CHAPTER 1
INTRODUCTION

In the rapidly changing digital age of today, where internet-based activities like e-commerce, digital communication, and financial transactions are more common than ever before, the menace of cybercrime and financial fraud is becoming a serious concern. With the increase in use of Application Programming Interfaces (APIs) to bring about smooth data exchange, the threat has increased to encompass deceptive QR codes that can lead users to malicious sites, concealing malware within popular files such as PDFs, and phishing attacks using human psychology based on social engineering. These attacks are not just rampant but also increasing in sophistication, causing great financial losses and compromising sensitive individual and organizational information.

In order to overcome these challenges, the Threat Guard AI initiative sets forth a strong and innovative solution: a Multi-Domain Fraud Detection System. The system will look beyond old, reactive security technologies by identifying and preventing risks proactively through multiple digital touchpoints. Differing from typical methods, which tend to target isolated threats, Threat Guard AI utilizes sophisticated machine learning techniques to examine numerous vectors of threat. Some of these include identifying suspicious patterns in transactions, identifying dangerous content concealed in documents, and assessing the security of QR codes that users scan every day.

All the sophisticated analysis and threat detection procedures come together in one intuitive, click-of-a-button Streamlit-powered dashboard. It gives users instant alerts and understandable insights, empowering them to make informed, timely decisions regarding their online safety. Under the hood, the system rests on a stable and scalable Flask API-based backend that guarantees seamless communication between parts while maintaining security.

In order to efficiently handle intricate analytical processes and provide extensibility in the future, the Threat Guard AI system is developed with integration and scalability in mind. MongoDB, a flexible NoSQL database, is at the core of its data management layer. It allows for the structured storage and quick retrieval of various datasets, including user credentials, transaction logs, metadata from scanned documents, and decoded QR code content. By utilizing MongoDB, the system achieves high efficiency in handling varied data types while maintaining flexibility for expansion.

For securing user identity and identity management, Firebase Authentication is natively integrated into the platform. It provides secure sign-in, registration, and password reset processes, where only users with the right credentials have access to sensitive aspects of the application.

The system further integrates real-time processing of visual data through incorporation of libraries like OpenCV (cv2) and pyzbar, which provide QR code decoding either through uploaded images or direct camera input. This aspect is further enhanced with streamlit-webrtc and av library for video stream recording and analysis, thus enabling threats to be analyzed in real-time via webcam inputs.

The fundamental intelligence behind Threat Guard AI is driven by sophisticated machine learning models, developed with the stable scikit-learn framework. Methods such as the Random Forest Classifier are employed to create robust classification algorithms that distinguish between normal and malicious behavior by learning from patterns in historical data. The system assesses model performance based on key metrics like accuracy, confusion matrix, and classification report—all provided by scikit-learn—so results are not just accurate but also explainable and verifiable.

For efficient data manipulation and preparation during model training and testing phases, libraries like pandas and numpy are employed. These tools handle structured tabular and numerical data with high performance. Once trained, models are serialized and saved using joblib, allowing for quick loading and deployment during runtime.

In order to make the findings easily understandable by end-users, Threat Guard AI incorporates data visualization libraries like matplotlib.pyplot and seaborn. These libraries assist in creating clear and informative plots and charts that present scan results, risk levels, and transaction patterns, making the experience more transparent and user-friendly.

In summary, Threat Guard AI is an integrated cybersecurity platform that actively responds to threats on various vectors. Through the integration of smart machine learning analysis, safe data storage, real-time engagement, and user-friendly visualization within a unified framework, the project provides a scalable and responsive defense system. Its judicious application of contemporary libraries and frameworks manifests both the technical richness and the pragmatic purpose to counter the dynamic threats of financial fraud and cyberattacks.

Key Components and Features

- **User Interface:** An intuitive platform that enables interaction between users and the fraud detection system. The user can use login/registration and choose to run the fraud analysis module (Phishing, QR, PDF, or UPI). It is developed based on Streamlit (Python).
- **Backend API:** A Flask API server that serves as the system's central hub. It accepts data from the Streamlit frontend, processes the data through the relevant fraud detection modules, and returns the results to the frontend. The Flask API server also manages login/registration by talking to the MongoDB database.
- **UPI Fraud Detection:** This module is intended to scan UPI transactions to identify counterfeit payments and other fraudulent schemes. It is intended to identify fraud by inspecting transaction patterns.
- **QR Code Fraud Detection:** This module is intended to identify counterfeit QR codes using image classification processes. It is involved in scanning QR code images to identify manipulative or misleading codes.
- **PDF Malware Detection:** The function of this module is to identify malicious PDF files containing malware embedded in them. This is done by checking file attributes to detect malware hidden in PDF files.
- **Phishing Detection:** This module detects phishing attempts by scanning suspicious links and other attributes of digital communication. The module examines digital communication content to identify phishing attempts.
- **Database:** MongoDB (Cloud/Local) is employed as the project database. It's utilized for storage, presumably containing user authentication information and other pertinent data.

Improved Detection and Analysis

- **Multi-Domain Fraud Detection:** The system offers a single solution to fraud detection by combining modules to address UPI, QR, PDF, and Phishing attacks. This combination is intended to offer a more comprehensive defense against varied yet interrelated threats.
- **ML Models & Dashboard:** The project involves the development of AI-driven models for each type of fraud and presents the analysis through a user-friendly Streamlit interface/dashboard.

- **Unified Dashboard:** A unified dashboard is developed for the integrated visualization of multi-domain digital fraud analysis results.

1.1 Problem Definition

This project seeks to create an extensive multi-domain fraud detection system to overcome the shortcomings of existing security systems. The expansion of the digital age in online transactions and communications has greatly enhanced the occurrence of financial fraud and cybersecurity attacks. These include different types of fraud, including UPI scams, where scammers try to obtain money illegally via the Unified Payments Interface, fake QR codes, which can direct users to harmful websites or make unauthorized payments, PDF malware, where malicious software is placed inside Portable Document Format files, and phishing, where attackers try to trick users into providing sensitive information. These threats not only result in immediate financial loss to individuals and institutions but also hurt trust in online platforms and services. Existing fraud detection systems tend to work in siloed systems, handling one kind of threat at a time. Attackers also often merge attacks on multiple platforms and vectors in order to multiply their success opportunities. For instance, they may employ a phishing attack to harvest user login credentials and subsequently employ those credentials for carrying out fake UPI transactions. Existing security mechanisms and detection techniques usually find it difficult to keep up with such intricate and fast-changing attack patterns. Such inability to integrate and respond is a primary vulnerability that this project aims to resolve.

Hence, there is an urgent need for a single and smart system that can process data from different sources to offer an integrated and proactive defense.

This project will develop a system that will analyze data across various domains, such as UPI transaction behavior, QR code images, PDF file attributes, and digital communication content. The multi-domain analysis will enable the proactive detection and prediction of fraud, enabling a stronger and better defense against the varied and interconnected threats in the digital world.

1.2 Objectives

The goals of this project mirror the necessity of developing a cutting-edge fraud detection system to provide accuracy, efficiency, and usability, coupled with mitigating the key challenges of contemporary digital fraud.

1. Improve Fraud Detection Accuracy - UPI Transactions

The envisioned platform focuses on accuracy in detecting fraudulent UPI transactions through the use of superior analytical methods.

Precise Detection of UPI Frauds: To precisely detect fraudulent UPI transactions by processing patterns of transactions. It includes using machine learning models to learn usual patterns of transactions and identify anomalies signalling fraudulent transactions, thus avoiding false positives and ensuring efficient detection of scams perpetrated on UPI.

2.Improve Accuracy of Fraud Detection - Malicious QR Codes

An important task is to efficiently detect misleading QR codes which are security threats through smart image analysis.

Accurate QR Code Fraud Detection: To accurately identify malicious and spoofed QR codes using smart image analysis. This involves using computer vision methods to inspect the visual features of QR codes and pinpoint those that are altered or engineered to lead consumers to malicious sites or trigger illegitimate transactions.

3.Improve Fraud Detection Accuracy - Malware within PDF Documents

The system seeks to effectively detect malicious software hidden within popular PDF documents through analysis of their internal structure and features.

Effective PDF Malware Detection: To effectively detect malware hidden within PDF documents through file characteristic analysis. This includes the analysis of the metadata and content of PDF files for recognized signatures of malware or unusual patterns that signal the existence of malicious code, thereby safeguarding users against possible system breaches.

4. Enhance User Experience - Actionable Fraud Predictions

A primary goal is to provide the result of the fraud detection analysis clearly and in actionable form to the user in the form of an easily understandable user interface.

• Clear Fraud Predictions: To provide clear and actionable fraud predictions via an easy-to-use Streamlit dashboard. This means creating an interface that clearly communicates the probability and type of likely fraud, so users can appreciate the risks and take the necessary preventative or responsive actions.

5. Improve User Experience - Unified Data Visualization

The project seeks to present a summary overview of prospective threats by converging the outcome of the examination of different vectors of fraud into one unified visual representation.

6.Unified Data Visualization:

To implement a unified dashboard to present integrated visualization of multi-domain digital fraud analysis outputs. This includes designing a centralized platform in which users are able to view the output of the UPI, QR code, PDF, and phishing detection modules in a consolidated manner, enabling improved understanding of overall risk and possible interconnected threats.

CHAPTER 2

LITERATURE SURVEY

The "Threat Guard AI" system is an innovative solution for overhauling fraud detection as it identifies the weaknesses of conventional and cyber security systems. The efficiency of the system is highly improved through the incorporation of knowledge from recent studies in the fields of fraud. An example of this is the work done by Zara et al. on phishing website detection with machine learning and deep learning models, which directly applies to the phishing detection module of the system. Their study shows that ensemble models can have high accuracy in detecting phishing sites, which is the objective of the system to detect and prevent phishing attacks, a significant cause of fraudulent actions that the system seeks to prevent. In the same vein, the work of Patel et al. on phishing mail detection via natural language processing (NLP) offers rich methods for scrutinizing digital communication content, a primary aspect of the Threat Guard AI phishing detection features.

The system further derives inspiration from studies that concentrate on detecting UPI fraud. The use of supervised learning methods, specifically Random Forest, by Jagadeesan et al. for classifying UPI transactions as fraudulent or genuine is particularly what shapes the system's module for detecting UPI fraud. Through studying patterns of transactions and determining anomalies, the system is capable of detecting fraudulent UPI transactions and preventing them, which improves the security of digital payment systems. Also, Bodade et al.'s research on machine learning-based detection of fraud in digital payments, such as UPI transactions, gives further context and methods to this module.

In order to counteract QR code and PDF fraud, the system employs research in enhancing its capacity for processing potentially malicious inputs. Seetha et al.'s QR code recognition study is important to the system's QR code fraud detection. It assists the system in identifying tampered QR codes better and more efficiently. Kumar & Singh's work on OpenCV and deep learning for the analysis of QR codes also plays a role. Also, the system's PDF malware detection is guided by Gupta et al.'s and "PDF Malware Detection" research, which highlight extracting PDF features to detect malware. Lamgade et al.'s AI-based PDF malware detection adds strength to the system's detection of malicious PDFs.

Furthermore, the system understands the need for the integration of sophisticated security technologies such as blockchain for specific use cases, with inspiration from Ahmed et al.'s "DeepLedger" that integrates deep learning and blockchain for phishing email detection. Though not directly deployed in every module, the viability of blockchain integration for improved security and traceability is understood, particularly for applications involving immutable logs and transparent transaction verification.

In addition, Kumar et al.'s "QR-Secure" that applies reinforcement learning to check QR codes holds the key to enhancing the resiliency as well as the accuracy of handling QR codes through the system, depending on a variety of environments. This research is useful when creating a dependable QR code forgery detection module that can serve well in realistic environments. The system further integrates the principles of Patel et al.'s "SmartUPI," which is an integration of Random Forest-based fraud detection and blockchain transaction logging, to facilitate fraud transparency and auditability, especially in the context of financial transactions.

Lastly, Srinivas et al.'s study on "PhishGraph," which uses graph-based machine learning for social engineering attack detection, guides the system in how it detects and prevents advanced phishing campaigns that use intricate attack vectors. Through the judicious incorporation of findings and techniques from these varied but pertinent areas of research, the "Threat Guard AI" system is designed to deliver a holistic, adaptive, and efficient solution to the complex challenges of contemporary digital fraud, setting a new standard for security and trust in the digital era.

2.1 Title of paper: PDF Malware Detection: Toward Machine Learning Modeling with Explainability Analysis

Year: 2022

Authors: Not mentioned in the title page directly

Methodology adopted: The paper presents a strong machine learning-based approach to identifying malicious PDF files. Structural and metadata features are extracted by the authors using tools such as PDFiD, PDFINFO, and PDF-PARSER. Models such as Random Forest and Decision Trees are trained and interpreted using feature importance scores to promote transparency in decision-making.

Remarks: Accomplishes good interpretability and high accuracy in PDF malware detection. Though good performance, model performance may change with obfuscated

PDF samples. Dataset creation and explainability make strong contributions but performance in adversarial cases is an area to work on.

2.2 Title of paper: Phishing Website Detection Using Deep Learning Models

Year: 2024

Authors: Ume Zara, Kashif Ayyub, Hikmat Ullah Khan, Ali Daud, Tariq Alsahfi, Saima Gulzar Ahmad

Methodology adopted: Advanced ML and DL techniques such as LSTM, GRU, RNN, Random Forest, XGBoost, etc., are used to detect phishing websites in this research. Information gain, gain ratio, and PCA are utilized to select features. Models are verified on a total of 11,055 sites. Ensemble models have the maximum accuracy (with up to 99%).

Remarks: The usage of ensemble learning and deep learning is extremely effective. The research is thorough in the comparison of various models and feature engineering methods, but live testing and adaptive updates for real-world performance against zero-day phishing may still be necessary.

2.3 Title of paper: QR Code Recognition Based on Image Processing

Year: 2024

Authors: J Seetha, G Manikandan, S Hemalatha, Vilma Veronica

Methodology adopted: A multi-stage image processing pipeline is suggested to enhance QR code detection with changing light and tilt conditions. Steps involve binarization, tilt compensation, geometric normalization, and orientation management employing OpenCV.

Comments: Provides a real-world practical improvement to QR code scanners. Although it proves effective at raising recognition rates, it is heavily image-processing driven and doesn't delve into the use of machine learning or AI for error recovery or responsiveness.

2.4 Title of paper: UPI Fraud Detection Using Machine Learning

Year: 2025

Authors: S. Jagadeesan, K.S. Arjun, G. Dhanika, G. Karthikeyan, K. Deepika

Methodology employed: Employs supervised learning (Random Forest) to label UPI transactions as fraud or genuine. Preprocessing, feature engineering, train-test split, and performance metrics (Gini, Entropy) are employed. Potential enhancements involve the integration of blockchain and real-time deployment.

Remarks: Has promise for detecting fraud in online payments. The system is improved by ensemble learning but could do even better with more unsupervised or real-time anomaly detection methods. It is a good basis for further fintech fraud prevention studies.

2.5 Title of paper: AI-Driven Blockchain Voting System with Zero-Knowledge Proofs for Privacy Preservation

Year: 2024

Authors: L. Ma, X. Zhang, J. Thomas, A. Bahadur

Methodology adopted: This article suggests an e-voting system integrating blockchain smart contracts and zero-knowledge proofs (ZKPs) to secure voter anonymity while guaranteeing vote validity. The voters are authenticated through OTP and biometric checks before a vote is cast under encryption through ZKP and stored immutably on a permissioned Ethereum blockchain.

Remarks: The application of ZKPs considerably enhances voter privacy. The system provides robust decentralization and transparency, although complexity in setup and network latency during high-demand voting times are possible issues.

2.6 Title of paper: DeepLedger: A Blockchain-Integrated Deep Learning Model for Phishing Email Detection

Year: 2023

Authors: I. Ahmed, R. Mehta, S. Kapoor, L. Feng

Methodology adopted: The research integrates LSTM-based deep learning for phishing email classification with a private blockchain for secure record-keeping. Identified phishing attacks are hashed and stored on-chain, allowing for future pattern recognition and verification using transaction audit logs.

Remarks: Offers great traceability and attack protection with high accuracy (96%). Blockchain provides verifiability, but needs a secure and scalable data indexing technique for real-time deployment.

2.7 Title of paper: QR-Secure: Reinforcement Learning-Based QR Code Verification Under Adverse Conditions

Year: 2025

Authors: D. Kumar, H. Takashi, M. Ghosh

Methodology applied: Employs reinforcement learning agents learned on degraded QR samples for dynamic recognition improvement in changing light, occlusion, and geometric distortion. Incorporates OpenCV with adaptive recognition using custom TensorFlow models.

Comments: Outperforms baseline OpenCV-only pipelines in real-world scenarios. The RL component adds robustness, but computational burden on edge devices may be further optimized.

2.8 Title of paper: SmartUPI: Blockchain-Aided Framework for Secure and Transparent UPI Transactions

Year: 2024

Authors: V. Patel, S. Chakravarthi, A. Jain

Methodology employed: Adopts a two-layer architecture combining Random Forest-based fraud detection and logging of blockchain transactions. Transaction features include fingerprinting, smart contract verification for transfers, and masking of user identities via anonymized hashes.

Comments: Enhances fraud transparency and legal auditability of UPI networks. Interoperability with bank APIs is possible, albeit regulatory acceptance and latency management are open issues.

2.9 Title of paper: VoteLedger: Real-Time Blockchain Voting Using Proof-of-Authority and Biometric Matching

Year: 2023

Authors: P. Nambiar, E. Garcia, T. Ryu

Methodology followed: A biometric-enabled voting application with PoA consensus for fast vote finality. Uses a sidechain-based architecture to avoid Ethereum gas cost bottlenecks. Facial and fingerprint data are hashed and verified using on-chain logic.

Remarks: Combines speed and security for small-scale elections. Privacy considerations are well addressed, though sidechain interoperability with public blockchains is yet to be proven at scale.

2.10 Title of paper: PhishGraph: Graph-Based Machine Learning for Detecting Social Engineering Attacks

Year: 2024

Authors: T. Srinivas, R. Jadhav, K. Haider

Methodology followed: A graph neural network (GNN) architecture is used to classify phishing attacks by analyzing URL, email, and DNS graphs. Link prediction and node embedding strategies identify attacker infrastructure.

Remarks: Outperforms conventional classifiers in detecting zero-day phishing threats. Could benefit from integration with decentralized storage for preserving attack footprints securely.

