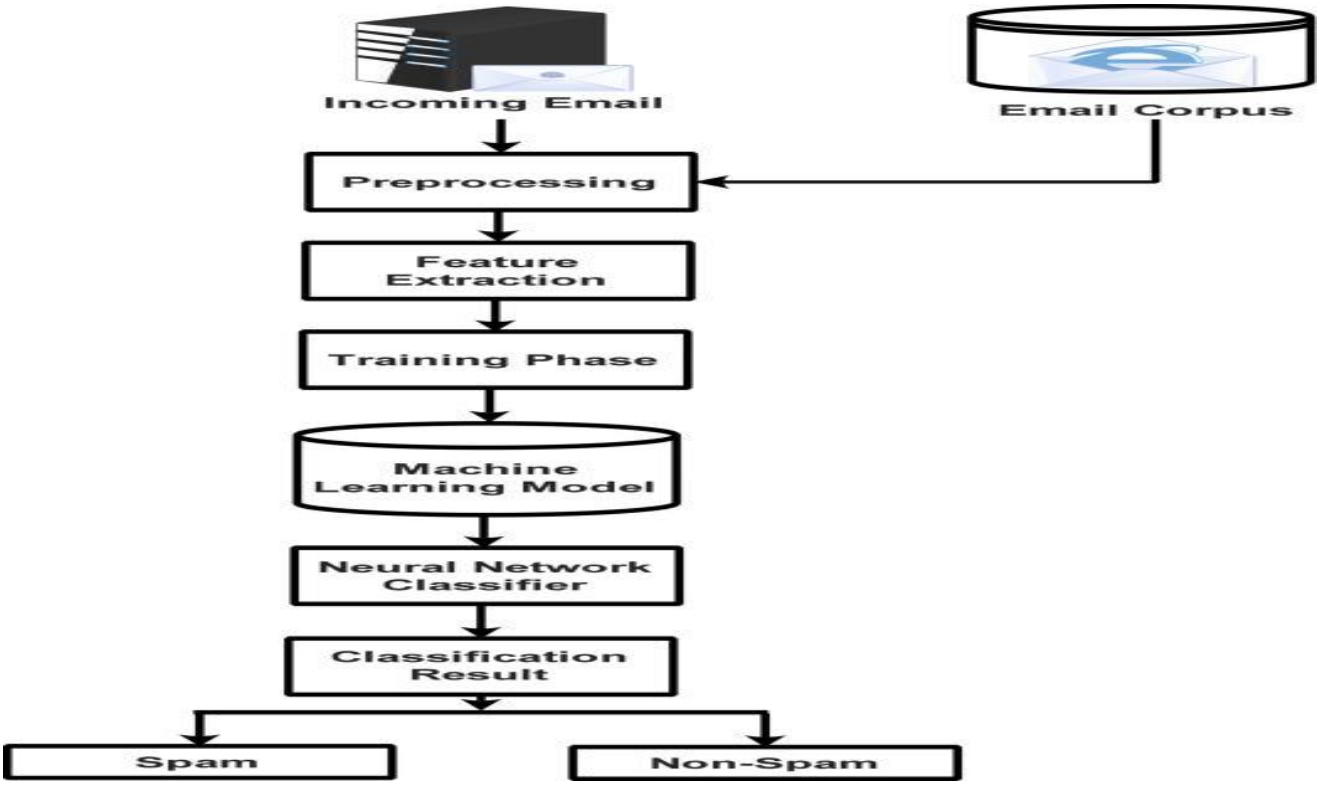
Although this method is easy to implement, it lacks flexibility and fails to adapt to the constantly changing spam patterns. As a result, it produces a high rate of false positives (legitimate emails flagged as spam) (Miller, 2020).

### 2.3.2 Machine Learning-Based Filtering

Machine learning-based filtering uses algorithms that learn from historical email data. These algorithms are trained on labeled datasets containing examples of both spam and ham emails. After training, they can automatically classify new emails based on learned patterns.

Common steps include preprocessing, tokenization, feature extraction (e.g., word frequency, sender reputation), and model training. Popular algorithms include Naïve Bayes, Support Vector Machines (SVM), and Decision Trees.

As shown in Figure 2.3, a typical machine learning spam filter consists of data preprocessing, feature extraction, model training, and classification layers.




***Source:*** Uploaded by Elsevier Ltd 2019

**Figure 2.3: Architecture of a Machine Learning Spam Filter**

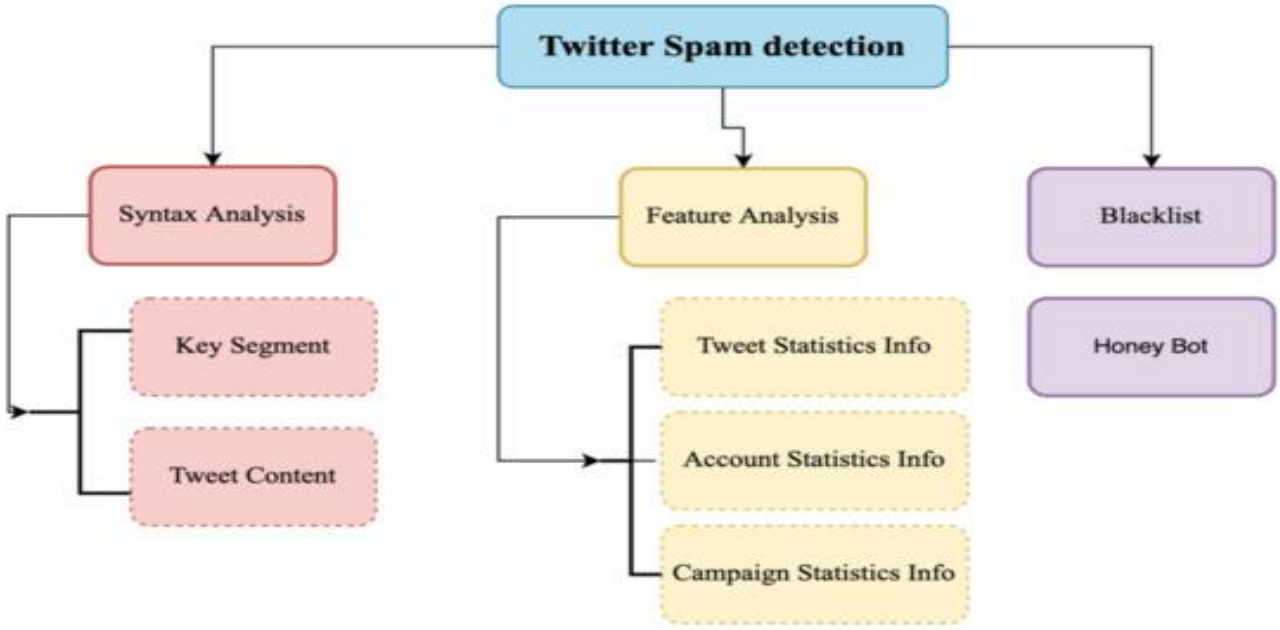
Machine learning filters are more adaptive than rule-based systems and can generalize to unseen spam types once trained on sufficient data.

### 2.3.3 Deep Learning-Based Filtering

Deep learning has significantly improved spam detection by modeling complex relationships between email words and patterns. Techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks can automatically extract and learn features from large datasets without manual intervention (Wang & Li, 2021).

These models capture contextual relationships within the email text, detecting spam even when spammers deliberately alter spelling or word order.

Figure 2.4 shows the architecture of an Artificial Neural Network (ANN) used for spam detection.




***Source:*** Amira Abdel Wahab & Mhammed Mostafa 2022

**Figure 2.4: Artificial Neural Network Model for Spam Detection**

### 2.3.4 Hybrid Systems

Hybrid systems combine multiple spam detection techniques such as rule-based filtering, machine learning, and deep learning to achieve higher accuracy. For instance, a hybrid model may use a rule-based pre-filter to remove obvious spam and then apply a machine learning model for fine-grained classification.

These systems provide better adaptability and robustness but may require more computational power and longer training times (Eze & Obinna, 2022).

## 2.4 Email Features for Spam Classification

Email features are the measurable attributes or properties extracted from an email message to assist a spam detection model in determining whether the message is spam or legitimate. The effectiveness of any spam classification system depends largely on the quality and relevance of these features (Sahin & Duman, 2020). Features can be broadly divided into header features, content-based features, and behavioral or link-based features.

### 2.4.1 Header Features

Header features are derived from the metadata of an email—the "envelope" information that includes details such as sender addresses, recipient address, subject line, date, and mail server routes. Some typical header indicators of spam include:

- **Sender address anomalies:** Domains with random characters (e.g., xytq@randommail.com ) or addresses not matching the display name often signal spam (Al-Qurishi et al., 2021).
- **Reply-to mismatch:** Spammers may use different "reply-to" and "from" addresses to divert responses.

- **Subject-line irregularities:** Excessive capitalization, exclamation marks, or trigger words such as “FREE,” “WINNER,” or “URGENT” tend to increase the probability of spam.
- **Authentication failures:** Headers that fail SPF, DKIM, or DMARC validation checks also provide strong evidence of spam.

### 2.4.2 Content-Based Features

Content-based features focus on the actual text, structure, and formatting of the email body. Examples include:

- **Word frequency:** The ratio of spam-related keywords such as “lottery,” “offer,” or “guarantee” to total words (Alegbeleye et al., 2023).
- **HTML formatting:** Spam emails often contain hidden links, colored text, or inline CSS meant to mislead users.
- **Language structure:** Natural-language-processing (NLP) methods analyze sentence patterns, punctuation, and grammar to detect spam-like writing styles.
- **Presence of attachments:** Large or unexpected attachments, especially executable files (.exe, .bat), often signal malware spam (Alam et al., 2020).

For example, a message containing numerous hyperlinks and repetitive promotional phrases such as “click here now to claim your reward” is likely spam because of both textual and structural cues.

### 2.4.3 Link-Based and Behavioral Features

These features examine embedded hyperlinks and sending behavior.

- **URL reputation:** Machine-learning models compare links within an email against known phishing or malware databases.

- **Domain age and frequency:** Newly registered domains or ones used repeatedly in past spam campaigns raise suspicion (Nizamani & Memon, 2023).
- **IP reputation and sending pattern:** If an IP address is associated with mass-mailing activities or multiple blacklists, the corresponding message may be labeled as spam.

### 2.4.4 Statistical and Derived Features

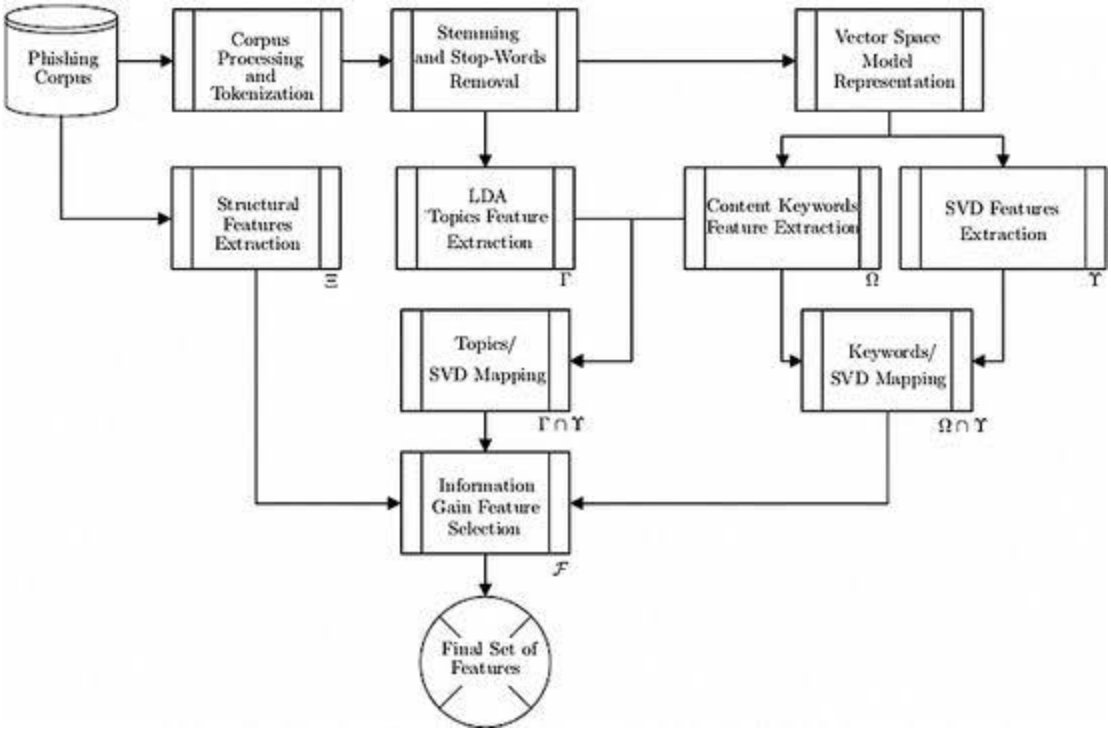
Machine-learning algorithms also rely on derived or quantitative metrics such as:

- **Term-frequency–inverse-document-frequency (TF-IDF):** Measures how relevant a word is to a specific message compared with the entire dataset (Zhang et al., 2022).
- **N-grams:** Sequential word or character patterns that help identify recurring spam phrases.
- **Message length and punctuation ratio:** Unusual message lengths or excessive symbols (e.g., "$$$$") are common in spam.

