DEVELOPMENT OF BANK ALERT FRAUD DETECTION SYSTEM USING MACHINE LEARNING ALGORITHMS

BY

NDUBUISI CHINENYE ROSEMARY

GOU/U22/CSC/858

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY,

GODFREY OKOYE UNIVERSITY, ENUGU STATE. IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF SCIENCE (B.SC), DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: DR. U. F. UKEBANAMA

JUNE, 2026.

APPROVAL PAGE

This research, titled "DEVELOPMENT OF BANK ALERT FRAUD DETECTION SYSTEM USING MACHINE LEARNING ALGORITHMS," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu, Nigeria.

Dr. U. F. Ukebanama ....

Supervisor. Signature/Date

External Examiner. ....

External Examiner Signature/Date

Dr. Frank Okebanama ....

HOD, Department of Computer Science and Mathematics Signature/Date

Prof .I. A. Ajah ....

Dean, Faculty of Computing and Information Technology Signature/Date

CERTIFICATION

I certify that I completed the research included herein and that it was primarily based on my own observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

NDUBUISI CHINENYE ROSEMARY

GOU/U22/CSC/858

DEDICATION

This project is dedicated to God Almighty, whose grace and wisdom sustained me through every stage of this work, and to my beloved family, whose unwavering love, encouragement, and sacrifices made this academic journey possible.

ACKNOWLEDGEMENTS

In the first place, I render all the praise and honor to Almighty God for the power, wisdom, guidance, and protection which He bestowed upon me throughout this research process, without which this study could not have been achieved.

I thank Dr. U. F. Okebanama for the invaluable assistance, encouragement, and constructive feedback he gave me in carrying out this project. I also want to thank Engr. Camilus Njoku and Engr. Kingsley Chimaeze for the valuable advice, direction, and assistance in helping me achieve the correct procedure to complete this research work successfully.

I want to thank the Head of Department and all lecturers in the Department of Computer Science and Mathematics for their invaluable assistance in ensuring my academic success.

I also want to thank my father for sponsoring my academic career as well as giving me all the necessary support needed. Without his assistance, encouragement, and belief in me, my academic achievements might never have been achieved.

Last but not least, I appreciate my entire family, relatives, and friends who played some part in ensuring that this study was completed.

ABSTRACT

However, the increasing incidence of bank alert fraud, whereby individuals receive fraudulent SMS alerts from unknown senders or even fake banks, poses a potential danger to the use of bank alerting systems. Despite the availability of existing literature on transaction fraud detection using data mining techniques, there are few studies that have been conducted on the classification of user-received SMS alerts. Consequently, this research aims to develop a bank alert fraud detection system for using machine learning algorithms. In this study, a dataset consisting of 600 SMS messages was collected manually. The messages were labeled into three categories: real, fake, and suspicious bank alerts. The dataset was collected manually via gathering of SMS messages from personal acquaintances, family members, and friends, as well as some publicly available SMS messages on social networking websites. The dataset includes 200 samples for each of the classes, after which it is further preprocessed, cleaned, and enlarged to 2,700 SMS messages. This feature set was extracted via TF-IDF Vectorization and Custom Feature Extraction. Three different models, including logistic regression, random forest, and XGBoost, were trained and tested with the data. The performance of the Logistic Regression classifier is better than that of others. It has an accuracy of 97.6%, precision of 97.7%, recall of 97.6%, and F1-score of 97.6%. The application is capable of detecting bank SMS alert fraud. The front-end was made using React Native, and the back-end was coded in FastAPI. Supabase will be used for the database management system. The user can decide to type in his/her message for verification. These messages can be classified either as 'real', 'fake', or 'suspicious'. If users feel like the classification algorithm used in this study has given a false label, they can provide feedback through API points in order to make the classifier better. This study does not need any SMS interception feature since it violates the terms set by both Android and the Google Play Store.

TABLE OF CONTENTS

CHAPTER TWO: LITERATURE REVIEW

CHAPTER THREE: SYSTEM ANALYSIS AND DESIGN

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

The emergence of mobile banking in Nigeria in the past decade has changed the dynamics of money management for citizens. Banks such as GTBank, UBA, Access Bank, Zenith Bank, and First Bank now issue text messages immediately whenever there is any transaction carried out from the customer's account. Known as bank alerts, these messages have become a common practice among Nigerians and an easy way of verifying payment, especially in the daily transactions of business dealings. Nigerian businessmen, sellers, and other entrepreneurs use these messages to verify payment before providing the buyer with goods or services (Abayomi-Alli et al., 2022).

But with the increase in reliance on such alerts, a major risk has emerged. Scammers have used this weakness by designing fake SMS alerts that resemble the actual alert messages sent by banks. This is done to convince the receiver that the amount was actually withdrawn even though the transaction never occurred. The effects have been disastrous, with many people suffering huge losses due to these scams (Okonkwo et al., 2022).

Apart from false alerts, another form of suspicious messages has cropped up. The distinguishing factor here is that although the message does not seem blatantly fake, there is something strange about it which renders its validity doubtful. Often, such messages cause perplexity on part of the user and prevent them from discerning the authenticity of the message without approaching the concerned bank (Adetunji et al., 2023).

The level of fraud cases in Nigeria’s banking industry has been increasing notwithstanding the measures that have been put in place by bodies like the CBN and NCC in their attempt to curb this trend. From the NIBSS report of 2023, there have been losses amounting to billions of naira annually from cases of electronic fraud. In most cases, losses experienced arise from social engineering attacks which include fake bank alerts (Abayomi-Alli et al., 2022).

Machine learning, one of the subfields of artificial intelligence, is effective in classifying texts. Some of its uses in classification tasks include detecting spam emails, conducting sentiment analyses and identifying phishing messages. Some of these approaches include Logistic

Regression, Random Forest and XGBoost, and they have all been successful in similar use cases of detecting frauds with high precision. Using natural language processing (NLP), specifically the term frequency-inverse document frequency (TF-IDF), model training has become efficient in this respect (Adetunji et al., 2023).

This research work suggests creating a mobile application for detecting bank SMS alerts as either real, fake, or suspicious by implementing machine learning for classifying the messages. This approach is simple and user-friendly because each user can copy his or her bank SMS alert into the app or web app, and get the classification result instantly, including the level of the model's certainty. The app is built based on React Native (frontend), FastAPI (backend API), and Supabase (database). Additionally, there is also the feedback function included to allow users to report wrong results of the classification process.

1.2 Statement of the Problem

The following problems motivated this study:

1. No labeled dataset of Nigerian bank SMS alerts exists covering all three message types such as real, fake, and suspicious.
2. Existing fraud detection systems focus only on transaction-level analysis and cannot classify SMS messages received on users' phones.
3. Many Nigerian traders and market sellers accept bank alerts as immediate proof of payment without any verification.
4. No user-facing tool exists to help individuals verify the authenticity of a bank alert before acting on it.

1.3 Aim and Objectives of the Study

The main aim of this study is to develop a bank alert fraud detection system using machine learning algorithms.

To achieve this aim, the following specific objectives were pursued:

1. Identify the factors that determine whether a bank SMS alert is real, fake, or suspicious.

2. Collect and prepare a dataset of Nigerian bank SMS alerts categorized as Real, Fake, and Suspicious.
3. Design and develop a machine learning-based bank alert fraud detection system
4. Implement the system as a mobile and web-based application.
5. Test and evaluate the system using performance metrics such as accuracy, precision, recall, and F1-score.

1.4 Significance of the Study

The importance of this study comes from the practical implementation which it offers to mitigate the issue of fraudulent transaction alerts being sent to Nigerian bank customers. Using the system developed in this research, a bank customer will be able to check if the alert he/she receives is legitimate, illegitimate, or doubtful, thus preventing him/her from releasing any goods, making money transfers, and conducting other actions associated with financial risk taking.

Moreover, this study proves useful for small entrepreneurs, POS service providers, tradesmen, and others involved in cashless business operations because the automated analysis of transaction alerts would help them avoid scams and use their devices more securely.

Furthermore, this study is also important from the perspective of contributions that it makes into the national financial and cybersecurity environment as well as into its development as far as it develops an efficient solution that works within specific parameters of communication used by Nigerian financial institutions and cybercriminals.

On an academic level, the current research is significant for contributing to machine learning, mobile security, cybersecurity, and fraud detection areas because it adds new data sets, models, and findings to existing research on SMS-based fraud detection systems.

1.5 Scope of the Study

The focus of this study is on the classification of SMS and alerts from mobile banking in the Nigerian banking sector. It is designed to classify these types of messages into three categories: authentic, fake, and suspicious.

The system would target some of the popular banks within Nigeria. These include GTBank, UBA, Access Bank, Zenith Bank, and FirstBank because these banks are the most popular banks in Nigeria. The dataset was collected based on the format followed by these banks.

In addition, the research did not cover e-mail scams related to online banking, scams through phone calls, social media banking frauds, or any attacks targeting the bank system directly. Secondly, automatic interception of SMS is not covered in this work because it is impossible for an unknown developer to accomplish such a task both on the Android platform and the Google Play Store platform. Users will be required to copy alerts and paste them in the app.

1.6 Limitation of the Study

Just like any other research, this project was performed under some limitations that had some impacts on the results obtained during the process.

First, one limitation is the number of messages in the dataset. There were only 600 messages used in this research. Although the dataset was augmented to obtain 2,700 messages, using an organically produced dataset would have been more effective and increased the generality of the models.

Also, it was very difficult to access banking API and transactional data due to strict regulations on banking information in Nigeria. Therefore, no actual banking information was used. The data obtained was from personal sources.

Furthermore, the lack of an automated SMS interception system in the application is a constraint. The Android OS prevents unauthorized access to a phone user's SMS inbox for security purposes. Users will have to manually copy and paste their alerts into the application; this might make using the program slightly inconvenient.

Finally, it should be noted that the system was never tested under actual operational conditions where real-life users used it during the course of this research project. The testing took place using test inputs, and although the findings have been positive, more testing under actual conditions is required before any conclusions can be drawn.

1.7 Operational Definition of Terms

The following terms are used frequently throughout this research and are defined here for clarity:

1. Bank Alert: An SMS message that is sent to the user of a bank informing him/her about an event taking place in his/her bank account like a withdrawal or any other activity.
2. Fake Alert: This type of SMS message is sent to the victim with the intention of tricking him/her into believing that a transaction has taken place by mimicking a real bank message.
3. Suspicious Alert: An SMS message that resembles a bank alert but fails to provide adequate information or uses suspicious language and patterns.
4. Real Alert: A real bank alert communicated using the legitimate communication platform of a recognized bank, which includes proper information related to the transactions and account numbers.
5. Machine Learning (ML): A branch of artificial intelligence that facilitates learning by machines through data and improvement in task performance without specific programming.
6. Natural Language Processing (NLP): A branch of artificial intelligence dealing with the interactions between computers and human languages so that computers can read and understand texts.
7. TF-IDF (Term Frequency–Inverse Document Frequency): A metric statistic used in natural language processing to measure the significance of a word in a document concerning other documents.
8. Logistic Regression: A predictive analytic model that works by taking independent variables into account and predicting the value of a dependent variable.
9. Random Forest: An ensemble learning method that involves generating many decision trees and returning their majority votes.
10. XGBoost: It is an enhanced version of the gradient boosting algorithm known for its high speed and efficiency in both classification and regression problems .
11. TF-IDF Vectorization: Transformation of textual data into numeric vectors through the use of the TF-IDF score.
12. Sender ID: is the identification name of the entity who will appear to be the sender of an SMS text, and in the case of Real bank messages, it refers to the name of the bank.

13. Confidence Score: It is the numeric score generated by a machine learning model reflecting the confidence level of the predictions made; measured in percentage.
14. React Native: A free and open-source mobile app development framework introduced by Facebook that allows building native applications using JavaScript and React.
15. FastAPI: State-of-the-art web framework for creating Python-based APIs.
    Supabase: Backend as a service (BaaS) open-source framework that provides database, authentication, and storage capabilities on top of PostgreSQL.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter focuses on the review of the literature available in the area of the development of a bank alert fraud detection system using machine learning approach. This review shall enable the researcher to comprehend the achievements made by previous researchers within this area regarding methodology adopted, results obtained and shortcomings in their studies. Thus, through this study, the researcher intends to add value to this field of study by contributing to the existing literature.

The chapter is divided into five sections. First, section 2.2 presents the theoretical framework of this research study. Secondly, an extensive review of 20 related studies is provided in section 2.3. Thirdly, all related studies are summarized in a tabular form in section 2.4. Lastly, section 2.5 presents the research gap of this study.

2.2 Theoretical Background

2.2.1 Overview of Bank Alert Fraud in Nigeria

The bank alert fraud is another type of social engineering where the perpetrator sends fake SMS messages intended to resemble bank messages in order to dupe the recipient into believing that he has received payment for an imaginary transaction. In Nigeria, this kind of scam is common since bank alerts have been known to act as proof of payment in various daily business activities (Okonkwo et al., 2022). The tendency by Nigerian businessmen and street vendors to rely on bank alerts as proof of payment constitutes their weakness which the fraudsters have found ways to exploit.

The number of incidences of such scams has been alarming as the annual losses reported due to electronic fraud in Nigeria have increased each year according to NIBSS (2023). While there have been various public awareness campaigns about this issue by the Central Bank of Nigeria urging customers to be vigilant (CBN, 2022), it continues since there is no reliable method to verify these payments through automatic means.

2.2.2 SMS Classification and NLP

Classification of SMS refers to assigning each SMS to an existing pre-defined category. In this case, the categories include real, fake, or suspicious SMS. The common practice is to transform the SMS into numerical data and use NLP models to train the machine learning algorithms for SMS classification (Adetunji et al., 2023). TF-IDF is the most popular feature extraction technique for this task, where the numerical values will depend on how frequently the terms occur in the text belonging to a certain class. However, at the same time, the terms should occur less often in the overall set of SMS, giving the algorithm an opportunity to highlight more discriminative language (Makinde et al., 2024). The structural features such as the number of characters, the proportion of capital letters, URL usage, and numerical content can serve as additional indicators.

2.2.3 Machine Learning for Fraud Detection

Approaches using supervised machine learning have been the most popular methods used for solving fraud detection problems in literature. Some of these algorithms that include Logistic Regression, Random Forest, and XGBoost have all shown to be very successful in different text classification problems (Makinde et al., 2024). The logistic regression approach uses linear separation in high-dimensional spaces and is quite easy to interpret (Makinde et al., 2024). The random forest algorithm constructs multiple decision trees from bootstrapped datasets while selecting random features during training, making it resistant to overfitting (Makinde et al., 2024). Lastly, the XGBoost algorithm uses gradient boosting, which involves building multiple trees by improving upon previous tree mistakes and achieving state-of-the-art results (Shafiq et al., 2024).

2.2.4 Logistic Regression, Random Forest, and XGBoost

The Logistic Regression algorithm is another simple yet powerful algorithm that is often used for classification problems. It tries to identify the optimal linear separator between the classes. Though being among the most elementary approaches to machine learning, it has demonstrated excellent performance on text classification problems, particularly when used in conjunction with TF-IDF features. The advantages of the approach are high speed, interpretability, and computational efficiency on high-dimensional sparse data.

The Random Forest approach uses several decision trees trained during the training process and produces a final output through the process of majority voting among the predictions from individual trees. It is insensitive to noise, can effectively work with high-dimensional data sets, and is less prone to overfitting than a single decision tree.

The acronym XGBoost means Extreme Gradient Boosting; it is a gradient boosting method that creates decision trees sequentially, where each subsequent tree improves on the shortcomings of the former tree. This algorithm is known to be very fast and highly performant with a good combination of various types of features like combining textual TF-IDF features with numerical ones. XGBoost has always performed well in various researches concerning fraud detection.

2.2.5 Mobile Application Development for Security Systems

The use of machine learning algorithms in the form of mobile applications represents an active field of research in cybersecurity. React Native allows the implementation of cross-platform applications for both Android and iOS based on one single JavaScript code, which makes it convenient to use for security software aimed at wide audiences. The FastAPI package facilitates high-speed operations using Python and enables the provision of REST services and data validation. Supabase is a scalable database based on PostgreSQL and can be used for storing logs and correction records for the purpose of refining the model.

2.3 Review of Related Works

This section includes fifteen relevant studies conducted from 2020 to 2026 on SMS fraud detection, phishing classification, natural language processing for financial fraud detection, and security of mobile banking.

The researchers, Makkar and Kumar (2020), introduced a deep learning framework that can be used in IoT environments to detect web spam and phishing attacks. In their experiment, machine learning algorithms were used to study URLs, website features, and web traffic behavior with a view to distinguishing between malicious and non-malicious websites. They found out that the proposed machine learning algorithm was able to detect malicious websites with high accuracy rates exceeding 96%. It is important to note that the researchers mainly concentrated on web

phishing and URL detection. Also, the web features obtained from the experiment cannot be applied in SMS fraud detection.

Tingfeng et al. (2020) designed a scheme for SMS fraud detection in the sphere of mobile banking with the help of BiLSTM model and attention mechanism. Applying the framework to 50,000 Chinese bank messages resulted in an F1-score of 98.3%. In such a way, the method allows one to focus on the significant parts of each message. Nevertheless, it can be noted that there are several critical limitations: linguistic specificity (only Chinese banking vocabulary); annotated set of data and considerable expenses.

In this research by Zulkifli et al. (2021), the detection of phishing SMS was studied in the context of mobile banking of Malaysia using 3,000 samples labeled in relation to whether they were phishing or not. Among the algorithms used, the best performance was exhibited by Random Forest, whose accuracy was 95.8%, outperforming both Naive Bayes and Decision Tree. The importance of suspicious URLs, unfamiliar sender names, and urgency of the message is noted as discriminating features between the classes. Disadvantages in the study include cultural specificities, binary classification, and no three-class classification.

The paper by Bagui et al. (2021) examined the issue of phishing detection via emails, using a collection consisting of 18,000 samples. Textual and structural features, such as link presence and distribution of HTML tags, were combined to yield more than 98% precision rate for both XGBoost and Random Forest. The significance of adding structural features to the text-based set is demonstrated here. The drawback of the paper is its focus on emails, rather than on SMS, which differ greatly structurally.

A study was done on the detection of phishing through SMS for Indian mobile banking applications by Priya and Karthikeyan (2021). They extracted some structural features such as message length, number of digits, link count, uppercase percentage, and keyword frequency using 2000 messages. The algorithm used to achieve 94.2% accuracy was Random Forest. Some of the limitations for this research include its applicability to Indian banking, classification of data into two classes, and not comparing with XGBoost and logistic regression.

The mixed-method study carried out by Abayomi-Alli et al. (2022) involved 200 participants, who were customers of Nigerian banks, and an analysis of 500 fraudulent transactions conducted

by the Economic and Financial Crimes Commission (EFCC). It revealed that more than 40% of the participants had received one or more fake bank alert SMS, while the traders and market sellers appeared to be the most vulnerable groups. Although the study provided some background information for the current research, it does not offer a technical system for detecting phishers.