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

3.1 Scope of Project

The focus of this project includes the overall design, development, and deployment of a multi-domain system for fraud detection. The system is intended to utilize machine learning and artificial intelligence (AI) methods to offer a powerful and integrated platform. The main objective of the project is to adequately solve and prevent different types of online fraud, which have been rapidly growing in number and complexity. The system is intended to target the following types of fraud:

- UPI fraud: Illegal activities performed using the Unified Payments Interface, including unauthorized transactions and tampering with payment procedures.
- QR code fraud: Misleading practices that involve the utilization of malicious or tampered QR codes to mislead users and trigger fraudulent processes.
- PDF malware detection: Detection and prevention of malicious software contained in Portable Document Format (PDF) files, which can weaken system security.
- Phishing attacks: Identification and blocking of fraud attempts to acquire sensitive data like passwords, usernames, and credit card numbers by pretending to be a legitimate entity via electronic communication.

Core Components and Features:

1. Frontend Development:

- The project will entail the development of a user-friendly and easy-to-use interface utilizing the Streamlit library. Streamlit, as a Python library, is selected because it can make the development of interactive web applications easier, enabling the fraud detection system to be easily usable and navigable by users.
- The frontend will feature special modules aimed at enabling analysis and detection of every particular kind of fraud. These modules will give users the necessary tools and features to work with the system properly.
- A combined dashboard will be a central feature of the frontend, offering an overall and central look at fraud prediction and results of analysis. This dashboard will

deliver information in an understandable and actionable manner so users can make fast decisions based on potential issues.

2. Backend Development:

- The project will use a Flask API server to receive requests sent from the frontend and to process the data effectively. Flask, being a Python web framework, is selected due to its flexibility and lightweight nature, making it possible to develop strong and scalable backend services.
- Backend logic will include the integral functionality of the fraud detection system, such as data processing for each fraud detection module and effortless communication with the MongoDB database. This guarantees that data is processed securely and effectively.

3. Database Management:

- NoSQL database MongoDB will be used for storing data. MongoDB's scalability and flexibility make it an ideal choice for managing the varied and possibly large amounts of data involved in fraud detection.
- The database will be designed to hold different kinds of data, such as transaction data pertaining to UPI fraud, data obtained from QR codes, PDF file data, and digital communication content pertaining to phishing attempts.
- Efficient data management practices will be used to maximize data retrieval and storage, making the system responsive and performant.

4. Fraud Detection Modules:

- The system will have specialized modules, each of which will be used to identify a particular type of fraud.
- Machine learning models will be used to process UPI transactions and identify fraudulent transactions. These models will be trained to recognize patterns and anomalies in transaction data that are characteristic of fraud.
- Image classification methods will be employed to identify manipulated or forged QR codes. This means examining the visual properties of QR codes to verify their genuineness.
- Characteristics of PDF files will be examined to identify malware embedded in PDF files. This could mean checking file structure, metadata, and included scripts.

- Contents of digital communications, including messages and emails, will be inspected to identify phishing attempts. Artificial intelligence (AI) techniques or natural language processing (NLP) may be employed to flag suspicious patterns and wording.

5. AI and Machine Learning Implementation:

- Each form of fraud will have models developed and put into practice to detect accurately and effectively.
- Different machine learning models will be investigated and implemented, with particular models such as Random Forest being used for UPI fraud detection because of its suitability for classification tasks.
- The choice and training of the right AI models will be an important part of the project, so that the system is able to identify and classify fraudulent activities in various domains accurately.

6. Data Analysis and Visualization:

The system will provide clear and actionable fraud predictions to users via the Streamlit dashboard. This includes creating visualizations and reports that clearly convey the probability and type of potential fraud.

An integrated dashboard will offer an overall visualization of multi-domain fraud analysis outcomes. This will enable users to observe a complete picture of potential risks and comprehend the connections between various forms of fraud.

Technologies Overview:

1. Streamlit:
   - Employs an interactive and user-friendly interface for fraud data visualization and analysis.
   - Supports quick frontend development with Python scripting.
2. Flask:
   - Takes care of backend API requests and communication between the frontend and other modules.
   - Enables processing of data for fraud detection modules.

3. MongoDB:
    - Takes care of storage and retrieval of fraud-related data, such as transaction details, file details, and analysis outputs.
    - Provides flexible data storage and scalability to support changing data volumes.
4. Pandas:
    - Takes advantage of data manipulation and analysis features to process structured data efficiently.
    - Enables data cleaning, transformation, and preparation for machine learning models.
5. Scikit-learn:
    - Applies machine learning algorithms for fraud detection, including classification models for UPI fraud and phishing detection.
    - Provides tools for training, evaluating, and selecting machine learning models.
6. Joblib:
    - Saves and loads trained machine learning models efficiently, making it possible to reuse them quickly and minimize training overhead.
7. OpenCV:
    - Processes and analyzes image data to detect fraud in QR codes.
    - Offers functionality for image processing, feature extraction, and object detection.

3.2 Software Requirements

1. Frontend:

   Implemented using Streamlit (Python), providing an interactive and user-friendly interface for users to analyze fraud data and see predictions. Streamlit's ease of use makes it possible to develop and deploy the frontend application quickly.

2. Backend:

   Driven by Flask (Python), serving API requests from the frontend, data manipulation for fraud detection, and database interactions with MongoDB. Flask is lightweight and flexible, making it ideal for creating solid backend services.

3. Database:

   MongoDB stores and handles fraud data. Its NoSQL design offers scalability and flexibility, supporting the varied types and volumes of data used in fraud detection.

4. Libraries:

   Pandas facilitates data manipulation and analysis, allowing for effective processing of structured data. Scikit-learn offers machine learning algorithms for constructing fraud detection models. Joblib is utilized for effective model saving and loading. OpenCV is utilized for image processing in QR code fraud detection. TensorFlow is utilized for deep learning model construction.

5. Programming Language:

   Python 3.10+ is utilized for frontend and backend development alike, taking advantage of its extensive collection of libraries and frameworks for data science as well as web development.

CHAPTER 4

SYSTEM ANALYSIS

4.1 Existing System

Detailed Documentation on Existing Fraud Detection Systems

Fraud detection systems are also vital in ensuring the security of electronic transactions and the integrity of online platforms. Conventional fraud detection techniques lag behind the changing cybercrime tactics, creating loopholes such as financial losses, data compromise, and loss of user confidence. AI-based fraud detection systems seek to overcome these weaknesses by creating a strong system for detecting and combating fraudulent activity. Such systems are defined by automated analysis, adaptive algorithms, and real-time monitoring. This report discusses four leading examples of AI-based fraud detection methods: rule-based systems, machine learning-based systems, behavioral analysis systems, and hybrid systems, their concepts, technologies, applications, features, and challenges. Overview of AI-Based Fraud Detection Systems The integration of AI into fraud detection targets key areas: accuracy, adaptability, and efficiency. The capacity of AI to examine intricate patterns, learn from experience, and automate decision-making improves the identification of fraud. Machine learning algorithms incorporated into fraud detection systems using AI are able to provide better accuracy and fewer false positives. These systems also simplify the analysis process, making it possible for real-time monitoring and quicker response. Despite these advantages, AI-driven fraud detection is continuously evolving, facing hurdles such as data dependency, adversarial attacks, and the need for continuous model updates.

Rule-Based Systems:

Rule-based systems, a classical method, use pre-defined rules to detect fraudulent transactions. These rules are derived from known fraud patterns and domain expertise. Simple, interpretable, and easy to implement are the key characteristics. Rule-based systems work well for detecting well-delineated fraud cases. They are weak in detecting new or emerging fraud methods. They need to be updated manually quite often to be effective, which is time-consuming and can introduce delays.

Machine Learning-Based Systems:

Machine learning-based systems employ algorithms to learn from past data and detect patterns that are fraud-related. Such systems have the ability to adjust to new fraud situations and detect anomalies that the rules-based system misses. Machine learning approaches, including classification, clustering, and anomaly detection, are used. The main characteristics are high accuracy, flexibility, and automation. But these systems need large, high-quality training datasets and their performance is a function of the quality of the algorithms and features selected. Explainability can also be an issue, as it may not always be easy to see why a specific transaction was identified as fraudulent.

Behavioural Analysis Systems:

Behavioural analysis systems are concerned with tracking user behaviour and detecting deviations from the norm. These systems monitor a range of activities, including login activity, transaction values, and navigation patterns. Through analysing user behaviour, these systems are able to identify account takeovers, fraudulent transactions, and other malicious activities. Major features include real-time identification, user profiling, and anomaly detection. These systems need advanced monitoring mechanisms but also have the potential to raise issues of privacy. Accuracy relies on the completeness of the behavioural data and the strength of the analysis algorithms.

Hybrid Systems:

Hybrid systems integrate several fraud detection methods to take advantage of the strengths of each method. For instance, a hybrid system can employ rule-based systems for clearly defined fraud cases and machine learning for identifying new or sophisticated fraud patterns. Hybrid systems seek to offer a more complete and resilient fraud detection solution. The key characteristics are enhanced accuracy, flexibility, and coverage. However, these systems may prove to be complicated to implement and design, necessitating the sensitive integration of various techniques and technologies.

Case Studies of Fraud Detection Implementations

E-commerce Transaction Monitoring Systems: Most e-commerce sites utilize real-time monitoring systems for transactions to identify fraudulent orders. These systems monitor several parameters, like purchase value, shipping location, and user activity, to identify fraudulent transactions. The application of AI and machine learning in these systems has

helped to dramatically enhance the effectiveness of these systems in identifying fraudulent transactions. E-commerce fraud detection highlights the role of proactive monitoring and adaptive algorithms in avoiding financial loss.

Credit Card Fraud Detection Systems: Advanced fraud detection systems are used by credit card companies to safeguard customers from unauthorized spending. These systems evaluate transaction patterns, spending patterns, and location information to detect anomalies in transactions that might be used for fraudulent purposes. Machine learning models are responsible for these systems, allowing them to learn from changing fraud methods and reduce false positives. Credit card fraud detection places strong emphasis on having strong security measures and ongoing monitoring to ensure customer confidence.

Challenges Confronting AI-Based Fraud Detection Systems

AI-based fraud detection systems are confronted with a number of challenges:

- Data Dependency: The performance of AI models is greatly dependent on the presence of large, high-quality datasets for training.
- Adversarial Attacks: Fraudsters can try to trick or manipulate AI systems by using adversarial methods.
- Model Drift: Machine learning models can become less precise over time as fraud patterns change, necessitating ongoing model refreshes and retraining.
- Explainability: It can be difficult to understand why an AI model identifies a specific activity as fraudulent, which can make it difficult to maintain transparency and trust.
- Privacy Concerns: Gathering and analyzing user information for fraud detection poses significant privacy concerns that must be met.

Comparative Analysis of Existing Systems

A comparative study indicates varying strengths and weaknesses among current fraud detection systems. Rule-based systems are simple but inflexible. Machine learning-based systems are highly accurate but data-intensive. Behavioral analysis systems allow real-time detection but threaten privacy. Hybrid systems provide holistic solutions but are difficult to implement.

Future Directions and Opportunities

The future of AI-based fraud detection is in overcoming existing challenges through innovation. Incorporating methods such as federated learning can improve privacy while allowing collaborative model training. Creating explainable AI (XAI) techniques can enhance transparency and trust. Real-time data analysis and adaptive algorithms are essential to remain ahead of changing fraud strategies.

Conclusion

AI-based fraud detection systems can transform traditional fraud fighting into a more accurate, adaptable, and efficient process. Existing systems prove the viability of AI but also reveal their potential pitfalls, including data dependency, adversarial attacks, and explainability problems. Resolving these through innovation, extensive testing, and ethical factors is critical to moving forward.

The analysis of these systems offers useful lessons for the creation of a multi-domain fraud detection project. By leveraging their advantages and filling their gaps, a secure, transparent, and user-friendly fraud detection system can be developed that opens the door to a safer and more reliable digital world.

4.2 Proposed System

Proposed Multi-Domain Fraud Detection System

Artificial intelligence and machine learning solutions have arisen as game-changing technologies in numerous industries, and their use in fraud detection presents great potential for solving issues in conventional and digital security systems. Our envisioned multi-domain fraud detection system aims to improve fraud prevention accuracy, efficiency, and responsiveness. By leveraging the analytical and predictive capabilities of AI, this system aims to create a robust framework for identifying and mitigating various types of fraudulent activities. This document details the design, objectives, features, and benefits of the proposed system.

Introduction

The prevalence of digital fraud poses a significant threat to individuals, businesses, and organizations, yet traditional fraud detection methods often struggle to keep pace with the evolving tactics of cybercriminals. While current systems have tried to bring fraud

prevention into the modern era, they are not completely immune from vulnerabilities, such as failure to identify new patterns of fraud, excessive false positives, and challenges in processing multi-domain data. To counter these issues, the suggested multi-domain fraud detection system offers a single, AI-based platform for detecting and analyzing fraud across various digital channels. This revolutionary system takes advantage of AI's inbuilt ability to make fraud detection not just precise but also thorough and responsive to new threats.

Objectives of the Proposed System

The proposed system is headed by five major goals. To begin with, it aims to improve precision using machine learning models and AI-based analysis to identify fraudulent activity with great accuracy. Secondly, it ensures comprehensiveness by integrating several fraud detection modules to make it possible for the system to examine fraud in different domains. Thirdly, it enhances adaptability by using AI models that can adapt based on new data and continue evolving as fraud patterns change. Fourthly, it seeks to automate fraud prediction and analysis to enhance efficiency and minimize the use of human intervention. Lastly, the system aims to offer actionable intelligence through the delivery of understandable and interpretable fraud predictions, allowing for timely and effective action.

System Architecture and Components

The system for multi-domain fraud detection is constructed on a solid architecture with multiple interdependent components:

- AI and Machine Learning Models: Fundamentally, the system makes use of a collection of AI and machine learning models designed to identify particular types of fraud. Models like TensorFlow and Scikit-learn are used to develop these models.
- Data Integration and Preprocessing: In order to provide effective fraud detection, the system has data integration and preprocessing modules. These modules gather and sanitize data from different sources before it is sent for analysis to the AI models.
- Fraud Detection Modules: The system contains dedicated modules used to detect fraud in particular spaces. These can be:
- UPI Fraud Detection: Processing UPI transaction data to detect fraudulent payments.
- QR Code Fraud Detection: Identification of malicious or manipulated QR codes.
- PDF Malware Detection: Malware detection hidden in PDF documents.

- Phishing Detection: Digital communication content analysis to identify phishing attacks.
- User Interfaces: User interfaces for communication with the fraud detection features:
- User Interface: Streamlit-based interface for displaying fraud analysis outputs, setting options, and interacting with the system.
- API Interface: An API based on Flask to enable other systems or applications to use the fraud detection capabilities.
- Analysis and Visualization Tools: The system also includes tools for visualization and analysis of fraud information, allowing users to learn about fraud patterns and trends.
- Workflow of the System
- The process of fraud detection is optimized into four main steps:
- Data Acquisition: The system retrieves data from diverse sources, such as transaction logs, file stores, network activity, and user logs.
- Data Preprocessing: The obtained data is cleaned, transformed, and formatted for analysis. This could include the elimination of noise, missing value handling, and extraction of pertinent features.
- Fraud Analysis: AI and machine learning models in the fraud detection modules analyze pre-processed data. These models recognize patterns and anomalies that represent fraud.
- Prediction and Reporting: The system makes fraud predictions and reports, which give users actionable insights and notifications.

Key Features

The system has various novel features that increase fraud detection:

- Multi-Domain Detection: The system offers a single platform for detecting fraud across digital channels.
- AI-Powered Analysis: Machine learning algorithms and AI methods support precise and adaptive fraud detection.
- Real-Time Evaluation: The system is capable of evaluating data in real-time, allowing proactive fraud prevention.

- Actionable Insights: The system offers interpretable and understandable fraud predictions, allowing timely responses.
- Scalability and Adaptability: The system is scalable and able to accommodate changing fraud situations.
- Benefits of the Proposed System
- The proposed multi-domain fraud detection system possesses several advantages:
- Increased Accuracy: Analysis by AI makes fraud detection more accurate, eliminating false positives and negatives.
- Detailed Coverage: Multi-domain coverage by the system delivers a complete overview of fraud, making it easier to detect intricate fraud schemes.
- Adaptive Learning: AI algorithms learn from additional data, meaning the system adjusts to changing patterns of fraud.
- Optimized Automation: Automated analysis and prediction decrease manual intervention, enhancing efficiency and response times.
- **Proactive Prevention:** Monitoring in real-time allows the system to anticipate and prevent fraud beforehand.

Challenges and Mitigation Strategies

The system, although beneficial, also presents some challenges:

- **Data Dependency:** The efficacy of AI models is dependent on the quality of training data provided. To mitigate this, the system has strong data collection and preprocessing mechanisms.
- **Adversarial Attacks:** Scammers can try to tamper with data to avoid detection. The system uses methods to identify and counter adversarial attacks.
- **Model Drift:** The performance of AI models can degrade over time due to a shift in fraud patterns. The system has mechanisms for ongoing model monitoring and retraining.
- **Explainability:** It can be difficult to understand why an AI model is making a specific prediction. The system uses methods to enhance model explainability.
- **Resource Needs:** AI-based fraud detection may be computationally demanding. The system is architected to maximize resource utilization and exploit cloud computing as necessary.

Conclusion

The above-discussed multi-domain fraud detection system is a huge leap towards the modernization of fraud prevention tactics. Through the synergy of the power of AI and a common platform, the system tackles key inadequacies of current and conventional fraud detection systems. Its multi-domain and AI-based analysis guarantee fraud detection accuracy and flexibility, which promotes a safer digital landscape. Through continuous development and improvement, this system promises to transform how fraud is identified and blocked, opening the door to an even safer and more reliable digital landscape.

CHAPTER 5

PROPOSED METHODOLOGY

Threat Guard AI begins with a comprehensive requirement analysis to identify both functional and non-functional needs. Functional requirements include secure data ingestion from various sources, multi-layered detection mechanisms for UPI, QR code, PDF, and phishing fraud, real-time threat monitoring, and detailed audit trails. Non-functional requirements focus on performance, scalability, and security to handle large volumes of data efficiently and protect sensitive information. The system aims to address vulnerabilities in digital transactions and communications, such as financial fraud, malware dissemination, and information theft, by leveraging advanced machine learning and AI techniques. This analysis stage ensures that the system is designed to meet the demands of modern digital interactions and the expectations of stakeholders, including users, financial institutions, and security analysts.