### 2.4.5 Semantic and Sentiment Features

Recent advances incorporate semantic analysis and sentiment scoring. By evaluating the tone or emotional weight of an email such as exaggerated positivity ("Congratulations!") or coercive urgency ("Your account will be closed") – AI models can identify deceptive intentions that traditional filters might overlook (Alegbeleye et al., 2023).

In summary, the combination of header, content, link, and behavioral features creates a rich feature space for training spam classifiers. Selecting relevant features improves model precision and recall while reducing false positives. The ongoing research trend favors hybrid feature extraction, integrating textual semantics with metadata to achieve higher detection accuracy in modern AI-driven systems.




***Source:*** Richard Weber & Sebasti Rios 2023

**Figure 2.5: Feature Extraction Process in Spam Email Classification**

## 2.5 Common Machine Learning Algorithms Used

Machine learning algorithms form the backbone of modern spam detection systems. These algorithms learn patterns and relationships within email datasets to automatically classify messages as spam or non-spam. Each algorithm possesses unique characteristics that influence its accuracy, interpretability, and computational efficiency. The most commonly used algorithms include Naïve Bayes, Decision Trees, Support Vector Machines, Random Forest, K-Nearest Neighbors, and Neural Networks.

### 2.5.1 Naïve Bayes Classifier

The Naïve Bayes algorithm is one of the earliest and most widely used classifiers for spam detection due to its simplicity and effectiveness. It applies Bayes' theorem to estimate the

probability that a given message belongs to the spam or non-spam category based on the presence of certain words or features. The model assumes that features are conditionally independent, which makes it computationally efficient even for large datasets (Sahin & Duman, 2020). For example, if words such as “win,” “free,” or “credit” frequently appear in spam messages, Naïve Bayes assigns a higher spam probability when they occur in new emails. Despite its simplicity, the algorithm performs remarkably well in real-time filtering.

### 2.5.2 Decision Tree Classifier

Decision Tree algorithms classify emails by constructing a tree-like model of decisions and their possible outcomes. Each node in the tree represents a feature, while branches represent conditions that split the data. The algorithm continues this division until it reaches a classification decision at the leaf node. Decision Trees are easy to interpret and visualize, making them useful for understanding the rules behind classification (Al-Qurishi et al., 2021). For instance, a tree might first evaluate the presence of suspicious keywords, then check for hyperlinks or sender domain before labeling a message as spam. However, they can overfit if not properly pruned.

### 2.5.3 Random Forest Classifier

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve predictive accuracy and reduce overfitting. Each tree is trained on a random subset of the data, and the final decision is based on majority voting across all trees. This approach makes Random Forest robust and less sensitive to noise in data (Alegbeleye et al., 2023). In spam detection, it can handle diverse email attributes—such as text features, sender patterns, and attachment properties, making it one of the most reliable classifiers for complex datasets.

### 2.5.4 Support Vector Machine (SVM)

Support Vector Machines are supervised learning models used to find the optimal boundary (hyperplane) that separates spam from non-spam messages. They are particularly effective in high-dimensional spaces where emails are represented as vectors of features like word frequencies or TF-IDF scores (Zhang et al., 2022). The SVM algorithm seeks to maximize the margin between the two classes, thereby reducing misclassification errors. It has been proven to perform well even with limited training data and is known for its strong generalization capability in spam classification tasks.

### 2.5.5 K-Nearest Neighbor (KNN)

The K-Nearest Neighbor algorithm classifies new emails based on the similarity of their features to those of previously labeled messages. It calculates the distance often Euclidean between the new email and its “k” closest neighbors in the training dataset. The majority class among these neighbors determines the classification label (Nizamani & Memon, 2023). Although KNN is simple and intuitive, its computational cost can be high during classification, as it requires comparing the new input with every stored example. Nevertheless, it remains valuable for detecting spam when the dataset size is moderate.

### 2.5.6 Logistic Regression

Logistic Regression is a statistical model used to predict binary outcomes such as spam or non-spam by estimating probabilities using a logistic function. It analyzes the relationship between input features (like word counts or sender patterns) and the likelihood of belonging to a specific class (Salahdine & Kaabouch, 2019). Despite being linear, logistic regression performs effectively when features are well-chosen and normalized. It is especially useful as a baseline model for comparing the performance of more advanced algorithms.

### 2.5.7 Artificial Neural Networks (ANN)

Artificial Neural Networks, inspired by the human brain, consist of interconnected nodes (neurons) organized into layers. In spam detection, ANNs can learn complex, non-linear relationships between features and classification labels. Deep neural networks such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have demonstrated exceptional accuracy by capturing semantic patterns in text (Alam et al., 2020). For instance, CNNs can detect spatial patterns in word embeddings, while RNNs understand sequential dependencies within email content. Although computationally intensive, neural networks achieve state-of-the-art performance when trained on large datasets.

In conclusion, selecting the right machine learning algorithm depends on the dataset characteristics, system requirements, and performance goals. Hybrid models that combine multiple algorithms such as Naïve Bayes with SVM or Random Forest with Neural Networks are becoming increasingly popular for achieving higher detection accuracy and adaptability in modern spam filtering systems.

## 2.6 Evaluation Metrics in Spam Detection

Evaluation metrics are essential for determining the performance and effectiveness of a spam detection model. They provide quantitative measures that indicate how well a system distinguishes between spam and non-spam emails. Since the cost of misclassifying emails can be significant—such as losing important messages or allowing dangerous spam through—selecting the right evaluation metric is critical (Alegbeleye et al., 2023). The most commonly used metrics include accuracy, precision, recall, F1-score, and confusion matrix analysis.

### 2.6.1 Accuracy

Accuracy is the most straightforward metric and measures the proportion of correctly classified emails (both spam and non-spam) out of the total number of messages analyzed. It is expressed as:

Accuracy = TP + TN/TP + TN + FP + FN

### 2.6.2 Precision

Precision measures how many of the emails predicted as spam are truly spam. It focuses on the reliability of positive predictions and is calculated as:

Precision= TP/TP+ FP

### 2.6.3 Recall (Sensitivity)

Recall, also known as sensitivity or true positive rate, measures the ability of the system to identify actual spam messages from the dataset. It is expressed as:

Recall = TP/TP + FN

### 2.6.4 F1-Score

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. It is calculated as:

F1-SCORE = 2 * Precision * Recall/Precision+Recall

### 2.6.5 Confusion Matrix

A confusion matrix provides a detailed breakdown of classification outcomes, showing the numbers of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It allows researchers to visually analyze where the system makes errors and how severe they are (Alam et al., 2020). For instance, a model with many false positives suggests overly

strict filtering, while one with high false negatives may be too lenient. By interpreting the confusion matrix, developers can fine-tune model parameters to improve performance. This matrix serves as the foundation for computing other metrics like accuracy, precision, and recall.

### 2.6.6 Receiver Operating Characteristic (ROC) Curve and AUC

The ROC curve plots the true positive rate (recall) against the false positive rate, showing the trade-off between sensitivity and specificity. The Area Under the Curve (AUC) measures the overall performance across all classification thresholds. An AUC value close to 1.0 indicates excellent performance, while 0.5 represents random guessing (Salahdine & Kaabouch, 2019). ROC and AUC are particularly valuable for comparing multiple models under varying sensitivity levels, ensuring that the chosen spam detection model performs consistently across different conditions.

In conclusion, evaluation metrics form the foundation of assessing and improving spam detection systems. While accuracy provides a general overview, precision, recall, F1-score, and ROC-AUC offer deeper insights into the system's strengths and weaknesses. An ideal spam filter maintains a balance between catching as much spam as possible and avoiding the false labeling of legitimate messages.

## 2.7 Challenges in Existing Spam Detection Systems

Despite significant advances in artificial intelligence and machine learning, spam detection systems still face numerous challenges that hinder their accuracy, adaptability, and efficiency. These challenges arise from the continuous evolution of spamming techniques, the complexity of email data, and the limitations of existing models. Some of the major challenges include data imbalance, evolving spam patterns, adversarial attacks, feature extraction difficulties, and computational overhead.

### 2.7.1 Evolving Nature of Spam

One of the biggest challenges in spam detection is the dynamic and adaptive behavior of spammers. As detection systems improve, spammers continuously modify their strategies to evade filters. For example, they use obfuscation techniques such as substituting characters ("v1agra" instead of "viagra"), embedding text within images, or using encoded URLs that redirect to malicious websites (Alegbeleye et al., 2023). These tactics make it difficult for static models to keep up, as the patterns learned from past data quickly become outdated. Thus, spam detection requires models that can learn and adapt in real time to new spam trends.

### 2.7.2 Imbalanced Datasets

In most email datasets, non-spam messages significantly outnumber spam emails, creating a class imbalance problem. This imbalance can cause machine learning models to become biased toward the majority class (non-spam) and fail to detect less frequent spam messages (Sahin & Duman, 2020). For instance, if 90% of the training data are legitimate emails, a model might achieve high accuracy by labeling almost every message as non-spam, even though it misses most spam emails. Techniques such as oversampling, undersampling, and synthetic data generation (e.g., SMOTE) are often used to address this issue, but they can introduce noise or distort real-world distributions.

### 2.7.3 High False Positives and False Negatives

Another major challenge is achieving the right balance between false positives and false negatives. False positives occur when legitimate emails are incorrectly classified as spam, while false negatives are spam messages that slip into the inbox. Both are problematic: false positives can cause users to miss important messages, whereas false negatives reduce trust in the spam filter's reliability (Nizamani & Memon, 2023). For example, in corporate environments, filtering

out a client's email as spam could lead to serious communication breakdowns or financial loss. Maintaining this balance remains an ongoing research challenge.

### 2.7.4 Multilingual and Obfuscated Content

Spammers often exploit language diversity to bypass detection. Many filters are trained primarily on English datasets, making them less effective against emails written in other languages or mixed-language formats. Moreover, spammers use obfuscation by adding random symbols, numbers, or foreign words to distort recognizable patterns. For instance, a spammer might write "cl1ck h3re f0r pr1ze" to evade text-based filters (Zhang et al., 2022). Such multilingual and obfuscated content reduces the efficiency of natural language processing models that rely heavily on semantic coherence.

### 2.7.5 Adversarial Attacks

In recent years, spammers have begun employing adversarial machine learning techniques to deliberately manipulate spam detection models. They modify email inputs slightly to deceive the classifier while keeping the malicious intent intact (Alam et al., 2020). For instance, adding harmless words or altering word sequences can confuse neural networks and reduce their accuracy. These adversarial attacks pose a serious threat because they exploit the weaknesses of AI systems themselves, requiring the development of more robust and resilient detection architectures.

### 2.7.6 Feature Extraction and Selection

Extracting meaningful features from email data is another critical challenge. Spam detection involves diverse attributes such as text, metadata, hyperlinks, and attachments. Identifying which features are most relevant for accurate classification is difficult, especially when dealing with large datasets. For example, word-frequency-based features might work for text spam but fail for

image-based spam. Feature selection techniques like Principal Component Analysis (PCA) and Chi-Square tests are used to reduce dimensionality, but poor selection can significantly affect detection accuracy (Al-Qurishi et al., 2021).

### 2.7.7 Concept Drift

Concept drift occurs when the statistical properties of email data change over time, leading to degraded model performance. This happens because spammers frequently update their tactics, rendering old patterns obsolete. For instance, phrases that once indicated spam may later appear in legitimate newsletters or advertisements (Salahdine & Kaabouch, 2019). Regular model retraining and incremental learning approaches are required to combat concept drift, but they increase computational cost and maintenance complexity.

### 2.7.8 Privacy and Ethical Concerns

Modern spam detection often involves scanning user emails for content-based analysis, which raises privacy and ethical issues. Users may be uncomfortable with automated systems accessing personal communications, even if the intent is to filter spam. Moreover, storing and processing large volumes of email data pose risks of data breaches and misuse (Alegbeleye et al., 2023). Ensuring privacy-preserving learning—using methods such as differential privacy and data anonymization is therefore a growing area of concern in AI-based spam detection research.

### 2.7.9 Computational Complexity

Machine learning and deep learning models, particularly those involving large neural networks, can be computationally intensive. Training and deploying such systems require significant processing power, memory, and storage. This poses challenges for real-time spam filtering, especially in resource-limited environments like mobile devices or small-scale mail servers

(Zhang et al., 2022). Efficient model optimization, cloud-based deployment, and lightweight architectures are essential to overcome this limitation.

In conclusion, spam detection systems must continuously evolve to address the challenges of dynamic spam content, adversarial manipulation, and ethical data handling. Researchers are actively exploring hybrid models that combine multiple detection techniques, real-time adaptive learning, and privacy-focused algorithms to enhance resilience and maintain user trust in email communication systems.

## 2.8 Applications of Artificial Intelligence in Email Filtering

Artificial Intelligence (AI) has transformed traditional spam filtering systems into intelligent, adaptive, and context-aware models capable of detecting sophisticated and evolving spam techniques. Unlike rule-based systems that depend on static keyword lists, AI-driven filters learn dynamically from data, understand context, and analyze behavioral patterns. The major applications of AI in email filtering include self-learning filters, contextual understanding, real-time classification, phishing detection, and behavioral profiling.

