SafeScan : Proactive Fraud Detection in Digital Payments using ML

The Project report Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

Submitted By

JILLIDIMUDI LALITHA VASAVI (21NM1A1218)

KRISHNA VINEETHA PATNAIK KUPPILI (21NM1A1223)

MUNAKALA PRIYA (21NM1A1231)

PANDURI RAKSHITHA RATNA SAI (21NM1A1239)

Under the Guidance of

Mrs. B. Sudha Madhuri

Assistant Professor

Department of Information Technology

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTU-GV)

Kapujaggarajupeta, Vadlapudi Post, Visakhapatnam, Andhra Pradesh.

2020-2024

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTU-GV)

Kapujaggarajupeta, Vadlapudi post, Visakhapatnam-530049

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report titled "**SafeScan : Proactive Fraud Detection in Digital Payments using ML"** is being submitted by **JILLIDIMUDI LALITHA VASAVI (21NM1A1218), KRISHNA VINEETHA PATNAIK KUPPILI (21NM1A1223), MUNAKALA PRIYA (21NM1A1231), PANDURI RAKSHITHA RATNA SAI (21NM1A1239),** in B. Tech IV II semester Information Technology is a record bonafied work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide	**Head of the Department**
Mrs. B. Sudha Madhuri	**Dr. B. Siva Lakshmi**
Assistant Professor	Associate Professor
Department of IT	Department of IT

External Examiner

DECLARATION

We hereby declare that this project titled **SafeScan : Proactive Fraud Detection in Digital Payments using ML** is the original work done by us in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology, Jawaharlal Nehru Technological University – Gurajada Vizianagaram. This project report has not been previously submitted to any other university/Institution for the award of any other degree.

JILLIDIMUDI LALITHA VASAVI (21NM1A1218)

KRISHNA VINEETHA PATNAIK KUPPILI (21NM1A1223)

MUNAKALA PRIYA (21NM1A1231)

PANDURI RAKSHITHA RATNA SAI (21NM1A1239)

ACKNOWLEDGEMENT

We express our grateful thanks to **Professor**, **Dr. B. ARUNDATHI**, **Principal** of Vignan's Institute of Engineering for Women, for the overall valuable support he provided during the project work and also during our course of study.

We sincerely and honestly thank **Associate Professor**, **Dr. T. SANDHYA KUMARI**, **Dean of Academics**, Vignan's Institute of Engineering for Women for her valuable advices and encouragement for the project work.

We have the pleasure of extending our sincere gratitude to **Associate Professor**, **Dr. D. VIJAYA LAKSHMI**, **Dean of R&D cell**, Vignan's Institute of Engineering for Women.

We take the privilege to express our profound sense of gratitude to **Associate Professor**, **Dr. B. SIVA LAKSHMI**, **Head of the Department**, Information Technology, Vignan's Institute of Engineering for Women

We extend a heartful thanks to our guide **Mrs. B. SUDHA MADHURI, Assistant Professor of department**, Information Technology, Vignan's Institute of Engineering for Women for his co-operation and for the perfection of our project.

We sincerely thank all the Staff Members of the Department for giving us their support for the completion of this project.

We would like to express our sincere thanks to our parents who encouraged us throughout our educational endeavour and to do this project work.

Finally, we would like to express our thanks to library staff, who have supported us in the accomplishment of the project.

JILLIDIMUDI LALITHA VASAVI (21NM1A1218)

KRISHNA VINEETHA PATNAIK KUPPILI (21NM1A1223)

MUNAKALA PRIYA (21NM1A1231)

PANDURI RAKSHITHA RATNA SAI (21NM1A1239)

CONTENTS

ABSTRACT

Fraud in digital payments poses a significant threat to user security, requiring robust measures for prevention and detection. The proposed project, SafeScan, offers a comprehensive solution to address these challenges by developing an integrated system for proactive fraud detection in digital payments. The system employs advanced validation mechanisms to verify QR codes and UPI IDs, ensuring the authenticity of payment details while detecting tampering or spoofing. Concurrently, Machine Learning algorithms are utilized to analyze transaction patterns, including frequency, amount, and user behaviour, to identify anomalies indicative of fraudulent activity. SafeScan also features an automated response system that blocks transactions flagged as suspicious and notifies users in real-time, empowering them to take immediate action. Additionally, the system provides an admin dashboard for real-time monitoring ensuring adaptability to emerging fraud tactics. By transmitting transactional and fraud detection data to a centralized platform for visualization and analysis, users can remotely monitor activity and make informed decisions. SafeScan aims to revolutionize digital payment security by offering an intelligent, scalable, and user-friendly solution, thereby fostering trust and confidence in the financial ecosystem.