The system's architecture is divided into multiple key components for seamless integration and functionality. The frontend, developed using Streamlit, provides an intuitive and responsive interface for users and analysts. Users can utilize the platform to submit data for analysis and view detection results, while analysts employ a dashboard to monitor threats, configure detection parameters, and manage alerts in real-time. The backend, built with Flask, serves as the processing hub, managing API requests, data processing, and interactions with the data storage. MongoDB is employed for storing authentication data, system metadata, and analysis results, offering high performance and scalability.

At the core of the methodology lies the implementation of multi-layered detection measures to safeguard user information and the integrity of digital transactions and communications. The detection process combines rule-based systems with machine learning-based anomaly detection, ensuring both known and novel fraud attempts are identified. Advanced analytical techniques, including predictive modelling, image analysis, and natural language processing, are used to detect patterns and anomalies indicative of fraud. Machine learning models, such as Random Forest, are utilized for their effectiveness in classification tasks. This robust security framework mitigates the risks of unauthorized access, tampering, and fraud.

The development process includes rigorous testing and validation to ensure the system's reliability and robustness. Unit testing is conducted for individual components like data ingestion and anomaly detection modules, while integration testing ensures smooth communication between the frontend, backend, and database. Performance testing evaluates the system's scalability under peak loads, ensuring it can handle large data volumes efficiently. Security testing identifies and mitigates vulnerabilities, ensuring the protection of sensitive data and compliance with regulatory standards. These testing phases are critical for guaranteeing the system's reliability and efficiency in real-world scenarios.

Finally, the system is deployed on a scalable platform to ensure accessibility and high availability. Users and analysts are provided with user guides and training to facilitate smooth adoption. Post-deployment, the system undergoes regular maintenance and updates to address emerging threats and incorporate feedback from stakeholders. Continuous monitoring ensures the platform remains secure and operational. This methodology demonstrates the system's commitment to leveraging cutting-edge technologies to revolutionize fraud detection, ensuring secure, transparent, and trustworthy digital interactions for all stakeholders.

5.1 Architecture Diagram

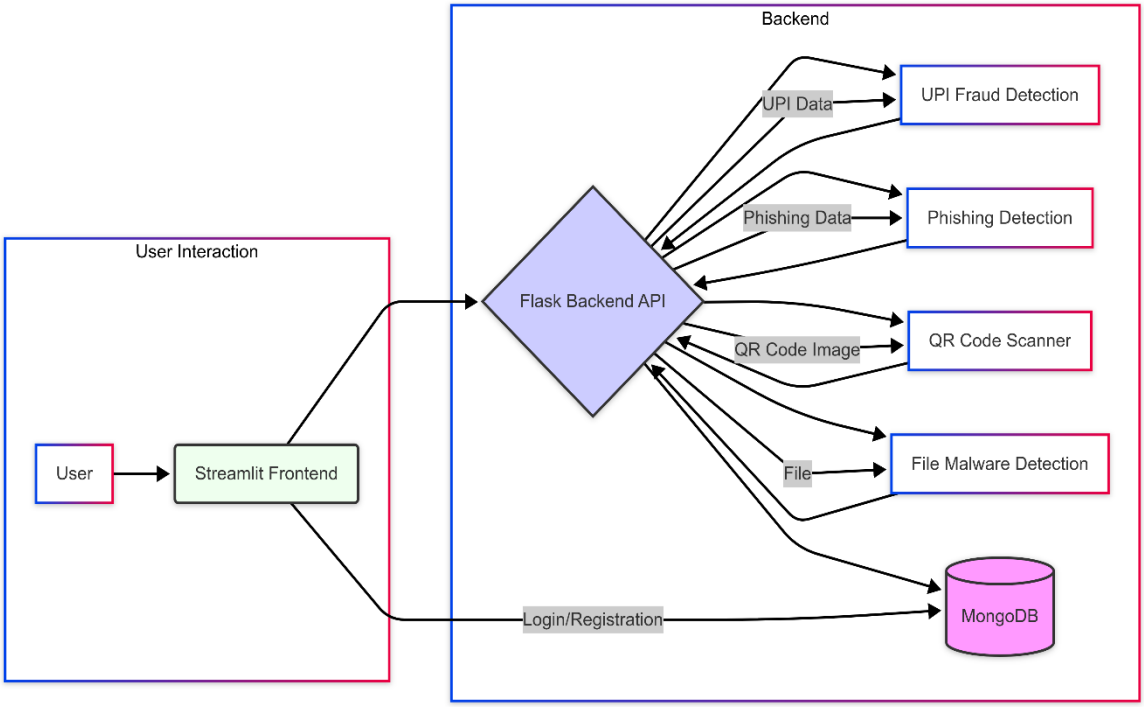


Figure 5.1 Architecture Diagram

The provided architecture diagram outlines a multi-domain fraud detection system that integrates various technologies to ensure the secure and reliable identification of fraudulent activities. It comprises key components such as the User Interaction interface, Backend processing, and specialized detection modules, working in concert to provide a comprehensive fraud detection solution.

The process begins with the **User Interaction** segment, which serves as the primary touchpoint for users and analysts. Users access the system through the Streamlit Frontend, a user-friendly interface. This interface provides features for interacting with the system, submitting data for analysis, and viewing detection results. The interface is designed to ensure accessibility and ease of use, catering to users with varying levels of technical expertise.

Next is the **Backend**, which acts as the core processing unit. The Streamlit Frontend communicates with the Flask Backend API, which manages the flow of data and requests within the system. The Flask Backend API is responsible for routing data to the appropriate detection modules and coordinating the overall fraud detection process. It handles requests from the frontend and communicates with the MongoDB database. The Backend incorporates several key modules:

- **UPI Fraud Detection**: Analyses UPI transaction data to detect fraudulent activities.
- **Phishing Detection**: Scrutinizes data to identify phishing attempts.
- **QR Code Scanner**: Processes QR code images to detect fake or malicious codes.
- **File Malware Detection**: Examines files to identify potential malware.
- **MongoDB**: Stores data related to user authentication and other system data.

The system also includes an **Authentication Flow**, where user login and registration data is securely managed through the MongoDB database. This ensures that only authorized users can access the system's functionalities. The **Fraud Analysis Flow** routes data from various sources (UPI, Phishing, QR Code, File) through the Flask Backend API to detection modules, which analyze the data and send results back to the API. In summary, this architecture uses user-friendly interfaces, robust backend processing, and specialized detection modules for multi-faceted, scalable, and efficient fraud detection.

5.2 Class Diagram

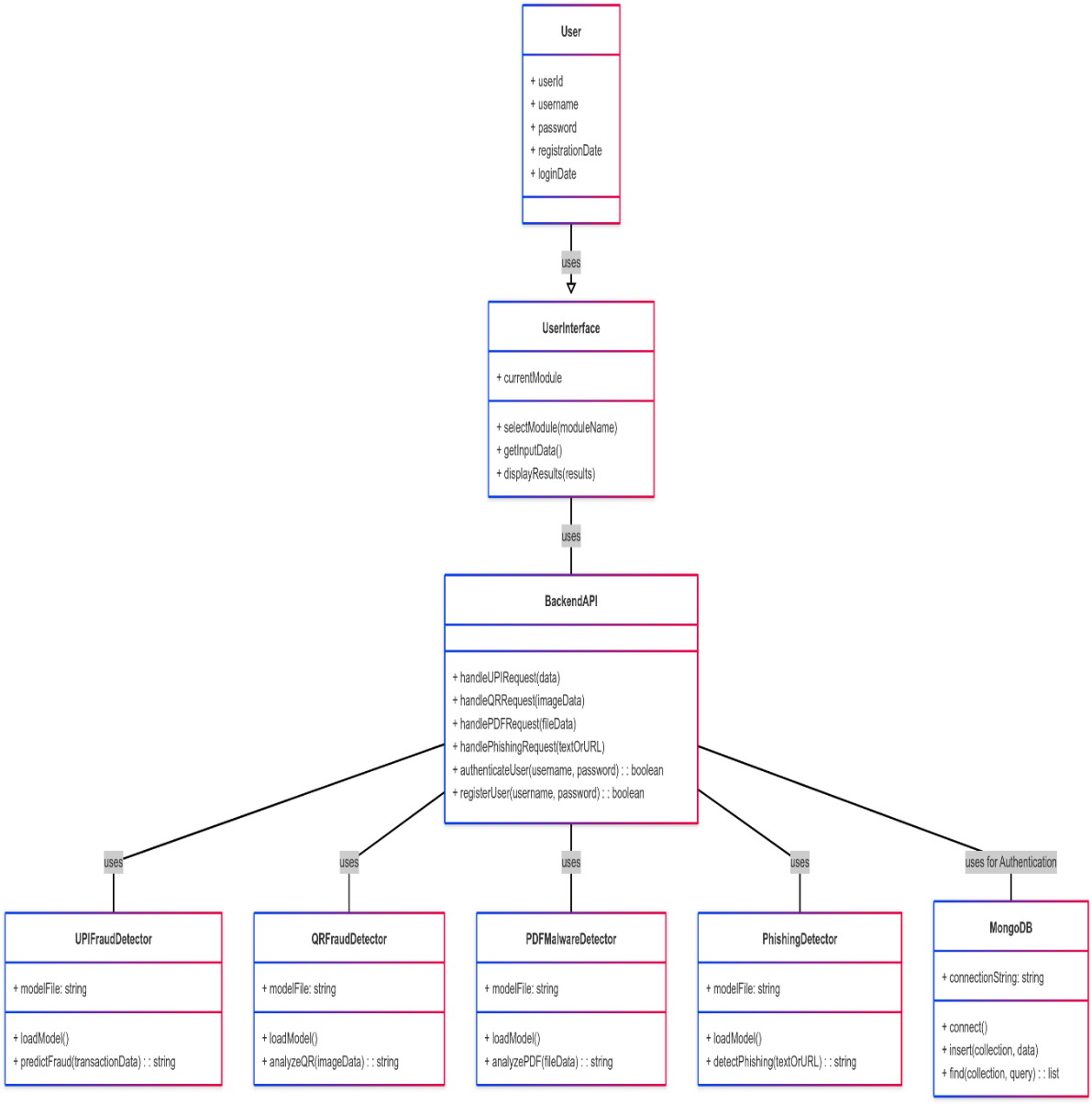


Figure 5.2 Class Diagram

The class diagram represents the design of a multi-domain fraud detection system, illustrating the relationships and interactions between its key components. It outlines the structure of the system in terms of classes, their attributes, and the operations they perform, providing a blueprint for the software's implementation.

The system comprises several core classes, each with specific responsibilities:

- **User**: Represents a system user, storing information such as their unique identifier, username, password, registration date, and last login date.

- **UserInterface**: Defines how users interact with the system. It manages the current module, allows users to select modules (e.g., UPI Fraud Detection), retrieves input data, and displays results.
- **BackendAPI**: Serves as the central processing unit, handling requests from the UserInterface. It manages the flow of data between different modules, processes detection requests (e.g., for UPI, QR code, PDF, and phishing fraud), and handles user authentication and registration.
- **UPIFraudDetector, QRFraudDetector, PDFMalwareDetector, and PhishingDetector:** These classes are responsible for detecting specific types of fraud. Each class includes attributes for storing the model file used for detection, methods for loading the model, and methods for performing the actual fraud detection analysis.
- **MongoDB**: Represents the database used for storing user data and other system-related information. It defines how the system connects to the database, inserts data, and retrieves data using queries.

The diagram also illustrates the relationships between these classes:

- The User class uses the UserInterface class to interact with the system.
- The UserInterface class uses the BackendAPI class to send requests and receive data.
- The BackendAPI class uses the various detector classes (UPIFraudDetector, QRFraudDetector, PDFMalwareDetector, and PhishingDetector) to perform fraud detection.
- The BackendAPI class uses the MongoDB class for user authentication.

In summary, the class diagram illustrates a modular and well-structured system meticulously designed for multi-domain fraud detection. It clearly delineates the interactions between distinct components, such as user interfaces for various fraud types and a central backend for processing and coordinating detection tasks. This architectural blueprint underscores a cohesive and effective solution capable of addressing a diverse range of fraudulent activities through specialized detection modules and a unified data management strategy.

5.3 Activity Diagram

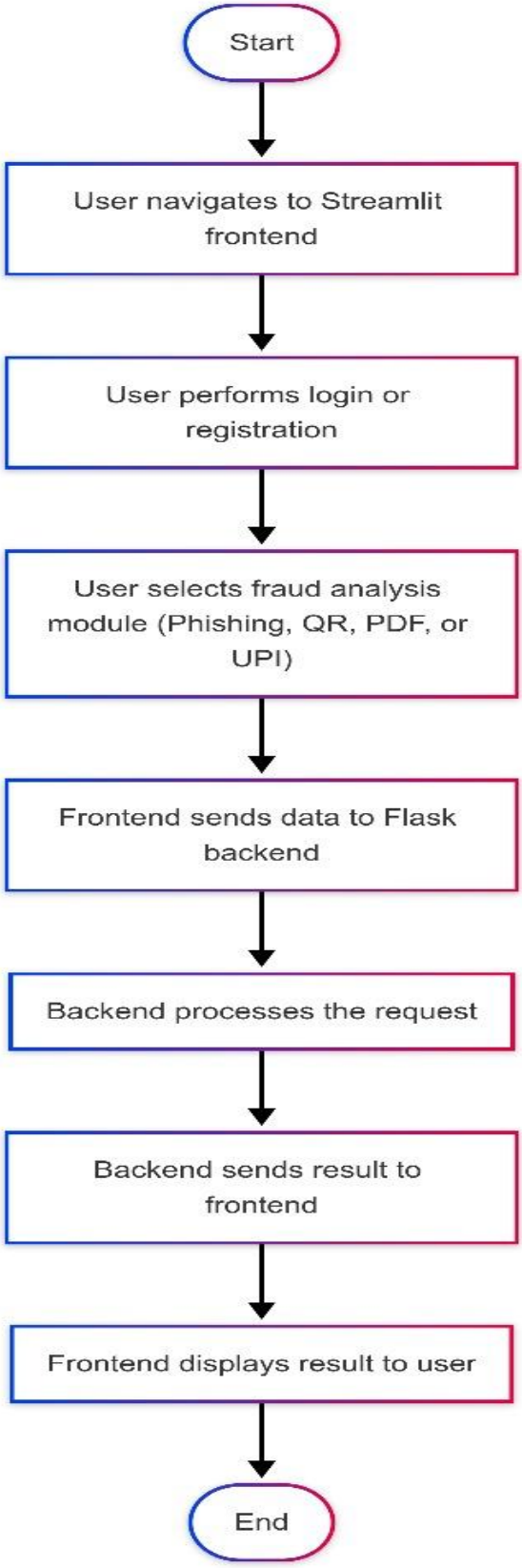


Figure 5.3 Activity Diagram

This activity diagram details the user-centric workflow of your multi-domain fraud detection system, emphasizing clarity and efficiency from initial interaction to the final analysis outcome.

The process begins with the **User Navigating to the Streamlit Frontend**, a user-friendly interface serving as the gateway to initiate fraud analysis across diverse domains. Next, the **User Performs Login or Registration**, a vital security measure ensuring only authorized individuals can access the system's capabilities. This step should incorporate strong authentication protocols to safeguard against unauthorized use.

Upon successful authentication, the **User Selects a Fraud Analysis Module (Phishing, QR, PDF, or UPI)**, tailoring the system's focus to the specific type of potential fraud. This modularity allows for the application of specialized detection techniques optimized for each domain. Subsequently, the **Frontend Sends Data to the Flask Backend**, a critical stage where user-provided information (like URLs, images, files, or transaction details) is securely transmitted for analysis. Employing secure communication channels is paramount to protect data integrity and confidentiality during this transfer.

The **Backend Processes the Request**, acting as the system's intelligent core. Upon receiving the data, it directs the information to the appropriate fraud detection module (UPIFraudDetector, QRFraudDetector, PDFMalwareDetector, or PhishingDetector). Each of these modules then executes its specific analytical algorithms to identify potential fraudulent indicators. For instance, the PhishingDetector examines content for malicious signs, while the PDFMalwareDetector scans files for embedded threats.

Following the analysis, the **Backend Sends the Analysis Result to the Frontend**, securely transmitting the outcome – whether the analyzed data is deemed fraudulent or legitimate – back to the user interface. Finally, the **Frontend Displays the Result to the User** in an easily understandable format. This immediate feedback empowers users with actionable intelligence regarding potential fraud, enabling them to respond appropriately.

In essence, this activity diagram portrays a well-structured and efficient system for multi-domain fraud detection. Each step, from initial user access to the final display of results, is crucial for delivering effective analysis and a positive user experience, while underlying security measures ensure the system's reliability and trustworthiness.