Okonkwo et al. (2022) focused on SMS bank alerts fraud in Nigeria, using 350 bank alert messages in the dataset with Naive Bayes classifiers and TF-IDF features. An accuracy rate of 82.4% was attained. The research represents the most relevant work related to Nigerian SMS but suffers from the small sample size, limited classification to binary, and lack of deployment in mobile apps and categorization of suspicious messages.

The article by Idrissi et al. (2023) designed multilingual SMS spam and phishing filter systems for three different languages, namely English, French, and Arabic. Using character-based n-grams and TF-IDF feature vectors with the Gradient Boosting algorithm on 8,000 samples, the average accuracy level reached 96.1%. This study proved the capability of using language-independent structural features. However, the limitations of the study include lack of African languages or Nigerian Pidgin English.

Adetunji et al. (2023) used machine learning algorithms on NLP-based approaches to classify fraudulent messages in Nigeria, gathering 800 financial messages from contacts on social media. After performing vectorization based on TF-IDF, Logistic Regression reached 91.3% accuracy, while Random Forest yielded 93.7% accuracy. This paper is Nigeria-related and applicable but suffers from the lack of size of dataset, a binary classification, and not being mobile-deployed.

In the paper, Makinde et al. (2024) created an ML-based mobile banking fraud detector for West Africa that relied on 2,500 messages from customers of Nigeria and Ghana. Using XGBoost, with TF-IDF and numeric data, they achieved 95.2% accuracy and a 94.8% F1-score. The paper is highly relevant but suffers from the lack of country uniformity of the dataset, a binary classification approach, and lack of mobile deployment.

Shafiq et al. (2024) evaluated Random Forest, XGBoost, SVM, and BERT embeddings in relation to the identification of SMS-based financial fraud in Pakistan with 4,200 banking alerts. XGBoost performed with the best accuracy of 96.7%. A key finding in this paper is the comparable performance of TF-IDF classifiers compared to BERT classifiers, but much more computationally efficient. The limitation of this study is the specific case of Pakistani banks and the use of only two classes for classification.

Combining metadata features with text TF-IDF was tested again by Bagui et al. (2021) in the area of SMS classifications, confirming that it always results in an improved classification accuracy. It also confirms the feature engineering approach in the current study.

A fraud detection system based on SMS received from Nigerian mobile banking services via a dataset of 1,200 alerts collected in Lagos and Abuja was created by Nwosu et al. (2025) using TF-IDF vectorization with crafted features that included the sender ID encoding and suspicious keyword detection. Random Forest outperformed Logistic Regression with an accuracy of 94.1% versus 93.5%. This paper represents the study most relevant to the present research in terms of the Nigerian context but is hindered by its binary classification problem, lack of mobile app, and user feedback.

As an extension of their multilingual classification framework, Idrissi et al. (2023) showed that character-based features are highly generalizable across languages in phishing detection, which supports language-invariant feature engineering as a promising approach when there is no significant labeled dataset available for a particular language. This study highlights the validity of the structured feature engineering used in this research.

Adewale et al. (2026) proposed an SMS fraud detection model for West African mobile banking, which used three labels of real, fraud, and unsure in classifying 2,000 SMSs from four West African nations. An XGBoost model combined with TF-IDF and structural information gained 96.3% precision rate and F1-score of 96.1%. It is the most relevant classification algorithm to the current study, and its weaknesses include non-Nigerian data set, web application only, and lack of user feedback system.

According to Mahmud et al. (2024), a hybrid deep learning model was proposed for smishing attack identification by analyzing SMS datasets available through mobile communication

applications. The researchers integrated CNNs and LSTM networks to enhance the smishing pattern detection process. Experimental results revealed that the hybrid approach attained an accuracy of 97.1%, outperforming machine learning techniques in detection efficiency. However, the research was more concerned with generic smishing attack detection and did not consider bank transaction messages as the area of focus. Further, the deep learning-based system was highly resource-intensive and was unable to be applied as a lightweight mobile application.

Alshinwan et al. (2025) proposed a smishing detection framework based on meta-heuristic optimization algorithms combined with machine learning techniques. The technique adopted optimized feature selection techniques in order to enhance the SMS phishing detection process. The method produced a high level of detection accuracy of greater than 96%, and handled data imbalance effectively. While producing good results, the research only concentrated on general SMS phishing and not on banking transaction alerts. Binary classification was employed and there was no suspicious alert class.

Ramanujam et al. (2025) conducted a comparative study on machine learning algorithms for smishing detection using SMS phishing datasets. The methods analyzed in the research were Logistic Regression, Support Vector Machine (SVM), and Random Forest in the detection of phishing messages. According to the results obtained, the highest accuracy was found to be more than 95% using the Random Forest algorithm. This research demonstrates the capability of machine learning in SMS fraud detection but concentrates only on phishing messages and credential stealing messages. Moreover, there is no mention of mobile device utilization in Nigeria.

Taylor and Robert (2025) developed a machine learning-based fraudulent SMS detection system for Chichewa language messages using Random Forest and Logistic Regression algorithms. The authors developed localized datasets, and proved that language-specific preprocessing greatly boosted the effectiveness of fraud detection. The accuracy for both models exceeded 96% using localized datasets. The study did not target SMS alert systems for financial transactions but aimed at detecting fraudulent messages in general. Moreover, the binary classification technique was used without implementing a mobile system.

Adewale et al. (2026) carried out a systematic review of SMS phishing attacks and defense techniques, examining recent advances in machine learning and deep learning approaches for

smishing detection. The review discovered that the most common methods employed for detecting SMS fraud were TF-IDF, Random Forest, XGBoost, and BERT-based algorithms. The research highlighted the increasing necessity for lightweight, understandable, and mobile fraud detection techniques in developing countries. Nonetheless, this research only conducted a literature review and did not apply a fraud detection technique in Nigeria's banking environment.

2.4 Summary of Literature Review

Table 2.1 below summarizes the fifteen related works reviewed in this chapter, showing the authors, what they did, the technique used, results achieved, and limitations identified.

Table 2.1: Summary of Related Works

S/N	Author(s) / Year	Work Done	Technique / Method	Results	Limitations
1	**Makkar & Kumar (2020)**	Deep learning framework for detecting web spam and phishing attacks in IoT environments using URL and webpage features.	Deep Learning on URL and webpage features	Accuracy above 96%	Focused on web phishing rather than SMS alerts; URL-based features unsuitable for plain-text bank SMS fraud detection; no mobile deployment.

2	**Tingfeng et al. (2020)**	SMS fraud detection for mobile banking using deep learning with attention mechanism on a large Chinese banking corpus of 50,000+ records.	BiLSTM(Bidirectional Long Short-Term Memory) + Attention Mechanism	F1-score: 98.3%	Specific to Chinese banking vocabulary; requires large annotated corpus; high computational cost; not replicable in Nigerian context.
3	**Zulkifli et al. (2021)**	SMS phishing detection in Malaysian mobile banking context using 3,000 labeled messages with feature analysis.	Random Forest with TF-IDF	Random Forest: 95.8% accuracy	Malaysian context; binary classification only; no three-class scheme for ambiguous messages; no mobile deployment.
4	**Bagui et al. (2021)**	Email phishing detection using 18,000+ phishing and legitimate emails, combining text content with structural features.	XGBoost, Random Forest on email + structural features	XGBoost & Random Forest: 98%+ precision	Email domain only; HTML structural features do not generalize to plain-text SMS; not applicable to bank alerts.

5	**Priya & Karthikeyan (2021)**	SMS phishing detection for Indian mobile banking using structural features: message length, digit count, link count, uppercase %, keyword frequency on 2,000 messages.	Random Forest + structural features	94.2% accuracy	Indian banking context; binary classification; no mobile application.
6	**Abayomi-Alli et al. (2022)**	Mixed-methods investigation into cybersecurity threats in Nigerian financial institutions; survey of 200 bank customers + analysis of 500 EFCC fraud cases.	Survey + EFCC case analysis (descriptive)	40%+ of respondents received a fake bank alert; traders identified as most victimized	Descriptive study only; no technical contribution.

7	**Okonkwo et al. (2022)**	First Nigerian study addressing SMS bank alert fraud; collected 350 messages and applied Naive Bayes with TF-IDF features.	Naive Bayes with TF-IDF	82.4% accuracy	Small dataset; binary classification only; no mobile deployment; no suspicious category.
8	**Idrissi et al. (2023)**	Multilingual SMS spam and phishing detection for English, French, and Arabic messages using character-level n-gram features.	Gradient Boosting + character-level n-gram TF-IDF (8,000 messages)	Average accuracy: 96.1%	No African language coverage; no deployment as usable application; binary classification only.
9	**Adetunji et al. (2023)**	NLP-based detection of fraudulent financial messages in Nigeria using 800 messages collected from social media contacts.	Logistic Regression, Random Forest with TF-IDF	Logistic Regression: 91.3%; Random Forest: 93.7% accuracy	Small dataset; binary classification; no mobile deployment.

10	**Makinde et al. (2024)**	Mobile banking fraud detection for West Africa using 2,500 SMS alerts from Nigerian and Ghanaian users combining TF-IDF with numeric features.	XGBoost + TF-IDF + numeric features	95.2% accuracy; 94.8% F1-score	Mixed-country dataset; binary classification; no mobile deployment.
11	**Shafiq et al. (2024)**	Comparative study of traditional ML vs BERT-based embeddings for SMS financial fraud in Pakistan using 4,200 banking alerts.	XGBoost, Random Forest, SVM, BERT embeddings	XGBoost: 96.7% accuracy	Pakistani banking context; binary classification only.

12	**Bagui et al. (2021b)**	Extended structural feature combination study showing metadata + TF-IDF text consistently improves classification across domains.	Structural + text TF-IDF features, RF	98%+ precision	Email context; findings applied but not directly transferable to SMS.
13	**Nwosu et al. (2025)**	Fraud detection for Nigerian mobile banking SMS using 1,200 alerts from Lagos and Abuja with hand-crafted features including sender ID encoding.	Logistic Regression, Random Forest, XGBoost + TF-IDF + sender ID + keyword features	RF: 94.1%; LR: 93.5% accuracy	Binary classification; no mobile application; no suspicious category.

14	**Idrissi et al. (2023)**	Multilingual extension showing character-level features generalize across languages.	Multilingual character-level TF-IDF	Average accuracy: 96.1%	No African language coverage; findings used to inform feature engineering strategy only.
15	**Adewale et al. (2026)**	SMS fraud detection for West African mobile banking using a three-class scheme (real, fraudulent, uncertain) across 2,000 messages.	XGBoost + TF-IDF + structural features, 3-class	96.3% accuracy; 96.1% F1-score	Not Nigeria-specific; web-only deployment.
16	**Mahmud et al. (2024)**	Hybrid deep learning approach for detecting smishing attacks from SMS datasets.	CNN + LSTM hybrid deep learning	97.1% accuracy	Generic smishing only; not bank-alert specific; computationally expensive; no lightweight mobile deployment.

17	**Alshinwan et al. (2025)**	Smishing detection using optimized feature selection and machine learning.	Meta-heuristic optimization + ML	Above 96% accuracy	Binary classification only; not focused on bank alerts; no suspicious category.
18	**Ramanujam et al. (2025)**	Comparative analysis of ML algorithms for SMS phishing detection.	Logistic Regression, SVM, Random Forest	Random Forest achieved 95%+ accuracy	Focused on phishing links; no Nigerian dataset; no mobile deployment.
19	**Taylor & Robert (2025)**	Fraudulent SMS detection using localized Chichewa datasets.	Random Forest, Logistic Regression	Accuracy above 96%	General SMS fraud only; binary classification; no mobile application.
20	**Adewale et al. (2026)**	Systematic review of SMS phishing detection approaches and datasets.	Review of TF-IDF, RF, XGBoost, BERT techniques	Identified high-performing ML/DL approaches	Review study only; no implemented system; not Nigeria-specific.

2.5 Research Gap

The literature review reveals five critical gaps in existing research:

- Most studies focus on email or web phishing, whose methods do not apply to bank SMS alert fraud.

- Previous research relies heavily on foreign datasets that don't include Nigerian specifics such as local bank names, sender IDs, Naira currency e.t.c
- Existing models strictly predict "Real" or "Fake," ignoring the crucial "Suspicious" category for ambiguous messages.
- There are no practical, mobile-based applications built for everyday Nigerian users to use in real-time.
- Previous models lack a way for users to report incorrect predictions, preventing the AI from learning new scam tactics.

This research project successfully addresses all the five research gaps. It creates a three-class labeled dataset specifically developed for Nigeria, comprising 2,700 bank SMS alerts; trains and tests three different classifiers; selects the most accurate classifier (Logistic Regression, with an accuracy of 97.6%); integrates the selected model into a mobile application developed using React Native and a web application created using React and FastAPI framework.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

This chapter discusses the analysis and design of the bank alert fraud detection system. First, there is an analysis of the current status and management of the alert fraud in banks, then, the objectives and requirements of the proposed system. Subsequently, there is the methodology of the design, the process of data collection, system modeling using UML diagrams, and the detailed design of the components of the system such as its database, I/O interface, and system architecture.

The methodology used in this research for designing the bank alert fraud detection system is OOAD. OOAD takes into consideration the object and interaction of objects. The methodology used for developing machine learning is CRISP-DM, which provides guidelines on data-driven projects' planning and execution (Makinde et al., 2024).

3.1 System Analysis

3.1.1 Analysis of the Existing System

Currently, there is no tool for use by mobile banking users in Nigeria that is designed to help automatically authenticate the genuineness of any received bank alert message. At present, the manner in which users deal with the issue is by being vigilant, calling customer care services at their respective banks, or through awareness campaigns (Okonkwo et al., 2022).

The typical reaction when a user receives a bank alert and seeks to authenticate its genuineness would be to look into the user notification bar on their phones for the existence of such an alert from the bank using its designated SMS sender name. Sometimes, users will even open their bank application to check their balances or transactions, especially where the amount and details align with the alert message received.

Although such techniques are effective in certain circumstances, they are ineffective in many others. While a seller is transacting in a market environment, it is impossible for him or her to launch a banking application, get it to load up and then double-check a transaction within mere seconds. It is under the pressure of the situation that the majority of sellers simply view an alert

on the buyer's device and consider it to be evidence of the transaction, precisely what scammers aim to achieve.

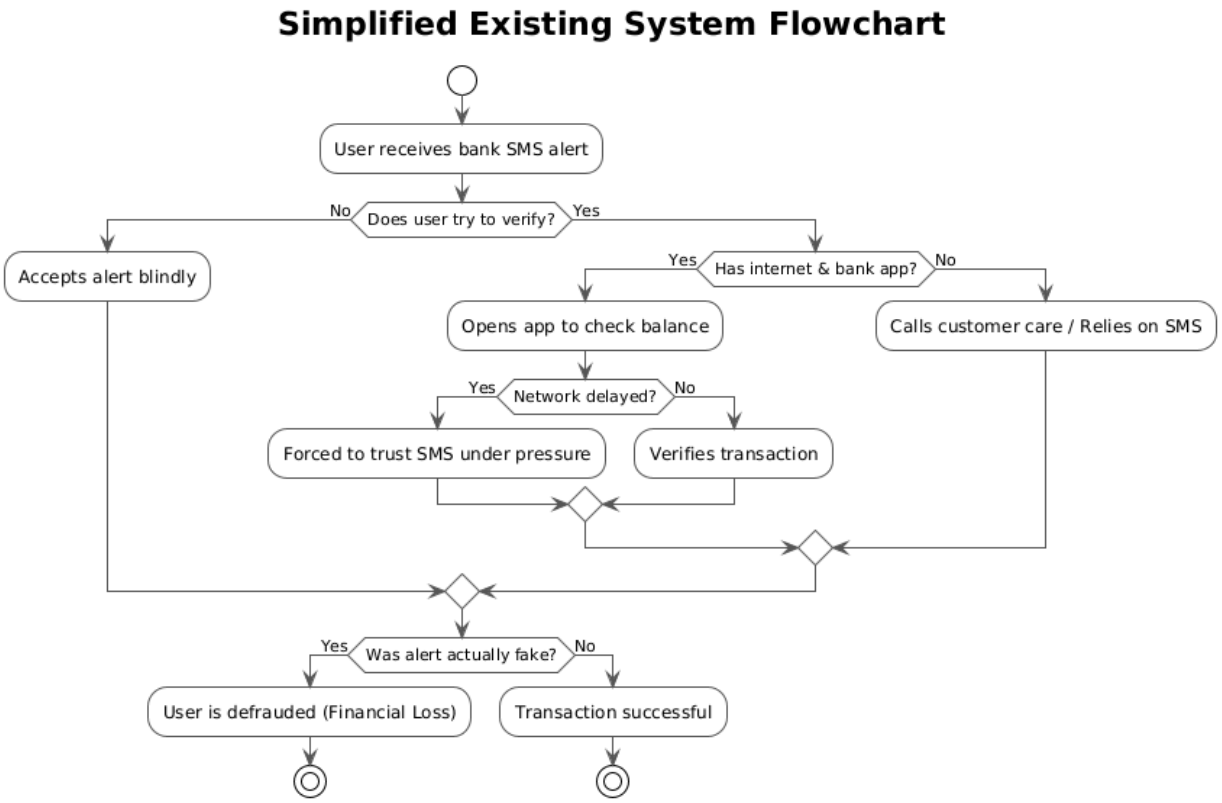

Figure 3.1: Flowchart of the Existing System

Moreover, not every consumer uses their phone with Internet connection during a transaction. Banking applications take quite a while to show recent transactions, meaning that a transaction notification just sent might not be shown immediately. Many elderly people and citizens in remote areas do not have access to banking applications and must depend solely on SMS notifications.

The Central Bank of Nigeria and the Nigerian Communications Commission sometimes issue advisories regarding the same; however, these are merely passive measures as they fail to provide users with an instrument for verifying the alerts instantly (CBN, 2022).

3.1.2 Limitations of the Existing System

Based on the analysis above, the following limitations were identified in the existing system:

- Checking the phone notification bar is unreliable because fraudsters can spoof SMS sender names to look exactly like a real bank, so a matching sender name does not guarantee the message is genuine.
- Opening a banking app to verify a transaction is impractical in fast-paced commercial settings, as a seller cannot afford to wait for an app to load and cross-check details before deciding whether to release goods to a buyer.
- There are places where a customer may not have access to internet services at that particular moment.
- Banking apps do not always reflect new transactions immediately, meaning a user may check their history and not see the transaction even if it is real, causing confusion.
- Many Nigerians, especially older people and those in rural areas, do not use mobile banking apps at all, leaving them with no option beyond reading the raw SMS message.
- CBN and NCC awareness campaigns are passive and do not provide any real-time or automated means of verifying a specific alert message.
- None of the existing approaches can classify an alert as suspicious. i.e a category that covers messages that are neither clearly real nor clearly fake but still carry enough uncertainty to warrant caution.

3.1.3 Analysis of the Proposed System

The proposed system tackles issues with the current approach in having a mobile application where anyone can take the text from the bank alert message and get the result through machine learning classification right away. The classes of the machine learning model for the classification are divided into real, fake, and suspicious.