LIST OF FIGURES

LIST OF SCREENS

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

Digital payments have revolutionized the way individuals and businesses handle financial transactions, offering unparalleled convenience, speed, and accessibility. The rise of platforms enabling cashless transactions has accelerated economic activities and facilitated seamless exchange of goods and services. However, the widespread adoption of these systems has also introduced new vulnerabilities, making them prime targets for fraudsters. Activities such as spoofed UPI IDs, tampered QR codes, phishing attempts, and other sophisticated techniques pose significant threats to users and financial institutions alike. To address these challenges, the proposed project, SafeScan, seeks to deliver an innovative, machine learning-driven solution for real-time fraud detection in digital payments.

Why Machine Learning for SafeScan?

Fraud detection in digital payments is a complex challenge due to the ever-evolving nature of fraudulent techniques. Traditional rule-based approaches, while effective to some extent, often struggle with new, sophisticated fraud patterns. Machine Learning (ML) offers a dynamic, data-driven approach to fraud detection, making SafeScan more efficient, accurate, and adaptive.

Pattern Recognition: Machine learning algorithms can identify hidden patterns in transactional data, detecting suspicious activities that might go unnoticed by conventional methods. By analyzing historical transaction data, ML models can recognize anomalies and flag potential fraud.

Adaptive Learning: Unlike static rule-based systems, ML continuously learns from new fraud patterns and adapts to emerging threats. This self-improving capability ensures that SafeScan remains effective even as fraudsters develop new techniques.

Scalability & Efficiency: With the increasing volume of digital transactions, manually analyzing each one is impractical. ML enables fast and automated fraud detection, making it scalable for large datasets while minimizing human intervention.

Feature Engineering for Fraud Detection: Machine learning models can extract meaningful features from transaction data, such as:

- **QR code metadata -** Verifying authenticity and checking for tampering.
- **UPI ID behaviour -** Detecting unusual transaction patterns or newly created fraudulent UPI IDs.
- **Transaction frequency -** Identifying deviations from normal user behaviour.

Reduction in False Positives: Traditional fraud detection systems often incorrectly block legitimate transactions, frustrating users. ML enhances fraud detection by improving precision, minimizing false positives, and ensuring smooth user experience.

Real-time Decision Making: ML-powered fraud detection enables instant risk assessment, preventing fraudulent transactions before they occur. This real-time decision-making capability ensures proactive fraud prevention rather than reactive reporting and virtual realms, ushering in a new era of transformative applications and solutions across various sectors and industries.