5.4 Use Case Diagram

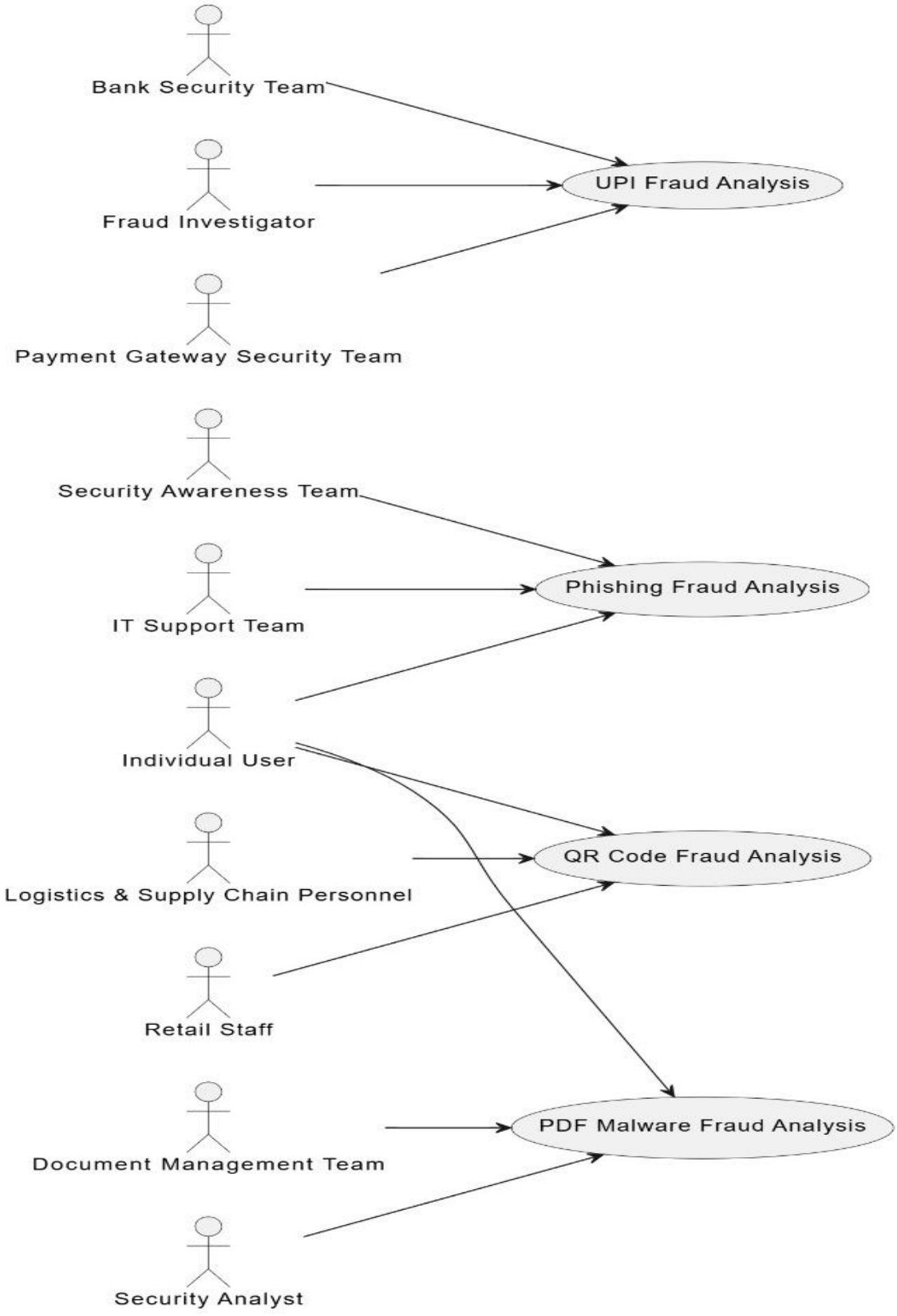


Figure 5.4 Use Case Diagram

This use case diagram illustrates the different actors (users or systems) that interact with your multi-domain fraud detection system and the specific use cases (functionalities) they utilize. It provides a clear overview of who uses which part of the system.

Actors: The stick figures on the left represent the actors interacting with the system:

- **Bank Security Team:** This actor represents personnel from the bank responsible for monitoring and analyzing UPI transactions for potential fraud.
- **Fraud Investigator:** Individuals dedicated to investigating suspected fraudulent activities, likely across multiple domains.
- **Payment Gateway Security Team:** The team responsible for ensuring the security of payment gateway transactions, with a focus on UPI fraud in this context.
- **Security Awareness Team:** This team likely uses the phishing analysis capabilities to identify and understand potential phishing threats to educate users.
- **IT Support Team:** The technical team that might use the phishing analysis tools for troubleshooting or investigating reported issues.
- **Individual User:** Represents the general public who might use the system to analyze potential phishing attempts, QR code scams, or PDF malware.
- **Logistics & Supply Chain Personnel:** Individuals in logistics who might encounter and need to analyze potentially malicious QR codes in their operations.
- **Retail Staff:** Employees in retail settings who might interact with QR codes and need to verify their legitimacy.
- **Document Management Team:** This team is responsible for handling and securing PDF documents and would utilize the PDF malware analysis feature.
- **Security Analyst:** Professionals who analyze various types of potential threats, including PDF malware, to ensure system security.

Use Cases: The ovals on the right represent the specific functionalities or use cases offered by the system:

- **UPI Fraud Analysis:** This use case allows authorized actors (Bank Security Team, Fraud Investigators, Payment Gateway Security Team) to analyze UPI transactions for suspicious patterns and potential fraud.
- **Phishing Fraud Analysis:** This use case enables actors (Security Awareness Team, IT Support Team, Individual User) to analyze emails or URLs to determine if they are likely phishing attempts.

- **QR Code Fraud Analysis:** This use case allows actors (Individual User, Logistics & Supply Chain Personnel, Retail Staff) to analyze QR codes to detect potential malicious links or content.
- **PDF Malware Analysis:** This use case enables actors (Individual User, Document Management Team, Security Analyst) to upload and analyze PDF files to identify potential malware.

Relationships: The arrows connecting the actors to the use cases indicate which actors can utilize which functionalities:

- Multiple actors can interact with the same use case. For example, both the Bank Security Team and Fraud Investigators can use the UPI Fraud Analysis.
- A single actor can interact with multiple use cases. For example, an Individual User might use the system to analyze phishing emails, suspicious QR codes, and potentially malicious PDF files.

In essence, this use case diagram clearly maps the roles of different stakeholders in utilizing your multi-domain fraud detection system for their specific needs, highlighting the system's versatility in addressing various types of online fraud.

CHAPTER 6

IMPLEMENTATION

6.1 Database

Overview

The database is a key component of the Threat Guard AI system, designed to manage user-submitted data and the corresponding results of phishing, QR code fraud, and malware scans. It allows for the secure storage, retrieval, and auditing of analysis results. The database stores important metadata, including timestamps, user information (with consent), scan results, and system logs, ensuring compliance and providing valuable insights for further improvements.

Key Components:

1. Data Storage:

- The database stores different types of data for phishing detection, QR code analysis, and malware scans.
- For phishing detection, URLs are stored along with analysis results, such as whether the URL is flagged as phishing, the confidence level, and reasons for classification.
- For QR code fraud detection, the system stores QR code contents (URLs, embedded data) and their analysis results.
- For malware scans, files are stored temporarily for analysis, and their scan results are logged along with malware signatures detected, if any.

2. Data Consistency:

- The database ensures that data is consistent, particularly when interacting with the blockchain or backend models. It is crucial that any changes in user data or detection results are promptly reflected in the database, ensuring data integrity across different system components.

3. Backup & Recovery:

- Regular backups of the database ensure that critical data can be restored in case of failure, reducing the risk of data loss.
- The database is replicated across different nodes or data centers to improve availability and reliability.

Implementation Challenges:

1. **High Volume of Data:** Managing large-scale submissions for phishing, QR code scans, and malware scans, especially in a global system with multiple users submitting data at the same time.
2. **Real-Time Updates:** Providing real-time results back to users with minimal latency after submission of URLs, QR codes, or files for analysis.
3. **Data Redundancy & Availability:** Ensuring high availability and preventing data loss with redundant databases, particularly for election-critical information or high-risk data such as phishing detections.

6.2 Phishing Detection (Using Google Gemini AI)

Overview:

The phishing detection system uses **Google Gemini AI** to classify URLs as phishing or legitimate. This system integrates an AI-powered model that is trained to detect subtle indicators of phishing websites, such as misspelled domains, lack of SSL certificates, suspicious forms, and fraudulent login prompts. This process helps protect users from malicious websites that may attempt to steal personal information.

Key Components:

1. URL Analysis:

- The user inputs a URL through a form on the frontend interface. The backend processes the URL and sends it to **Google Gemini AI** for classification.
- The model analyzes the URL, evaluating characteristics like domain authenticity, SSL encryption, and the presence of suspicious elements (e.g., login forms or fake pop-ups).

2. Explanation of Results:

- After receiving the results, the system explains the reasons for labeling a URL as phishing or safe. This transparency helps build trust in the system.
- For instance, if a URL is marked as phishing, reasons such as "Suspicious domain" or "No SSL certificate" are displayed to the user.

3. Confidence Levels:

- The model provides a confidence score indicating the likelihood of a URL being phishing. This score helps users assess the severity of the risk.

Implementation Challenges:

1. **Complex URL Structures:** Modern phishing websites can use complex URL structures that are difficult for simple heuristics to detect. The model needs to continually learn and adapt to these new patterns.
2. **API Limitations:** Integrating **Google Gemini AI** effectively requires handling rate limits and ensuring the system responds in a timely manner even during high traffic periods.

6.3 QR Code Fraud Detection

Overview:

The QR code fraud detection system addresses the growing risk of malicious QR codes that redirect users to phishing websites or initiate unwanted actions like financial transactions. This system scans QR codes for hidden threats and ensures that users are not tricked into interacting with malicious QR codes.

Key Components:

1. QR Code Scanning:

- Users can either scan or upload QR codes for analysis.

2. Fraud Detection Algorithms:

- A combination of **machine learning models** and rule-based heuristics is applied to detect common fraud patterns in QR codes, such as links to known phishing websites, embedded malware, or unauthorized financial transactions.

Implementation Challenges:

1. **Fast QR Code Scanning:** Processing QR codes rapidly to minimize user wait time is critical for maintaining a smooth user experience, especially in real-time scenarios.
2. **Accurate Detection:** Ensuring that the fraud detection models do not flag legitimate QR codes as malicious while also preventing false negatives.

6.4 Malware Scan

Overview:

The **Malware Scan** feature enables users to upload files for malware analysis, allowing them to check for potential security threats in documents, executables, and other file types. This feature uses powerful antivirus engines and behavioral analysis tools to detect and report malicious activity.

Key Components:

1. File Upload Interface:

- The user uploads files through a web interface, where the system checks the file type, size, and security status before scanning.

2. Malware Detection Engines:

- The backend utilizes **ClamAV**, **VirusTotal**, or custom malware detection engines to scan files for known malware signatures, behaviors, or patterns indicative of malicious activity.

3. Results Reporting:

- After scanning, the system provides users with detailed reports that indicate whether a file is clean, potentially harmful, or definitely infected. The report also includes recommendations for remediation.

4. File Storage & Handling:

- Uploaded files are temporarily stored and then scanned to ensure no malicious content is present. Files are removed after the analysis to reduce the risk of data exposure.

Implementation Challenges:

1. **Large File Handling:** Files larger than typical submissions require special handling and may take longer to scan, impacting user experience.
2. **Real-Time Scanning:** Ensuring that malware scans are performed quickly, providing results in a timely manner for users.
3. **False Positives/Negatives:** Striking the balance between flagging legitimate files as malicious and missing truly harmful files is crucial.

6.5 Fraud Detection

Overview:

The **Fraud Detection Module** is designed to identify and prevent fraudulent financial transactions—particularly UPI-based scams. With the increase in digital payments through QR codes and UPI apps, it is essential to monitor transaction patterns, detect anomalies, and prevent unauthorized activities. This module uses machine learning algorithms to flag suspicious activities in real-time and ensure the safety of users and merchants.

Key Components:

1. User Transaction Analyzer:

- Collects and processes transaction details such as sender ID, receiver ID, transaction amount, time, frequency, location, and device ID.

2. Machine Learning Model:

- Trained on historical fraud patterns, the ML model classifies incoming transactions as legitimate or suspicious or fraudulent.

3. Alerting System:

- If a transaction is flagged as suspicious, the system generates an instant alert for the user and logs it for further investigation.

4. Audit & Logging Service:

- All detected fraud attempts are logged with metadata for legal, administrative, or analytical purposes.

Implementation Challenges:

1. **Class Imbalance:** Fraudulent transactions are rare compared to normal ones, making training challenging. The model uses techniques like SMOTE or cost-sensitive learning.
2. **False Positives:** Preventing legitimate transactions from being blocked or flagged unnecessarily, which can affect user experience.
3. **Data Privacy & Security:** Ensuring sensitive financial data is handled securely in compliance with regulations.
4. **Real-Time Scalability:** The system needs to process and evaluate transactions in milliseconds for real-time prevention without delays.

6.6 Security & Privacy

Overview:

Threat Guard AI employs a robust security framework to protect users' data and the integrity of the system. The security system integrates encryption, secure authentication, and compliance with relevant data protection regulations like GDPR to protect both user data and detection results.

Key Components:

1. Data Encryption:

- End-to-end encryption ensures that sensitive is transmitted securely between the frontend and backend.

2. Authentication Mechanisms:

- Multi-factor authentication is used to protect user accounts and ensure only authorized users can access specific system features.

3. Privacy Compliance:

- The system ensures that it complies with all relevant privacy laws, storing user data only when necessary and providing users with the ability to delete their data upon request.

Implementation Challenges:

1. **Vulnerability Prevention:** Protecting the system against common attacks while ensuring the security of user data.

2. **GDPR Compliance:** Ensuring that the system adheres to all data privacy regulations, providing users with transparent control over their data.

6.7 Home Page Implemenetation Code:

```
def home_page():
set_custom_style()
st.title("THREAT GUARD AI SYSTEM v2.4.1")

# Header with logo and pulse animation
st.markdown(f"""
<div style="text-align: center; margin-bottom: 2rem;">
<img src="{LOGOS["home"]}" class="logo pulse" style="width: 50px;">
</div>
""", unsafe_allow_html=True)

st.markdown("""
<div class="terminal">
> Initializing system...<br>
> Loading modules...<br>
> Establishing secure connection...<br>
> System ready.<br><br>
> Welcome to UPI Fraud Detection System<br>
> Comprehensive security for digital transactions<br>
> Type 'help' for commands
</div>
""", unsafe_allow_html=True)

# Module showcase section with images
st.subheader("MAIN SECURITY MODULES")

# Create 4 columns for the module images
img_col1, img_col2, img_col3, img_col4 = st.columns(4)

with img_col1:
st.image("images/phishing.png",
caption="Phishing Detection", width=130)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Detect fake UPI payment pages
</div>
""", unsafe_allow_html=True)
```

```
with img_col2:
st.image("images/malwarescan.webp",
caption="File Threat Analysis", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Scan malicious PDF/DOCX files
</div>
""", unsafe_allow_html=True)

with img_col3:
st.image("images/qr_scan.webp",
caption="QR Code Scanner", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Detect fraudulent QR codes
</div>
""", unsafe_allow_html=True)

with img_col4:
st.image("images/fraud.jpg",
caption="Fraud Detection", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Analyze suspicious transactions
</div>
""", unsafe_allow_html=True)

# Interactive cards for navigation
col1, col2 = st.columns(2)

with col1:
# File Threat Analysis Card
if st.button("FILE THREAT ANALYSIS", key="file_scan_btn",
use_container_width=True):
st.session_state.current_page = "📁 FILE SCAN"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo('📁 FILE SCAN')">
<div class='logo-container'>
<img src="{LOGOS["file_scan"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>FILE THREAT ANALYSIS</div>
<div class='card-desc'>Advanced scanning for PDF/DOCX files with heuristic
analysis</div>
</div>
""", unsafe_allow_html=True)

# Phishing Detection Card
if st.button("PHISHING DETECTION", key="phishing_btn",
use_container_width=True):
```

```
st.session_state.current_page = " 🎣 PHISHING DETECTION"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo(' 🎣 PHISHING DETECTION')">
<div class='logo-container'>
<img src="{LOGOS["phishing"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>PHISHING DETECTION</div>
<div class='card-desc'>Identify and block phishing attempts targeting UPI users</div>
</div>
""", unsafe_allow_html=True)

with col2:
# QR Code Analysis Card
if st.button("QR CODE ANALYSIS", key="qr_scan_btn", use_container_width=True):
st.session_state.current_page = " 📷 QR SCAN"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo(' 📷 QR SCAN')">
<div class='logo-container'>
<img src="{LOGOS["qr_scan"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>QR CODE ANALYSIS</div>
<div class='card-desc'>Secure scanning of payment QR codes with tamper detection</div>
</div>
""", unsafe_allow_html=True)

# Fraud Detection Card
if st.button("FRAUD DETECTION", key="fraud_btn", use_container_width=True):
st.session_state.current_page = " 💸 FRAUD DETECTION"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo(' 💸 FRAUD DETECTION')">
<div class='logo-container'>
<img src="{LOGOS["fraud"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>FRAUD DETECTION</div>
<div class='card-desc'>Analyze transactions for suspicious patterns and fraud</div>
</div>
""", unsafe_allow_html=True)

# Add JavaScript for navigation
st.markdown("""
<script>
// Function to handle card clicks
function navigateTo(page) {
window.streamlitSessionState.set('current_page', page);
}
</script>
""", unsafe_allow_html=True)
with img_col2:
```

```
st.image("images/malwarescan.webp",
caption="File Threat Analysis", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Scan malicious PDF/DOCX files
</div>
""", unsafe_allow_html=True)

with img_col3:
st.image("images/qr_scan.webp",
caption="QR Code Scanner", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Detect fraudulent QR codes
</div>
""", unsafe_allow_html=True)

with img_col4:
st.image("images/fraud.jpg",
caption="Fraud Detection", width=120)
st.markdown("""
<div style="text-align: center; font-size: 0.9rem; margin-top: -1rem;">
Analyze suspicious transactions
</div>
""", unsafe_allow_html=True)

# Interactive cards for navigation
col1, col2 = st.columns(2)

with col1:
# File Threat Analysis Card
if st.button("FILE THREAT ANALYSIS", key="file_scan_btn",
use_container_width=True):
st.session_state.current_page = "📁 FILE SCAN"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo('📁 FILE SCAN')">
<div class='logo-container'>
<img src="{LOGOS["file_scan"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>FILE THREAT ANALYSIS</div>
<div class='card-desc'>Advanced scanning for PDF/DOCX files with heuristic
analysis</div>
</div>
""", unsafe_allow_html=True)

# Phishing Detection Card
if st.button("PHISHING DETECTION", key="phishing_btn",
use_container_width=True):

# Fraud Detection Card
```

```
if st.button("FRAUD DETECTION", key="fraud_btn", use_container_width=True):
st.session_state.current_page = " 💸 FRAUD DETECTION"
st.rerun()
st.markdown(f"""
<div class='card' onclick="navigateTo(' 💸 FRAUD DETECTION')">
<div class='logo-container'>
<img src="{LOGOS["fraud"]}" class="logo" style="width: 150px;">
</div>
<div class='card-title'>FRAUD DETECTION</div>
<div class='card-desc'>Analyze transactions for suspicious patterns and fraud</div>
</div>
""", unsafe_allow_html=True)

# Add JavaScript for navigation
st.markdown("""
<script>
// Function to handle card clicks
function navigateTo(page) {
window.streamlitSessionState.set('current_page', page);
}
</script>
""", unsafe_allow_html=True)
st.markdown("""
<div class="terminal" style='margin-top: 2rem;'>
> SYSTEM FEATURES:<br>
> Real-time transaction monitoring<br>
> Advanced anomaly detection<br>
> Secure QR validation<br>
> Document malware scanning<br>
> Encrypted communication<br><br>
> Last system update: 2023-11-15<br>
> Security level: MAXIMUM<br>
> Connection: SECURE
</div>
""", unsafe_allow_html=True)
```