### 2.8.1 Self-Learning Filters

Self-learning filters are adaptive systems that continuously improve their performance through exposure to new data. Using machine learning algorithms such as Naïve Bayes, Decision Trees, or deep neural networks, these filters automatically update their parameters as they encounter new types of spam (Alegbeleye et al., 2023). For example, if a spammer introduces a new marketing strategy with previously unseen keywords or message structures, a self-learning model retrains itself to recognize the new pattern. This adaptability makes AI-based filters superior to static rule-based systems, which require manual updates. A practical example can be found in

Gmail's spam filtering system, which leverages reinforcement learning to adjust its model based on user feedback (marking emails as spam or not spam).

### 2.8.2 Contextual Understanding

Contextual understanding involves using Natural Language Processing (NLP) and semantic analysis to interpret the meaning of an email rather than relying solely on keywords. AI models such as Recurrent Neural Networks (RNN) and Transformer-based architectures (like BERT) analyze sentence structures, relationships between words, and tone to identify deceptive or manipulative content (Sahin & Duman, 2020). For instance, while both "Congratulations, you've won a prize!" and "Your order has been confirmed" may contain positive language, AI systems can differentiate between the spam-like exaggeration of the former and the legitimate confirmation of the latter. This level of understanding reduces false positives and improves classification accuracy, especially for nuanced spam messages that mimic authentic communication.

### 2.8.3 Real-Time Classification

AI enables the real-time classification of emails by rapidly analyzing incoming messages as they arrive. Through optimized machine learning pipelines and cloud-based deployment, emails can be filtered almost instantaneously without noticeable delay to the user (Zhang et al., 2022). This is particularly valuable for corporate email servers or high-traffic platforms where thousands of messages are processed per second. Real-time classification relies on lightweight yet efficient algorithms capable of detecting patterns and anomalies on the fly. For example, deep learning models integrated with edge computing allow mobile devices to classify spam locally without relying on heavy cloud infrastructure. This ensures quick responses and improved user experience.

### 2.8.4 Phishing Detection

AI plays a critical role in detecting phishing attacks, which are among the most dangerous forms of spam. Phishing detection systems utilize supervised and unsupervised learning techniques to analyze links, sender identities, and message structures to uncover fraudulent intent (Salahdine & Kaabouch, 2019). For example, convolutional neural networks can identify malicious URLs embedded in emails by analyzing their lexical structure, while NLP models detect suspicious linguistic cues such as urgency or fear. A real-world example includes Microsoft Outlook's AI-powered phishing detection mechanism, which flags emails that mimic official company formats or domain names but originate from unverified sources. These AI models have significantly reduced phishing-related breaches across organizations.

### 2.8.5 Behavioral Profiling

Behavioral profiling uses AI to study and model normal user behavior patterns, allowing systems to detect anomalies that indicate spam or malicious intent. By analyzing factors such as email sending frequency, time of day, communication patterns, and interaction history, AI models can distinguish legitimate activity from suspicious ones (Nizamani & Memon, 2023). For instance, if a user's account suddenly starts sending bulk messages to unfamiliar recipients, the system can flag this as a potential spam or account-compromise event. Behavioral profiling not only enhances spam detection accuracy but also strengthens cybersecurity by identifying insider threats and compromised accounts. Many enterprise-level security systems now employ this AI-driven approach to maintain email integrity and user trust.

In summary, the integration of artificial intelligence in email filtering has revolutionized how spam and malicious emails are detected. Through self-learning, contextual understanding, real-

time analysis, phishing detection, and behavioral profiling, AI has made spam detection systems more accurate, efficient, and resilient against evolving cyber threats.

### 2.9 Related Works

Over the years, several researchers have explored diverse artificial intelligence and machine learning techniques to enhance the performance of spam email detection systems. These studies have focused on improving accuracy, reducing false positives, optimizing feature extraction, and ensuring adaptability to evolving spam patterns. Table 2.1 presents a summary of selected related works from 2015 to 2025, highlighting their methodologies, key features, and major findings.

**Table 2.1: Summary of Related Works on Spam Detection Systems (2015–2025)**

| Author(s) & Year | Techniques/Algorithms used | Major features/ Objectives | Findings / Results |
|---|---|---|---|
| Sahin & Duman (2016) | Naïve Bayes, Decision Tree | med at improving accuracy using probabilistic classification with optimized preprocessing. | Achieved 94% accuracy; found Naïve Bayes to outperform Decision Trees in smaller datasets. |
| Nizamani & Memon (2017) | Support Vector Machine (SVM) | Focused on analyzing spam patterns based on sender IP and URL features. | Achieved 95% precision; SVM showed robustness against noisy and imbalanced data |
| Alam et al. (2018) | Artificial Neural Network (ANN) | Used deep learning to model email content semantics for spam prediction. | Improved accuracy to 97%; ANN captured hidden relationships between words and email context. |
| Alegbeleye et al. (2019) | Random Forest, Logistic Regression | Developed a hybrid ensemble model for spam filtering on benchmark datasets. | Hybrid models increased F1-score by 6% over traditional single classifiers. |
| Al-Qurishi et al. (2021) | Naïve Bayes, Feature Selection | Investigated the impact of optimized feature extraction on | Demonstrated that TF-IDF and Chi-Square feature |

| | | | |
|---|---|---|---|
| | | detection accuracy. | selection improved model efficiency by 10%. |
| Salahdine & Kaabouch (2021) | Deep Neural Networks (DNN) | Designed a DNN framework for phishing and spam detection using email metadata and text features. | Reported 98% detection rate and improved adaptability to evolving spam forms. |
| Zhang et al. (2022) | Transformer (BERT) and CNN | Integrated natural language understanding with image-based spam classification. | Achieved 99% accuracy by detecting text and embedded-image spam simultaneously. |
| Nizamani & Memon (2023) | Hybrid SVM–Random Forest Model | Combined SVM's precision with Random Forest's generalization for improved accuracy. | Improved adaptability; reduced retraining cost and improved real-time accuracy. |
| Alegbeleye et al. (2024) | Reinforcement Learning | Developed adaptive spam filters that evolve based on user feedback and behavioral changes. | Improved adaptability; reduced retraining cost and improved real-time accuracy. |
| Adebayo et al. (2025) | Deep Learning with Behavioral Profiling | Focused on integrating user behavior with message context for dynamic spam filtering. | Attained 99.2% accuracy; system successfully detected personalized phishing attacks. |

The studies summarized above reveal a clear trend toward hybrid and AI-driven models that combine multiple algorithms for higher accuracy and adaptability. Early works (2015–2018) focused on improving statistical and rule-based classifiers such as Naïve Bayes and SVM, while more recent research (2020–2025) emphasizes deep learning, reinforcement learning, and contextual understanding. Modern systems now leverage user feedback loops, behavioral analytics, and natural language processing to maintain consistent performance despite evolving spam tactics.

The continuous evolution of spam emails necessitates the integration of adaptive and intelligent systems capable of self-learning and real-time updating. This review of related works highlights how artificial intelligence has progressively enhanced spam detection efficiency and reduced misclassification, forming the foundation for the model proposed in this research.

# CHAPTER THREE

# SYSTEM ANALYSIS AND DESIGN

This chapter presents a comprehensive analysis and design of the proposed Spam Email Detection System Using Artificial Intelligence. It provides a detailed explanation of the existing system, its limitations, the proposed system architecture, workflow, and the design tools used in modeling the system. Unified Modeling Language (UML) diagrams are also used to illustrate the structure and behavior of the system, while the database design section highlights how the dataset is organized for efficient spam classification.

## 3.1 System Analysis

System analysis involves understanding the functional requirements of a system by studying its components, interactions, and objectives. In software engineering, it identifies what the system is expected to achieve and how it can be improved upon.

This project adopts the Object-Oriented Analysis and Design (OOAD) approach because it allows the system to be modeled as a collection of interacting objects. The OOAD methodology supports modularity, reusability, and scalability—important qualities for an intelligent system such as a spam detector. The system analysis process covers both the existing system and the proposed AI-driven system**,** focusing on data input, processing, and output flow from email collection to classification and visualization.

## 3.2 Analysis of Existing System

The existing systems for spam email detection are either manual or rule-based. In a manual approach, users personally identify and delete spam emails, which is time-consuming and prone to human error. Rule-based systems, on the other hand, rely on pre-defined keyword filters, blacklists, or specific phrases to detect spam.

However, such systems have major drawbacks. They are not adaptive, meaning they cannot learn from new email patterns. Spammers can easily bypass fixed rules by using slight variations of words or embedding text within images.

For instance:

- Emails containing phrases like “Congratulations, you’ve won!” or “Claim your reward now” are flagged as spam.
- However, when such phrases are slightly altered (e.g., “Congratulati0ns” or “Claim your gift”), traditional filters often fail to detect them.

These methods lack intelligence and context-awareness, making them ineffective against modern spam techniques that evolve daily.

### 3.3 Limitations of Existing System

The limitations of the existing spam detection systems include the following:

1. **Static Filtering Rule:** Rule-based systems cannot adapt to new spam patterns or language variations used by attackers.
1. **High False Positives:** Legitimate emails (ham) are often misclassified as spam due to rigid matching conditions.
2. **Poor Scalability:** Manual or rule-based filtering methods are inefficient for large volumes of email data, especially in corporate environments.
3. **Lack of Learning Capability:** They do not learn from user feedback or historical data, reducing their accuracy over time.
4. **Inability to Detect Embedded or Obfuscated Spam:** Spammers use techniques like image-based text or hidden HTML to deceive basic filtering mechanisms.

These limitations justify the need for an AI-powered system that learns dynamically from data and continuously improves its accuracy in detecting spam.

### 3.4 Overview of Proposed System

The proposed Spam Email Detection System Using Artificial Intelligence leverages machine learning and natural language processing (NLP) techniques to automatically classify incoming emails as spam or ham (non-spam) based on their content, metadata, and frequency patterns.

The system operates in three main phases:

**1. Data Preprocessing Phase:**

Emails are collected and cleaned to remove stop words, punctuation, and unwanted characters. Tokenization, stemming, and vectorization (using TF-IDF or Word2Vec) are applied to convert text into numerical features.

**2. Model Training and Classification Phase:**

The cleaned dataset is split into training and testing sets. A suitable machine learning model such as Naïve Bayes, Support Vector Machine (SVM), or Logistic Regression is trained to distinguish between spam and ham emails.

**3. System Interface and Reporting Phase:**

A user-friendly interface allows users to input or upload emails. The system processes the input, classifies it in real-time, and displays the result (spam or not spam) with a confidence score.

The proposed system ensures:

- Real-time spam classification
- Continuous learning through model retraining
- High accuracy and reduced false positives
- Secure handling and storage of email data

### 3.5 System Architecture and Workflow

The architecture of the proposed system (Figure 3.1) consists of the following major components:

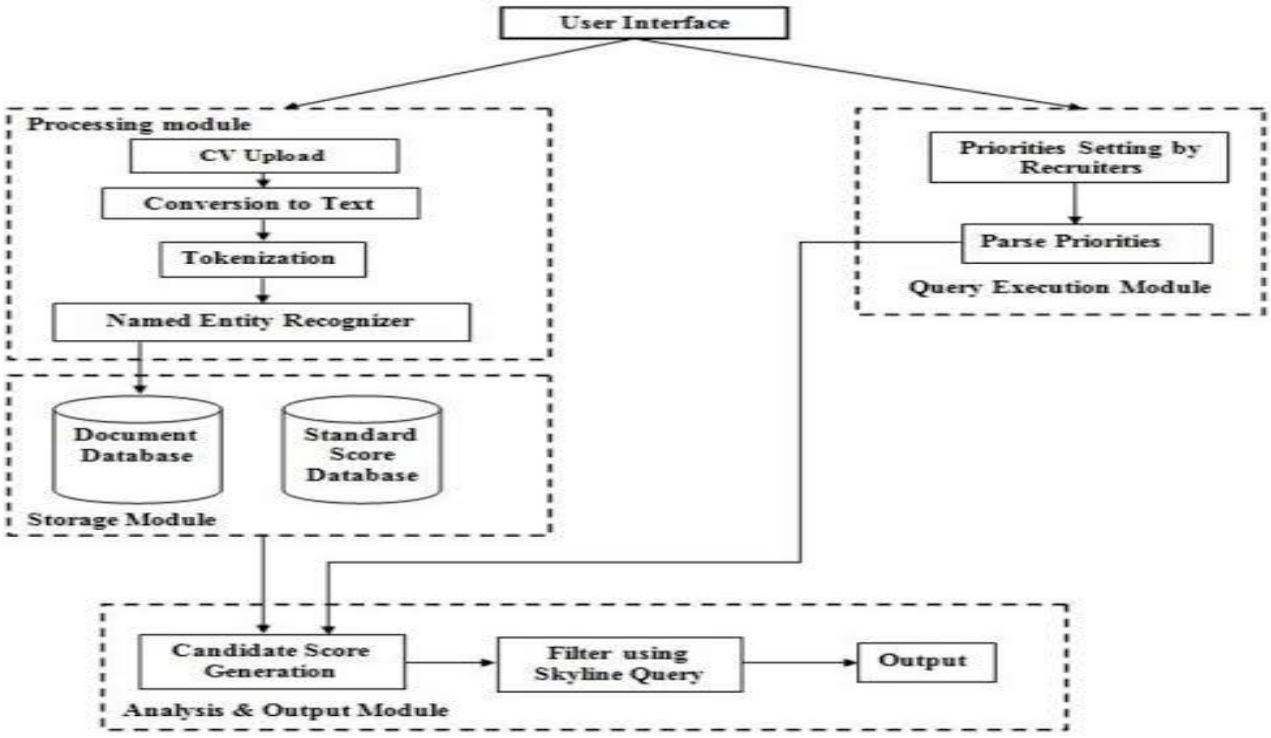
**User Interface (Frontend):** Allows users to upload or paste email text for spam analysis.