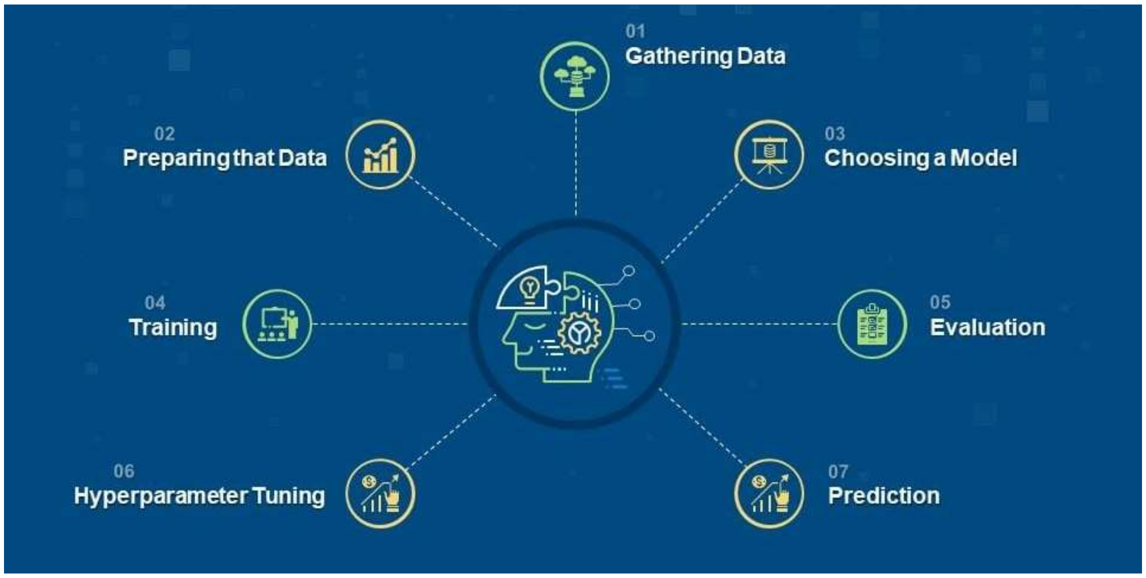


Figure 1.1: Image of How Machine Learning Analyzes

Some Other Applications of Machine Learning

- **Finance & Fraud Detection -** Credit card fraud detection, Loan approvals, Algorithmic trading in stock markets, Personalized financial recommendations, Risk management.
- **Healthcare & Medicine -** Disease prediction & diagnosis (e.g., cancer detection, diabetic retinopathy), Drug discovery & development, Personalized treatment plans.
- **Autonomous Vehicles & Transportation -** Self-driving cars, Traffic prediction & optimization, Route planning, Navigation systems, Autonomous drones & delivery robots.
- **Natural Language Processing (NLP) & AI Assistants -** Chatbots & virtual assistants (ChatGPT, Siri, etc.), Sentiment analysis in social media & reviews, Text summarization.
- **Cybersecurity & Threat Detection -** Malware detection & prevention, Phishing attack identification, Anomaly detection in network security, Spam filtering & bot detection.
- **Image & Video Processing -** Face recognition & authentication (Face ID, CCTV surveillance), Object detection & image segmentation, Deepfake detection.

THE MACHINE LEARNING MODEL

A machine learning model is a sophisticated mathematical representation of a real-world phenomenon. Unlike traditional software, which relies on explicit, rule-based programming, a machine learning model learns from data. This means it can identify patterns, make predictions, and even make decisions without being explicitly programmed for each specific scenario. Essentially, it mimics the human ability to learn from experience, recognize recurring patterns within data, and subsequently use this knowledge to make informed judgments or predictions about future events.

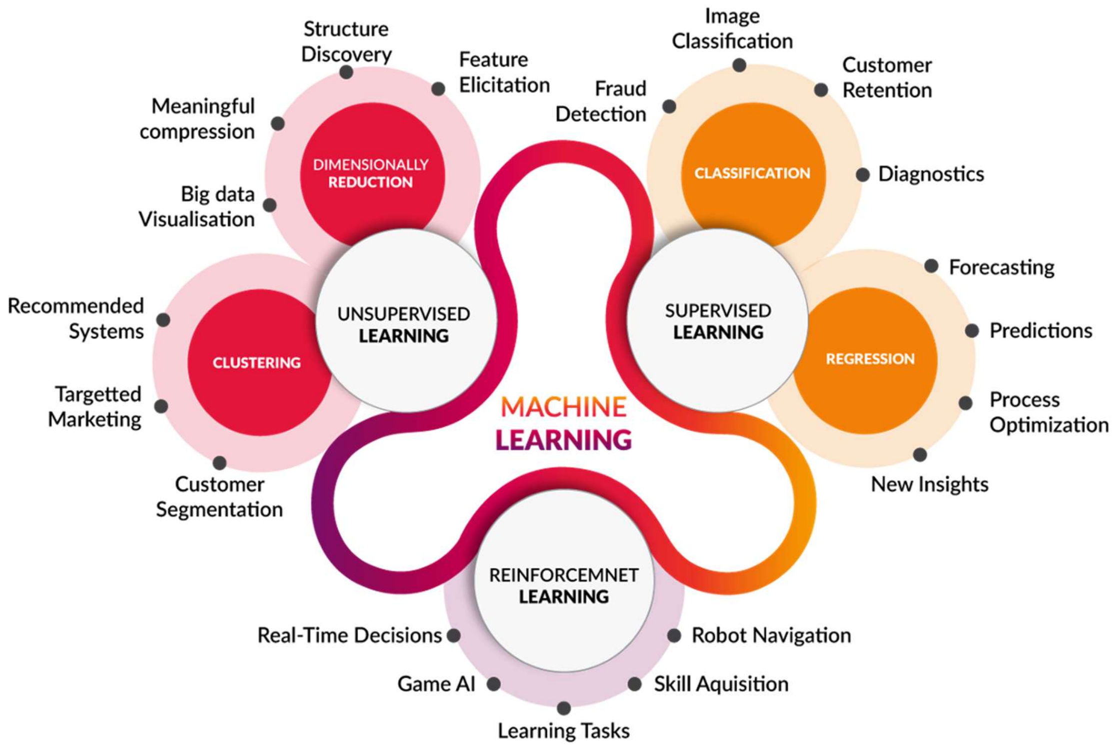


Figure 1.2: The Machine Learning Model

At its core, a machine learning model operates by identifying underlying patterns and relationships within a dataset. This process involves training the model on a vast amount of data, allowing it to adjust its internal parameters and improve its ability to make accurate predictions. The type of learning employed can vary significantly. In supervised learning, the model is trained on labelled data, where each data point is associated with a known outcome. This allows the model to learn the mapping between inputs and outputs, enabling it to predict outcomes for new, unseen data. Classification, which predicts categorical outcomes (e.g., spam/not spam, disease/no disease), and regression, which predicts continuous values (e.g., stock prices, house prices), are prominent examples of supervised learning tasks.