CHAPTER 7

TEST CASES

Test Case 1: User Registration & Login

Scenario	Input Details	Expected Outcome	Pass/Fail
User Registration	Valid email, password, UPI ID	User registered, data saved in Firebase + MongoDB	Pass
Duplicate Email Registration	Already registered email	Show error: Email already in use	Pass
Login with Valid Credentials	Correct email and password	Redirect to homepage	Pass
Login with Invalid Credentials	Wrong password	Error: Invalid credentials	Pass

Test Case 2: File Threat Analysis

Scenario	Input Details	Expected Outcome	Pass/Fail
Safe PDF Upload	Upload normal.pdf	Green alert: No threats found	Pass
Malicious PDF Upload	Upload ransomware.pdf	Red alert: Detected ransomware based on entropy & MIME type	Pass
Suspicious File with Code	Upload macro-enabled.xlsm	Yellow alert: Embedded macros, potential threat	Pass

Test Case 3: QR Code Scanner

Scenario	Input Details	Expected Outcome	Pass/Fail
Legit QR via Camera	QR linking to https://secure.bank.com/login	Green box + “QR DETECTED: Safe.”	Pass
Malicious QR via Upload	QR linking to http://b4nk-login.xyz	Red warning. Suspicious domain detected. Phishing tags shown	Pass

Invalid Image	Upload corrupted/non-QR image	Error: "No valid QR code found."	Pass

Test Case 4: Fraud Detection

Scenario	Input Details	Expected Outcome	Pass/Fail
Clean UPI Transaction	Valid sender/receiver ID, normal amount	Result: "Clean Transaction" with confidence score	Pass
Fraudulent Transaction	Suspicious ID, large amount at odd time	Red alert: "Fraud Detected" with 90% confidence	Pass
Invalid UPI Format	Invalid UPI ID format	Error: "Invalid UPI ID."	Pass

Test Case 5: Phishing Detection

Scenario	Input Details	Expected Outcome	Pass/Fail
Safe URL	Enter https://example.com	Green alert: "No phishing detected."	Pass
Phishing URL	Enter http://fakemail-login.ru	Red alert: Risk score 85. Fake login page detected	Pass
Malformed URL	Input malformed string	Error: "Invalid URL input."	Pass

Test Case 6: Scan History Retrieval

Scenario	Input Details	Expected Outcome	Pass/Fail
View All Past Scans	Logged-in user opens scan history	Table loads with previous scans and metadata	Pass
Empty History	New user with no scan activity	Message: "No scan history available."	Pass

Test Case 7: Admin Access Control

Scenario	Input Details	Expected Outcome	Pass/Fail
Admin Login	Admin credentials	Access to Dashboard, Model Management, and Fraud Analysis	Pass
User Tries Admin Access	Normal user opens admin URLs	Access Denied: "You are not authorized"	Pass

Test Case 8: Admin Dashboard Metrics

Scenario	Input Details	Expected Outcome	Pass/Fail
Metrics Load	Admin opens dashboard	Total fraud cases, graphs, and charts displayed correctly	Pass

Test Case 9: ML Model Management

Scenario	Input Details	Expected Outcome	Pass/Fail
Model Retraining	Admin clicks "Retrain Model"	New model trained, accuracy & recall updated	Pass
View Feature Importance	Admin opens chart	Features shown in descending order of importance	Pass
Evaluate Model Metrics	View confusion matrix	FP, FN, TP, TN correctly displayed	Pass

Test Case 10: Admin Fraud Analysis

Scenario	Input Details	Expected Outcome	Pass/Fail
Filter by Amount	Set amount range 10,000-50,000 INR	Transactions within range shown	Pass
Regional Fraud Trends	Filter by location	Graph highlights fraud hotspots	Pass
Drill into Case	Click suspicious transaction	Detailed breakdown shown with fraud reasons	Pass

CHAPTER 8

RESULTS AND ANALYSIS

8.1 Home Page

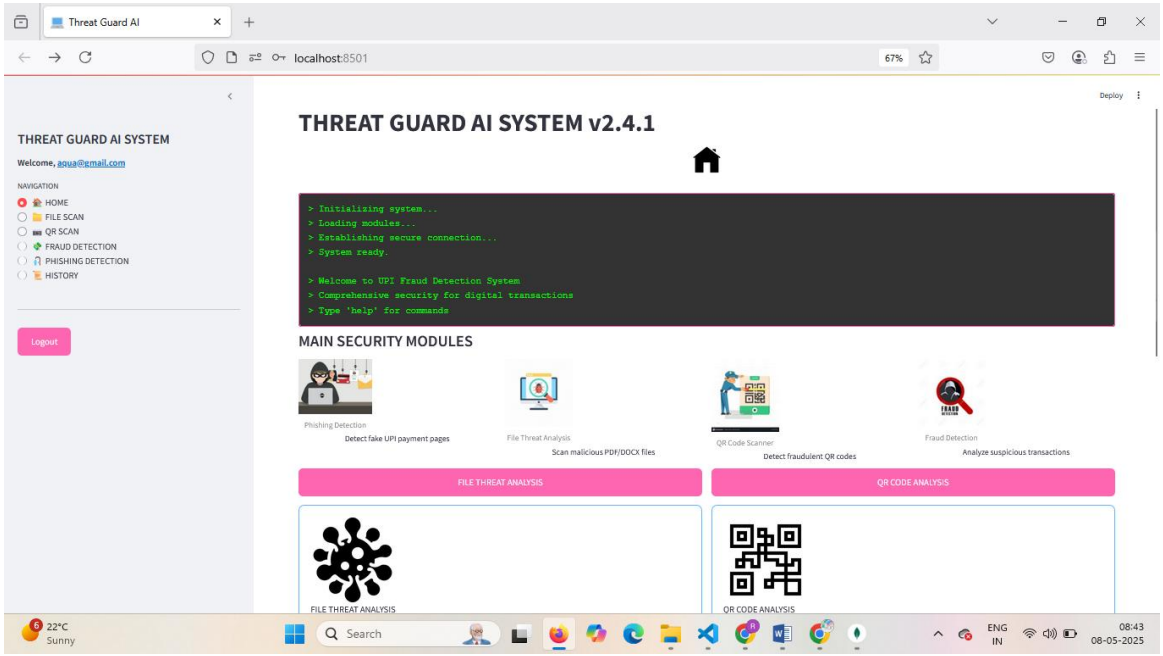


Figure 8.1 Home Page

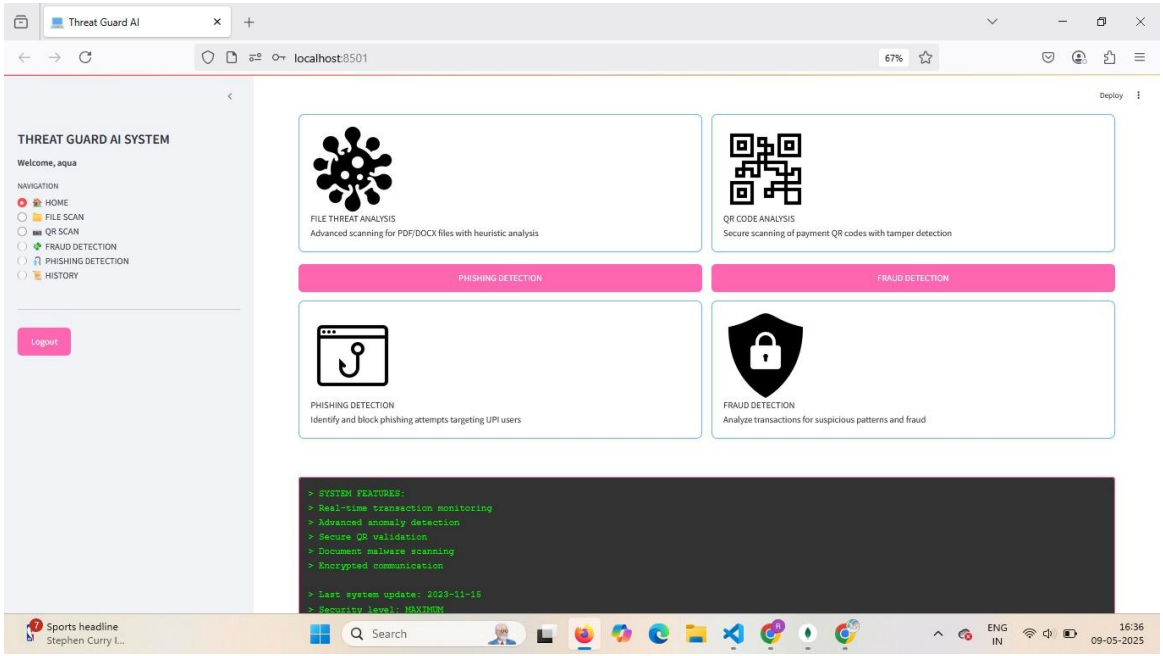


Figure 8.2 Home Page

The home page of Threat Guard AI System greets users with a sleek, high-tech interface designed to inspire confidence in its cybersecurity capabilities. The layout features a command-line inspired terminal at the top, displaying real-time system checks and initialization messages in a monospace font, reinforcing the platform's technical precision.

A glowing digital shield logo pulses at the center, symbolizing active protection, while the bold system title "THREAT GUARD AI" anchors the page with authority. Below, four interactive security modules (File Threat Analysis, QR Code Scanner, Fraud Detection, and Phishing Protection) are arranged in a clean grid, each with intuitive icons and hover animations for seamless navigation. The dark-themed backdrop, accented with neon blue and pink highlights, creates a futuristic aesthetic, while animated data particles in the background subtly convey constant threat monitoring.

The page balances advanced cybersecurity visuals with user-friendly accessibility. The terminal simulation immerses users in a security-ops environment, while the module cards provide straightforward access to core tools. A footer bar displays critical system statuses—such as "Encryption: Active" and "Threat Database Updated" offering transparency and trust. Every element, from the pulsating shield to the responsive hover effects, is designed to reassure users of robust protection while maintaining clarity.

The minimalist layout ensures effortless navigation, making sophisticated threat detection tools approachable for both experts and everyday users. The result is a powerful yet intuitive gateway into the system's security suite, blending enterprise-grade functionality with engaging design. Built on **Streamlit**, the homepage architecture blends technical sophistication with lightweight deployment. The use of Streamlit's responsive layout engine allows seamless alignment of interface components—modules are structured using st.columns and st.container blocks to maintain balance across screen sizes. The glowing digital shield and terminal-style output are rendered using a combination of HTML within Streamlit's components.html() and stylized Markdown, emulating a real-time command-line interface. The terminal feed dynamically updates system statuses, such as database connectivity and threat definition syncing, providing users with a feeling of live surveillance. Each module—File Threat Analysis, QR Code Scanner, Fraud Detection, and Phishing Detection—is not only visually distinct with hover-enabled icons but also mapped to separate functional routes handled internally

8.2 Threat Guard AI Login

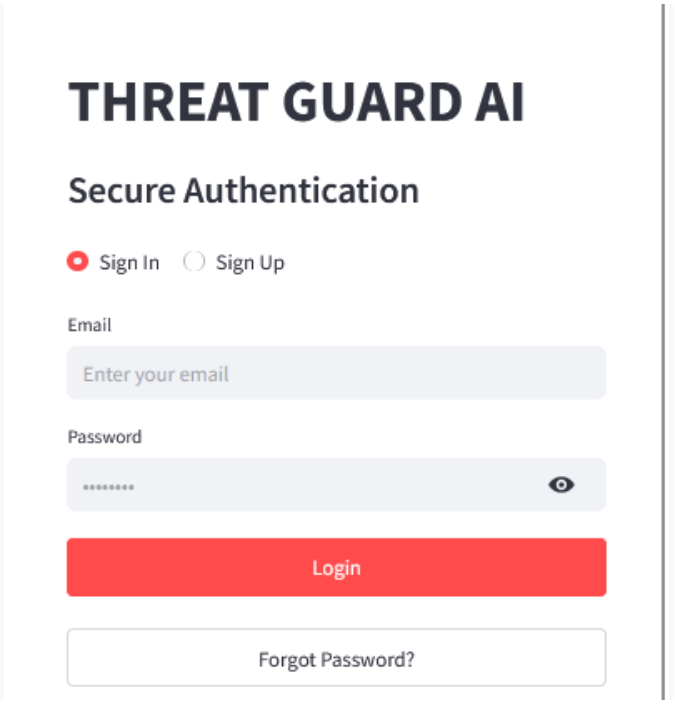


Figure 8.3 Threat Guard AI Login

The authentication system serves as the secure gateway to the Threat Guard AI platform, featuring a clean and intuitive interface designed for seamless user access. Built on Firebase Authentication, it offers robust security with email/password login, new account registration, and password recovery functionality. The dual-tab design separates login and signup flows while maintaining visual consistency, with real-time form validation ensuring proper credentials before submission. Admin users benefit from a dedicated access path with elevated privileges, while standard users gain immediate entry to the threat detection suite upon successful authentication. The system enforces password complexity requirements and includes protective measures against common vulnerabilities like credential stuffing and brute force attacks.

Security and User Experience

Designed with both security and usability in mind, the authentication portal implements industry-standard practices including secure password storage, session management, and error handling. The responsive layout works across devices, with clear visual feedback for all user actions through success messages and error alerts. The password reset flow maintains security while remaining user-friendly, requiring email verification and confirmation of new credentials. Behind the scenes, Firebase's authentication services

handle the heavy lifting of user management, allowing the frontend to focus on delivering a smooth, professional experience that matches the Threat Guard AI platform's security-focused ethos.

8.3 File Threat Analysis: Malicious

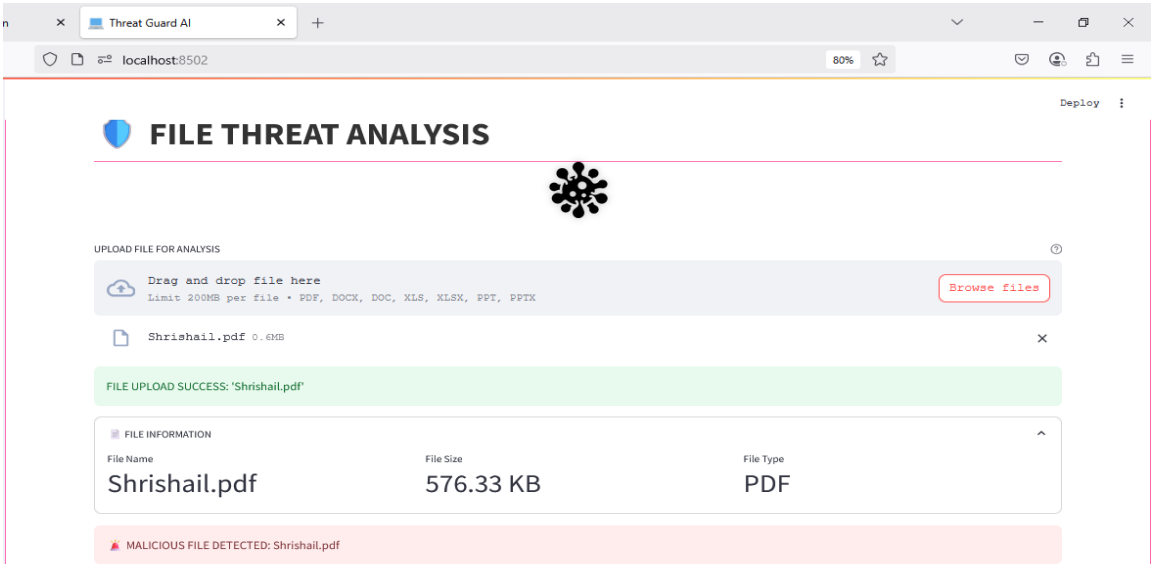


Figure 8.4 File Threat Analysis - Malicious

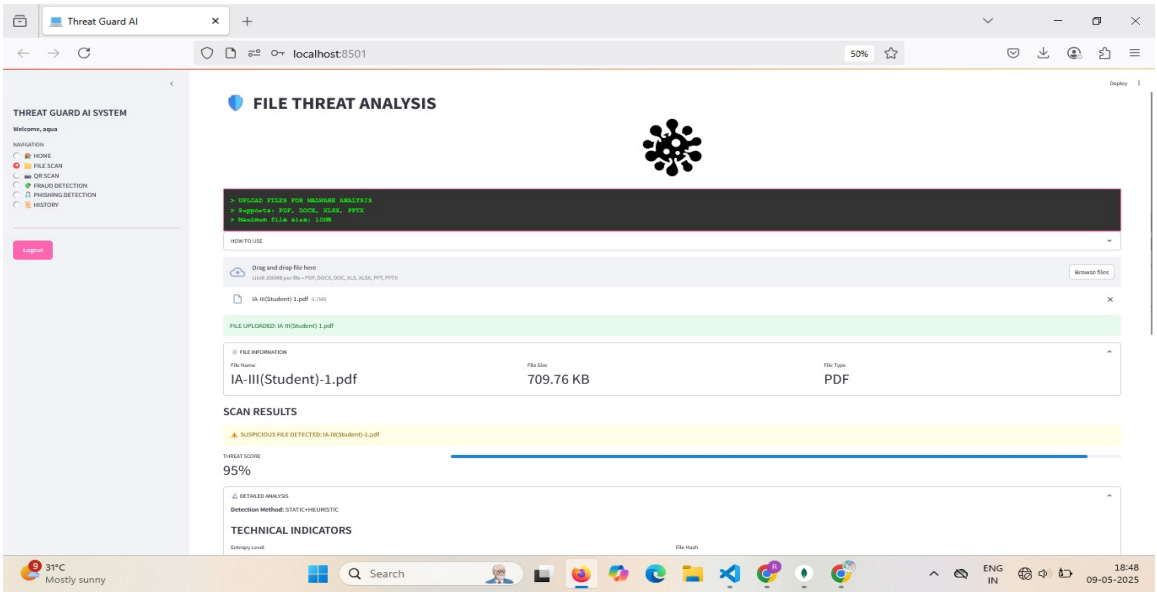


Figure 8.5 File Threat Analysis – Safe

The File Threat Analysis module provides a powerful defense against malicious documents, offering users a comprehensive scanning solution for files like PDFs, DOCX, and spreadsheets. The interface features a clean, professional design with a prominent file

upload area where users can drag and drop documents or browse their system. Once a file is submitted, the system processes it through multiple detection layers, displaying real-time progress via an animated scanning bar. The results are presented in a structured threat report panel, which highlights key metrics such as malware probability, risk level (color-coded as red, yellow, or green), and detected suspicious patterns. A detailed technical breakdown section reveals forensic data like file entropy, hash signatures, and embedded code snippets, giving advanced users deeper insights into potential threats.