The system is backed by a trained machine learning model served through a FastAPI backend API. A Supabase database stores user feedback reports. The feedback mechanism allows users to report incorrect predictions, which are stored for future model retraining.

3.2 System Requirements Specification

3.2.1 Functional Requirements

Table 3.1 presents the functional requirements of the bank alert fraud detection system

Table 3.1: Functional Requirements

ID	Requirement	Description
FR 1	Alert Classification	Accept a bank SMS alert as text input and classify it as Real, Fake, or Suspicious.
FR 2	Confidence Score	Display a confidence score alongside each classification result.
FR 3	Alert History	Maintain a history of all classified alerts accessible to the user.
FR 4	Feedback Submission	Allow users to report incorrect classifications via a 3-dot menu on history items.
FR 5	Feedback Storage	Store user feedback corrections in the feedback_reports database table.
FR 6	Manual Input	Allow users to manually paste alert messages into the text input field.
FR 7	History Display	Each history item shall show message text, label, confidence score, and timestamp.

3.2.2 Non-Functional Requirements

Table 3.2 presents the non-functional requirements of the system.

Table 3.2: Non-Functional Requirements

ID	Requirement	Description
NFR1	Performance	Return a classification result within 3 seconds under normal network conditions.
NFR2	Usability	Interface shall be accessible to users with minimal technical knowledge.
NFR3	Reliability	The backend API shall achieve at least 99% uptime during active periods.
NFR4	Security	User-submitted messages shall not be shared and shall be stored securely.
NFR5	Scalability	The backend shall handle multiple concurrent requests without degradation.
NFR6	Maintainability	The codebase shall be modular to allow model updates without major changes.

3.3 System Design Methodology

3.3.1 Object-Oriented Analysis and Design (OOAD)

For the design of this system, OOAD was selected as the system design technique. The main benefit of applying OOAD is that the system will be structured in an object-oriented manner whereby the system components will be objects that consist of attributes and functions, making the system modular, reusable, and providing documentation of how the components communicate with each other (Pressman & Maxim, 2020). The fraud detection system utilizing bank alerts consists of several system components, such as the mobile frontend, web frontend, FastAPI backend, machine learning training pipeline, and the Supabase database. These components have clearly defined tasks within the system.

3.3.2 CRISP-DM for Machine Learning Development

CRISP-DM process model refers to a process model that provides a structured yet flexible framework in planning and implementing a data mining and machine learning project. This framework was introduced by Wirth and Hipp in 2000 as a cross-industry non-domain specific framework. The CRISP-DM framework entails six key steps in developing a machine learning

project, namely, business understanding, data understanding, data preparation, modelling, evaluation and deployment. One of the greatest benefits associated with the use of the CRISP-DM process model is the ability to be flexible as one moves back and forth from the different phases based on the knowledge acquired. It is very appropriate when dealing with projects whose data keeps changing.
The CRISP-DM process model was used in guiding the machine learning part of the project. The six phases were executed as follows:

The definition of the business problem was made based on Business Understanding, where the type of problem was established as a three-class classification task. Target labels were Real, Fake, and Suspicious. Key questions to address were what language/structural indicators differentiate real and fake bank alerts and how alerts that have some indicators of being fake yet do not have all of them be handled. In this regard, the decision to use three classes as opposed to two came as a result of this very stage because many authentic Nigerian bank alerts had characteristics that could confuse a binary classifier.

The process of identifying the characteristics of the three types of SMS messages was carried out in the Data Understanding stage. For example, data about the distribution of message length, digits in messages, URLs, sender IDs formats, and keywords was extracted for each type. This exploratory step helped identify features that will prove useful in preparing data in the next step and the class imbalance.

Data Preparation was the most intensive step of the project, which included text normalization and cleaning, synthesis of artificial data for class imbalance problem, stratified train-test splitting with a proportion of 80/20 percentiles, TF-IDF vectorization that was done separately for the message body (resulting in 11,225 features) and sender ID attribute (102 features), and finally, the creation of twelve manually designed structured features measuring various characteristics of each message. The output of this step included concatenation of three vectorizations resulting in an 11,339-dimensional input for each data point.

Modelling utilized three supervised classifiers: Logistic Regression, Random Forest, and XGBoost, which were trained on pre-processed data. The same hyperparameters' configuration was used for all models in order to conduct a fair comparison. All models were trained on a

stratified 80-percentile training part of data and tested on 20 percentiles. No part of the test data was used in the training or tuning process.

The comparison of the results obtained by all three models was carried out using the following classification metrics which are traditionally used: accuracy, precision, recall, and F1 score, with each measure computed on a per-class basis to establish whether the proposed classifier exhibited different classification results when processing data related to the class Suspicious – the one that was believed to be the hardest for classification. The confusion matrices were also employed for the analysis of misclassification cases.

Model deployment entailed embedding the selected model into the FastAPI application's backend, followed by its serialisation into REST API endpoint format. Both mobile frontend using React Native technology and web frontend based on React library were linked to the backend, thus allowing end users to upload SMS messages and receive real-time classification with their corresponding confidence scores. The latter task also included lightweight model prediction validation on the deployed model against off-line test cases.

3.4 Data Collection and Preparation

3.4.1 Data Collection

A total of 600 SMS text messages used as bank alerts were collected manually from different sources including contacts, friends, family members, and publicly available texts on social networking sites. The collected messages were classified into three categories, where there are legitimate bank alert messages, phishing or fake messages, and suspicious messages.
These messages contained information about the mobile banking trends of some of the popular banks in Nigeria, for example, GTBank, UBA, Access Bank, Zenith Bank, and FirstBank.

3.4.2 Preprocessing and Augmentation

Prior to model training, the following preprocessing methods were used: Text normalization (fixing line breaks, extra spaces, punctuation errors, while retaining capitalization since capital ratio is a classification feature); Standardizing labels to three predefined values; Standardizing the Sender ID field to six predetermined values (GTBank, UBA, AccessBank, ZenithBank,

FirstBank, and Unknown), where unknown Sender IDs are categorized as Unknown; and Removal of exact duplicates.

The training data was expanded from 600 to 2,700 using controlled text variations (paraphrasing, synonyms, and formatting alterations) performed on each class separately to maintain the consistency of labels (Adetunji et al., 2023). In each class, 900 examples were created. Upon dividing the dataset into an 80% training set and a 20% testing set, there were 2,160 and 540 samples in the training and testing sets, respectively.

3.5 Dataset Description and Preprocessing

In this section, information about the dataset utilized during the process of model training and details regarding all preprocessing tasks conducted prior to using it in machine learning algorithms are provided.

3.5.1 Dataset Overview

In this research, the dataset used consisted of Nigerian bank SMS alert messages. A total of 600 messages were manually collected from personal contacts, friends, family members, and publicly shared examples on social media platforms. The messages were organized into three classes such as Real, fake, and suspicious, with 200 messages in each class. The dataset was then expanded to 2,700 samples through data augmentation, giving 900 samples per class. The final distribution is shown in Table 3.4 below.

Table 3.3: Dataset Class Distribution

Label	Number of Samples	Percentage
Real	900	33.3%
Fake	900	33.3%
Suspicious	900	33.3%
Total	2,700	100%

3.5.2 Text Preprocessing Steps

Before the dataset was used for model training, the following preprocessing steps were applied:

1. Text Normalization: Broken line breaks, double spaces, and inconsistent punctuation were removed from all messages. The original casing of each message was kept because the uppercase ratio is one of the classification features.
2. Label Standardization: All label values were checked and made consistent using three fixed values such as Real, fake, and suspicious.
3. Sender ID Standardization: All sender ID values were mapped to six standard values: GTBank, UBA, AccessBank, ZenithBank, FirstBank, and Unknown. Phone numbers and unrecognized values were all assigned to Unknown.
4. Duplicate Removal: Exact duplicate messages were identified and removed after normalization.
5. Train-Test Split: The dataset was split into 80% training data (2,160 samples) and 20% test data (540 samples) using stratified sampling so that all three classes are represented equally in both sets.

3.5.3 Feature Engineering

Twelve structured features were extracted from each message to complement TF-IDF text features. These features capture structural and linguistic patterns that vocabulary alone cannot represent. Table 3.5 describes each feature.

Table 3.4: Feature Engineering Summary — 12 Structured Features

Feature	Type	Description
message_length	Numeric	Total character count of the message.
word_count	Numeric	Total number of words in the message.
digit_count	Numeric	Number of digit characters; real alerts typically contain specific amounts and references.
uppercase_ratio	Float (0–1)	Proportion of uppercase characters; fake alerts often use all-caps for urgency.
punctuation_count	Numeric	Number of punctuation characters in the message.
has_url	Binary (0/1)	Whether the message contains a URL or suspicious link pattern.
has_amount	Binary (0/1)	Whether the message contains a naira amount (e.g., N5,000 or NGN).
has_account_reference	Binary (0/1)	Whether the message includes an account number or transaction reference (e.g., TXN123456).
suspicious_term_count	Numeric	Count of high-risk words: 'claim', 'verify', 'click', 'block', 'prize', 'urgent'.
mentions_bank	Binary (0/1)	Whether the message explicitly mentions a recognized Nigerian bank name.
sender_is_numeric	Binary (0/1)	Whether the sender ID is a phone number — a strong fraud indicator.
sender_length	Numeric	Character length of the sender ID; legitimate bank IDs tend to be short and consistent.

The final feature matrix for each message was assembled by combining three components: the SMS text TF-IDF vector (11,225 features), the sender ID TF-IDF vector (102 features), and the twelve structured features described above, giving a total of 11,339 features per sample.

To validate the relevance of the twelve structured features, Figure 3.2 presents the feature importance scores extracted from all three trained models to evaluate the impact of the custom feature engineering. For the Logistic Regression classifier, importance was derived from the mean absolute coefficient values averaged across the three output classes. For the ensemble

models, Random Forest and XGBoost, the built-in feature importance metrics based on impurity reduction and information gain were utilized, respectively. Despite the different mathematical foundations of these algorithms, the results were highly consistent across all three models.

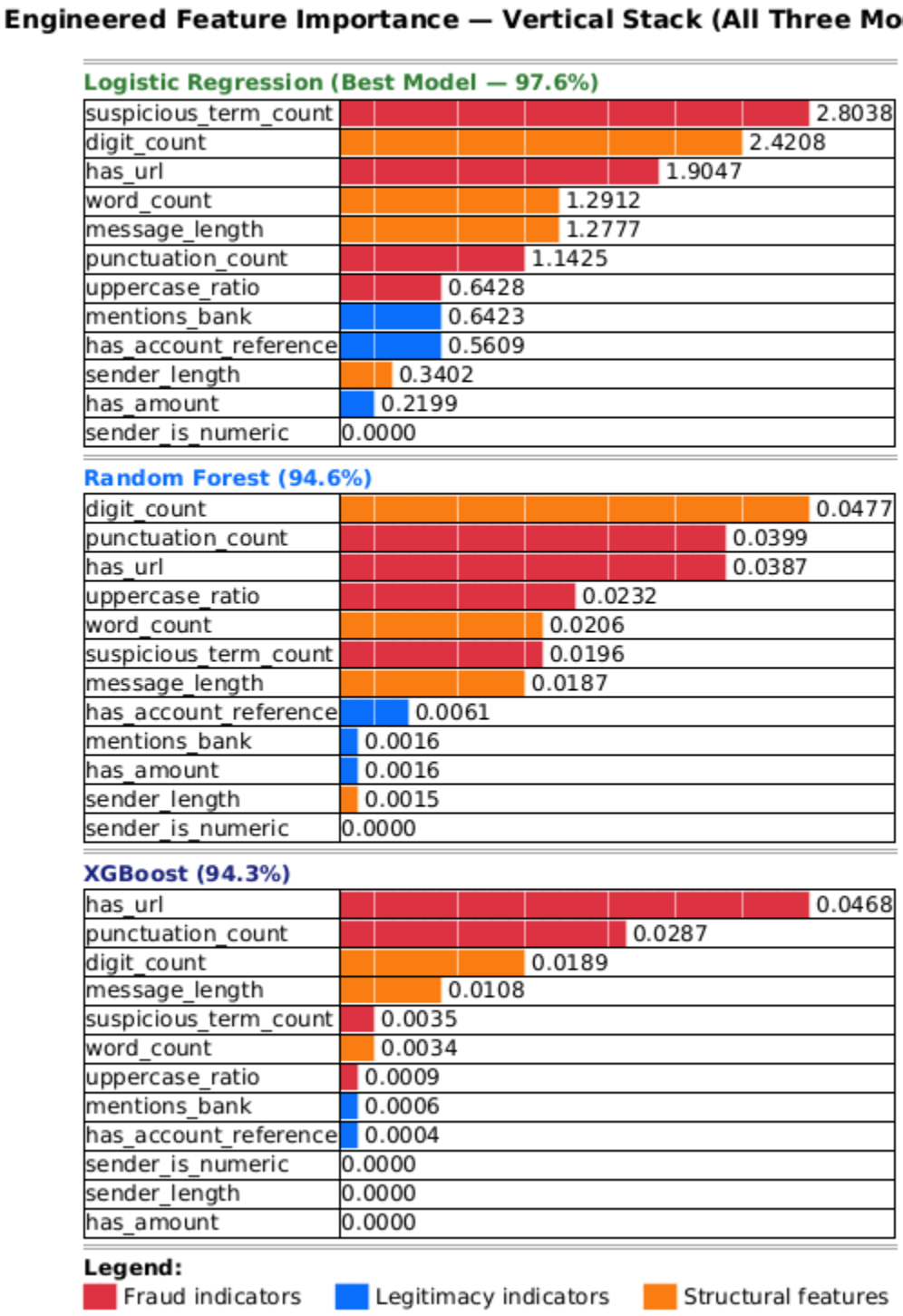

Figure 3.2: Engineered Feature Importance — All Three Models

Specifically, engineered structural features such as the presence of URLs, digit count, and suspicious term count consistently ranked as the most predictive indicators. This cross-model alignment mathematically confirms that the manual feature engineering contributed meaningfully to the system's overall classification accuracy.

3.6 System Modeling Using UML

3.6.1 Use Case Diagram

Figure 3.3 depicts a use case diagram illustrating the interactions between two primary actors: the User and the Admin, with the system.

The User actor interacts directly with three major use cases: Submit Bank Alert, View Alert History, and Report Incorrect Classification. The Admin actor manages the backend operations through two distinct use cases: Review Feedback Reports and Manage Model Data.

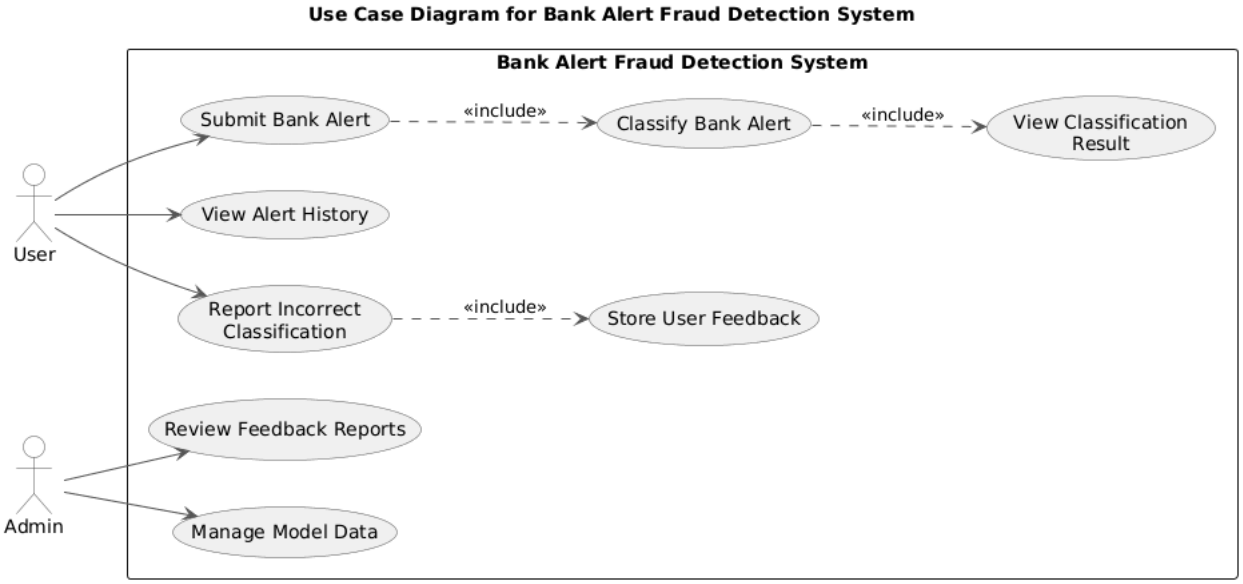

Figure 3.3: Use Case Diagram

3.6.2 Activity Diagram

The Activity Diagram in Figure 3.3 shows the flow of activities when a user submits a bank alert for classification. The user opens the application and pastes the alert text into the input field.

Upon tapping the Classify button, the message is sent to the FastAPI backend. The backend preprocesses the text, extracts features, passes the data through the trained model, and returns a prediction label and confidence score. The result is displayed to the user. If the user believes the result is incorrect, they can use the feedback option to submit a correction.

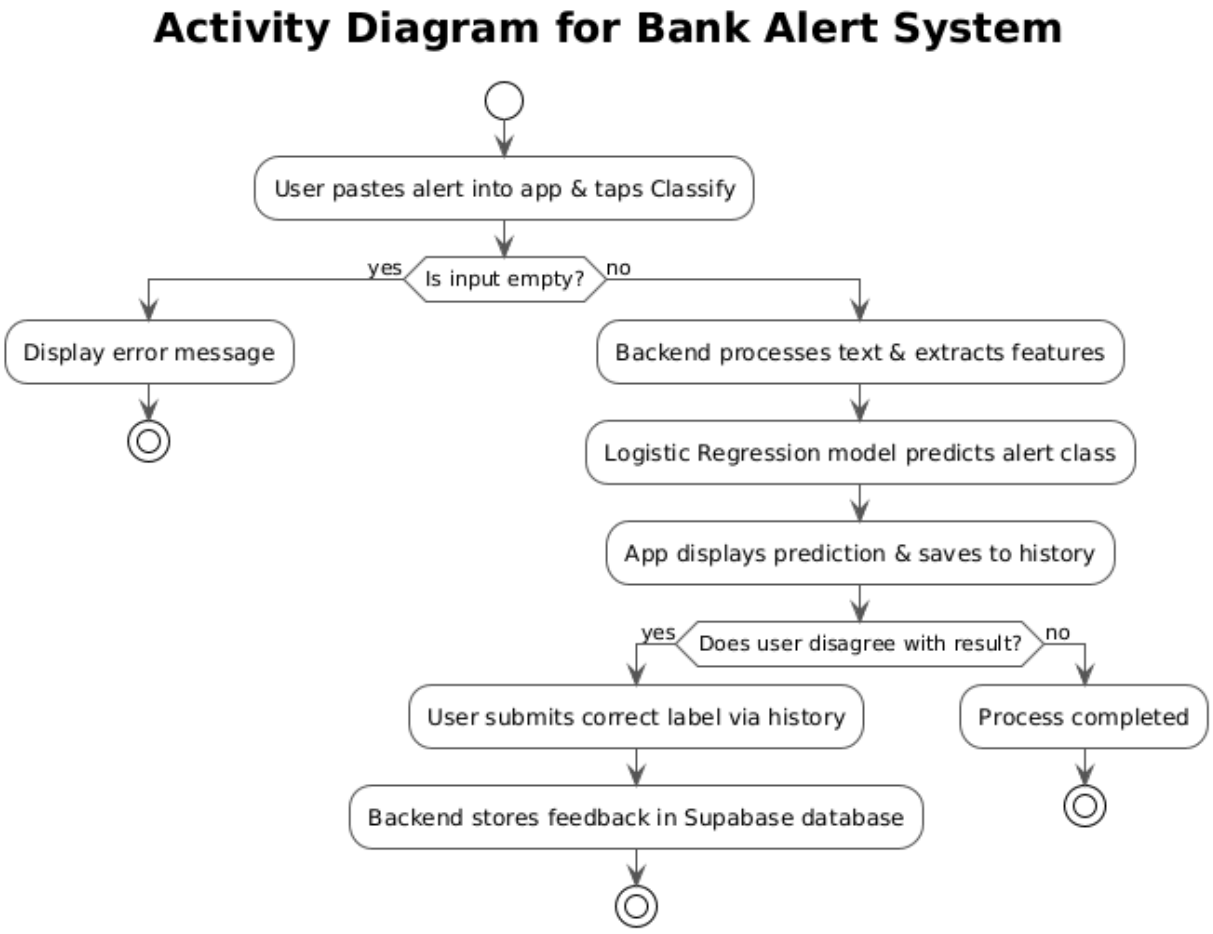

Figure 3.4: Activity Diagram

3.6.3 Sequence Diagram

Figure 3.4 represents the sequence diagram of the bank alert fraud detection system. The sequence diagram highlights the interaction between the components of the system during classification of the bank alert message submitted by the user.