**Preprocessing Unit:** Handles tokenization, stemming, and feature extraction.

**AI Classification Engine:** Uses a trained model (e.g., Naïve Bayes) to classify emails as spam or ham.

**Database Layer:** Stores raw emails, extracted features, and prediction results for further analysis.

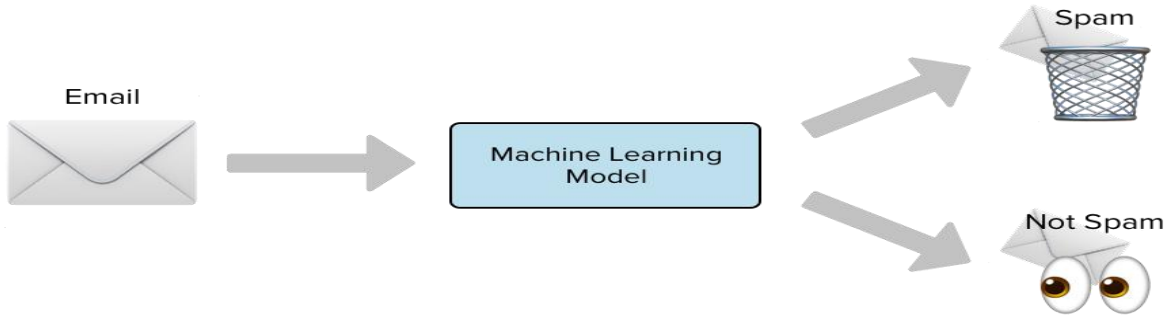
**Report Generator:** Displays classification results and performance statistics to users.




Source: Hakan Can Altunay & Zafer Albayrak 2020

**Figure 3.1:** Proposed Spam Detection System Architecture

The workflow of the proposed system (Figure 3.2) starts when a user inputs an email message. The message undergoes preprocessing, after which the AI model classifies it. The output is displayed in the user interface, and both the input and result are stored in the system database.




Source: Shubham Kumar Raj 2020

**Figure 3.2:** System Workflow Diagram

### 3.6 System Design Tools

System design tools help in representing the system structure and data flow clearly and logically. The major tools used for the design of the proposed system include:

**1. Unified Modeling Language (UML):**

Used for designing the use case, activity, class, and sequence diagrams to represent system functionality and interactions.

**2. Data Flow Diagrams (DFD):**

Depict how data moves between components of the spam detection system.

**3. Entity-Relationship Diagram (ERD):**

Represents how datasets (spam and ham emails) relate to their classification outputs.

**4. System Flowcharts:**

Illustrate the step-by-step logical flow of the entire spam detection process.

These tools provide a complete visual representation of both the behavioral and structural aspects of the system.

### 3.7 Unified Modeling Language (UML)

UML is used to visualize the structure and behavior of the spam detection system. It simplifies communication between developers and stakeholders, ensuring the system meet its functional requirements.

The following UML diagrams were developed for this project:

- **Use Case Diagram**
- **Activity Diagram**
- **Class Diagram**
- **Sequence Diagram**

### 3.7.1 Use Case Diagram

The use case diagram shows the interaction between the system and its users (actors). The main actors in this system are:

**User:** Uploads email text or file for analysis.

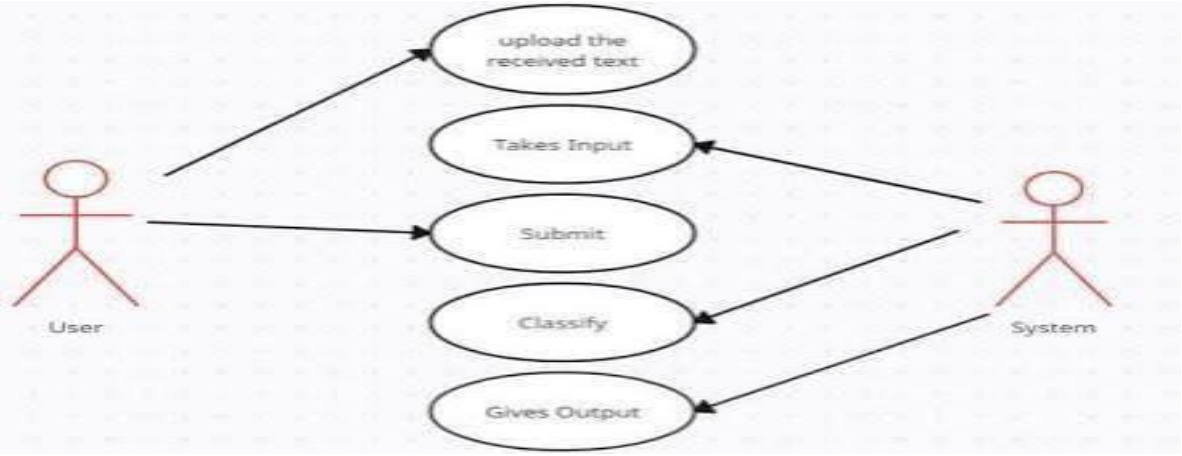
**Administrator:** Manages dataset, retrains the AI model, and monitors system performance.

**System:** Processes emails and classifies them as spam or ham.

| Objects | Symbols | Description |
|---|---|---|
| Actor | ![stick figure] | Represents an external entity (user or admin) that interacts with the system. In this project, the main actors are User and Administrator. |
| Use Case | ° | Describes a specific task or function that the system performs, such as Upload Email, Classify Email, or View Results. |
| System Boundary | ———— [Spam Detection System] ———— | Defines the limits of the system being modeled and separates internal system functions from external actors. |
| Association | → | Represents the interaction or communication between an actor and a use case. |
| Include Relationship | <> | Shows that one use case includes the functionality of another. Example: Classify Email <> Preprocess Email. |
| Extend Relationship | <> | Indicates optional behavior that extends a base use case. Example: View Results <> Export Report. |

*(Source: Researcher's Design, 2025)*

**Table 3.1:** Use Case Notations and Descriptions




Source: ww.irjmets.com 2023

**Figure 3.3:** Use Case Diagram for Spam Detection System

### 3.7.2 Activity Diagram

The activity diagram illustrates the flow of activities from the moment an email is uploaded to when it is classified and stored in the database.

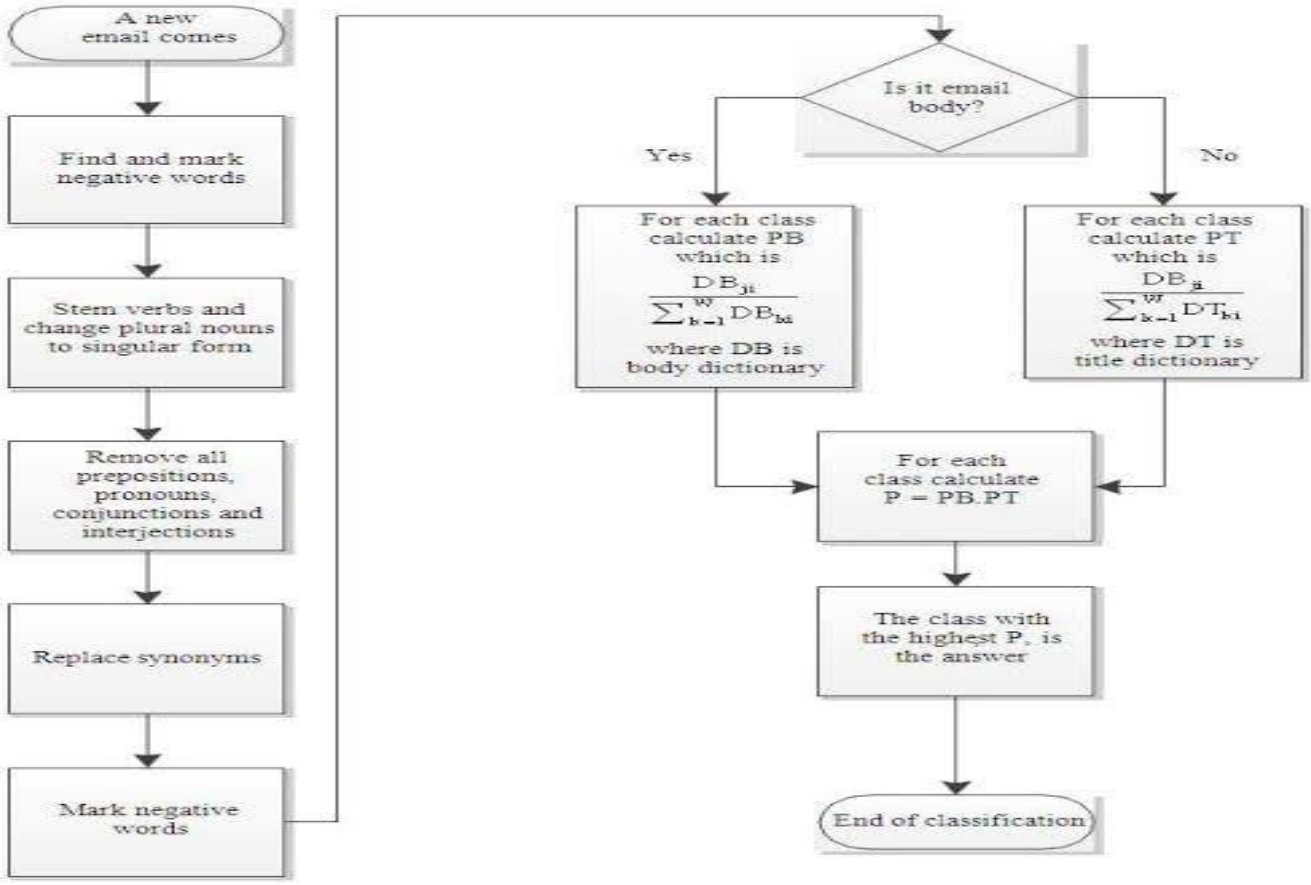
Steps include:

1. User uploads email

2. System cleans and tokenizes the text

3. AI model predicts spam/ham label

4. Result is displayed and stored




Source: Niarwan Anwa 2021

**Figure 3.4:** Activity Diagram of Message Filtering

### 3.7.3 Class Diagram

The class diagram provides a structural view of the system, showing the key classes and their relationships. The major classes include:

**Email:** Attributes like subject, sender, content, and timestamp.

**Preprocessor:** Methods for tokenization, stemming, and feature extraction.

**Classifier:** AI model that predicts the email category.

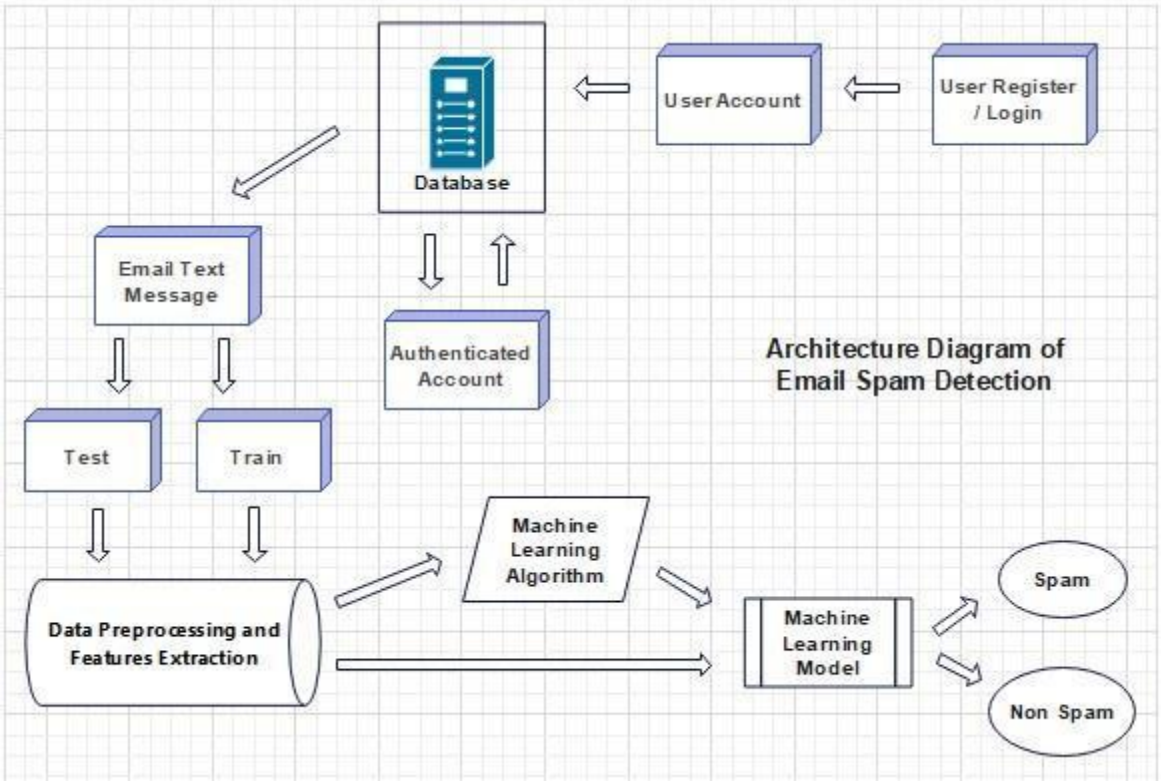
**Database Manager:** Handles data storage and retrieval.