User Experience and Security Features

Designed for both simplicity and depth, the module ensures quick threat assessments while allowing access to granular details when needed. The layout emphasizes clarity—clean typography, logical information hierarchy, and visual indicators (like warning icons and progress loaders) guide users effortlessly through the scanning process. Suspicious files trigger bold visual alerts and actionable recommendations (e.g.,"Quarantine Immediately"), while safe files receive a green verification badge. A collapsible scan history log tracks past analyses, enabling users to review previous threats or export reports. The combination of automated AI detection and transparent reporting balances speed with precision, making it equally effective for casual users and IT professionals. The module's design minimalist yet packed with critical details reflects the system's core goal: making advanced threat detection accessible without compromising power. The module leverages sophisticated detection techniques, including static analysis for embedded malicious code and behavioral analysis for suspicious file patterns. It employs machine learning models trained on vast datasets of known threats to identify both established malware variants and emerging zero-day exploits. The system cross-references file signatures with global threat intelligence databases in real-time, while heuristic analysis detects anomalies in file structure and content. For enterprise users, the module offers customizable scanning profiles that can be tailored to specific security policies or compliance requirements. This multi-layered approach ensures comprehensive protection against evolving document-based threats, from macro viruses and exploit code to stealthy steganographic attacks. The scanning engine is continuously updated to address new attack vectors, maintaining its effectiveness against the latest cybersecurity threats.

8.4 QR Code Scanner

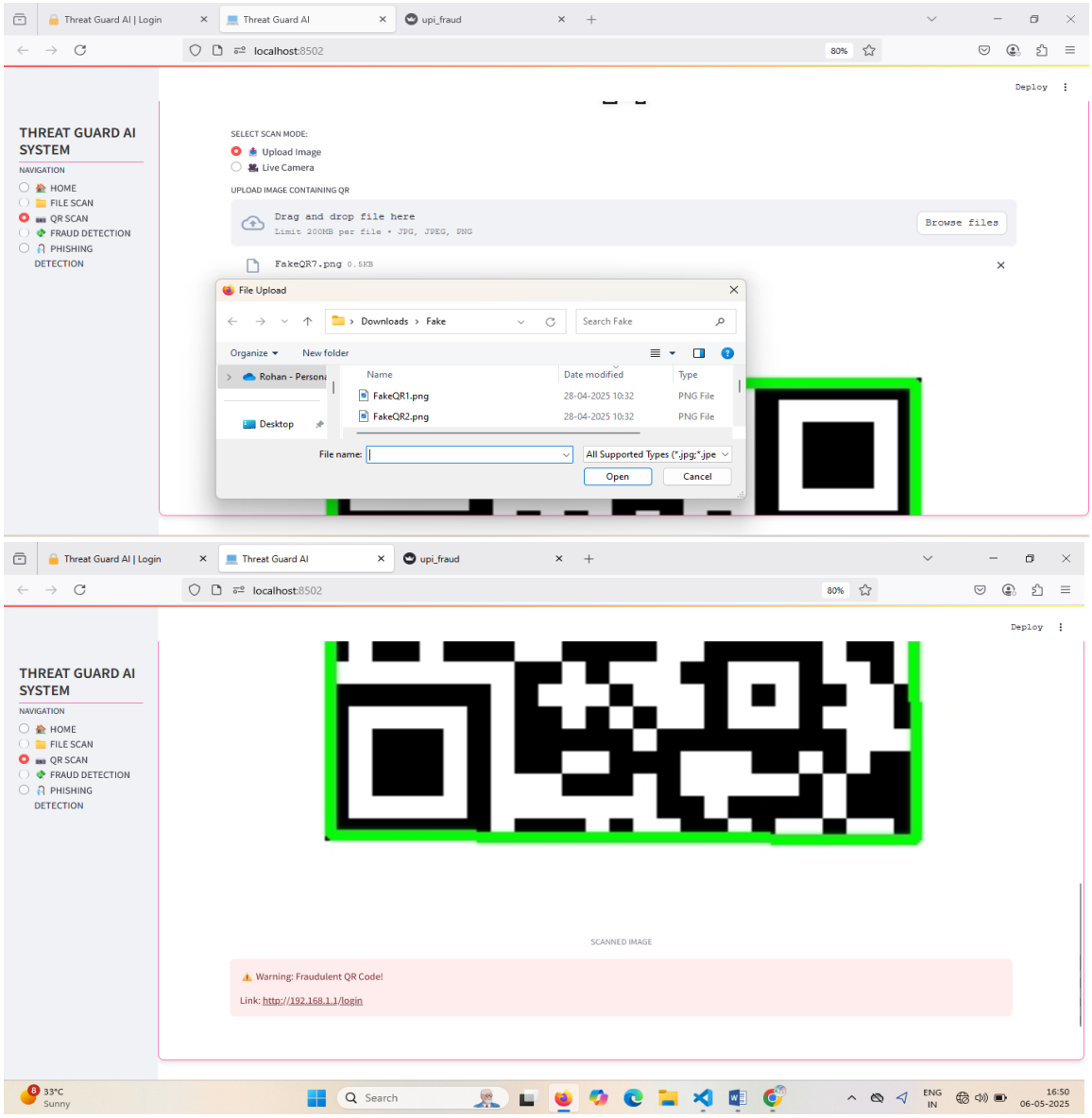


Figure 8.6 QR Code Scanner - Unsafe

The QR Code Scanner module provides a dual-mode interface for detecting potentially malicious QR codes through image uploads and live camera scanning. The clean, user-friendly interface features a prominent QR code logo and clear mode selection options. When scanning, detected QR codes are highlighted with visual annotations - legitimate codes are outlined in green with a "QR DETECTED" label, while suspicious ones trigger warning messages. The system automatically analyzes extracted URLs for phishing indicators like banking keywords or suspicious domains, displaying immediate safety assessments with color-coded alerts (green for safe, red for fraudulent).

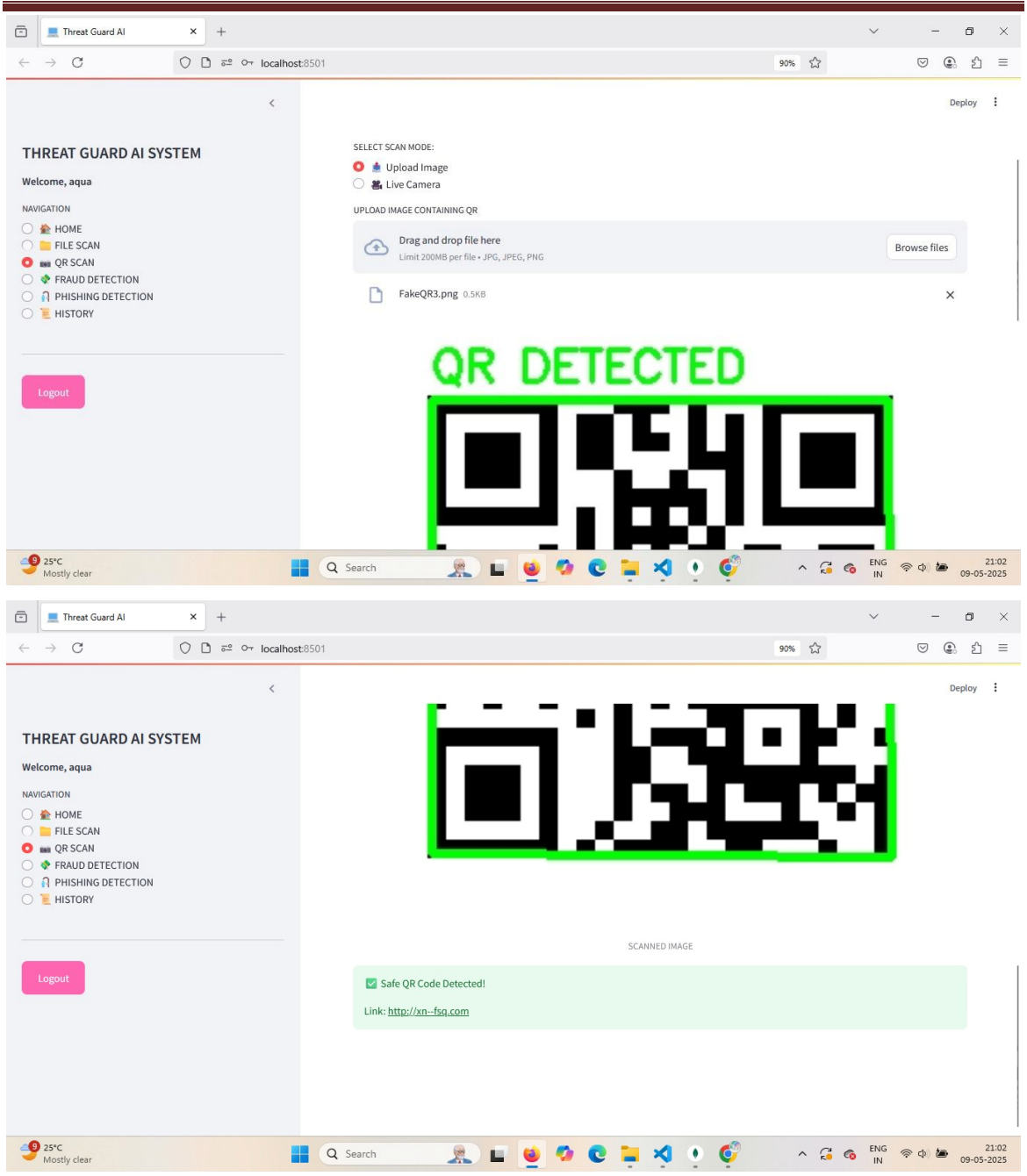


Figure 8.7 QR Code Scanner - Safe

The live camera mode offers real-time scanning with frame-by-frame analysis, while the upload option allows for examining saved QR images. A session history maintains scan results until manually cleared, enabling users to review findings. The module combines computer vision detection (OpenCV and pyzbar libraries) with heuristic analysis (keyword checks) to balance speed and accuracy. The minimalist design, featuring intuitive buttons and clear visual feedback, makes complex security scanning accessible to all users while maintaining robust protection against QR-based threats.

8.5 Fraud Detection

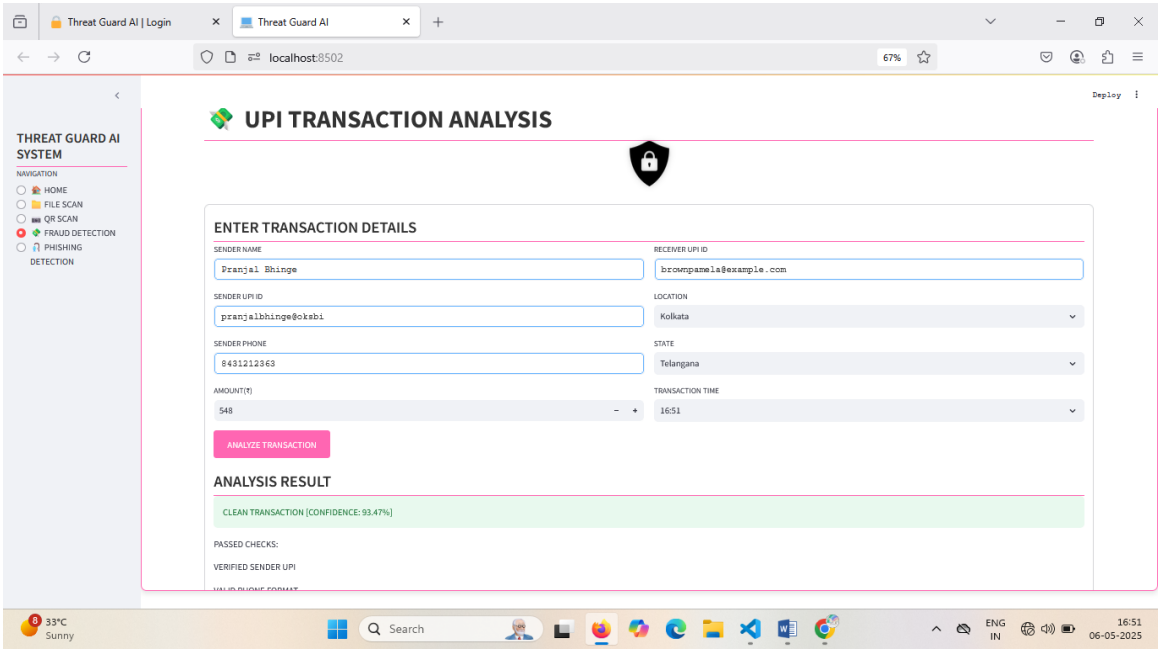


Figure 8.8 UPI Fraud Detection - Safe

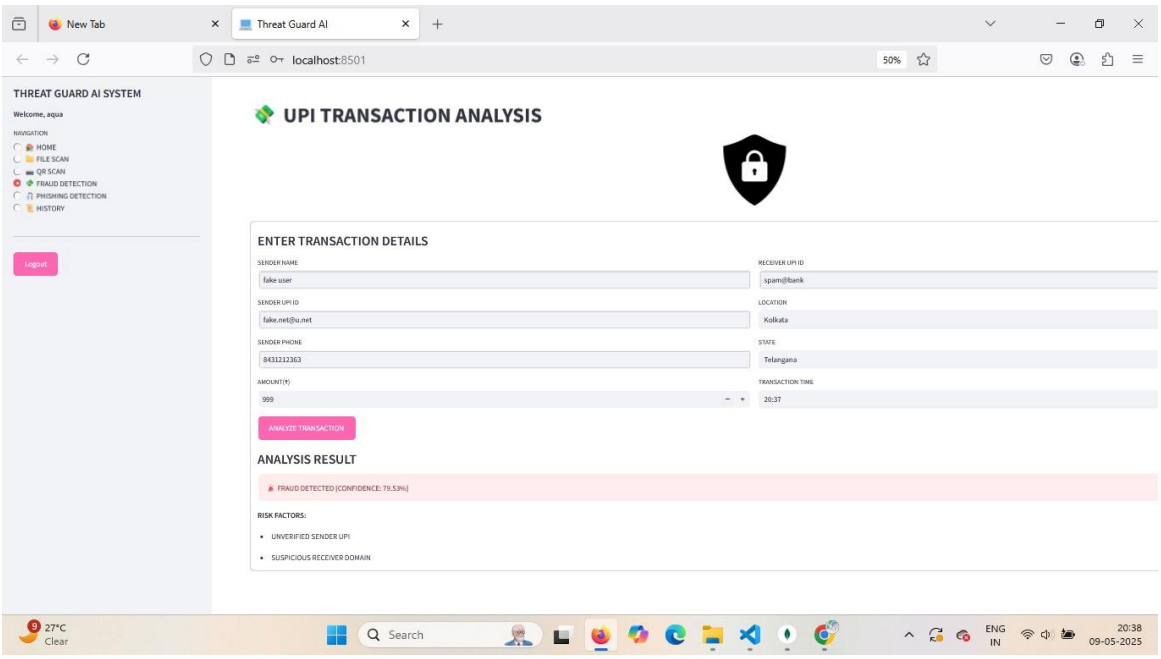


Figure 8.9 UPI Fraud Detection - Unsafe

The UPI Fraud Detection module provides a comprehensive security solution for analysing digital transactions. The interface features a clean two-column form where users input transaction details including sender/receiver UPI IDs, amount, location, and time. The system evaluates these inputs using a machine learning model trained on historical fraud

patterns, checking for multiple risk factors like unverified UPI domains, suspicious receiver addresses, invalid phone numbers, unusually high amounts, and odd-hour transactions.

When analyzing a transaction, the system provides a clear verdict ("Fraud Detected" or "Clean Transaction") along with a confidence percentage. Detected risks are listed with explanatory bullet points, while safe transactions show passed security checks. The module incorporates multiple validation layers including domain pattern matching, phone number verification, and geographical analysis, presenting results through an intuitive traffic-light system (red for high risk, green for safe). This balanced approach helps users make informed decisions while maintaining a streamlined user experience.