The process begins when the User pastes a bank alert message into the Mobile Application and clicks the Classify button. The Mobile Application sends a POST request to the FastAPI Backend, passing the message text and sender ID as the request body. The FastAPI Backend

receives the request and forwards the message to the Feature Extractor, which applies TF-IDF vectorization and extracts the twelve structured features. The Feature Extractor returns the combined feature matrix to the FastAPI Backend.

The predicted label is received back through the JSON object created by the FastAPI Backend from the ML Model after running the classifier pipeline, which also outputs the confidence score and class probabilities. The JSON response is returned from the FastAPI Backend to the Mobile Application, which displays the output of the predictor along with its confidence score.

In case the predicted label does not match what the user feels should be the label for that particular message, then they have the option of selecting the correct label through the 3-dot menu in the Alert History Screen of the Mobile Application. The Mobile Application creates a POST feedback API request and sends it to the FastAPI Backend. The FastAPI Backend saves this in the Supabase Database and returns a success message to the Mobile Application.

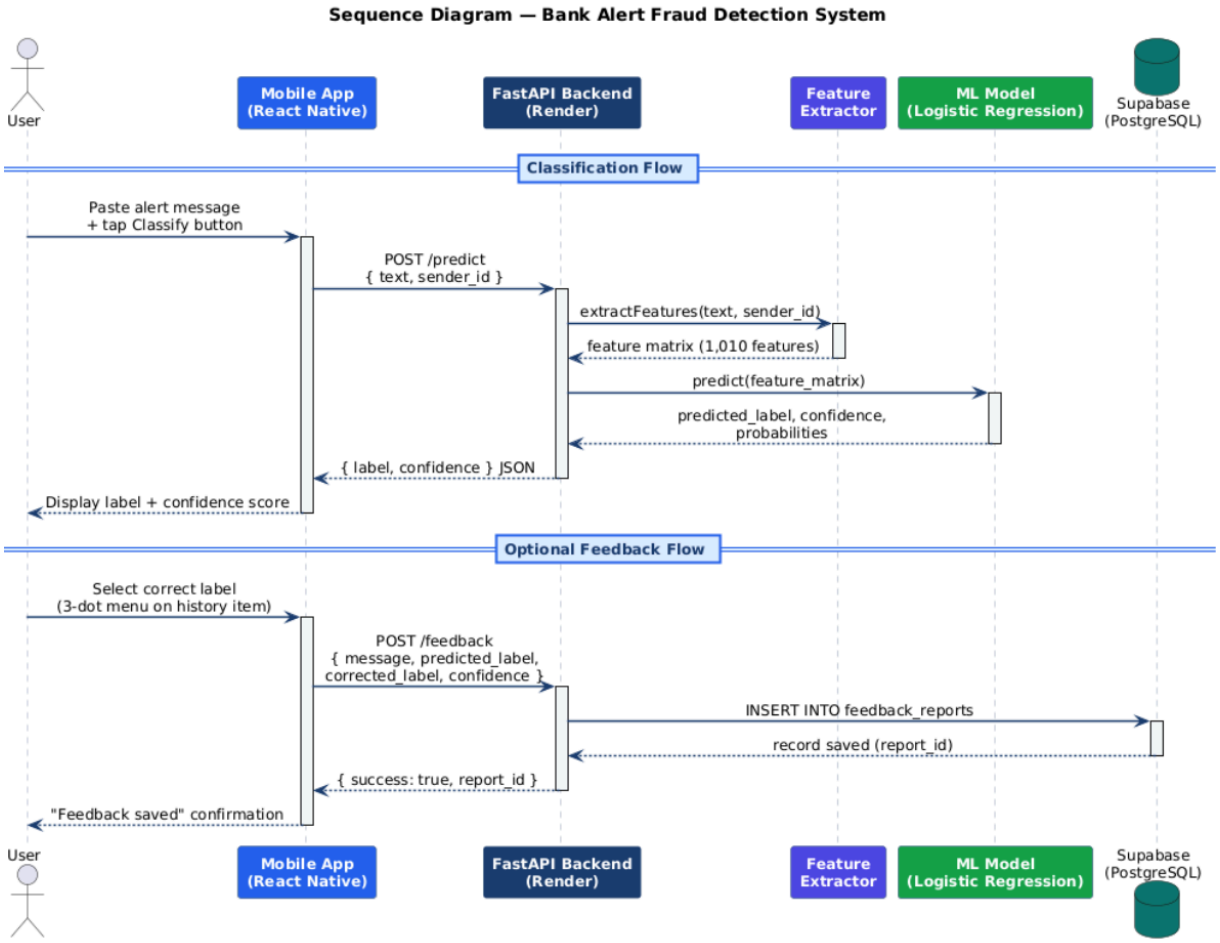

Figure 3.5: Sequence Diagram

3.6.4 Class Diagram

Figure 3.4 shows the class diagram representing the classes within the fraud detection system, including the methods of each class and relations between them. MobileApp is the core of this application that provides UI interaction and communication with APIController. Moreover, it creates AlertMessage objects that represent the user's input and creates HistoryItem objects that represent previous predictions made. APIController works as an intermediate layer that receives classification requests from mobile app and forwards user's feedback to appropriate parts of backend logic. Three types of classifiers (LogisticRegressionClassifier, RandomForestClassifier, and XGBoostClassifier) have been developed using BaseClassifier that includes general methods. They process the data provided by the FeatureExtractor by combining features into a matrix and providing PredictionResult consisting of prediction label and probability.

If the user makes corrections to previous predictions on the history screen, then FeedbackReport object is created that includes the original message, the mistake of the model, and the user's correction.

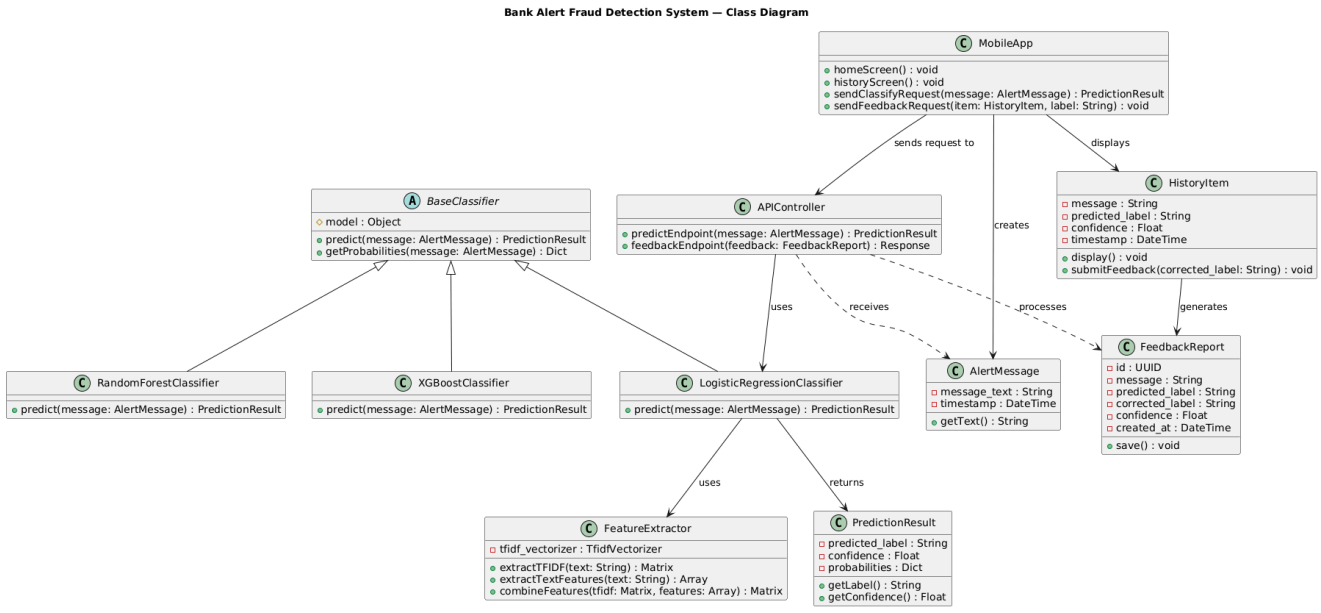

Figure 3.6: Class Diagram

3.6.5 Model Design Architecture

Figure 3.7 above demonstrates the unified machine learning classification pipeline that is common to all three classifiers used in this study. There are five consecutive stages in the pipeline. In Stage 1 (Input), the user inputs an unprocessed SMS notification from the Nigerian bank. In Stage 2 (Preprocessing), the text of the message undergoes lowercasing, normalization, and removal of URL tokens. In Stage 3 (Feature Extraction), three feature vectors are computed in parallel, including the TF-IDF representation of the text body (11,225 features based on unigrams, bigrams, and sublinear scaling), the TF-IDF vector for the sender ID (102 features), and 12 structured features (message length, proportion of uppercase letters, presence of URLs, proportion of digits, and count of suspicious keywords). These feature vectors are then concatenated into one feature vector with 11,339 dimensions for each message. In Stage 4 (Classification), this feature vector is used to make predictions using each classifier individually. For the fifth stage (Output), each model generates the predicted output in form of class labels (Real, Fake, and Suspicious) with their associated probability score. In the second part of the flow chart below, each model's pipeline is presented with the unique setup of the classification process. The classifier logistic regression utilizes the multinomial solver lbfgs with C=1.0 to achieve accuracy of 97.6%, random forest classifier employs aggregation of 200 bootstrap trees using voting strategy with 94.6% accuracy, while XGBoost employs 200 trees using gradient boosting at a learning rate of 0.1 with an accuracy of 94.3%.

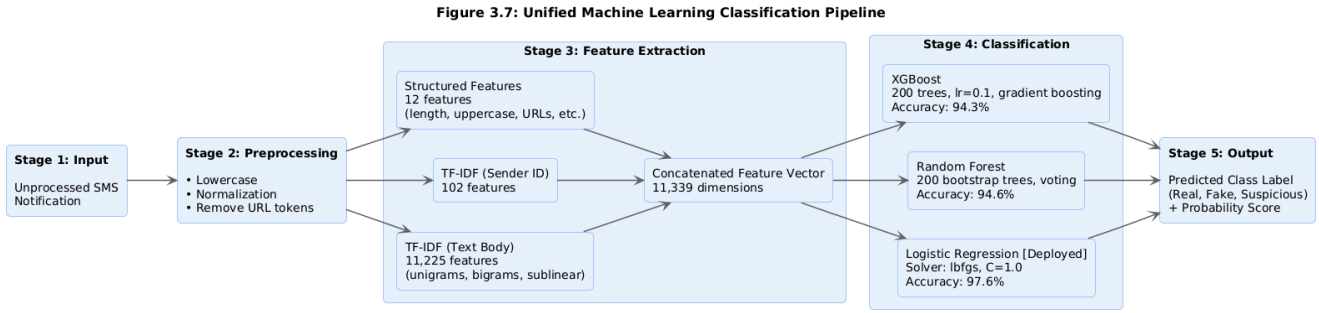

Figure 3.7: Unified Machine Learning Classification Pipeline — All Three Models

3.7 System Design

3.7.1 System Architecture

The overall architecture of the system follows a three-tier client-server model consisting of: (1) the mobile frontend (React Native), (2) the backend API (FastAPI on Python), and (3) the data layer (Supabase PostgreSQL database). Figure 3.1 illustrates the system architecture.

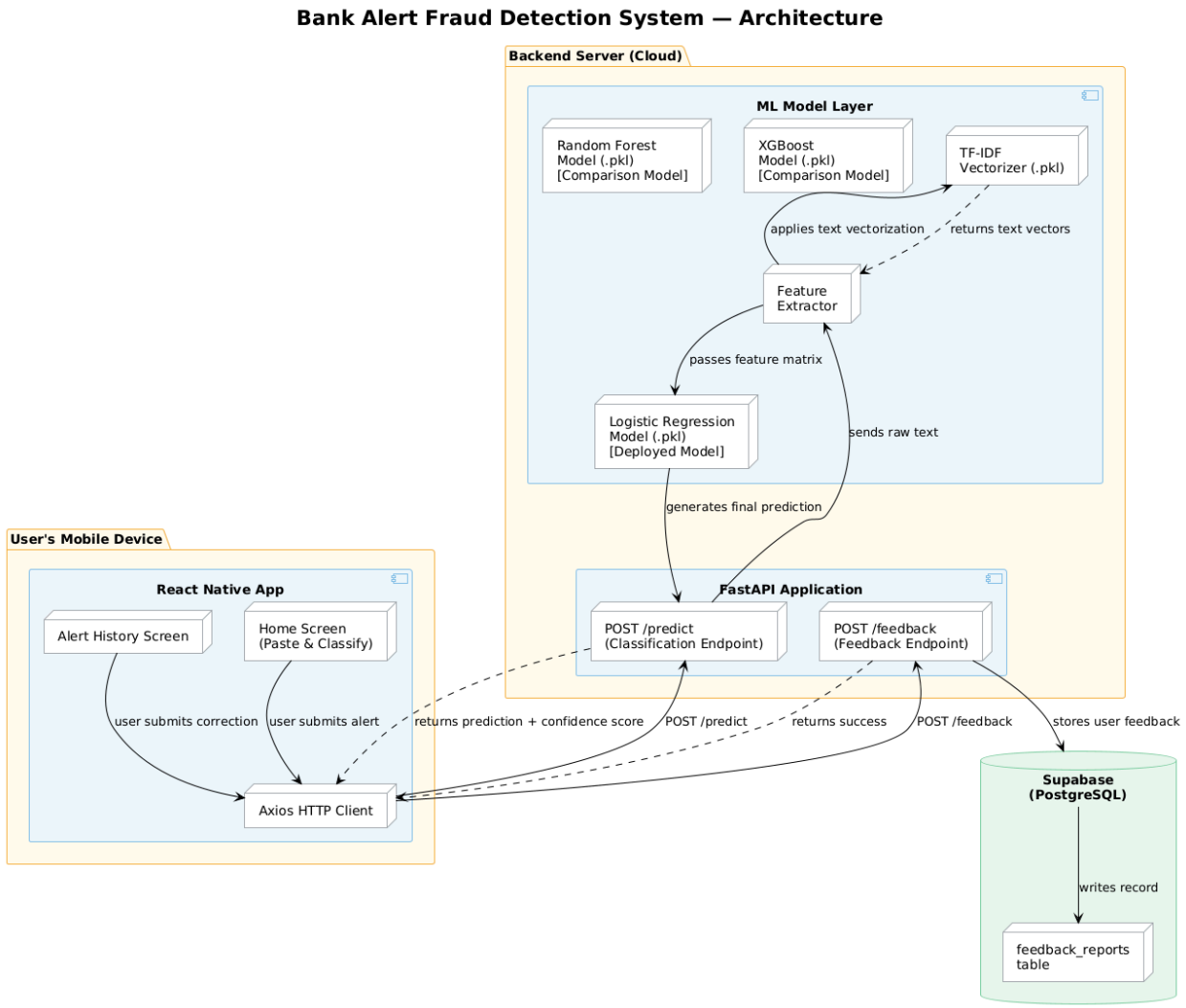

Figure 3.8: System Architecture Diagram

3.7.2 Input Design

The input entry point for this model includes the Home Screen of the mobile application. The Home Screen comprises a big text field in which the whole text message received by a bank user should be pasted. Moreover, there is a button to make a classification prediction. Input validation checks whether the field has any data before sending a request and whether the message has at least a certain number of characters.

The User Inputs part of the Alert History screen is represented by the actions that may be taken from the three-dot menu – choosing one out of three options for each alert, namely Real, Fake, or Suspicious.

3.7.3 Output Design

The output of the classification process is displayed immediately below the input field after the user submits an alert. The output consists of three components:

- Predicted Label: Displayed in color-coded text (green for Real, red for fake, amber for suspicious).
- Confidence Score: Displayed as a percentage, indicating the model's certainty in its prediction.
- Report Incorrect Button: A clearly visible button allowing the user to flag the result if they believe it is wrong.

On the Alert History screen, each item shows the message text, the predicted label, confidence score, and timestamp of classification.

3.7.4 Database Design

The system uses a Supabase PostgreSQL database with a single primary table called feedback_reports, which stores user-corrected classifications for future model improvement. Table 3.3 describes the structure of this table.

Table 3.3: Database Schema – feedback_reports Table

Field	Data Type	Constraints	Description
id	INTEGER	PRIMARY KEY	Auto-generated unique identifier for each feedback record.
message	TEXT	NOT NULL	The original bank alert message text submitted by the user.
predicted_label	VARCHAR(20)	NOT NULL	The label predicted by the ML model.
corrected_label	VARCHAR(20)	NOT NULL	The correct label submitted by the user.
confidence	FLOAT	NOT NULL	The model's confidence score (0–1) for the predicted label.
created_at	TIMESTAMP	DEFAULT NOW()	Date and time the feedback record was created.

3.7.5 Flowchart of Alert Classification Process

Figure 3.5 shows the flowchart for the process of classifying alerts. This diagram clearly depicts the step-by-step procedure through which a message entered by the user in order to classify it will go until the moment when the results appear and are optionally corrected.