Unsupervised learning, on the other hand, deals with unlabelled data, where the model must discover hidden patterns and structures within the data itself. Clustering, which groups similar data points together, and dimensionality reduction, which reduces the number of features while preserving essential information, are key techniques in unsupervised learning.

Deep learning, a subfield of machine learning, utilizes artificial neural networks with multiple layers to model complex patterns and relationships in data. These networks, inspired by the human brain's interconnected neurons, can learn intricate representations of data, enabling them to excel in tasks such as image and speech recognition, natural language processing, and playing games. Convolutional Neural Networks (CNNs), renowned for their ability to process images, and Recurrent Neural Networks (RNNs), adept at handling sequential data like time series and natural language, are prominent examples of deep learning architectures.

The effectiveness of a machine learning model hinges on several critical factors. The quality and quantity of the data used for training significantly impact its performance. High-quality data, free from biases and errors, is essential for building accurate and reliable models. Additionally, the selection of the appropriate model architecture is crucial. Different models are better suited for specific tasks and data types. For instance, decision trees may be suitable for simple classification problems, while deep neural networks may be necessary for complex image recognition tasks.

Furthermore, meticulous hyperparameter tuning is essential to optimize the model's performance. Hyperparameters are settings that control the learning process, such as the learning rate and the number of hidden layers in a neural network. Careful tuning of these parameters can significantly improve the model's accuracy and generalization ability.

Rigorous evaluation is another crucial step in the machine learning process. Various metrics, such as accuracy, precision, recall, and F1-score, are used to assess the model's performance on unseen data. This evaluation helps identify areas for improvement and ensures that the model meets the desired performance standards.

Finally, addressing potential biases in the data and model predictions is paramount. Biases can arise from various sources, such as skewed data representation or inherent biases in the algorithms themselves. It is crucial to identify and mitigate these biases to ensure fair and equitable outcomes.

Where is the term Machine Learning being used?

Machine learning (ML) has permeated virtually every facet of our modern lives, quietly yet profoundly shaping our interactions with technology and the world around us. At its core, ML empowers computers to learn from data, identify patterns, and make decisions without being explicitly programmed for each specific task. This transformative capability stems from sophisticated algorithms that enable machines to analyze vast datasets, extract meaningful insights, and build predictive models that can adapt and improve over time.

In our daily lives, ML is subtly yet pervasively present. Social media platforms leverage ML to curate personalized feeds, anticipating our interests and preferences based on our past interactions and online behavior. Search engines utilize intricate algorithms to understand our queries, rank search results, and deliver the most relevant information, considering factors like location, search history, and the context of our search. Online shopping experiences are also significantly influenced by ML, with platforms employing recommendation engines to suggest products that align with our individual tastes and browsing habits, enhancing our shopping experience, and driving sales.

Beyond the realm of personal technology, ML plays a critical role in advancing healthcare. In medical imaging, ML algorithms assist in the diagnosis of diseases like cancer by analyzing medical images such as X-rays and MRIs, identifying subtle patterns that may be imperceptible to the human eye. Furthermore, ML accelerates drug discovery by analyzing vast datasets of genetic information, molecular structures, and clinical trial data to identify promising drug candidates and predict their efficacy and safety.

The financial sector also heavily relies on ML to mitigate risks and optimize operations. Sophisticated algorithms are employed to detect and prevent fraudulent activities, such as credit card fraud and money laundering, by analyzing transaction patterns and identifying anomalies that may indicate suspicious behavior.

The applications of ML are constantly expanding, with new and innovative use cases emerging across various sectors. From artificial intelligence in manufacturing, where ML optimizes production processes and predicts equipment failures, to its applications in environmental science, where it helps to monitor climate change and predict natural disasters, ML is transforming industries and shaping the future of our world. As technology continues to advance, we can expect to witness even more remarkable applications of ML, further revolutionizing the way we live, work, and interact with the world around us.