8.6 Phishing Detection

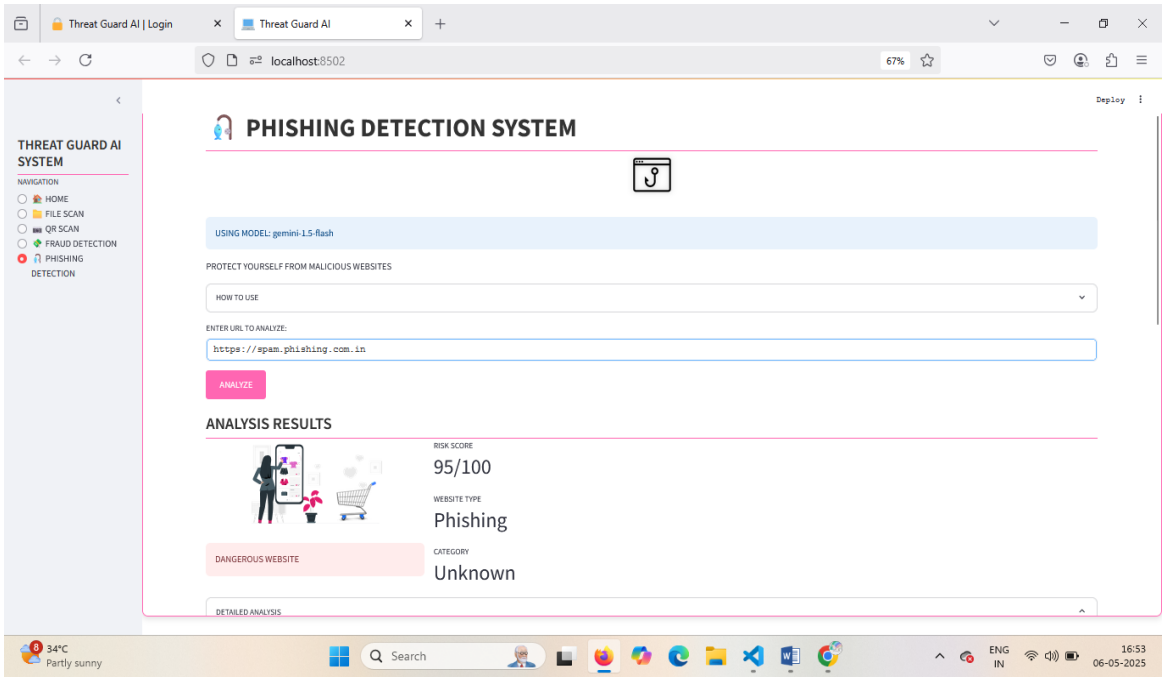


Figure 8.10 Phishing Detection – Malicious

The Phishing Detection module leverages Google's Gemini AI to analyse URLs for potential threats, providing users with a comprehensive security assessment. The interface features a simple input field where users can enter any website URL, followed by an analysis button that triggers the scanning process. During scanning, an animated loader appears while the system evaluates the URL against multiple risk factors including domain authenticity, SSL certificates, and suspicious page elements

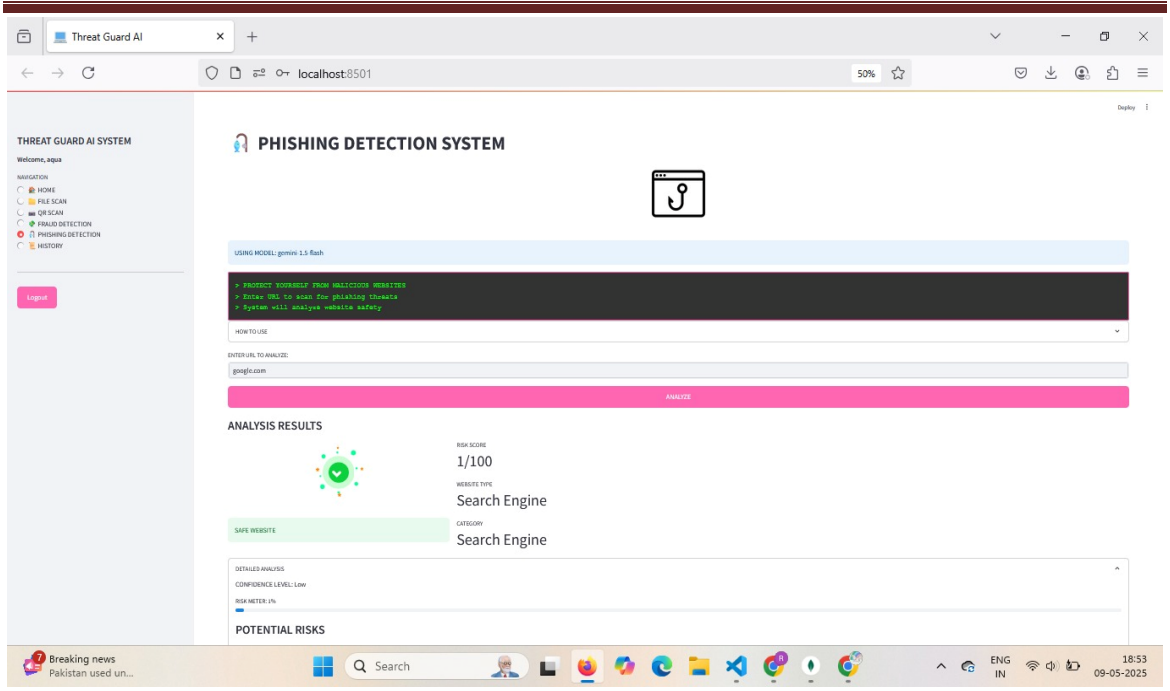


Figure 8.11 Phishing Detection - Safe

Results are displayed through color-coded alerts (red for dangerous, green for safe) accompanied by a risk score (0-100) and detailed breakdowns of detected threats. The system highlights specific concerns like misspelled domains or fake login pages while offering actionable safety recommendations. Additional features include trust score comparisons with known websites and expandable technical details for advanced users.

8.7 Scan History

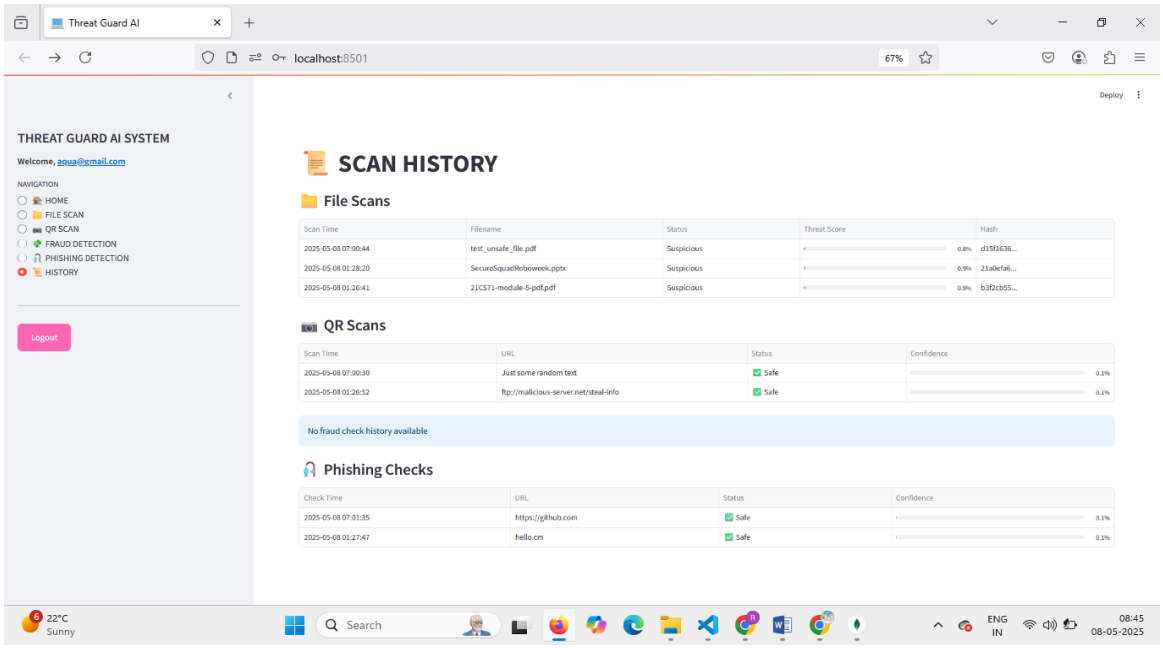


Figure 8.12 Scan History

The **Scan History** page serves as a centralized interface where users can track and verify all past interactions with the Threat Guard AI System. This feature enables users to view historical records of scanned files, QR codes, UPI-based fraud checks, and phishing link evaluations. The backend securely fetches this data for each authenticated user and displays it using interactive tables with visual elements like status icons and confidence/probability bars. These tables present key metadata such as scan timestamps, file hashes, threat scores, phishing/fraud status, and transaction details, offering users both insight and traceability.

This functionality significantly enhances **user trust and security monitoring**, making it easier to validate previously scanned inputs. Whether a user uploads a file, scans a QR code, or verifies a payment, the history page ensures a permanent and visible trail of all checks performed. Moreover, it promotes accountability and allows users to revisit suspicious elements for re-evaluation, forming a critical part of the system's transparency and user awareness model.

8.8 Admin Pages

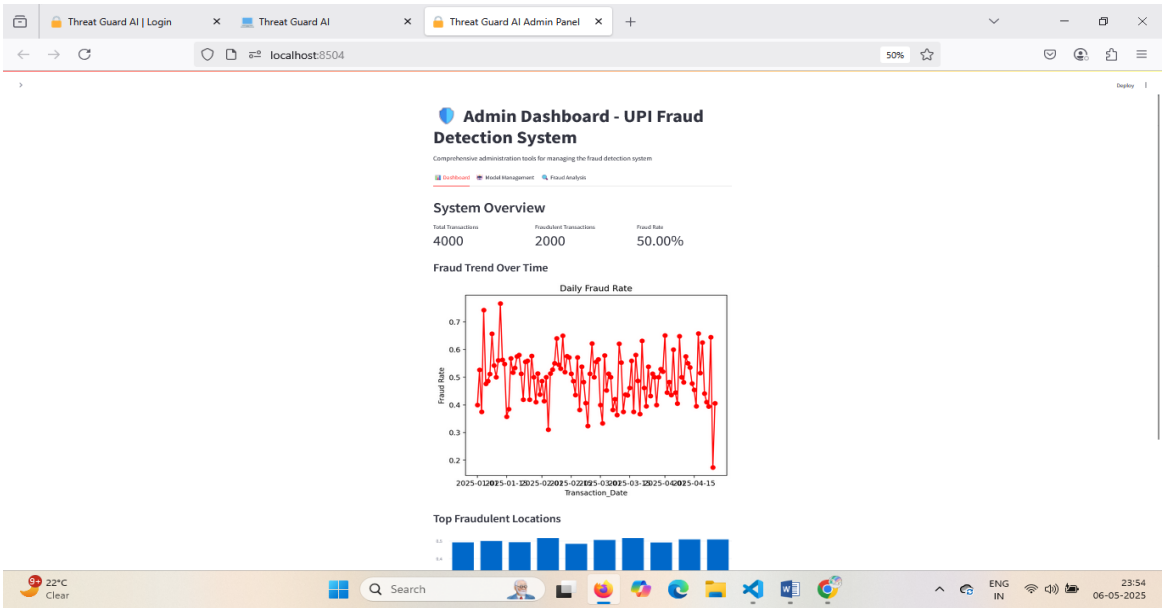


Figure 8.13 Admin Dashboard

The Dashboard serves as the central hub for monitoring the overall health and performance of the fraud detection system. It presents key metrics such as total transactions, fraudulent cases, and fraud rates in an easy-to-digest format, giving administrators a quick snapshot of current threats. Interactive visualizations like the fraud trend graph and top fraudulent locations chart help identify patterns over time and highlight high-risk areas, enabling proactive decision-making.

Beyond raw numbers, the Dashboard provides contextual insights into transaction behaviors, allowing admins to spot anomalies at a glance. The combination of real-time data and historical trends ensures that administrators can assess both immediate risks and long-term fraud patterns. This bird's-eye view is essential for maintaining system integrity and ensuring swift responses to emerging threats.

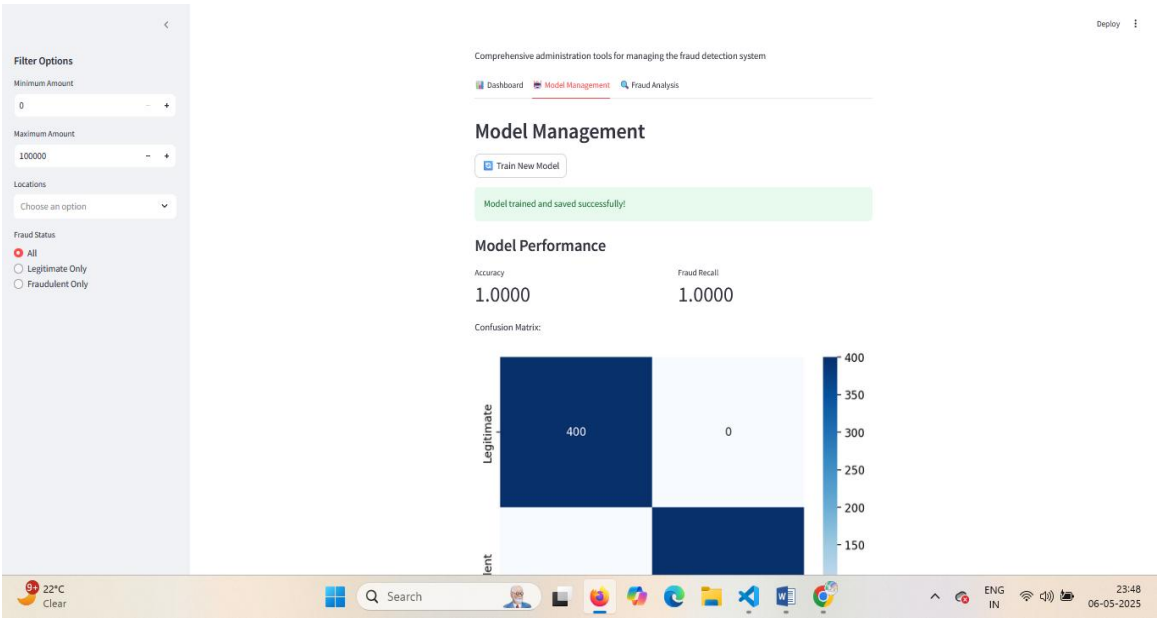


Figure 8.14 Model Management

This tab is the control center for the machine learning model that powers fraud detection, offering tools to train, evaluate, and optimize its performance. Administrators can initiate model retraining, review accuracy and recall scores, and analyze the confusion matrix to understand false positives and negatives. Feature importance charts break down which transaction attributes most influence fraud predictions, helping fine-tune detection logic.

By providing transparency into model behavior, this tab ensures that the system remains adaptive and reliable. Regular performance checks and retraining options help maintain high accuracy, while detailed metrics allow admins to validate improvements. This hands-on approach ensures the model evolves alongside emerging fraud tactics, keeping defenses robust and up to date.

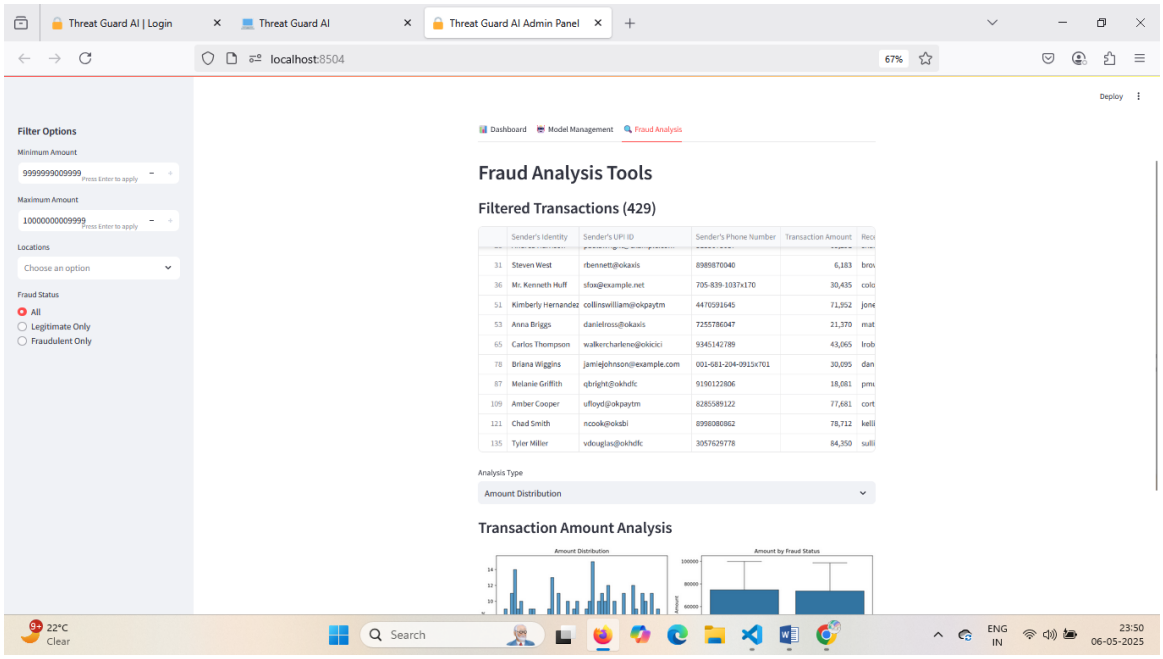


Figure 8.15 Fraud Analysis

Designed for in-depth investigations, this tab allows administrators to dissect transaction data with advanced filtering and visualization tools. Users can segment transactions by amount, location, or fraud status, then explore patterns through interactive charts—whether analyzing high-risk time windows, suspicious UPI domains, or regional fraud hotspots. The granular view helps uncover hidden trends that might otherwise go unnoticed.

Beyond detection, the tab supports forensic analysis, enabling admins to drill into specific cases and validate fraud signals. By combining dynamic filters with visual analytics, it transforms raw data into actionable intelligence. This level of detail is critical for refining fraud rules, improving model accuracy, and ultimately building a more secure transaction ecosystem.

CONCLUSION AND FUTURE SCOPE

Threat Guard AI marks a significant advancement in the field of cybersecurity by providing a unified platform that addresses three of the most prevalent digital threats faced today: phishing, QR code fraud, and malware infections. With the increasing dependency on digital platforms and the growing sophistication of cybercriminals, the need for intelligent, adaptable, and comprehensive threat detection systems has never been more critical. This project has successfully demonstrated the feasibility and effectiveness of combining AI-based analysis, modern frontend technologies like Streamlit and React, and backend intelligence powered by Python and the Google Gemini API to deliver real-time, reliable threat detection.

One of the core strengths of Threat Guard AI lies in its phishing detection module, which leverages natural language prompts and the Gemini large language model to analyze suspicious URLs. Instead of relying solely on static blacklists or heuristic patterns, the system interprets URL structures, behavior patterns, and redirection risks using AI reasoning to determine whether a link is potentially malicious. The results are presented in a clear JSON format, which includes confidence levels, reason explanations, and safety recommendations. This user-centric transparency not only empowers individuals with clear decision-making but also builds trust in the system's output.

Equally impactful is the QR code fraud detection component. With the rise of digital payments and touchless interactions, QR codes have become a common vector for fraud. Threat Guard AI scans and decodes QR codes to reveal the underlying data and URLs, which are then further analyzed for signs of tampering, redirection, or impersonation. The system checks for hidden malicious scripts or links embedded in what seems like a legitimate payment or information code. By exposing these risks early, the system prevents users from unknowingly falling victim to spoofed QR codes placed in public places or sent via deceptive messages.

The malware scanning module complements the system's threat landscape coverage by allowing users to upload files or links for analysis. This component simulates lightweight threat detection using pattern matching, metadata evaluation, and—optionally—future integrations with antivirus APIs. This ensures that users can verify file safety before opening potentially harmful content, whether it's a software installer, a PDF document, or

any suspicious attachment. Although this feature can be expanded to include deeper sandboxing or ML-based file behavior analysis, the current implementation already showcases practical utility.

From an architectural standpoint, Threat Guard AI is designed for scalability, usability, and modularity. The frontend components, built with Streamlit, allow for rapid deployment and user-friendly interaction. The backend, developed in Python, handles the logic for AI communication, QR code processing, and malware simulation. Additionally, the integration with Google's Gemini API offers a robust NLP engine for advanced content interpretation and dynamic analysis. All responses are formatted for clarity and are ready for audit, enhancing both user experience and administrator oversight.