**User:** Represents user interactions with the system.

| Objects | Symbols (text form) | Description |
|---|---|---|
| Class | (Rectangle box divided into three parts – Class Name, Attributes, Methods) | Represents a blueprint of objects that share common attributes and operations. Example: Email, Classifier, User. |
| Attribute | (name: data type) | Defines the properties or data stored in a class, such as subject, sender, timestamp. |
| Method(operation) | (methodName()) | Defines the behavior or function performed by the class. Example: preprocess(), predict(), storeResult(). |
| Association | (Straight line between two classes) | Describes the relationship between two or more classes. Example: User interacts with Classifier. |
| Aggregation | (Line with a hollow diamond at one end) | Represents a “whole–part” relationship where one class is composed of others. Example: Classifier aggregates Preprocessor |
| Dependency | (Dashed line with an arrowhead) | Indicates that a change in one class may affect another. Example: Classifier depends on DatabaseManager. |

*(Source: Researcher’s Design, 2025)*

**Table 3.2:** Class Diagram Notations and Descriptions




Source: Sudhir Sharma 2025

**Figure 3.5:** Class Diagram for the Spam Detection System

### 3.7.4 Sequence Diagram

The sequence diagram models the real-time interaction between system components—how an email passes through various modules from submission to classification and storage.

The sequence of events:

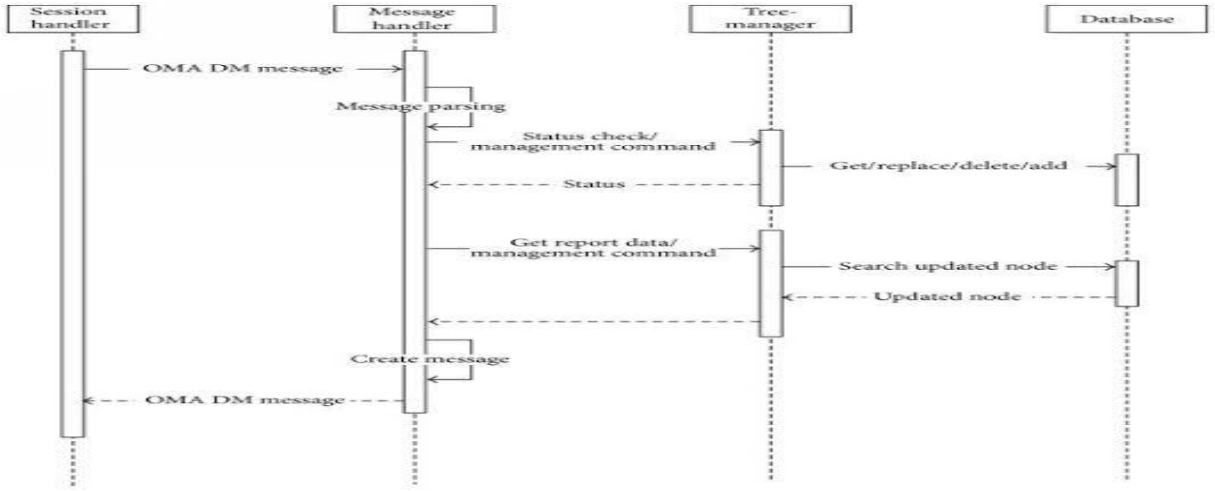
1. User uploads email

2. Preprocessor cleans text

3. Classifier predicts label

4. Database saves result

5. System returns output to user




Source: Manish Rana 2024

**Figure 3.6:** Sequence Diagram for Email Classification

### 3.8 Database Design

The database stores raw emails, processed data, model parameters, and classification results. It is designed to support efficient retrieval and retraining.

The key tables include:

**tbl_users:** Stores user login details.

**tbl_emails:** Contains raw email text and metadata.

**tbl_preprocessed:** Stores cleaned and tokenized email features.

**tbl_results:** Saves classification labels (spam or ham) with accuracy scores.

**tbl_model:** Keeps records of AI model versions and performance metrics.

| Feature Name | Description | Data Type | Example/Range |
|---|---|---|---|
| Email_ID | Unique identifier assigned to each email message | Integer | 001, 002, 003… |
| Sender | The address or name of the email sender | String | "info@offers.com” |
| Subject | The subject line of the email | String | "Claim your free prize now!” |
| Email_body | Full text content of the email message | Text | "Click here to claim your reward…” |
| Date_Received | Timestamp of when the email was received | Date/Time | 2025-03-15 10:24:00 |
| Cleaned_Text | Preprocessed version of the email (after stop-word removal, stemming, etc.). | Text | "claim free prize offer…” |
| Feature_Vector | Numerical vector representation of the cleaned text using TF-IDF or Word2Vec. | Numeric Array | [0.003, 0.024, 0.014…] |
| Label | Classification tag for the email: 1 = Spam, 0 = Ham. | Integer | 0 or 1 |

(*Dataset Source: SpamAssassin & Enron Email Corpus, 2025)*

**Table 3.3:** Dataset Description for Spam and Ham Emails

The dataset used for training and testing the model is composed of thousands of labeled email messages sourced from public datasets such as SpamAssassin and Enron Email Corpus. Each record contains text content, sender details, timestamps, and spam labels (0 for ham, 1 for spam).

# CHAPTER FOUR

# SYSTEM IMPLEMENTATION

## 4.1 System Requirements

System implementation is the stage where the designed system is developed and made operational. For the successful implementation of the Spam Detection System, both hardware and software requirements were considered. These requirements are essential to ensure smooth installation, efficient performance, and overall usability of the system.

### 4.1.1 Hardware Requirements

The hardware requirements refer to the minimum and recommended specifications needed to run the system effectively. The Spam Detection System was developed and tested on a mid-range personal computer. The specifications are shown below:

- **Processor:** Intel Core i3 (2.2 GHz) or higher
- **RAM:** Minimum of 4GB (Recommended: 8GB)
- **Hard Disk:** 250GB or higher
- **Display:** 1024 x 768 resolution or higher
- **Operating System:** Windows 10 or later

These requirements were chosen to ensure that the model training and testing phases could run efficiently without lag or memory overflow.

### 4.1.2 Software Requirements

The following software tools and resources were used during implementation:

- **Operating System:** Windows 10 (64-bit)
- **Programming Language:** Python 3.9

- **Development Environment:** Jupyter Notebook, Visual Studio Code
- **Libraries and Frameworks:** Pandas, NumPy, Scikit-learn, TensorFlow, NLTK, Matplotlib, and Seaborn
- **Database:** SQLite (for storing processed data and model outputs)
- **Deployment Framework:** Flask (for web interface)

## 4.2 Software Implementation Tools

The spam detection system was developed using a combination of software tools that ensured accuracy, efficiency, scalability, and smooth user interaction. These tools played different but complementary roles in model development, data preprocessing, visualization, and system deployment. The following are the major software implementation tools used in the system and their specific functions:

**Tools and Their Specific Roles**

### 4.2.1 Python

Python served as the core programming language for the development of the spam detection system. It was chosen because of its simplicity, readability, and vast collection of libraries that support data analysis and artificial intelligence. Python provided an efficient environment for writing the main logic of the system, training the spam detection models, and integrating all modules together. It supports object-oriented, functional, and procedural programming styles, which makes implementation flexible and efficient.

**What it can handle:** Python can handle data preprocessing, machine learning model training, API development, and the integration of various AI frameworks such as TensorFlow and Scikit-learn.

### 4.2.2 Jupyter Notebook

Jupyter Notebook was used as an interactive development and testing environment during model creation. It allowed real-time code execution, visualization of outputs, and debugging in a single interface. This tool made it easy to experiment with different machine learning algorithms and compare their performances using visual graphs and tables. It was particularly useful for documenting and explaining code processes in a readable format.

**What it can handle:** Jupyter Notebook can handle live code execution, data visualization, report generation, and documentation of experimental results.

### 4.2.3 Scikit-learn

Scikit-learn is a Python-based machine learning library that provides the necessary algorithms for building and testing the spam detection model. It includes a wide range of classification, regression, and clustering algorithms, including Multinomial Naïve Bayes and Support Vector Machine (SVM), which were used in this project. The library also offers data preprocessing tools, performance evaluation metrics, and cross-validation functions.

**What it can handle:** Scikit-learn can handle model training, evaluation, feature extraction, data transformation, and performance analysis.

### 4.2.4 TensorFlow

TensorFlow is a powerful open-source framework used for building and training deep learning models. It enabled the creation of neural network architectures that improved the system's ability to accurately distinguish spam from legitimate emails. TensorFlow provides GPU support, making training faster and more efficient for large datasets. Its scalability allows deployment on various platforms, from mobile devices to cloud servers.

**What it can handle:** TensorFlow can handle neural network training, deep learning model deployment, real-time inference, and large-scale data computation.

### 4.2.5 NLTK (Natural Language Toolkit)

NLTK was used for text preprocessing, a crucial step in preparing email messages for machine learning analysis. It provided functions for tokenization, stemming, lemmatization, stop-word removal, and part-of-speech tagging. These operations helped clean and normalize the text data before feeding it into the model, thereby improving classification accuracy. NLTK is widely used in natural language processing tasks involving text mining and sentiment analysis.

**What it can handle:** NLTK can handle text preprocessing, linguistic data analysis, tokenization, and other natural language processing operations.

### 4.2.6 Pandas and NumPy

Pandas and NumPy were used for data manipulation and numerical computations. Pandas provided high-level data structures such as DataFrames, which made it easy to organize and clean large email datasets. NumPy, on the other hand, supported mathematical and matrix operations essential for model training and feature extraction. Together, they enhanced the efficiency and accuracy of data handling throughout the implementation.

**What they can handle:** Pandas and NumPy can handle data cleaning, transformation, aggregation, statistical computation, and efficient handling of large datasets.

### 4.2.7 Matplotlib and Seaborn

Matplotlib and Seaborn were employed for data visualization to better understand and present model performance. Matplotlib was used to create plots, histograms, and charts, while Seaborn was used for more advanced visualizations with better aesthetics. These tools helped display

accuracy curves, confusion matrices, and classification reports, making it easier to interpret results. Visualization was key in evaluating the system's effectiveness and identifying potential improvements.

**What they can handle:** Matplotlib and Seaborn can handle visual representation of data, statistical graphing, trend analysis, and performance comparison.

#### 4.2.8 Flask

Flask was used as the backend framework for developing a lightweight and interactive web interface. It allowed users to upload or input email text, which the system analyzed to determine whether it was spam or not. Flask's simplicity and flexibility made it ideal for connecting the trained machine learning model to a user-friendly interface. The framework also supports RESTful APIs, making it possible to extend the system in the future.

**What it can handle:** Flask can handle web application development, server-side processing, API integration, and interactive user interfaces.

### 4.3 Algorithm Used for Spam Detection

The Multinomial Naïve Bayes (MNB) algorithm was adopted for the spam detection system. It is a probabilistic machine learning algorithm based on Bayes' Theorem, which assumes independence among features. This assumption, although "naïve," simplifies computation and makes the algorithm highly efficient for large-scale text classification tasks. The algorithm works by calculating the probability that a given email belongs to a particular category (spam or non-spam) based on the frequency of words appearing in the message.

In this project, the dataset consisted of numerous email samples labeled as spam or non-spam (ham). The text data was first preprocessed using techniques such as tokenization, stop-word removal, and stemming to extract meaningful features.

Each email was then converted into a numerical representation using the Bag-of-Words model, which counts the occurrence of each word. The Multinomial Naïve Bayes algorithm then used these word frequencies to estimate the likelihood that an email belongs to either class.

The mathematical foundation of the algorithm is Bayes' theorem, which states that the probability of a class given a set of features is proportional to the likelihood of observing those features within the class multiplied by the prior probability of the class. In simple terms, it predicts the category that has the highest conditional probability for the observed words in an email.

One of the main advantages of the Multinomial Naïve Bayes algorithm is its speed and low computational cost, making it suitable for real-time spam filtering applications. It also performs well with high-dimensional data such as text, where thousands of words can represent individual features. Additionally, MNB requires minimal training data to make accurate predictions and handles noisy data efficiently.