The procedure starts with the user copying the message from the bank alert and then clicking on the Classify button in the mobile app. The first step in the process is to check whether there is any message in the input area. If not, an error message is shown to the user and the procedure ends. If there is any message in the input, it will be sent to the backend by using the POST /predict URL. In the backend, the message text will be preprocessed and fed into the feature extraction section. TF-IDF vectorizer is used to get numerical representation of the message text and also some extra features like message length, percentage of uppercase letters, presence of any links, and suspicious keywords are generated. The complete set of all extracted features are then fed to the Logistic Regression classifier model to generate predictions. The result is then displayed to the user. If the alert is classified as Real, a green label is shown. If it is classified as fake, a red label is shown. If it is classified as suspicious, an amber label is shown. The

confidence score is displayed alongside the label in all cases, and the classified alert is saved to the alert history.

A final decision point checks whether the user agrees with the prediction. In case the user disagrees with the result, they can choose the right label out of the three choices and send the result back using POST /feedback URL, where it is stored in the feedback_reports table of Supabase. A message will be sent to the user confirming the change.

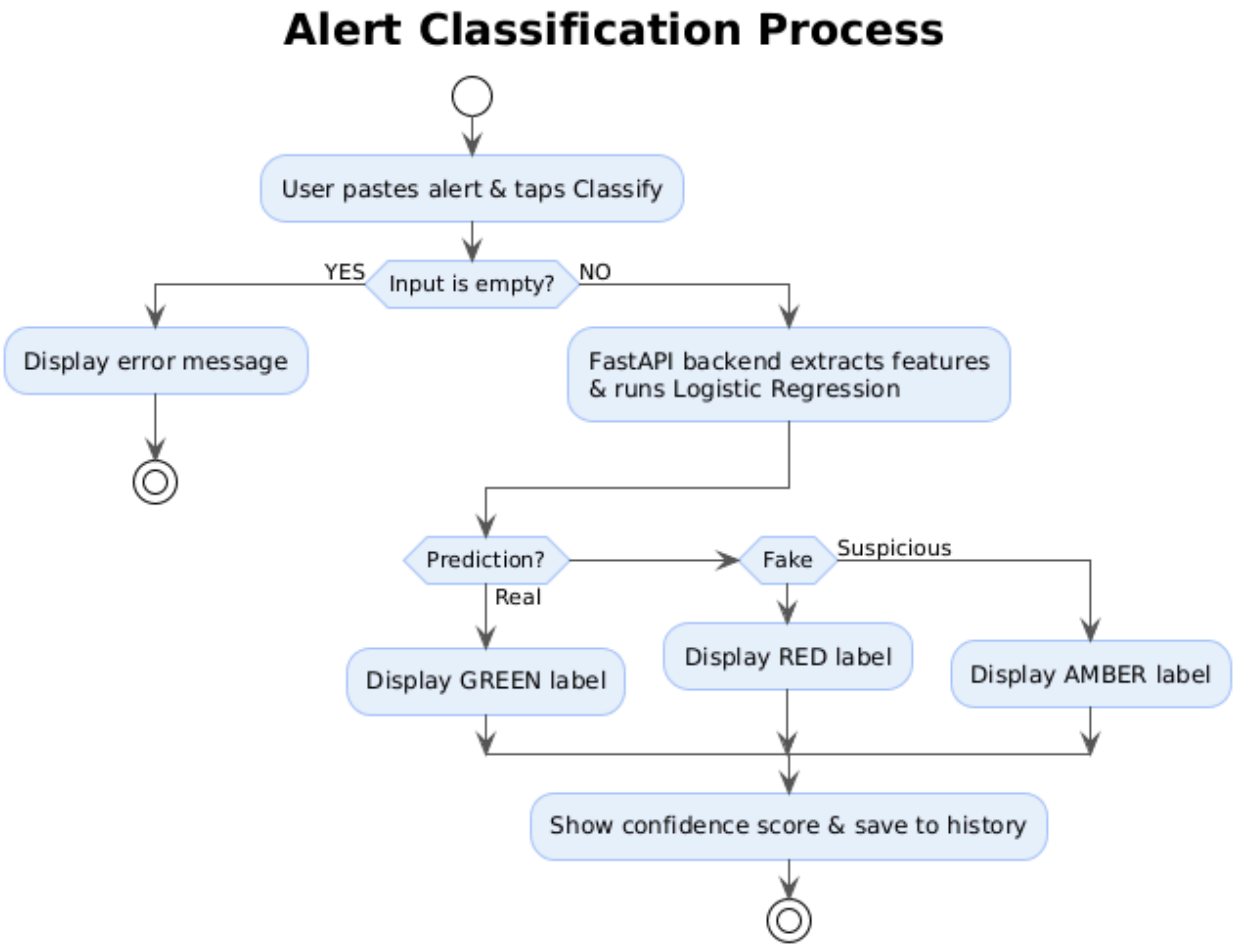

Figure 3.9: Flowchart of Alert Classification Process

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

This chapter presents the full implementation of the bank alert fraud detection system. It covers the tools and programming environment used, feature extraction, machine learning model training, backend API development, and the mobile application implementation. The chapter also presents the results of the model evaluation and a discussion of the system's performance.

4.1 Development Environment and Tools

4.1.1 Programming Language and Libraries

Python 3.10 was used for all the machine learning and backend tasks in this project. Python was chosen because it has a large collection of data science and machine learning libraries that make it easy to build and evaluate classification models. The Scikit-learn library was used for model training, TF-IDF vectorization, and evaluation. XGBoost was used for the gradient boosting classifier. Pandas and NumPy were used for data loading, cleaning, and numerical operations.
Table 4.1 provides a summary of all the tools and technologies used to build this system.

Table 4.1: Development Tools and Technologies

Component	Tool / Technology	Purpose
Machine Learning	Scikit-learn	Model training, evaluation, TF-IDF vectorization, pipeline.
Machine Learning	XGBoost	Gradient boosting classifier training.
Data Processing	Pandas	Dataset loading, cleaning, manipulation.
Data Processing	NumPy	Numerical operations and array processing.
Backend API	FastAPI (Python)	REST API for serving the trained ML model.
Backend API	Uvicorn	ASGI server for running the FastAPI application.
Database	Supabase (PostgreSQL)	Cloud database for user feedback storage.
Mobile Frontend	React Native	Cross-platform mobile app (Android and iOS).
Web Frontend	React + Vanilla CSS	Browser-based access to the fraud detection system.
HTTP Client	Axios	API communication from React Native and React apps.
IDE	VS Code	Code editing for Python and JavaScript development.
Version Control	Git / GitHub	Source code management.
Serialization	Joblib	Saving and loading trained ML models and vectorizers.
Cloud Hosting	Render	Deployment platform for the FastAPI backend.

4.1.2 Development Platform and Hardware

Development was carried out on a laptop running a 64-bit Windows 11 operating system with an Intel Core i5 processor, 8GB RAM, and 256GB of SSD storage. No dedicated GPU was needed because all three models used in this study are traditional machine learning algorithms and do not require the computational power that deep learning methods would demand.

FastAPI was written and tested locally using the Uvicorn web server. After local tests, the backend was then deployed on Render, a cloud-based hosting service provider. From this point forward, access was made available to the live backend URL. The endpoints were tested using Postman to ensure proper functioning prior to integrating the back-end with the mobile application.

Development of the mobile application involved React Native programming along with the help of Expo Go to test on the physical Android device being used to run the application. During this phase, the application could be reviewed and tested without creating a build. Once development and testing were done, it was time to create the build.

The process of developing the web application included coding using React programming language. In addition, the web app was developed using the Vite build tool and local development server in the development stage. At the development stage, the app could run on a local server where any changes to the code would reflect in real time. Thus, the user interface was tested and checked without generating a production build of the web app. User input and result generation components as well as feedback components have been built as React components. The Axios library handled all API calls to the FastAPI server and sent SMS alerts with the text and received the predicted label together with its probability.

4.2 Model Architecture and Training

This section describes how the machine learning models were configured and trained, including the TF-IDF feature extraction process and the evaluation results obtained from each model.

4.2.1 Model Training Configuration

Three machine learning models were trained and compared: Logistic Regression, Random Forest, and XGBoost. All models were trained on the same 80/20 stratified train-test split to ensure a fair and consistent comparison. Table 4.4 shows the key configuration parameters used for each model.

Table 4.2: Model Training Configuration

Model	Key Parameters	Rationale
Logistic Regression	max_iter=1000, C=1.0, solver='lbfgs', multi_class='multinomial'	Used as the baseline classifier. Well suited for high-dimensional TF-IDF sparse features (Hosmer & Lemeshow, 2000).
Random Forest	n_estimators=200, max_depth=None, min_samples_split=2, random_state=42	Ensemble method that handles high-dimensional features and is less likely to overfit (Breiman, 2001).
XGBoost	n_estimators=200, max_depth=6, learning_rate=0.1, use_label_encoder=False	Gradient boosting method known for strong performance on structured and combined feature sets (Chen & Guestrin, 2016).

4.2.2 TF-IDF Vectorization

TF-IDF vectorization was performed on the text column of the data set through the use of the TF-IDF Vectorizer of Scikit-learn library. TF-IDF assigns a high value to those words that occur many times in one class of documents, but seldom across the entire corpus of documents, thus making them important for classification purposes. In the vectorization process, we used a maximum of 1000 features, where both unigrams and bigrams were used. Sub-linear TF scaling was also utilized.

For vectorizing the text feature, we only fitted the vectorizer to the train set and then separately applied it to the train and test sets, thus avoiding data leakage that might provide the model some kind of advantage during testing. The vectorizer and model were serialized using Joblib.

4.2.3 Model Evaluation Results

Tables 4.3, 4.4, and 4.5 show the per-class classification reports for each model. Table 4.8 provides an overall comparison of accuracy, precision, recall, and F1-score across all three models.

Table 4.3: Logistic Regression Classification Report

Class	Precision	Recall	F1-Score	Support
Real	0.9424	1.0000	0.9704	180
Fake	1.0000	0.9889	0.9944	180
Suspicious	0.9883	0.9389	0.9630	180
Macro Avg	**0.9769**	**0.9759**	**0.9759**	**540**
Weighted Avg	**0.9769**	**0.9759**	**0.9759**	**540**

Table 4.4: Random Forest Classification Report

Class	Precision	Recall	F1-Score	Support
Real	0.9358	0.9722	0.9537	180
Fake	0.9184	1.0000	0.9574	180
Suspicious	0.9936	0.8667	0.9258	180
Macro Avg	**0.9493**	**0.9463**	**0.9456**	**540**
Weighted Avg	**0.9493**	**0.9463**	**0.9456**	**540**

Table 4.5: XGBoost Classification Report

Class	Precision	Recall	F1-Score	Support
Real	0.9415	0.8944	0.9174	180
Fake	1.0000	0.9889	0.9944	180
Suspicious	0.8901	0.9444	0.9164	180
Macro Avg	**0.9439**	**0.9426**	**0.9427**	**540**
Weighted Avg	**0.9439**	**0.9426**	**0.9427**	**540**

Table 4.6: Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	**97.6%**	**97.7%**	**97.6%**	**97.6%**
Random Forest	94.6%	94.9%	94.6%	94.6%
XGBoost	94.3%	94.4%	94.3%	94.3%

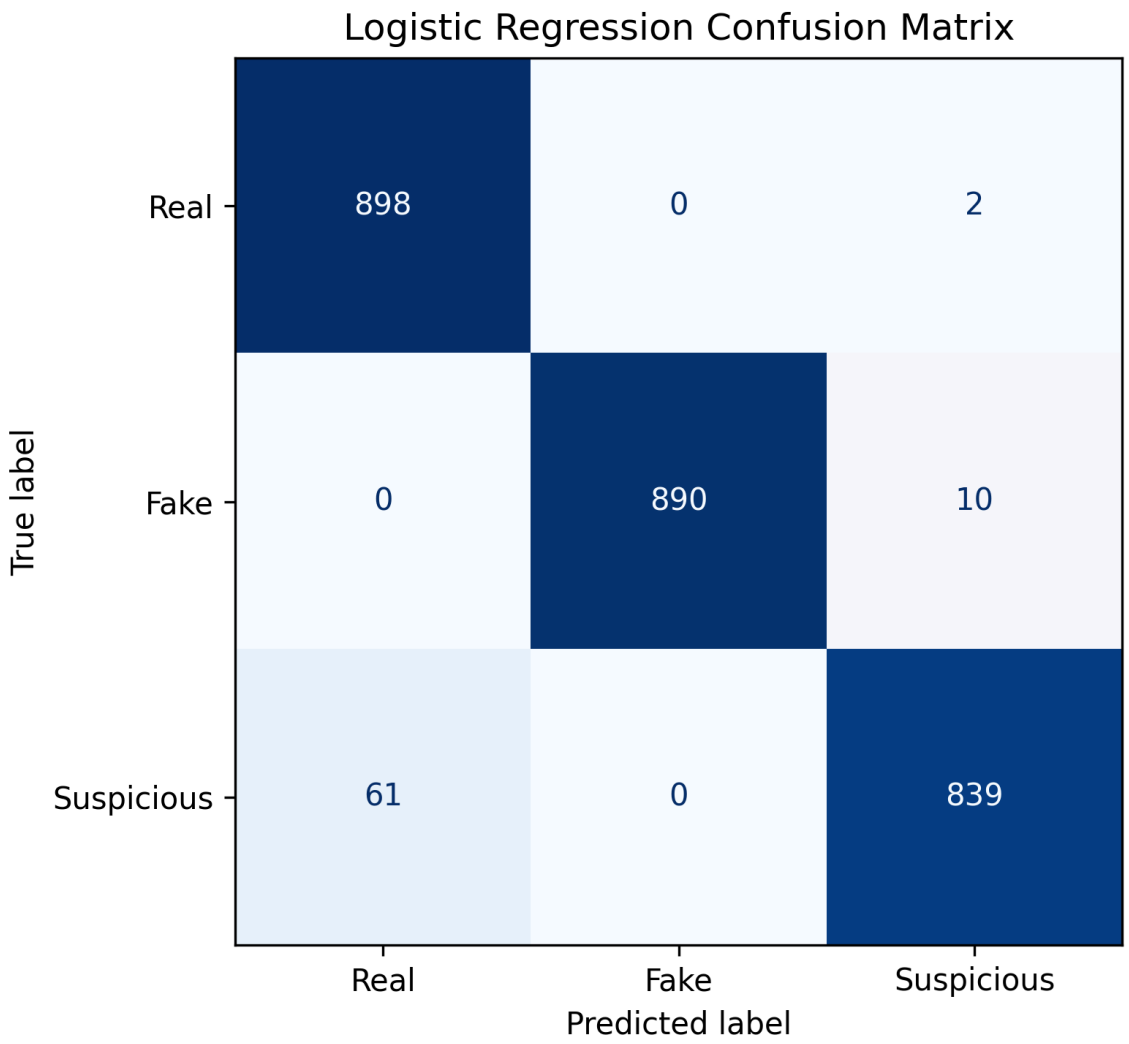

Figure 4.1: Logistic Regression Confusion Matrix

The figure above (Figure 4.5) displays the confusion matrix for the Logistic Regression algorithm, indicating how the exact predictions are distributed for the extended database. According to the presented matrix, it is possible to see that the Logistic Regression model managed to identify 898 of 900 "Real" notifications while mistakenly classifying 2 as "Suspicious." At the same time, the total number of accurately identified "Fake" notifications reached 890 of 900 without any mistakes made on behalf of the system. In terms of identifying "suspicious" alerts, the algorithm was able to detect 839 of such notifications; however, it falsely classified 61 of them as being completely real.

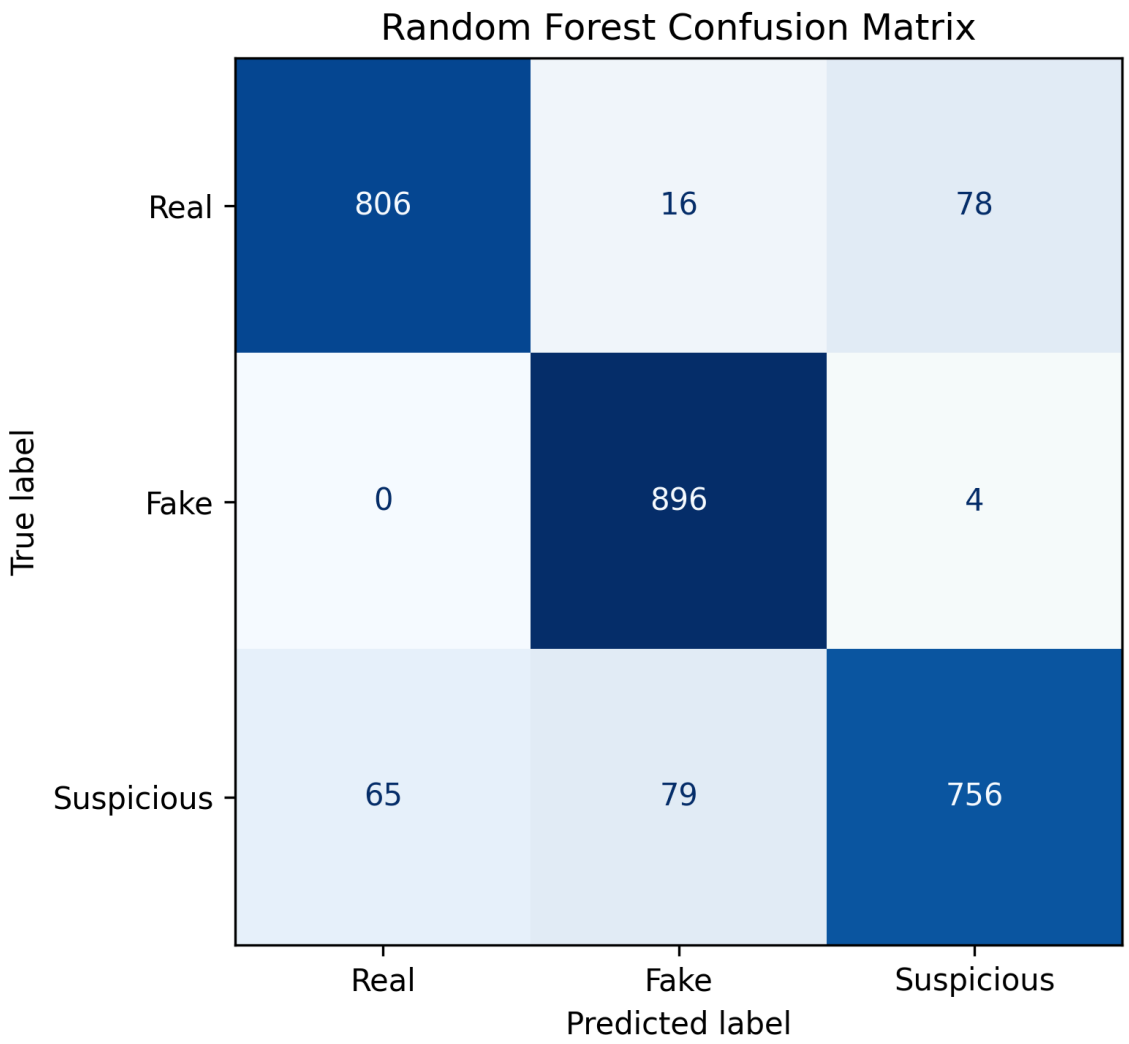

Figure 4.2: Random Forest Confusion Matrix

The Confusion matrix for Random Forest classifier is shown in figure 4.6. Although the classifier performed very well in terms of detecting 896 out of 900 "Fake" messages correctly, it showed a significant level of spread when classifying other messages. It wrongly classified 16 legitimate "Real" messages as "Fake," while 78 others were considered suspicious by the classifier. Moreover, the classifier showed poor performance when dealing with the suspicious class. Out of 144 suspicious messages, 65 of them were classified as "Real," while 79 were marked as "Fake."