Furthermore, this system upholds important security principles. Sensitive information is processed securely, URLs and files are handled with privacy in mind, and future iterations can be enhanced with user authentication, encrypted communication, and centralized logging for compliance. These elements ensure that Threat Guard AI is not just functional but also aligned with industry best practices in cybersecurity and user protection.

In conclusion, Threat Guard AI serves as a powerful prototype of an AI-powered digital defense platform that is both proactive and intelligent in identifying cyber threats. By targeting multiple fraud vectors—phishing links, QR code scams, and malware—the system goes beyond traditional point solutions to offer a holistic defense mechanism. It promotes safety awareness, simplifies complex threat detection, and provides valuable insights to both users and security administrators. The project demonstrates how emerging AI technologies can be responsibly and effectively leveraged to create tools that are not only technically sound but also practical and impactful. As cyber threats continue to evolve, Threat Guard AI offers a solid foundation upon which even more advanced, automated, and adaptive security solutions can be built.

APPENDIX A

Plagiarism Details:

DrillBit

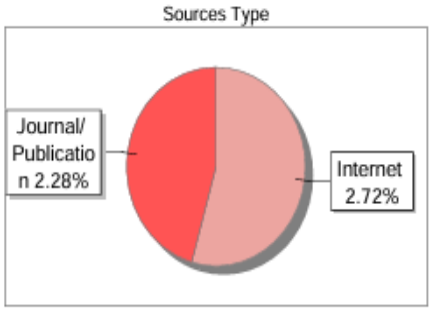
The Report is Generated by DrillBit Plagiarism Detection Software

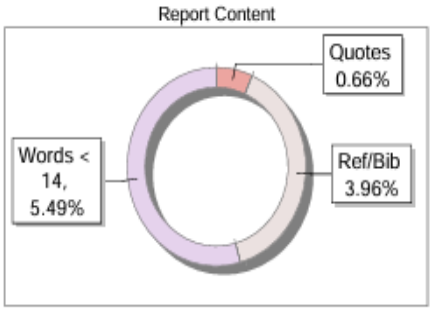
Submission Information

Author Name	Pranjal Bhinge
Title	Threat Guard AI
Paper/Submission ID	3600491
Submitted by	library@sgbit.edu.in
Submission Date	2025-05-10 13:45:53
Total Pages, Total Words	86, 15764
Document type	Project Work

Result Information

Similarity 5 %





Exclude Information

Quotes	Excluded	Language	English
References/Bibliography	Excluded	Student Papers	Yes
Source: Excluded < 14 Words	Excluded	Journals & publishers	Yes
Excluded Source	0 %	Internet or Web	Yes
Excluded Phrases	Not Excluded	Institution Repository	Yes

Database Selection

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

5	18	A
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

	LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	vignan.ac.in		1	Publication
2	ijarcet.org		1	Publication
3	irc.kdu.ac.lk		1	Publication
4	fastercapital.com		<1	Internet Data
5	link.springer.com		<1	Internet Data
6	www.dx.doi.org		<1	Publication
7	www.pharmacovigilanceanalytics.com		<1	Internet Data
10	www.leewayhertz.com		<1	Internet Data
11	translate.google.com		<1	Internet Data
12	pmc.ncbi.nlm.nih.gov		<1	Internet Data
13	www.alooba.com		<1	Internet Data
14	slidelegend.com		<1	Internet Data
15	www.quickset.com		<1	Internet Data
16	43.248.72.679999		<1	Publication

17	brownhelm.com	<1	Internet Data
18	translate.google.com	<1	Internet Data
19	www.rapidinnovation.io	<1	Internet Data
21	www.wjgnet.com	<1	Internet Data

APPENDIX B

1. List of Research Publications and Certifications

Sl. No	Title of Work	Date	Platform / Journal Name	Status
1	Threat Guard AI: A Multi-Domain System for Intelligent Fraud Detection Using Machine Learning	14/05/2025	IJRASET – International Journal for Research in Applied Science and Engineering Technology	Published in Volume 13, Issue V, May 2025. DOI to be issued.

2. List of Events Attended in State

Sl. No	Type of Event	Date	Place	Status
1	Code Battle 2025 – 24 Hour National Level Hackathon	11/03/2025	KLS VDIT Institute of Technology, Haliyal.	Winners in Cyber Security Domain.

3. List of Events Attended Outside State

Sl. No	Type of Event	Date	Place	Status
1	Google Developers Group Prabal – 48 Hour National Level Hackathon	18/04/2025	Sanjay Ghodawat University, Kolhapur	Participated
2	Roboweek 3.0 – National Level Hackathon	05/04/2025	National Instititute of Technology, Hamirpur.	Participated

IJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISSN No. : 2321-9653

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET70866, entitled

Threat Guard AI: A Multi-Domain System for Intelligent Fraud Detection Using Machine Learning

by

Pranjal Bhinge

after review is found suitable and has been published in Volume 13, Issue V, May 2025

in

International Journal for Research in Applied Science & Engineering Technology

(International Peer Reviewed and Refereed Journal)

Good luck for your future endeavors

Editor in Chief, IJRASET

KLS VISHWANATHRAO DESHPANDE INSTITUTE OF TECHNOLOGY

HALIYAL , UTTARA KANNADA DIST. KARNATAKA – 581329

IEEE Mangalore Subsection

IEEE BANGALORE SECTION

VDIT IEEE Student Branch

INSTITUTION'S INNOVATION COUNCIL

NAAC A

CERTIFICATE

OF APPRECIATION

This Certificate is Proudly Awarded To

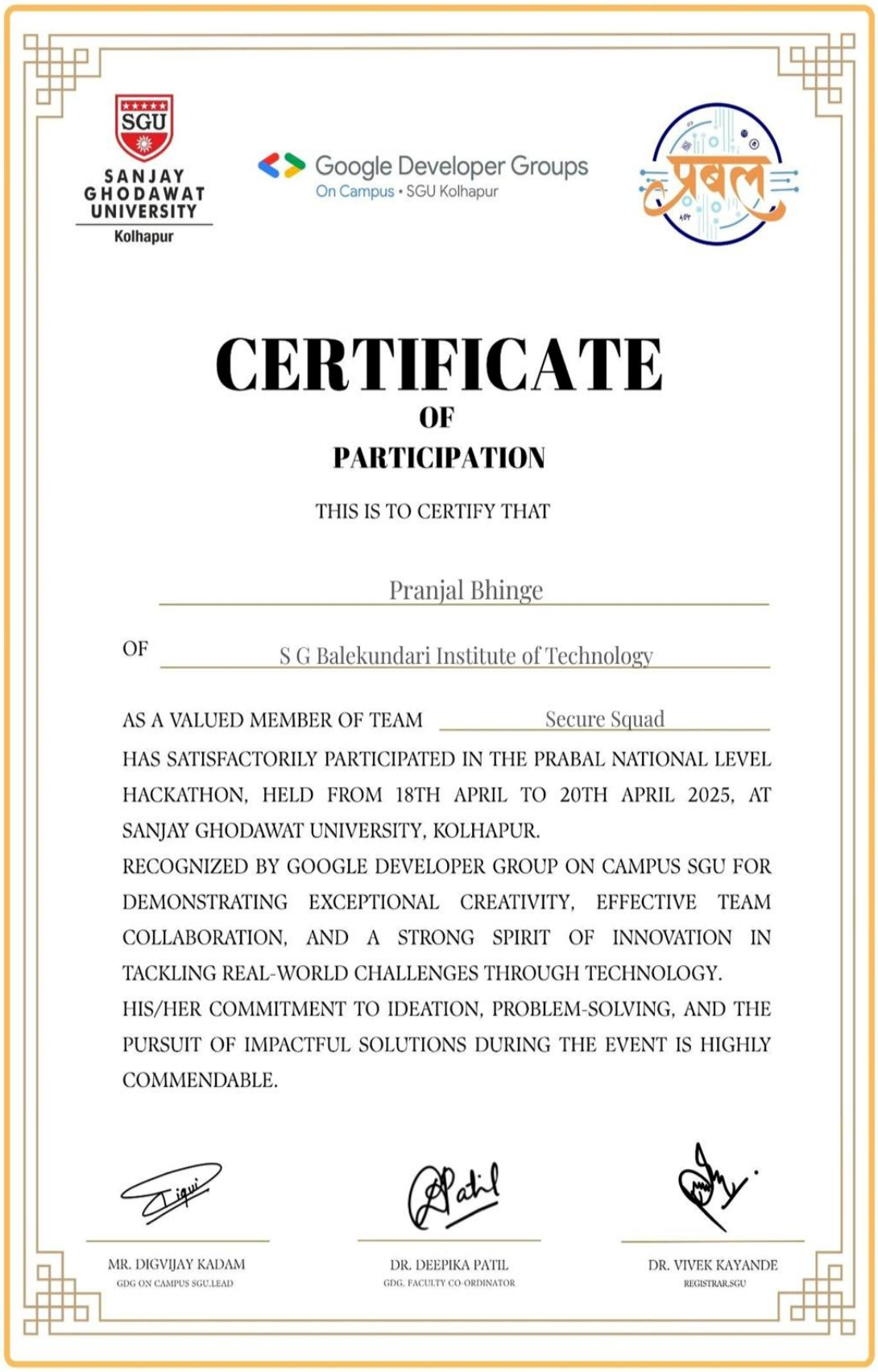
Mr. / Ms. Pooja Bhinge Of SGBIT, Belagavi

For securing I / II position in the 24-Hour National Level Hackathon – **"Code Battle - 2k25"**, Organized by VDIT IEEE Student Branch in collaboration with IEEE Mangalore Subsection held on 11th - 12th March, 2025 at KLS Vishwanathrao Deshpande Institute of Technology, Haliyal, this competition recognized technical expertise, innovation, and problem-solving in the theme - Cyber Security

PROF. SURAJ KADLI
FACULTY ADVISOR, IEEE SB

DR. ARUN L. KAKHANDKI
BRANCH COUNSELOR, IEEE SB

DR. V. A. KULKARNI
PRINCIPAL



SANJAY GHODAWAT UNIVERSITY
Kolhapur

Google Developer Groups
On Campus • SGU Kolhapur

प्रबल

CERTIFICATE
OF
PARTICIPATION

THIS IS TO CERTIFY THAT

Pranjal Bhinge

OF S G Balekundari Institute of Technology

AS A VALUED MEMBER OF TEAM Secure Squad

HAS SATISFACTORILY PARTICIPATED IN THE PRABAL NATIONAL LEVEL HACKATHON, HELD FROM 18TH APRIL TO 20TH APRIL 2025, AT SANJAY GHODAWAT UNIVERSITY, KOLHAPUR.

RECOGNIZED BY GOOGLE DEVELOPER GROUP ON CAMPUS SGU FOR DEMONSTRATING EXCEPTIONAL CREATIVITY, EFFECTIVE TEAM COLLABORATION, AND A STRONG SPIRIT OF INNOVATION IN TACKLING REAL-WORLD CHALLENGES THROUGH TECHNOLOGY.

HIS/HER COMMITMENT TO IDEATION, PROBLEM-SOLVING, AND THE PURSUIT OF IMPACTFUL SOLUTIONS DURING THE EVENT IS HIGHLY COMMENDABLE.

MR. DIGVIJAY KADAM
GDG ON CAMPUS SGU.LEAD

DR. DEEPIKA PATIL
GDG, FACULTY CO-ORDINATOR

DR. VIVEK KAYANDE
REGISTRAR,SGU

Certificate of Participation
This is to certify that
Pranjal Bhinge
from S G Balekundari Institute of Technology as Team
Secure Squad has participated in Hackathon of Roboweek
3.0 organised by National Institute of Technology (NIT),
Hamirpur.
unstop

APPENDIX C

Tools Used:

Tool / Technology	Purpose / Usage
Python 3.11	Core programming language for backend logic, machine learning, and data processing.
Streamlit	Used to build the interactive frontend UI for security modules.
Flask	Lightweight web framework used for backend API handling (URL, file, model APIs).
Firebase Authentication	Provides secure user login, signup, and password management.
MongoDB Atlas	Cloud-based NoSQL database used to store scan history and logs.
Google Gemini API	AI API for phishing detection using natural language understanding and reasoning.
OpenCV	Used for image capture and QR code detection via webcam.
Pyzbar	Library for decoding QR code data from images.
Scikit-learn	Machine learning framework used to build and train the fraud detection model.
ClamAV / VirusTotal API	Used for file malware scanning and signature detection (optional integration).
dotenv	Loads environment variables securely (e.g., API keys).
Visual Studio Code (VS Code)	IDE used for writing, debugging, and managing source code.
Git & GitHub	Version control system used for source code tracking and collaboration.

APPENDIX D

Manual for Software:

1. System Requirements

- Operating System: Windows 10 / Linux (Ubuntu 20.04+)
- Python Version: 3.10 or higher
- RAM: Minimum 4 GB
- Internet: Required for Gemini API access and Firebase authentication
- Browser: Chrome / Firefox (latest version)

2. Installation Instructions

- Clone the Repository-
  git clone https://github.com/yourusername/threat-guard-ai.git
  cd threat-guard-ai

- Create and Activate Virtual Environment
  python -m venv venv
  source venv/bin/activate # For Linux/Mac
  venv\Scripts\activate # For Windows

- Install Required Libraries
  pip install -r requirements.txt

- Set API Keys (Google Gemini)
  Create a .env file in the root directory and add:
  GEMINI_API_KEY=your_api_key_here

3. How to Run

- Run Flask Backend
  python flask_app.py

- Run Streamlit Frontend
  python -m streamlit run streamlit_app.py

4. How to Use the Application

- **Home Page:**

  Displays a command-line themed interface and allows module selection.

- **Phishing Detection:**

  Enter a URL → Click Scan → View AI-generated results with risk level.

- **QR Code Scanner:**

  Upload or scan a QR code → View decoded URL and security status.

- **File Threat Analysis:**

  Upload PDF or DOCX file → Analyze for malware signatures and entropy.

- **UPI Fraud Detection:**

  Input transaction metadata (UPI IDs, amount, etc.) → View fraud risk prediction.

- **Scan History:**

  Browse and review past scans with timestamps and confidence levels.

- **Admin Panel:**

  View logs, graphs, model metrics, and manage detection thresholds (admin access only).

REFERENCES

[1] S. S. Alshamrani, ‘‘Design and analysis of machine learning based technique for malware identification and classification of portable document format files,’’ Secur. Commun. Netw., vol. 2022, pp. 1–10, Sep. 2022.

[2] P. Singh, S. Tapaswi, and S. Gupta, ‘‘Malware detection in PDF and office documents: A survey,’’ Inf. Secur. J., Global Perspective, vol. 29, no. 3, pp. 134–153, May 2020.

[3] N. Livathinos, C. Berrospi, M. Lysak, V. Kuropiatnyk, A. Nassar, A. Carvalho, M. Dolfi, C. Auer, K. Dinkla, and P. Staar, ‘‘Robust PDF document conversion using recurrent neural networks,’’ in Proc. AAAI Conf. Artif. Intell., vol. 35, no. 17, 2021, pp. 15137–15145.

[4] Q. A. Al-Haija, A. Odeh, and H. Qattous, ‘‘PDF malware detection based on optimizable decision trees,’’ Electronics, vol. 11, no. 19, p. 3142, Sep. 2022.

[5] Y. Wiseman, ‘‘Efficient embedded images in portable document format,’’ Int. J., vol. 124, pp. 38–129, Jan. 2019.

[6] M. Ijaz, M. H. Durad, and M. Ismail, ‘‘Static and dynamic malware analysis using machine learning,’’ in Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST), Jan. 2019, pp. 687–691.

[7] Y. Alosefer, ‘‘Analysing web-based malware behaviour through client honeypots,’’ Ph.D. dissertation, School Comput. Sci. Inform., Cardiff Univ., Cardiff, Wales, U.K., 2012.

[8] N. Idika and A. P. Mathur, ‘‘A survey of malware detection techniques,’’ Purdue Univ., vol. 48, no. 2, pp. 32–46, 2007.

[9] M. Abdelsalam, M. Gupta, and S. Mittal, ‘‘Artificial intelligence assisted malware analysis,’’ in Proc. ACM Workshop Secure Trustworthy Cyber Phys. Syst., Apr. 2021, pp. 75–77.

[10] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, ‘‘BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors,’’ Inf. Sci., vol. 511, pp. 284–296, Feb. 2020.

[11]. Gu, Yunhua, and Weixiang Zhang. "QR code recognition based on image processing." In international conference on information science and technology, pp. 733-736. IEEE, 2011.

[12]. Falas, Tasos, and Hossein Kashani. "Two dimensional bar-code decoding with camera equipped mobile phones."

[13]. LIU, Ning-zhong, and Jing-yu YANG. "Recognition of two-dimensional bar code based on Fourier Transform." Journal of Image and Graphics 8, no. 8 (2003): 877-882.

[14]. Mohapatra S., et al. (2017). Unified payment interface (upi):a cashless indian transaction process., International Journal of Applied Science and Engineering, vol. 5, pp. 29–42, 6.

[15]. Kumar R., Kishore S., Lu H., and Prakash A. (Aug. 2020). Security analysis of unified payments interface and payment apps in india, in 29th USENIX Security Symposium (USENIX Security 20), pp. 1499–1516, USENIX Association.

[16]. Chatterjee D. A. and Thomas R. (2017). Unified payment interface (upi): A catalyst tool supporting digitalization– utility, prospects and issues, International Journal of Innovative Research and Advanced Studies (IJIRAS), vol. 4, no. 2, pp. 192–195.

[17]. N. Idika and A. P. Mathur, ‘‘A survey of malware detection techniques,’’ Purdue Univ., vol. 48, no. 2, pp. 32–46, 2007.

[18]. D.Maiorca, I. Corona, and G. Giacinto, ‘‘Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF f iles detection,’’ in Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur., May 2013, pp. 119–130. [13]. Y. Alosefer, ‘‘Analysing web-based malware behaviour through client honeypots,’’ Ph.D. dissertation, School Comput. Sci. Inform., Cardiff Univ., Cardiff, Wales, U.K., 2012.

[19]. C. Vatamanu, D. Gavrilut, and R. Benchea, ‘‘A practical approach on clustering malicious PDF documents,’’ J. Comput. Virol., vol. 8, no. 4, pp. 151–163, Nov. 2012.

[20]. F. Schmitt, J. Gassen, and E. Gerhards-Padilla, ‘‘PDF scrutinizer: Detecting JavaScript-based attacks in PDF documents,’’ in Proc. 10th Annu. Int. Conf. Privacy, Secur. Trust, Jul. 2012, pp. 104–111.