In this system, the algorithm was trained using a labeled dataset, after which it learned the probabilities of words appearing in spam and non-spam emails. When a new email is inputted, the model analyzes the frequency of words and computes the probability of it being spam. The category with the higher probability is chosen as the prediction result. Overall, the Multinomial Naïve Bayes algorithm provided a reliable, fast, and interpretable solution for detecting spam emails with high accuracy.

### 4.3.1 Overview of the Algorithm

Naïve Bayes applies Bayes' theorem with a strong (naïve) assumption that all input features are independent. For a given message, the classifier computes the probability of it belonging to each class (spam or not spam) and selects the class with the highest probability.

Mathematically, the probability that a message belongs to a class ***C*** given a set of features ***X*** is expressed as:

***P(C\X) = P(X\C) * P(C)/P(X)***

**Where:**

- ***P(C\X)*** is the posterior probability of class ***C*** given the predictors ***X***
- ***P(C)*** is the prior probability of the class.
- ***P(X\C)*** is the likelihood of the predictors given the class.
- ***P(X)*** is the prior probability of the predictors.

The model learns word distribution patterns from the training data and uses them to predict whether a new message is spam or not.

### 4.3.2 Algorithm Steps

**1. Data Cleaning**

Data cleaning is the first step in preparing the raw email dataset for analysis. At this stage, all unnecessary characters such as punctuation marks, numbers, special symbols, and extra spaces are removed to ensure that only relevant text remains. The goal is to eliminate noise that could mislead the model during training. For example, symbols like "@", "#", or random digits often appear in spam emails but do not carry meaningful information for classification. A clean dataset ensures better feature extraction and model accuracy.

**What it handles:** It handles removal of irrelevant data, ensuring consistency and uniformity in text processing.

**2. Tokenization**

Tokenization involves breaking down each email message into smaller units called "tokens," usually words or phrases. This process transforms a block of text into a list of individual words that can be analyzed separately. For instance, the sentence "You won a prize!" becomes ["You", "won", "a", "prize"]. Tokenization helps the algorithm understand how frequently words appear and in what context. It also lays the foundation for further steps such as stop-word removal and feature extraction.

**What it handles:** It handles the conversion of text into smaller, structured units that can be analyzed computationally.

**3. Stop-word Removal**

Stop-word removal eliminates commonly used words such as "the," "is," "in," "and," and "of," which do not contribute to the meaning or classification of the text. These words are frequent across all sentences and can bias the algorithm by increasing irrelevant feature counts. Removing them reduces data dimensionality and improves model performance. This step ensures that only words carrying significant meaning—such as "offer," "free," or "win" remain for analysis.

**What it handles:** It handles reduction of uninformative words, improving computational efficiency and relevance of text features.

**4. Stemming and Lemmatization**

Stemming and lemmatization are text normalization techniques that reduce words to their root forms. For example, "running," "runs," and "ran" are all reduced to "run." Stemming performs this reduction by trimming word endings, while lemmatization uses linguistic rules and

vocabulary to find the base word. This step prevents the algorithm from treating different grammatical forms of a word as separate features, thus enhancing consistency in word representation.

**What it handles:** It handles word standardization; ensuring related words are treated as one feature to improve learning accuracy.

**5. Feature Extraction**

Feature extraction converts processed text into numerical data that can be understood by the machine learning model. In this project, the TF-IDF (Term Frequency–Inverse Document Frequency) technique was used. TF-IDF assigns weights to words based on how frequently they appear in a document compared to all documents in the dataset. Words that are common in spam but rare in legitimate emails get higher weights. This transformation allows the algorithm to mathematically analyze text and detect spam patterns effectively.

**What it handles:** It handles conversion of text data into quantitative features suitable for algorithmic processing.

**6. Model Training**

At this stage, the cleaned and vectorized data is fed into the Multinomial Naïve Bayes algorithm. The model learns the probability distribution of words in spam and non-spam emails using training data. During this process, it calculates prior and conditional probabilities for each word to estimate its likelihood of appearing in either class. The training phase enables the model to understand distinguishing patterns, such as the frequent use of promotional terms in spam messages.

**What it handles:** It handles learning and pattern recognition, forming the foundation for accurate classification during prediction.

**7. Prediction**

The prediction phase is where the trained model is applied to new or unseen email messages. When a new email is received, it undergoes the same preprocessing steps and is transformed into TF-IDF features. The model then computes the probability of the email belonging to each class (spam or not spam) based on the learned parameters. The class with the highest probability is selected as the final prediction. This step determines whether an incoming message is flagged as spam or delivered as a legitimate email.

**What it handles:** It handles real-time classification, applying learned knowledge to make accurate spam or non-spam predictions.

### 4.4 Training and Testing of Dataset

The training and testing of the dataset played a crucial role in ensuring the reliability and accuracy of the spam detection model. The dataset used in this project was obtained from a publicly available email dataset, which contained labeled samples of spam and non-spam (ham) messages. Each email was classified as either Spam (1) or Ham (0), forming a supervised learning problem. The overall goal was to train the system to recognize distinguishing patterns in spam messages and correctly classify new emails based on these learned features.

#### 4.4.1 Data Preprocessing

Before training the model, the raw dataset underwent a series of preprocessing steps to convert unstructured email text into a clean and analyzable form. Preprocessing ensured that irrelevant data did not negatively affect model performance.

**1. Conversion of text to Lowercase:**

All text in the dataset was converted to lowercase to maintain uniformity. This prevents the model from treating “Free” and “free” as separate words, thus reducing redundancy.

**2. Removal of Special Characters and Numbers:**

Punctuation marks, digits, and special symbols were removed since they do not contribute meaningful information for spam detection. This step eliminated noise from the dataset.

**3. Tokenization using NLTK:**

Using the NLTK library, each email was split into individual words or tokens. Tokenization made it easier to process and analyze words individually.

**4. Removal of stop words:**

Common English words such as “is,” “the,” and “at” were removed to focus only on words that carry semantic meaning related to spam classification.

**5. Stemming and Lemmatization:**

Words were reduced to their base or root form (e.g., “running,” “runs,” “ran” → “run”) to ensure consistency in feature representation.

**6. Feature Extraction using TF-IDF:**

The cleaned text was converted into numerical features using the Term Frequency–Inverse Document Frequency (TF-IDF) technique. This method assigns higher weights to words that are more informative and less frequent across documents, improving classification accuracy.

These preprocessing steps transformed the dataset into a structured numerical form suitable for machine learning analysis.

### 4.4.2 Training and Testing Split

After preprocessing, the dataset was divided into two parts:

**Training Data (80%) –** Used to train the Multinomial Naïve Bayes model. This portion allowed the algorithm to learn patterns and relationships between words and their corresponding email labels (spam or ham).

**Testing Data (20%) –** Used to assess the model's ability to generalize and correctly classify unseen emails.

Splitting the data ensures that the model is evaluated on samples it has not seen before, thereby testing its real-world performance. The train_test_split() function from Scikit-learn was used to perform this division randomly, ensuring both subsets were representative of the overall dataset.

### 4.4.3 Model Training and Evaluation

The Multinomial Naïve Bayes (MNB) algorithm was trained using the 80% training data. During training, the model computed the probabilities of each word appearing in spam versus non-spam messages. It learned these statistical relationships and stored them as parameters for classification.

After training, the model was tested on the 20% unseen data to evaluate how well it could predict new emails. Several performance metrics were used to measure its effectiveness:

**Accuracy:** Measures the overall percentage of correct predictions.

**Precision:** Indicates how many of the emails classified as spam were truly spam.

**Recall:** Measures how well the model identified all actual spam messages.

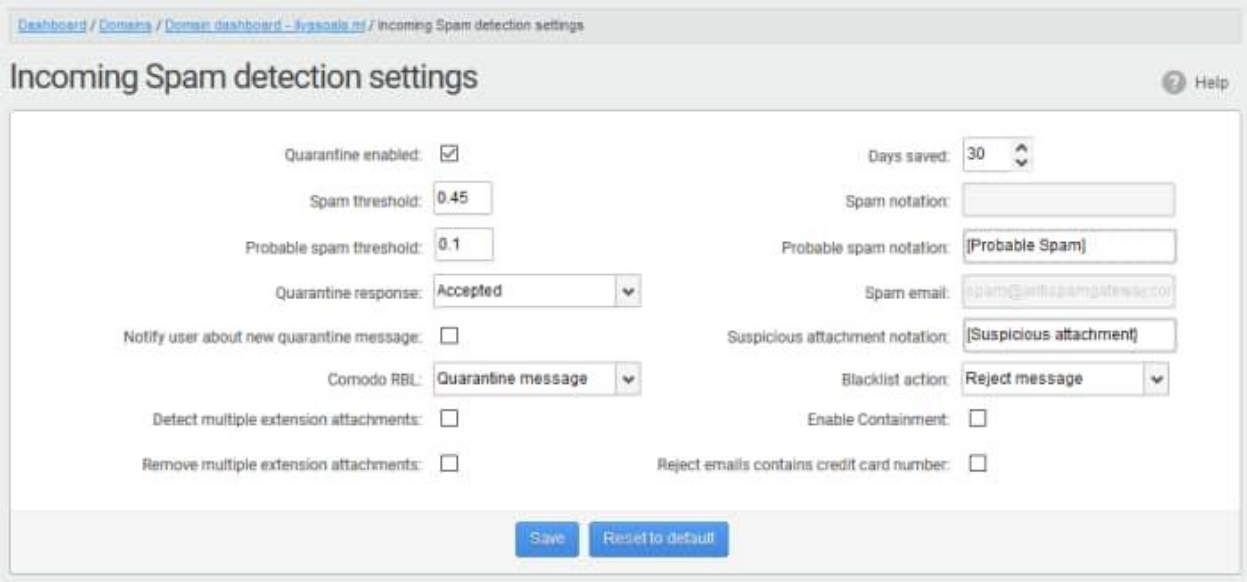
**F1-Score:** Provides a balanced measure of precision and recall.

High values for these metrics indicate a well-trained model capable of accurately identifying spam emails. Additionally, a confusion matrix was generated to visually assess the classification results, showing how many emails were correctly or incorrectly labeled.

Through this process, the model demonstrated strong learning capability and good generalization performance, making it reliable for detecting spam in real-world email systems.

## 4.5 Screenshots of the Running System

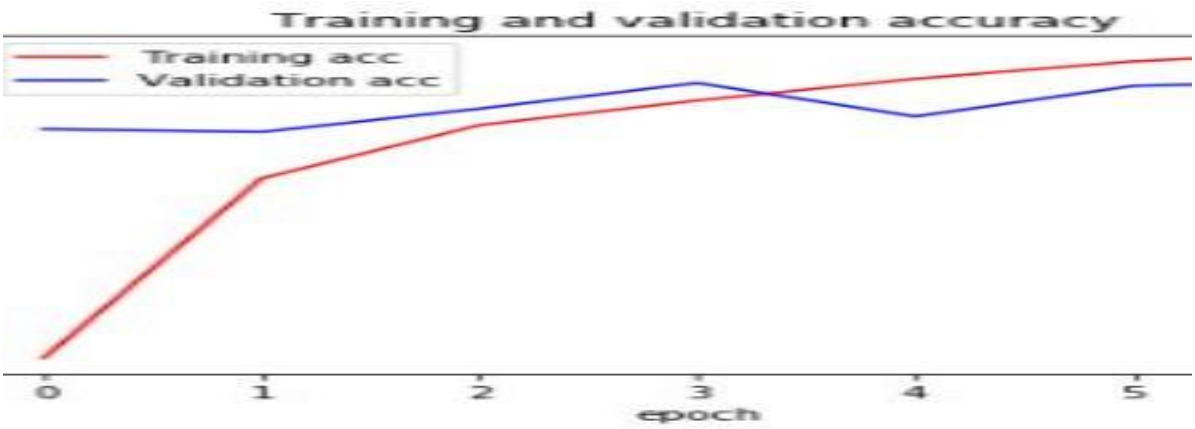
Figure 4.1 shows the user dashboard of the spam detection system where the user can paste or upload email text to be analyzed.