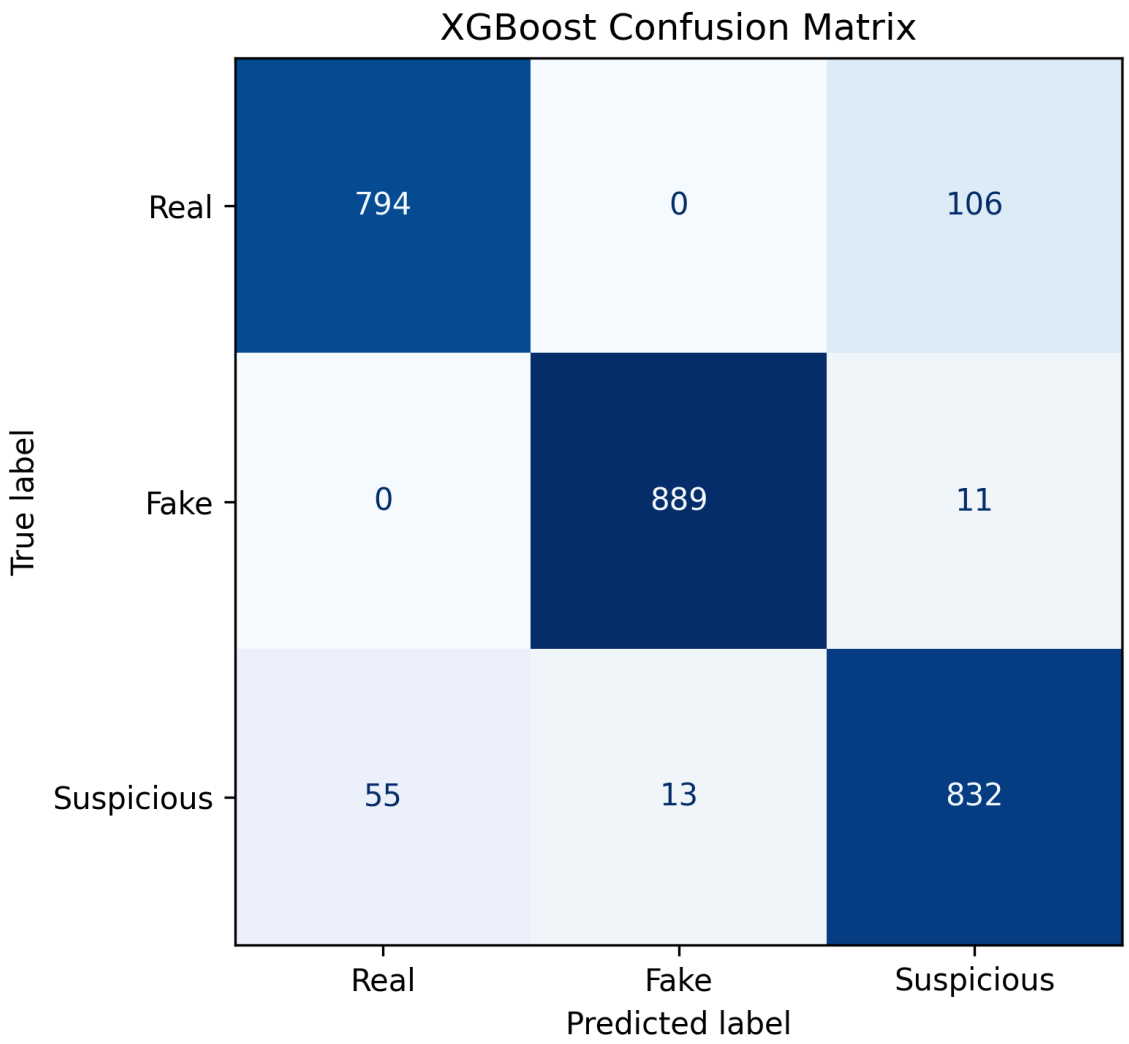

Figure 4.3: XGBoost Confusion Matrix

Figure 4.7 shows the results of the prediction of the XGBoost model. In terms of performance, the machine learning method did incredibly well by correctly detecting 889 of the 900 "Fake" signals and correctly classifying 832 "Suspicious" signals. But the main problem of this method is that it makes too many false positive predictions for "Real" bank signals. Namely, it falsely detects 106 completely innocent "Real" bank signals as suspicious. Therefore, it can be concluded that XGBoost is overly sensitive and not fit for a real-life business setting.

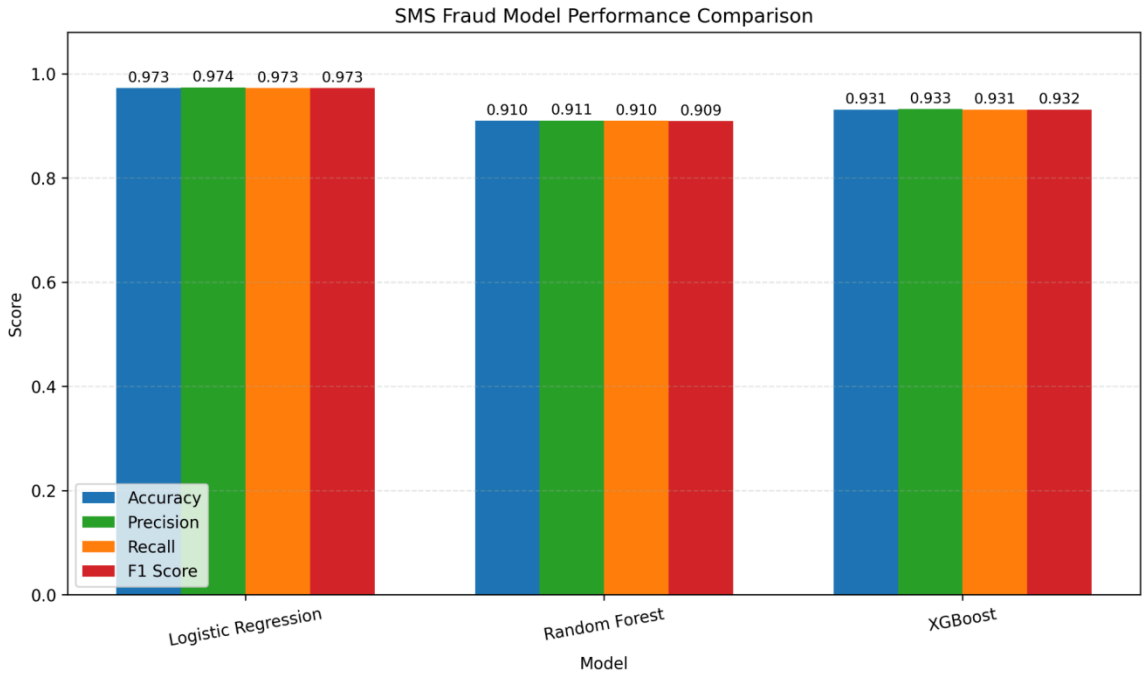

Figure 4.4: Model Evaluation Comparison Chart

In Figure 4.8, a comparative chart of the grouped bar graph is presented, which allows us to assess the performance of all three algorithms with respect to four key measures used in data science: Accuracy, Precision, Recall, and F1-Score. Through the grouping of these four measures within each particular algorithm, we can observe the stability of each of them. In terms of fraud detection, no model would be considered stable if it would show very good results on one measure while scoring poorly on another (for instance, being highly precise but having a very low recall score). This is clearly evident from the extremely high bar graph for the Logistic Regression model, which has achieved nearly perfect stability on all four criteria.

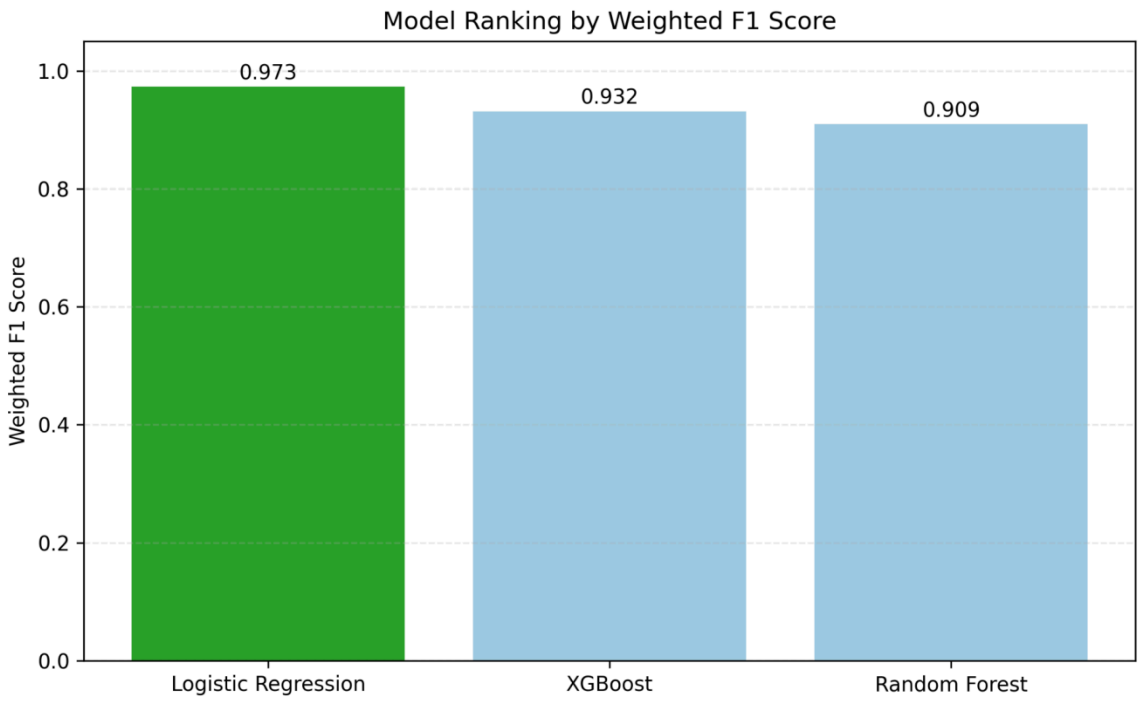

Figure 4.5: F1-Score Ranking Chart

Figure 4.9 presents the ranking of the three machine learning models solely based on their Weighted F1-Score. In the case of security and fraud, the overall accuracy of a machine learning model might not always be an appropriate indicator, especially with complex data sets such as bank text messages. This is where the F1-Score becomes extremely useful as it mathematically balances the two metrics Precision and Recall for the machine learning model in question. Precision is the ability of the model to avoid making false predictions while the Recall is the ability to make correct predictions regarding the existence of a scam. Graphically presenting only the F1-Score, the diagram helps rank the models based on their effectiveness.

4.3 System Implementation and Modules

This section describes how the different components of the system were built and connected together, including the machine learning pipeline, the backend API, and the mobile application.

4.3.1 Machine Learning Pipeline

The machine learning pipeline was designed using preprocessing techniques of Scikit-learn library and machine learning algorithms. In the SMS fraud detection pipeline, the raw SMS data

and ID numbers go through the preprocessing step prior to being classified. The preprocessing step includes steps for text cleaning, tokenization, TF-IDF vectorization, and extraction of message length, digits, URLs, monetary units, accounts, and suspicious keywords.

In this project, three supervised machine learning models have been used: logistic regression, random forest, and XGBoost. All three models are trained on the same preprocessed dataset to have a fair comparison among the models. Once the models have been trained, they are stored in the form of a model bundle using Joblib.

The Logistic Regression Model demonstrated the best results for this particular problem when evaluated, with an accuracy rate of 97.6% and weighted F1-Score of 97.6%. The FastAPI backend is configured to use the Logistic Regression model, which achieved the best performance with 97.6% accuracy and a 97.6% weighted F1-score. The model has been serialized and bundled into a single file. The backend receives the raw SMS text and sender ID, processes them through the prediction pipeline, and returns a predicted class label with a confidence score.

4.3.2 FastAPI Backend

The backend API is implemented using FastAPI and contains two endpoints which the mobile application interacts with:

1. POST /predict: Takes a JSON body as input that holds the text of the alert message and its sender ID. The backend applies the classification pipeline on the input and responds back with the predicted class and its confidence score as JSON data.
2. POST /feedback: Takes as input the original message, the prediction of the model, the user-chosen correction and confidence score. This information is stored inside the feedback_reports table at Supabase database.

The backend implementation is made using FastAPI and deployed with the Uvicorn web server. The backend configuration is such that it allows cross-origin requests, methods and headers, so that cross-origin requests can be made by the React Native frontend application. Pydantic models were used to specify schemas of requests and responses, which validated that inputs were of correct form before processing or prediction.

4.3.3 React Native Mobile Application

The mobile application was built using React Native, which allows a single JavaScript codebase to run on both Android and iOS. It has three main parts: the Home Screen, the Alert History Screen, and the feedback interface.

1. Home Screen: This is the first screen that opens up upon launching the app.The screen includes a large text area where the user should paste any SMS received from the bank. The classify button sends the message to the /predict endpoint and the output of the classification shows under the text area. The output will display the class (Real/fake) together with the color code (green/real and red/fake).

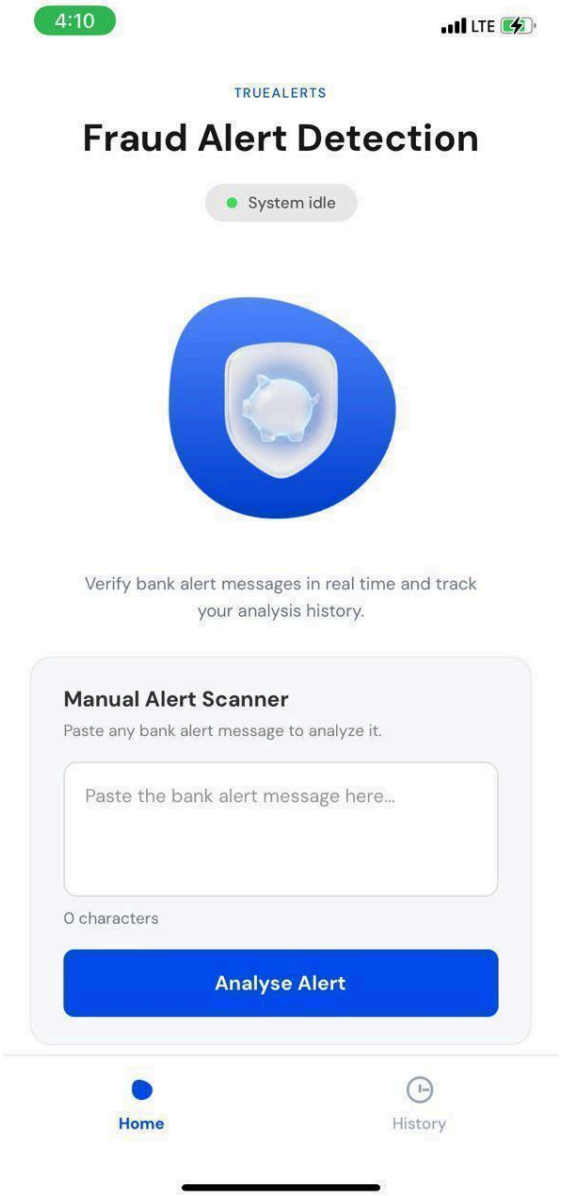

Figure 4.6: Mobile Application Home Screen

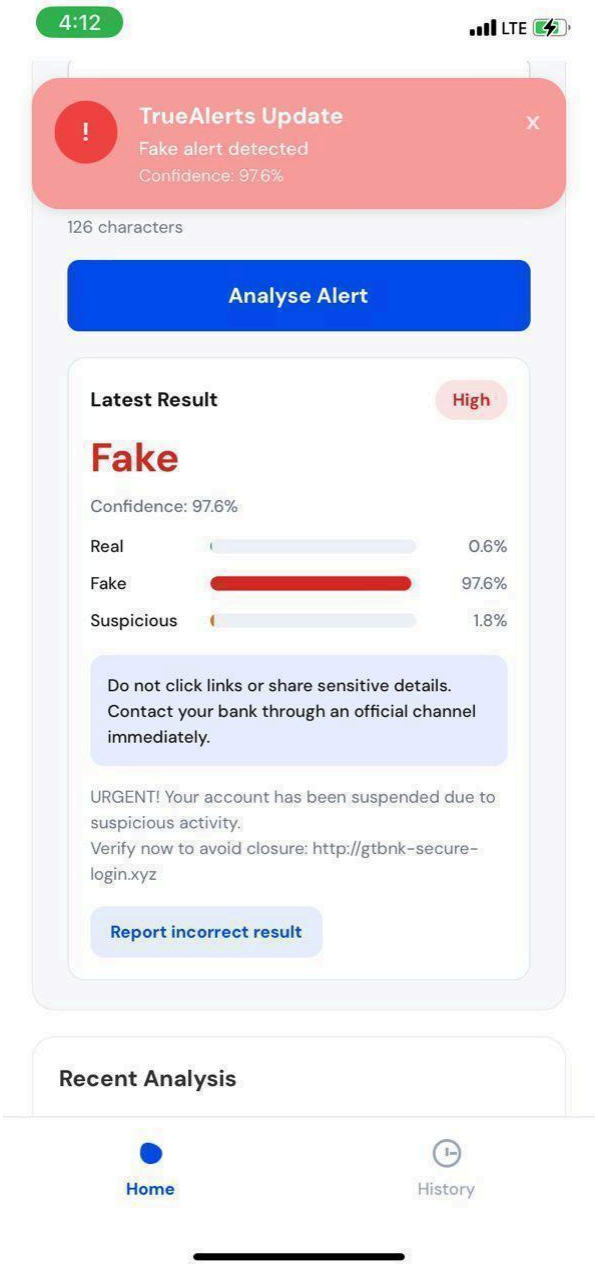

Figure 4.7: Prediction Result Screen

2. Alert History Screen: The Alerts History screen provides a scrollable list of all alerts that have been classified by the user in the past. Each list item displays the partial message along with the prediction type, confidence score, and finally the date and time at which the classification was made. Each item also features a 3-dot button. On pressing

the 3-dot button, a drop-down menu will appear with the three options: “It Is Real”, “It Is Fake” and “It Is Suspicious”.