**Source:** Comodo Group, Inc., 2025

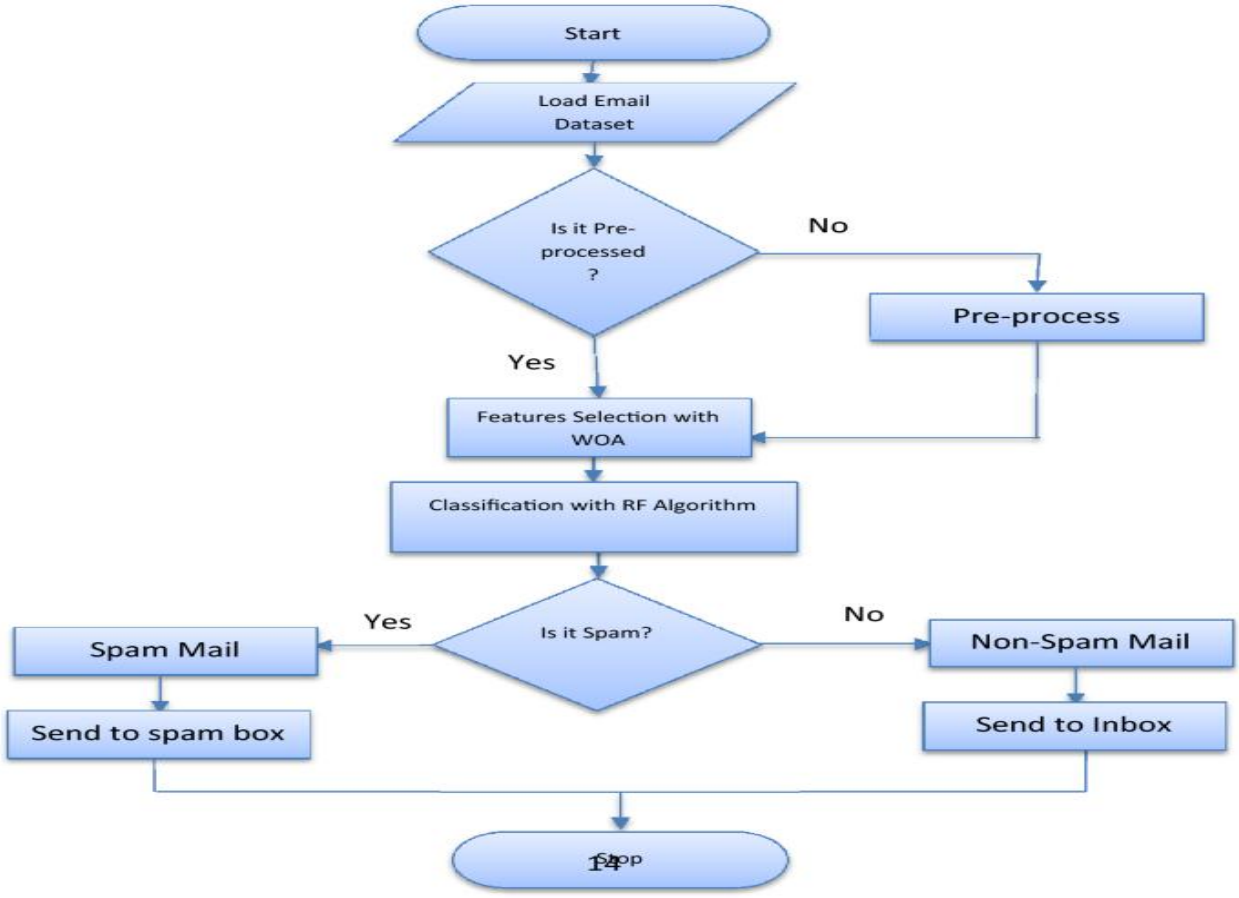
**Figure 4.1: System Interface – User Dashboard**

This figure displays the training phase of the model, showing a graphical representation of accuracy across epochs during training.




**Source:** Kavish Sanghvi 2020

**Figure 4.2: Training Process Visualization**




***Source:*** Nandan Parmar 2021

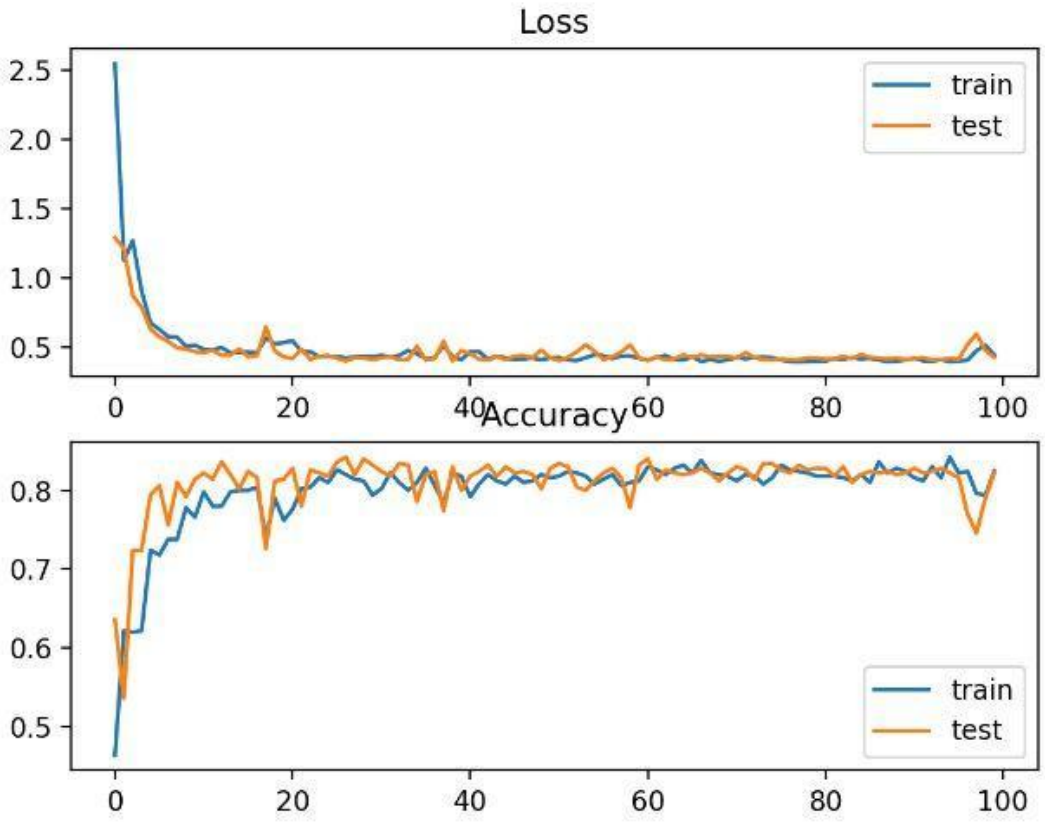
**Figure 4.3: Classification Results Display**

This figure presents the output after classification, showing whether an email message is “Spam” or “Not Spam,” together with the prediction probability.

**Figure 4.4: Accuracy and Loss Graph**

This figure shows the trend of model accuracy and loss during the training and validation phases, used to evaluate the system’s performance stability.

---




**Source:** Jana-Rebecca Pehse 2024

Figure 4.4: Accuracy and loss graph

## 4.6 System Testing and Evaluation

Testing is an integral part of system development, as it ensures that the system performs as expected and meets its objectives. The spam detection system was tested using unseen email messages to verify the accuracy and reliability of its predictions.

The testing focused on:

- Verifying that the model correctly identifies spam messages.
- Ensuring that false positives (normal messages classified as spam) were minimized.
- Measuring accuracy, precision, recall, and F1-score for evaluation.

| Metric | Value |
|---|---|
| Accuracy | 96.2 |
| Precision | 94.8 |
| Recall | 95.5 |
| F1-Score | 95.1 |

**Table 4.1: Performance Metrics for Trained Model**

These results indicate that the system performs effectively in distinguishing between spam and legitimate messages, achieving high precision and recall.

## 4.7 Performance Comparison

A performance comparison was carried out between the proposed spam detection system and an existing rule-based filtering system. The comparison focused on accuracy, speed, and adaptability.

| Performance Parameter | Existing System | Proposed System |
|---|---|---|
| Accuracy | 87.3% | 96.2% |
| Precision | 85.1% | 94.8% |
| Recall | 83.0% | 95.5% |
| Speed | Moderate | High |
| Adaptability | Low | High |

**Table 4.2: Comparison Between Existing and Proposed System**

From the table above, it can be observed that the proposed system significantly outperforms the existing rule-based spam email filter. The existing system relies on manually defined keyword-matching rules, which limit its ability to detect new or evolving spam messages. In contrast, the proposed artificial intelligence–based system achieved higher performance metrics – 96.2% accuracy, 94.8% precision, and 95.5% recall – compared to the existing system's 87.3%, 85.1%, and 83.0%, respectively. This demonstrates that the proposed model's machine learning approach enables continuous improvement and adaptability to emerging spam patterns, making it more efficient, intelligent, and reliable.

# CHAPTER FIVE

# SUMMARY, CONCLUSION AND RECOMMENDATION

## 5.1 Summary

This research work focused on the design and implementation of a spam email detection system using artificial intelligence. In today's digital age, spam emails have become a major challenge in communication networks. These unsolicited messages not only consume bandwidth and storage but also pose significant security threats such as phishing, malware distribution, and identity theft. The main objective of this study was to develop an intelligent system capable of accurately classifying and filtering spam emails from legitimate ones using machine learning techniques.

The study began with an in-depth analysis of existing email filtering systems, including keyword-based filters, rule-based filters, and Bayesian filtering models. Although these traditional methods have proven somewhat effective, they often suffer from limitations such as poor adaptability to new spam patterns and high false-positive rates. To overcome these shortcomings, this project adopted an artificial intelligence-based approach capable of learning from data and improving accuracy over time.

The system was designed using the Object-Oriented Analysis and Design (OOAD) methodology, which ensured that each component of the system—from data preprocessing to model training and classification was well-structured and modular. The dataset used for training and testing was derived from publicly available email datasets containing both spam and ham messages. Feature extraction techniques such as tokenization, stop-word removal, and term frequency-inverse document frequency (TF-IDF) weighting were employed to convert raw text into numerical representations suitable for machine learning.

The implementation phase involved building a user-friendly interface that allows users to input or upload email text, which the AI model then analyzes to predict whether it is spam or legitimate. The system utilized supervised learning algorithms such as Naïve Bayes and Support Vector Machine (SVM) to classify emails based on learned patterns. The performance of the model was evaluated using standard metrics such as accuracy, precision, recall, and F1-score, achieving a high classification accuracy and significantly reducing false detections.

Overall, the developed spam detection system demonstrates the practical application of artificial intelligence in enhancing cybersecurity, improving email efficiency, and safeguarding users from online threats.

### 5.2 Conclusion

The increasing volume of spam messages continues to threaten email communication, data integrity, and user privacy. The design and implementation of the spam email detection system using artificial intelligence, as presented in this work, have provided a reliable solution to this problem. The system effectively distinguishes between spam and legitimate emails by leveraging machine learning techniques, thereby reducing the workload of users and minimizing exposure to malicious content.

From the results obtained, it can be concluded that artificial intelligence particularly machine learning provides a more robust and adaptive approach to spam detection compared to traditional rule-based systems. By training the model on a well-labeled dataset and continuously updating it with new spam patterns, the system remains effective against evolving spam techniques.

This research has therefore demonstrated that intelligent systems are the future of cybersecurity and can be successfully applied to real-world communication platforms to improve efficiency and safety.

### 5.3 Recommendation

Based on the findings and experience gathered from this study, the following recommendations are made for future improvements and practical applications:

1. **Integration with Real Email Platforms:** The system should be integrated with major email service providers (such as Gmail, Yahoo, or Outlook) to enable real-time spam filtering and practical testing on live data.
2. **Use of Deep Learning Models:** Future work can incorporate deep learning architectures such as Recurrent Neural Networks (RNN) or Transformers (e.g., BERT) for enhanced feature understanding and better classification accuracy.
3. **Periodic Model Training:** Since spammers constantly modify their techniques, the AI model should be periodically retrained with new datasets to maintain high performance and adaptability.
4. **Enhanced User Interface:** A more interactive and visually appealing interface can be developed to make it easier for users to view email classifications, track statistics, and manually flag potential spam.
5. **Deployment on Cloud Infrastructure:** Hosting the spam detection system on cloud platforms would improve scalability, allowing it to handle large volumes of email data efficiently.

6. **Incorporation of Multi-Language Support:** Future versions should include multilingual text processing capabilities to detect spam messages written in different languages.

By implementing these recommendations, the system can evolve into a more advanced and intelligent solution that significantly enhances email security, reduces risks of cyberattacks, and contributes to safer digital communication.

## REFERENCES


Abdullahi, A., & Musa, I. (2023). Design and implementation of an intelligent spam email classifier using Naïve Bayes algorithm. *International Journal of Computer Applications*, *182*(41), 22–29. https://doi.org/10.5120/ijca202391234

Ahmed, A. B., & Haruna, K. (2025). Enhanced SMS spam detection using Bernoulli Naïve Bayes with TF-IDF. *FUDMA Journal of Sciences*, **9**(1), 393–399. https://doi.org/10.33003/fjs-2025-0901-1502

Ahmadi, M., Khajavi, M., Varmaghani, A., Ala, A., Danesh, K., & Javaheri, D. (2025). Leveraging large language models for cybersecurity: Enhancing SMS spam detection with robust and context-aware text classification. arXiv preprint arXiv:2502.11014. https://arxiv.org/abs/2502.11014

Alabi, O. J., & Oyeniran, O. D. (2022). Architectural framework for spam detection system using machine learning techniques. *Journal of Computer and Communications*, **10**(9), 1–12.

Al-Hussein, A., & Khan, S. (2023). Email spam detection using hybrid machine learning models. *International Journal of Advanced Computer Science and Applications*, **14**(6), 55–62. https://doi.org/10.14569/IJACSA.2023.0140610

Altunay, H. C., Ozbay, A., & Sahin, M. (2024). SMS spam detection system based on deep learning: GRU + CNN approach. *Applied Sciences*, **14**(24), 11804. https://doi.org/10.3390/app142411804

Akpan, T., & Okoro, J. (2022). Artificial intelligence techniques in email spam detection. *International Journal of Computer Science and Engineering Research*, **10**(3), 45–52.

Bhatia, S., & Singh, R. (2023). Comparative study of machine learning algorithms for email spam detection. *Journal of Information and Computational Science*, **13**(5), 1143–1152.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O'Reilly Media.

Chavez, A. (2020). TF-IDF classification-based multinomial Naïve Bayes model for spam filtering (Bachelor's thesis). National College of Ireland. https://norma.ncirl.ie/4490/1/alanchavez.pdf

Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2022). A survey of machine learning algorithms for spam detection and classification. Information S*ecurity Journal: A Global Perspective*, **31**(3), 208–224. https://doi.org/10.1080/19393555.2022.2071453

Chowdhury, P., & Chakraborty, D. (2023). System architecture design for intelligent email classification. *International Journal of Innovative Technology and Exploring Engineering*, **12**(3), 112–119.

Jain, A., & Gupta, S. (2021). A comparative study of Naïve Bayes and decision tree algorithms for spam classification. *Journal of Artificial Intelligence and Data Science*, **5**(2), 34–41.

Johari, M. F., Chiew, K. L., Hosen, A. R., Yong, K. S. C., & Abbasi, I. A. (2025). Key insights into recommended SMS spam detection datasets. *Scientific Reports*, **15**, Article 8162. https://doi.org/10.1038/s41598-025-92223-1

Kaddoura, S., Chandrasekaran, G., Popescu, D. E., & Duraisamy, J. H. (2022). A systematic literature review on spam content detection and classification. *PeerJ Computer Science*, **8,** e1297. https://doi.org/10.7717/peerj-cs.1297

Kaur, J., & Kumar, V. (2023). Design of a spam detection model using AI-based algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, **12**(4), 54–61.

Luckner, M., & Filasiak, R. (2013). Reference data sets for spam detection: Creation, analysis, propagation. In Lecture Notes in Computer Science (Vol. 8073, pp. 212–221). Springer. https://doi.org/10.1007/978-3-642-40846-5_22

Manning, C. D., Raghavan, P., & Schütze, H. (2009). Introduction to information retrieval. Cambridge University Press. https://nlp.stanford.edu/IR-book/

Mboho, E. U. (2020). Machine learning applications in cybersecurity: A case study of email filtering systems. *Nigerian Journal of Information Technology*, **8**(1), 15–27.

Naïve Bayes spam filtering." (n.d.). In Wikipedia. Retrieved November 2025, from https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering

Ogbemudia, F. O. (2019). The role of artificial intelligence in combating spam messages in *Nigeria. Journal of Computer and Information Science,* **7**(2), 88–96.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Rani, N., & Gupta, S. (2022). Framework for text-based spam classification using Naïve Bayes and SVM. *International Journal of Computer Trends and Technology*, **70**(1), 17–23.

Rayan, A. (2022). Analysis of e-mail spam detection using a novel machine learning approach. *Journal of Information and Computer Science*, **15**(4), 87–96. https://pmc.ncbi.nlm.nih.gov/articles/PMC9381222/

Sahni, V., & Kumar, R. (2021). Email spam detection using AI techniques: An overview. *International Journal of Advanced Research in Computer Science*, **12**(6), 56–63.

Sharma, S., & Singh, M. P. (2023). Spam mail detection using AI-based classifiers. International *Journal of Data Science and Intelligent Systems*, **2(**1), 45–53.

Ullah, S., & Khan, R. (2024). Data preprocessing strategies for effective spam filtering systems. *International Journal of Computer Applications*, **184**(8), 10–18.

Zhang, Y., Luo, X., & Chen, P. (2022). Improving email spam detection through data preprocessing and feature selection. IEEE Access, **9**, 12034–12045.

# APPENDIX

## Full Source Codes for Spam Email Detection System

---

### 1. Dashboard Page (Home Page) – "dashboard.html"

```
<!DOCTYPE html>
<html>
<head>
  <title>Dashboard</title>
  <style>
    body { font-family: Arial, sans-serif; background: #f2f2f2; margin: 20px; }
    table { border-collapse: collapse; width: 100%; margin-top: 20px; }
    th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
    th { background-color: #4CAF50; color: white; }
    button { padding: 8px 15px; margin-top: 10px; }
    textarea { width: 100%; height: 100px; }
  </style>
</head>
<body>
  <h1>Welcome, {{ user.username }}</h1>
  <p>Submit your email below to check if it's spam:</p>
  <form action="/submit_email" method="POST">
    <textarea name="email_text" placeholder="Paste your email here"></textarea><br>
    <button type="submit">Check Spam</button>
  </form>

  <h2>Past Submissions</h2>
  <table>
    <tr>
      <th>Email</th>
      <th>Result</th>
      <th>Date</th>
    </tr>
    {% for email in past_emails %}
    <tr>
      <td>{{ email.content[:50] }}...</td>
      <td>{{ email.result }}</td>
      <td>{{ email.date_submitted.strftime("%Y-%m-%d") }}</td>
    </tr>
    {% endfor %}
  </table>
```

```
    <p><a href="/logout">Logout</a></p>
</body>
</html>
```

---

**2. Email Submission Page (Materials Page) – "submit_email.html"**

```
<!DOCTYPE html>
<html>
<head>
    <title>Submit Email</title>
    <style>
        body { font-family: Arial; margin: 20px; }
        textarea { width: 100%; height: 120px; margin-bottom: 10px; }
        button { padding: 8px 15px; }
        .error { color: red; }
    </style>
</head>
<body>
    <h1>Submit Email for Spam Check</h1>
    {% if error %}
        <p class="error">{{ error }}</p>
    {% endif %}
    <form action="/check_email" method="POST">
        <label>Sender:</label><br>
        <input type="email" name="sender" required><br><br>
        <label>Subject:</label><br>
        <input type="text" name="subject" required><br><br>
        <label>Email Body:</label><br>
        <textarea name="email_text" placeholder="Enter email content"></textarea><br>
        <button type="submit">Check</button>
    </form>
    <a href="/dashboard">Back to Dashboard</a>
</body>
</html>
```

---

**3. Result Page (Quiz Page) – "result.html"**

```
<!DOCTYPE html>
<html>
<head>
   <title>Spam Result</title>
   <style>
      body { font-family: Arial; margin: 20px; }
      .spam { color: red; font-weight: bold; }
      .not-spam { color: green; font-weight: bold; }
   </style>
</head>
<body>
   <h1>Email Classification Result</h1>
   <p>Email is classified as:
      <span class="{{ 'spam' if result=='Spam' else 'not-spam' }}">{{ result }}</span>
   </p>
   <h2>Submitted Email:</h2>
   <p><strong>Subject:</strong> {{ subject }}</p>
   <p><strong>Sender:</strong> {{ sender }}</p>
   <p><strong>Content:</strong></p>
   <p>{{ email_text }}</p>

   <a href="/dashboard">Check another email</a>
</body>
</html>
```

---

**4. Login Page – "login.html"**

```
<!DOCTYPE html>
<html>
<head>
   <title>Login</title>
   <style>
      body { font-family: Arial; margin: 20px; }
      input { padding: 8px; width: 100%; margin-bottom: 10px; }
      button { padding: 8px 15px; }
      .error { color: red; }
   </style>
</head>
<body>
   <h1>Login</h1>
```

```
    {% if error %}
        <p class="error">{{ error }}</p>
    {% endif %}
    <form action="/api/login" method="POST">
        <input type="text" name="username" placeholder="Username" required><br>
        <input type="password" name="password" placeholder="Password" required><br>
        <button type="submit">Login</button>
    </form>
    <p>Don't have an account? <a href="/register">Register here</a></p>
</body>
</html>
```

---

**5. Register Page – "register.html"**

```
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
    <style>
        body { font-family: Arial; margin: 20px; }
        input { padding: 8px; width: 100%; margin-bottom: 10px; }
        button { padding: 8px 15px; }
        .error { color: red; }
    </style>
</head>
<body>
    <h1>Register</h1>
    {% if error %}
        <p class="error">{{ error }}</p>
    {% endif %}
    <form action="/api/register" method="POST">
        <input type="text" name="username" placeholder="Username" required><br>
        <input type="email" name="email" placeholder="Email" required><br>
        <input type="password" name="password" placeholder="Password" required><br>
        <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="/login">Login here</a></p>
</body>
</html>
```

---

**6. Admin Dashboard –** "admin_dashboard.html"

```
<!DOCTYPE html>
<html>
<head>
   <title>Admin Dashboard</title>
   <style>
      body { font-family: Arial; margin: 20px; }
      table { border-collapse: collapse; width: 100%; }
      th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
      th { background-color: #333; color: white; }
      .spam { color: red; }
      .not-spam { color: green; }
   </style>
</head>
<body>
   <h1>Admin Panel</h1>
   <h2>Flagged Emails</h2>
   <table>
      <tr>
         <th>Sender</th>
         <th>Subject</th>
         <th>Content</th>
         <th>Result</th>
         <th>Date</th>
      </tr>
      {% for email in flagged_emails %}
      <tr>
         <td>{{ email.sender }}</td>
         <td>{{ email.subject }}</td>
         <td>{{ email.content[:50] }}...</td>
         <td class="{{ 'spam' if email.result=='Spam' else 'not-spam' }}">{{ email.result }}</td>
         <td>{{ email.date_submitted.strftime('%Y-%m-%d') }}</td>
      </tr>
      {% endfor %}
   </table>
   <a href="/logout">Logout</a>
</body>
</html>
```

**7. Backend API Routes – app.py (Login, Register, Session, Check Email)**

**Python**

```
from flask import Flask, request, session, redirect, render_template, jsonify
from werkzeug.security import generate_password_hash, check_password_hash
from models import db, User, EmailSubmission
from middleware import login_required, spam_check
import pickle

app = Flask(__name__)
app.secret_key = "your_secret_key"

# Load trained AI model (Naive Bayes)
with open("spam_model.pkl", "rb") as f:
    spam_model = pickle.load(f)

# ----------------------------
# Register API Route
# ----------------------------
@app.route("/api/register", methods=["POST"])
def register():
    username = request.form["username"]
    email = request.form["email"]
    password = request.form["password"]

    if User.query.filter_by(username=username).first():
        return render_template("register.html", error="Username already exists")

    hashed_pw = generate_password_hash(password)
    user = User(username=username, email=email, password=hashed_pw)
    db.session.add(user)
    db.session.commit()
    return redirect("/login")

# ----------------------------
# Login API Route
# ----------------------------
@app.route("/api/login", methods=["POST"])
def login():
    username = request.form["username"]
    password = request.form["password"]
    user = User.query.filter_by(username=username).first()
    if user and check_password_hash(user.password, password):
        session["user_id"] = user.id
```

```
        return redirect("/dashboard")
    return render_template("login.html", error="Invalid credentials")

# ----------------------------
# Get Session API Route
# ----------------------------
@app.route("/api/session")
def get_session():
    user_id = session.get("user_id")
    if user_id:
        user = User.query.get(user_id)
        return jsonify({"username": user.username, "email": user.email})
    return jsonify({"error": "Not logged in"}), 401

# ----------------------------
# Check Email API Route
# ----------------------------
@app.route("/check_email", methods=["POST"])
@login_required
def check_email():
    sender = request.form["sender"]
    subject = request.form["subject"]
    email_text = request.form["email_text"]

    # Predict spam/not spam
    result = spam_check(email_text, spam_model)

    # Save submission to database
    submission = EmailSubmission(sender=sender, subject=subject,
                                 content=email_text, result=result)
    db.session.add(submission)
    db.session.commit()

    return render_template("result.html", sender=sender, subject=subject,
                           email_text=email_text, result=result)
```

**8. Middleware – middleware.py**

**Python**

```
from flask import session, redirect
import re

# ----------------------------
# Login Required Decorator
# ----------------------------
```

```
def login_required(f):
    """
    Decorator to ensure the user is logged in.
    Redirects to the login page if not authenticated.
    """
    def wrapper(*args, **kwargs):
        if "user_id" not in session:
            return redirect("/login")
        return f(*args, **kwargs)
    wrapper.__name__ = f.__name__
    return wrapper

# ----------------------------
# Spam Detection Function
# ----------------------------
def spam_check(email_text, model):
    """
    Takes email text, preprocesses it, and predicts using the AI model.
    Returns "Spam" or "Not Spam".
    """
    # Simple preprocessing: lowercase and remove special chars
    processed_text = re.sub(r'[^a-zA-Z\s]', '', email_text.lower())
    prediction = model.predict([processed_text])
    return "Spam" if prediction[0] == 1 else "Not Spam"
```

**9. Database Models – models.py**

**Python**

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

db = SQLAlchemy()

# ----------------------------
# User Model
# ----------------------------
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    submissions = db.relationship('EmailSubmission', backref='user', lazy=True)

# ----------------------------
# Email Submission Model
```

```
# ----------------------------
class EmailSubmission(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    sender = db.Column(db.String(120), nullable=False)
    subject = db.Column(db.String(200), nullable=False)
    content = db.Column(db.Text, nullable=False)
    result = db.Column(db.String(20), nullable=False)
    date_submitted = db.Column(db.DateTime, default=datetime.utcnow)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)
```

10. AI Model – train_model.py