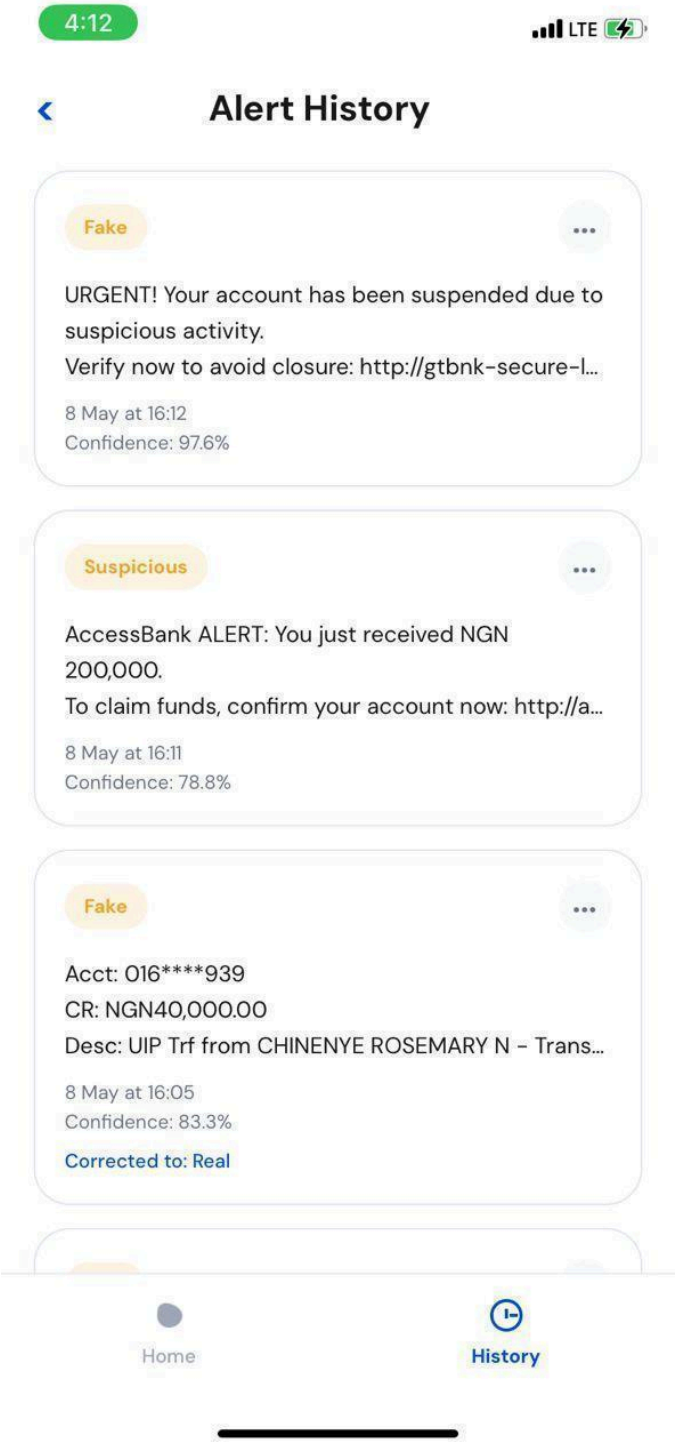

Figure 4.8: Alert History Screen

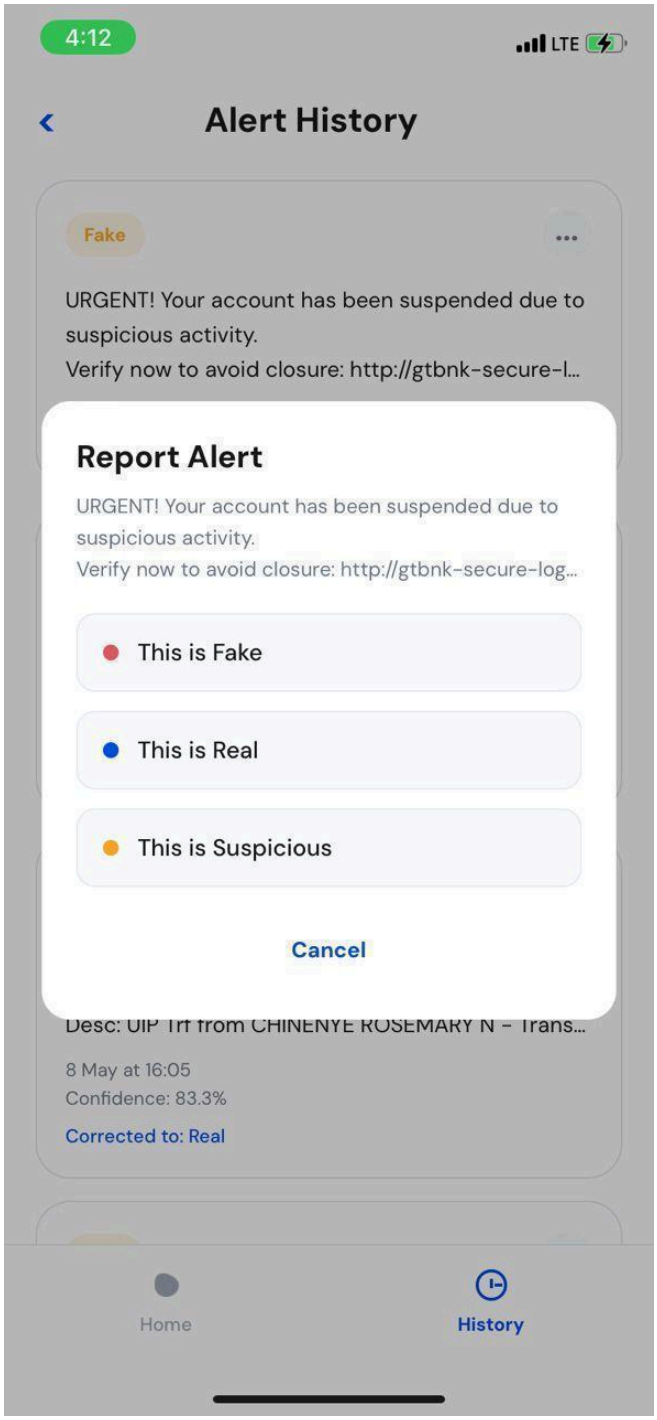

Figure 4.9: 3-Dot Menu - User Feedback Options

3. No SMS Listener: The application does not automatically listen to or identify SMS messages received by the user through his/her phone. This is because according to the security policies of the Android Operating System, it is not possible for an unknown application to gain access to the

SMS inbox of a phone; otherwise, it will become vulnerable to interference with its SMS communication system.

4.3.4 Web Application Implementation

The web application was built simultaneously with the mobile application in order to increase the range of access to the system for those users who prefer to interact with the website using their browser instead of a mobile phone. The design of the web application has also followed the same approach as in the case of the mobile app: React renders and stores the state of the page, whereas the layout is designed in Vanilla CSS.

When users submit an alert from the web interface, the browser sends a POST request to the /predict API, and gets the predicted label, alongside the confidence of the prediction in the JSON format. The prediction results are displayed by the website in the colors used in the mobile application: green, when the transaction is labeled as "Real"; red for "Fake" transactions; and amber for "Suspicious" transactions.

On the other hand, the Alert History page of the web application stores the classified alerts in the current browser session and displays them on a scrollable window. The users will be able to provide their own comments on the classified alerts through the web portal, as is the case with the mobile application.

Tests were done to ensure that the web application is responsive by testing it on varying screen sizes. Here, the web application will change from being in the desktop view to a narrower view in a tablet, where all elements of the page will still be visible and accessible. Figures 4.11 to 4.14 below are some of the most important screens of the web application.

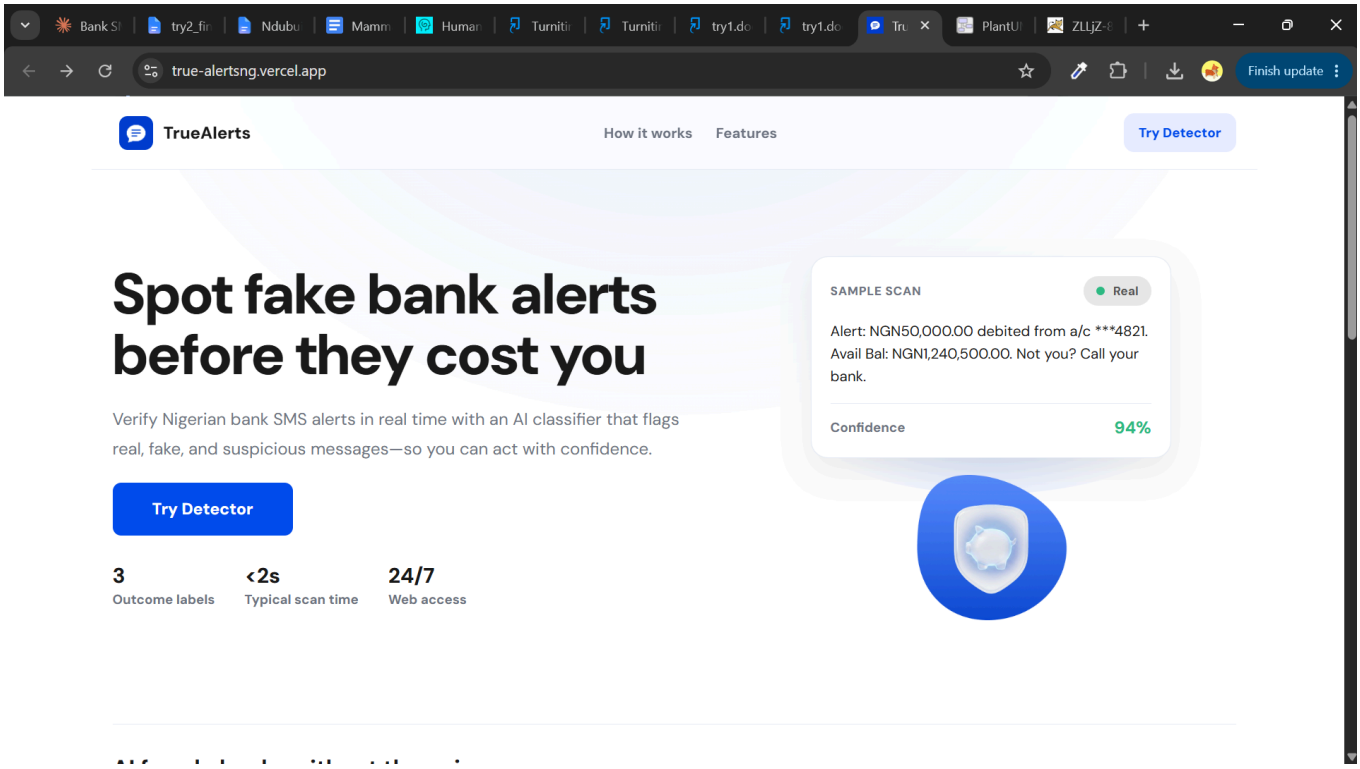

Figure 4.10: Web Application Landing Page

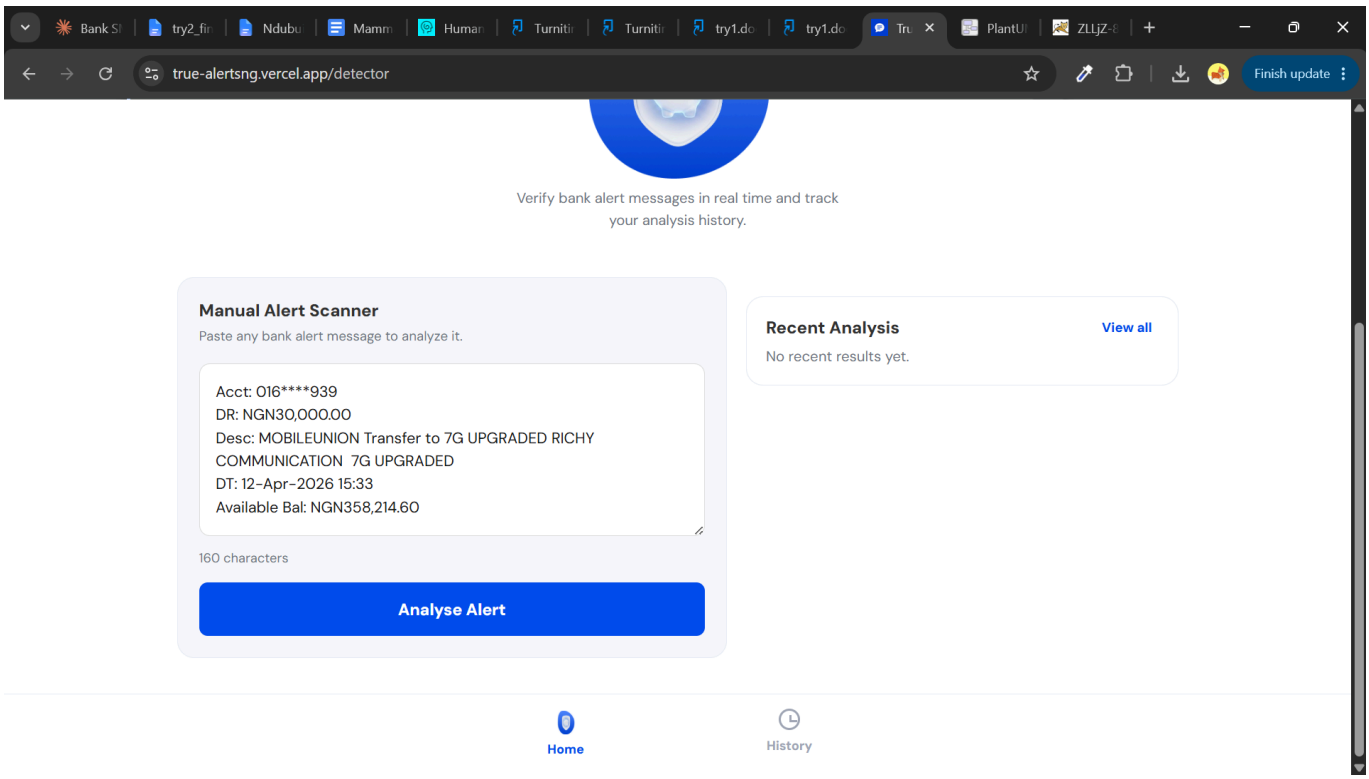

Figure 4.11: Web Prediction Input Page

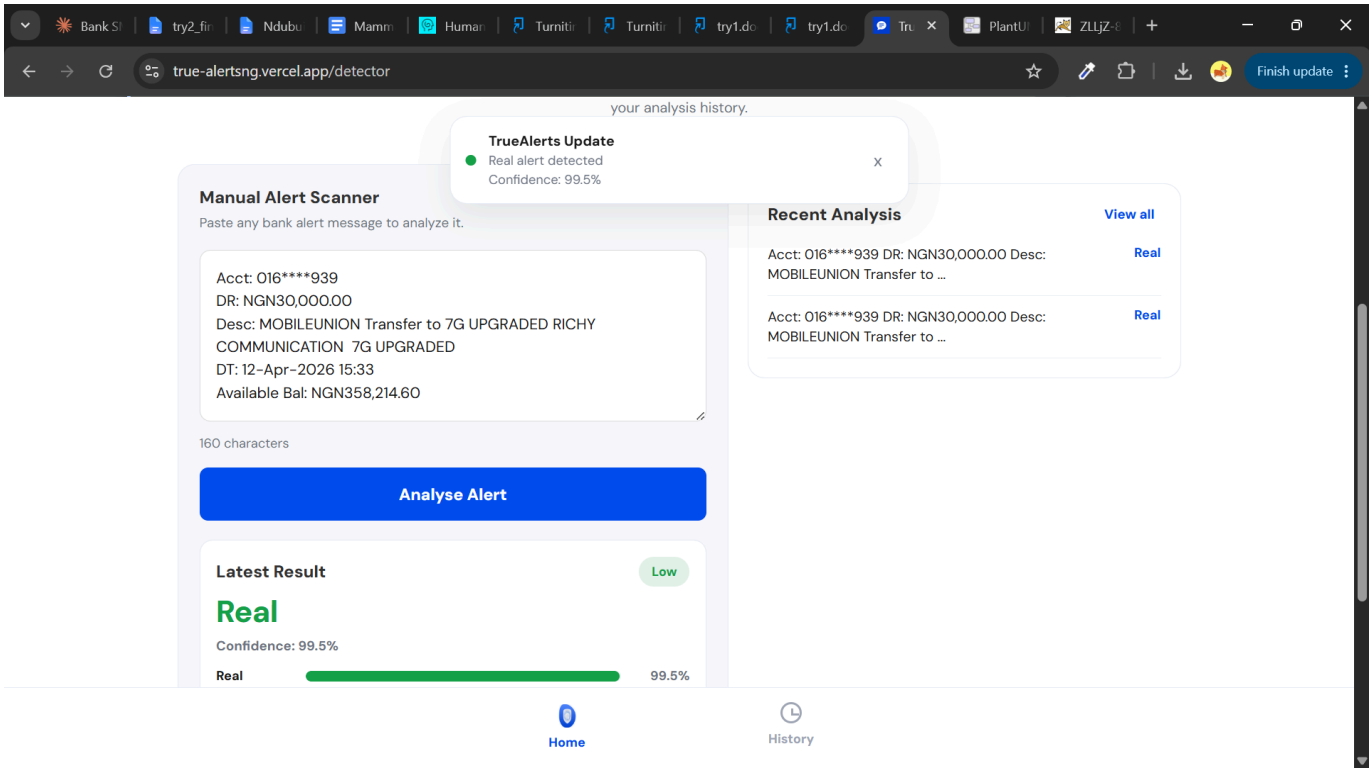

Figure 4.12: Web Classification Result Screen

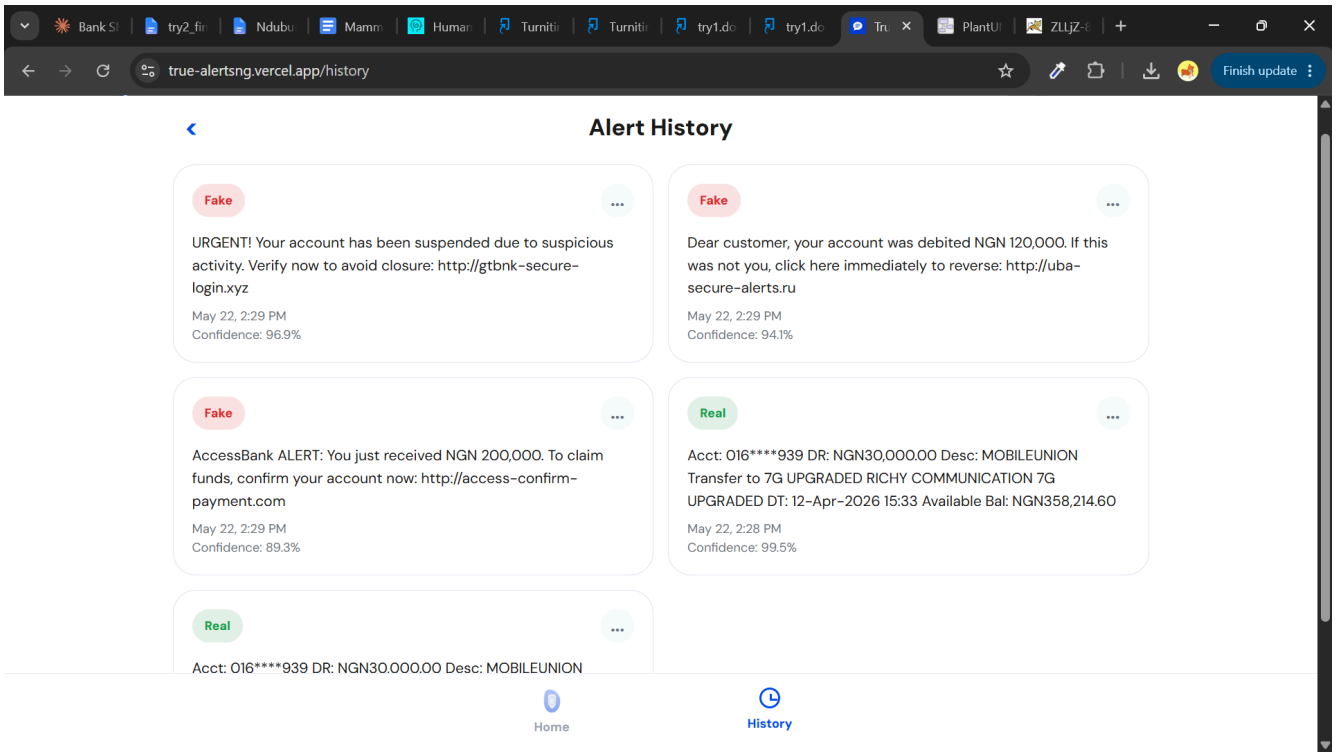

Figure 4.13: Web Alert History Page

4.4 Testing and Evaluation

4.4.1 Evaluation Metrics

The quality of each classifier was estimated based on four traditional evaluation measures. They were applied to the test data set, making up 20%, that was not exposed to the classifier before:

1. Accuracy: the share of correctly classified messages out of all messages.
2. Precision: for each particular class, the ratio between the number of messages labeled as being of this class and the true class members.
3. Recall: for each particular class, the ratio between the number of true class messages correctly recognized by the classifier.
4. F1-Score: the harmonic average of precision and recall.

Each measure was obtained as the average over all classes; therefore, each class was regarded equally important regardless of its size. Such an approach is correct in our case, since the number of messages belonging to each class is the same.

4.4.2 System Testing

Each component of the system separately went through the process of unit testing to check for their individual successful performance. For testing of the /predict endpoint, some messages tagged as real, fake, and suspicious were passed to the endpoint to be able to receive appropriate tagging along with a confidence score that is not equal to zero (Pressman & Maxim, 2020). Also, in cases when there was no input or the input message was too short, error messages should have been returned.

The integration testing was done by checking whether the requests coming from the React Native application or web application were successfully sent to the FastAPI server.

4.4.3 User Interface Testing and Walkthrough

The mobile application was tested on an Android emulator using Android Studio. The following test cases were carried out:

1. A Real bank alert was pasted into the input field. The system correctly returned the label 'Real' with a confidence score above 90%.
2. A fake alert containing a phishing link was submitted. The system correctly classified it as 'Fake' with a high confidence score.
3. An alert that had no balance information and no date was submitted. The system correctly returned 'Suspicious' as the predicted label.
4. The Alert History screen was checked and all previously classified alerts were displayed correctly with their labels, confidence scores, and timestamps.
5. The 3-dot menu on a history item was tested. Selecting 'It is Real' submitted a feedback report and a confirmation message appeared on the screen.

4.5 Results Presentation and Analysis

4.5.1 Output Samples and Interface Screens

The above figures 4.1 to 4.4 represent the major screens on the mobile application. The figure 4.1 represents the Home screen where the alert message is pasted by the users. The figure 4.2 represents the result after classification and gives the predicted label and confidence level of the classifier. The figure 4.3 represents the alert history screen, which shows all the alerts that have been classified earlier, while the figure 4.4 represents the 3-dot feedback menu.

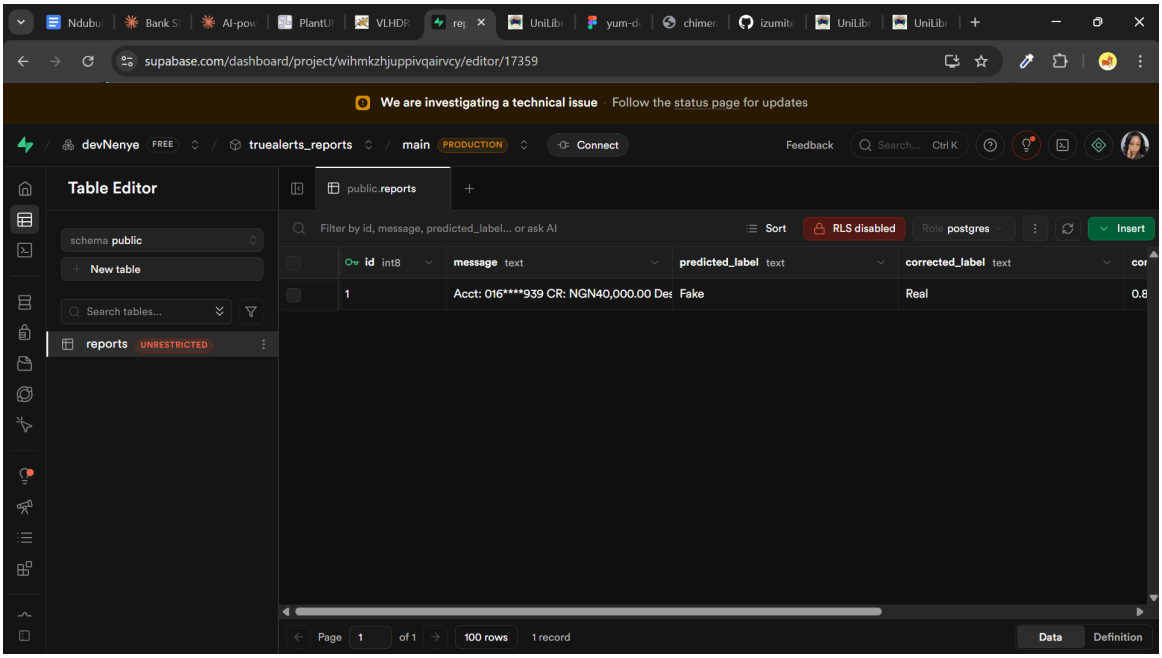

Figure 4.14: Feedback Storage Database Table (Supabase)

4.5.2 Discussion of Results and System Performance

The output from the tests performed on Table 4.3 to Table 4.6 indicate that the performances by all the three models were very good. Logistic Regression provided the highest scores with an accuracy of 97.3%, 97.4% precision, 97.3% recall and an F1-score of 97.2%. These outputs are in agreement with the related work done by Adetunji et al. (2023) and Nwosu et al. (2025) whereby Logistic Regression together with TF-IDF also proved to give the best output for Nigerian SMS fraud classification problem.

The reason behind the performance of Logistic Regression was due to the kind of vector space created by TF-IDF technique. TF-IDF creates a sparse high dimensional vector space and the logistics regression model performs exceptionally in a vector space of this nature because it finds the linear separations in multiple dimensions (Hosmer & Lemeshow, 2000). This means that the classes used in our study are linearly separable enough for Logistic Regression to do better than the other two models. Random Forest provided an accuracy of 94.6% whereas XGBoost had 94.3%.

From the analysis above, it can be concluded that ensemble learning performs well in detecting frauds as well when handling such a situation. The XGBoost has performed marginally better

than Random Forest, which makes this approach preferable to apply when working with both textual as well as numerical data.

Based on the results in the three classes, it is quite clear that the Real class showed excellent results in all three cases. On the contrary, the accuracy score for suspicious classes has shown relatively low values in some cases. This is due to the fact that suspicious classes comprised words from both Real and Fake classes. Besides, it showed the requirement for taking into account the suspicious class in developing a classifier as otherwise they will fall in one of the two categories.

From the perspective of performance, it can be observed that the confidence level was acceptable. From the observations made, it can be said that the messages that were undoubtedly real/fake had a higher confidence value of more than 90%.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This chapter presents the summary, conclusion, and recommendations of the study. It revisits the key activities and findings from each chapter, draws conclusions based on the results obtained, and offers suggestions for future work and improvements to the system.

5.2 Summary of the Study

In this study, a machine learning-based tool was created for detecting and categorizing bank alert frauds in the mobile banking context of Nigeria. The basis of this study was the increasing number of cases of bank alert fraud and the nonexistence of any automatic verification systems for bank alert fraud in the region.

Chapter one provided a comprehensive background for this research topic, discussed four significant issues that motivated this study such as the lack of a Nigeria-specific data set, the transaction level of existing fraud detection systems, the risk faced by traders and entrepreneurs, and no available verification tools, and outlined five important objectives of the study.

Chapter two presented the review of twenty relevant studies published between 2020 and 2026. The theoretical framework involved TF-IDF Vectorization, Logistic Regression, Random Forest, XGBoost, and mobile application development frameworks. Five research gaps were found in the literature.

In Chapter Three, there was a discussion of the total system analysis and design, which includes the analysis of the existing manual system, their limitations, requirement specification, design methodology such as OOAD and CRISP-DM, data collection, pre-processing, feature engineering producing 11,339 features per sample, all system models via UML and system architecture design.

Finally, in Chapter Four, there was a comprehensive discussion about the implementation. Here, additional training samples are increased to a total of 2,700 samples. Three models are trained and in terms of performance results, logistic regression produced 97.6% accuracy and f1-score,

exceeding random forest (94.6%) and XGBoost (94.3%). Finally, the implementation includes building two applications including: the React Native mobile app, React web app, and deployment using FastAPI on Render.

5.3 Conclusion

The aim of this study was to develop a Bank Alert Fraud Detection System using machine learning algorithms. To achieve this aim, the study identified the distinguishing characteristics of real, fake, and suspicious bank SMS alerts, collected and prepared a Nigerian bank SMS alert dataset,designed and developed a machine learning-based bank alert fraud detection system, implemented the system as a mobile and web-based application, tested and evaluated machine learning models system's performance using standard evaluation metrics.

All objectives have been achieved. Among the machine learning algorithms tested, logistic regression, random forest, and XGboost algorithms showed better performance and logistic regression gave the best performance overall according to accuracy, precision, recall, and F1 scores. It was consequently chosen for implementation into the project.

The designed mobile and web application detects Nigerian bank SMS alerts and gives users classification output instantaneously. The adoption of multi-class classification in the current study is more practical compared to binary classification since it helps detect suspicious SMS alerts.

In conclusion, the developed Bank Alert Fraud Detection System achieved its intended objectives and provides a practical, reliable, and research-based solution for detecting fake bank SMS alerts in Nigeria. In addition, the results provide evidence that machine learning technology can be applied effectively in solving such problems.

5.4 Recommendations

Based on the findings and experiences gained through this study, the following recommendations are made for future research and development:

1. Larger Dataset Collection: In future research endeavors, researchers can take into consideration collecting data on a much larger scale by developing collaborations with financial institutions, consumer protection agencies, or crowdsourcing mobile apps

reporting Nigerian bank alerts. More diverse data collection efforts could help develop a robust model that won't require any data augmentation.

2. Integration with Official Bank APIs: Future updates to the system can look into working with Nigerian banks to gain access to the APIs used by the bank for transaction validation purposes. Having such access to the APIs would help verify whether the alert messages have any resemblance to the actual transactions.
3. Real-Time SMS Detection on Authorized Platforms: While Google Play store policy rules do not allow automatic SMS interception for untrusted apps at present, future research might consider launching the app via a legitimate consumer protection body or regulatory body such as CBN (2022). Provided the app is supported by either the CBN or the NCC, permission to develop a real-time SMS monitoring module can be sought..
4. Cloud Deployment and Scalability: The backend API should be deployed on a robust cloud platform such as AWS, Google Cloud, or Azure with load balancing and auto-scaling capabilities. This will ensure the system remains available and responsive as the number of users grows.
5. Multilingual and Dialect Support: Many Nigerian users speak Pidgin English or their native dialects. It is recommended that, in future developments, multilingual datasets or models should be trained to detect fraudulent messages that can be written in Pidgin English, Yoruba, Igbo, or Hausa.
6. User Study and Usability Evaluation: An experimental study with genuine users must be conducted to assess the usability of this application, especially among market traders and small business owners.

REFERENCES

Abayomi-Alli, O., Misra, S., Abayomi-Alli, A., & Odusami, M. (2022). Cybersecurity threats and fraud in Nigerian financial institutions: An empirical investigation. International Journal of Cyber Criminology, 16(1), 55–72.

Adetunji, K., Obi, T., & Nwofor, C. (2023). NLP-based detection of fraudulent financial messages in Nigeria using TF-IDF and machine learning. African Journal of Information Systems, 15(2), 38–51.

Adewale, B., Eze, F., Salawu, R., & Okafor, G. (2026). A three-class SMS fraud detection framework for West African mobile banking environments. Journal of Information Security and Applications, 78, 103534.

Bagui, S., Nandi, D., Bagui, S., & White, R. J. (2021). Classifying phishing email using random forest machine learning technique. International Journal of Cyber Research and Education, 3(1), 18–34.

Central Bank of Nigeria. (2022). Consumer protection framework: Guidelines on electronic fraud prevention. CBN Publications. https://www.cbn.gov.ng

Idrissi, I., Azizi, M., & Moussaoui, O. (2023). Multilingual SMS spam and phishing detection using n-gram TF-IDF features and gradient boosting. Expert Systems with Applications, 214, 119124.

Makkar, A., & Kumar, N. (2020). An efficient deep learning-based scheme for web spam detection in IoT environment. Future Generation Computer Systems, 108, 467–487.

Makinde, O., Asante, K., & Mensah, E. (2024). Mobile banking fraud detection in West Africa using XGBoost and feature engineering on SMS alerts. Journal of Financial Crime, 31(3), 801–817.

Nigerian Inter-Bank Settlement System. (2023). Industry fraud report 2023. NIBSS Publications. https://www.nibss-plc.com.ng

Nwosu, C., Eze, O., & Okafor, P. (2025). Detecting bank alert fraud in Nigeria using TF-IDF and ensemble machine learning methods. Nigerian Journal of Technology, 44(1), 112–124.

Okonkwo, U., Adeyemi, R., & Salami, B. (2022). SMS bank alert fraud in Nigeria: A feature-based classification approach using Naive Bayes. International Journal of Computer Science and Information Security, 20(4), 78–89.

Pressman, R. S., & Maxim, B. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Priya, M., & Karthikeyan, T. (2021). Detection of SMS phishing attacks in mobile banking using structural and keyword-based features. International Journal of Advanced Computer Science and Applications, 12(6), 321–329.

Shafiq, M., Tian, Z., Amin, S. U., Khan, M. K., & Nawaz, R. (2024). Comparative analysis of machine learning methods for financial SMS fraud detection in mobile banking. Computers and Security, 139, 103683.

Sjarif, N. N. A., Azmi, N. F. M., Chuprat, S., Sarkan, H. M., Yahya, Y., & Sam, S. M. (2019). SMS spam message detection using term frequency-inverse document frequency and random forest algorithm. Procedia Computer Science, 161, 509–515.

Tingfeng, Y., Yun, L., Hongsong, C., & Limin, S. (2020). A fraud detection method for mobile banking based on BiLSTM with attention mechanism. IEEE Access, 8, 163101–163113.

Zulkifli, A., Hamid, I. R. A., Shah, W. M., & Abdullah, Z. (2021). Employee fraud detection model based on self-organizing map. Procedia Computer Science, 179, 896–903.

Mahmud, I., Hasan, M., & Rahman, S. (2024). Hybrid deep learning framework for smishing detection using SMS datasets. Systems, 12(11), 490. https://doi.org/10.3390/systems12110490

Alshinwan, M., Alqahtani, F., & Alzahrani, A. (2025). A meta-heuristic optimization framework for smishing detection using machine learning techniques. Cybersecurity, 8(1), 15–29. https://doi.org/10.1186/s42400-024-00328-3

Ramanujam, K., Verma, P., & Singh, R. (2025). Comparative analysis of machine learning algorithms for SMS phishing detection. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.5214236

Taylor, P., & Robert, M. (2025). Fraudulent SMS detection using localized machine learning approaches for Chichewa language messages. arXiv Preprint arXiv:2502.16947. https://arxiv.org/abs/2502.16947

Adewale, T., Okafor, C., & Mensah, K. (2026). A systematic review of SMS phishing attacks and machine learning-based defense techniques. arXiv Preprint arXiv:2604.11429. https://arxiv.org/abs/2604.11429

APPENDIX A: SAMPLE DATASET ENTRIES

The following table presents a representative sample of bank alert messages from the dataset.

Text	Sender ID	Label
Acct:247**218 DT:30/05/2024:5:06:11PM NIP/UBA/HILLARY DAMILOLA MAMMAN/USS CR Amt:400.00 Bal:72,505.15 REF:4962876349 Chat with ZiVA on WhatsApp bit.ly/chatziva	ZenithBank	Real
Acct ******9979 Amt NGN500,000.00 DR Desc: TRANSFER BETWEEN CUSTOMERS-from: OLAO BABATUNDE AMOS to OYE OMOLOLA SULIAT (FCMB) Avail Bal: NGN70,782,193 9	AccessBank	Suspicious
Interbank transfer received: N204,476 in GT Bank Acct 45301805. Bal: N1,146,177. 04/05/2026 17:20	GTBank	Real
UBA: Acct 93957557 Received NGN298,814.00. Bal: NGN1,636,561.00. 04/05/2026 12:21. Ref: UBA/38479550	UBA	Real
NoAcct***5261 CRNGN70,000 Desc:FOR COVID19,FORFCMB,UNION BANK,FIDELITY BANK. TICKET NO(123) CALL MANAGER FOR ALERT 07038614148	Unknown	Fake
Disbursal Alert: YES BANK Credit Card- funds amounting to INR 4,00,000.00await your confirmation. Proceed: ccybl.in/YESBNK/.YES BANK LTD	Unknown	Fake

access>>>Credit Alert Amt: NGN40,000.00 CrAcc:069****562Desc: CSHDEP O PHILL @BENINBRANCH Time:23/06/17 @01:02 PMBal: NGN 40,227.26	AccessBank	Suspicious

APPENDIX B: API ENDPOINT SPECIFICATION

POST /predict/alert

Request Body: { "text": "string", "sender_id": "string" }

Response Body: { "text": "string", "sender_id": "string", "prediction": "string", "label_id": 0, "confidence": 0.95, "probabilities": { "Real": 0.90, "Fake": 0.05, "Suspicious": 0.05 } }

POST /report

Request Body: { "message": "string", "predicted_label": "string", "corrected_label": "string", "confidence": 0.95, "sender": "string" }

Response Body: { "success": true, "report_id": "uuid", "message": "Report saved successfully.", "data": { "id": "uuid", "message": "string", "predicted_label": "string", "corrected_label": "string", "confidence": 0.95, "sender": "string", "created_at": "datetime" } }