Scope and Benefits of Machine Learning

Machine Learning (ML) encompasses a broad spectrum of applications, impacting numerous industries and aspects of our daily lives. In healthcare, ML algorithms analyze medical images to assist in disease diagnosis, accelerate drug discovery, and personalize treatment plans. Within the financial sector, ML powers fraud detection systems, informs credit scoring models, and drives algorithmic trading strategies. E-commerce thrives on ML-powered recommendation engines that personalize product suggestions and optimize marketing campaigns.

The benefits of ML are multifaceted. By analyzing vast datasets and identifying intricate patterns, ML algorithms enhance accuracy and precision in various tasks, from medical diagnoses to fraud detection. Furthermore, ML automates repetitive tasks, freeing up human resources for more creative and strategic endeavours. This automation leads to increased efficiency and cost savings across various industries.

Moreover, ML empowers businesses and organizations to make data-driven decisions, providing valuable insights and actionable recommendations. In healthcare, it enables personalized medicine, while in finance, it strengthens risk management strategies. By personalizing experiences, such as product recommendations and customer service interactions, ML enhances customer satisfaction and loyalty. Ultimately, the adoption of ML fosters innovation and provides a competitive advantage by enabling the development of new products, services, and business models.

Issues and challenges of Machine Learning

- Data Sparsity
- Data Noise and Inconsistency
- Overfitting
- Underfitting
- Interpretability/Explainability (Black Box Problem)
- Privacy
- Job Displacement
- Misuse
- High Computational Costs
- Scalability
- Lack of Skilled Professionals

Life-Cycle of Machine learning

The lifecycle of a Machine Learning model typically involves several stages: data collection and preparation, where high-quality data is gathered, cleaned, and prepared for training; model selection and training, where appropriate ML algorithms are chosen and trained on the prepared data; model evaluation, where the model's performance is rigorously assessed using relevant metrics; model deployment, where the trained model is integrated into a real-world application; and finally, monitoring and maintenance, where the deployed model is continuously monitored for performance degradation and undergoes regular retraining and updates to ensure ongoing accuracy and adaptability to evolving data patterns.

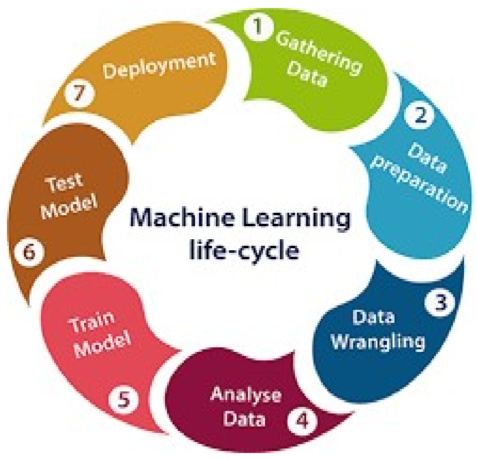


Figure 1.3: Life-cycle of Machine Learning

1.1 MOTIVATION

The increasing dependency on digital payment systems has led to a surge in fraudulent activities, causing significant financial losses to individuals and businesses. Existing fraud detection systems often fail to operate in real-time and struggle to adapt to evolving attack strategies. The motivation behind SafeScan stems from the need to provide a secure, intelligent, and proactive fraud detection mechanism that not only identifies fraudulent transactions but also prevents them before they occur. By leveraging the power of machine learning, this project aims to create an adaptive system that continuously learns from transaction patterns, ensuring enhanced security for users. SafeScan also addresses the lack of awareness among users by providing real-time alerts and actionable insights. Ultimately, the project aspires to strengthen trust in digital payment ecosystems by making transactions safer and more reliable, fostering widespread adoption and financial security.

1.2 PROBLEM DEFINITION

With the rapid growth of digital payments, financial fraud has become a pressing concern, resulting in significant monetary losses and loss of user trust. Fraudsters exploit vulnerabilities in payment systems by tampering with QR codes, spoofing UPI IDs, and using sophisticated phishing techniques. Existing fraud detection mechanisms primarily rely on predefined rule-based approaches, which struggle to keep up with evolving fraud patterns and often fail to provide real-time security. The SafeScan project aims to address these challenges by developing an intelligent fraud detection system that leverages machine learning algorithms for proactive fraud prevention. The system will analyze transaction patterns, validate QR codes and UPI IDs, and alert users in real-time if fraudulent activities are detected. By ensuring secure and transparent digital transactions, SafeScan aims to minimize fraud, protect users, and enhance the reliability of digital payment platforms.

1.3 OBJECTIVE OF SAFESCAN

The primary goal of SafeScan is to create a proactive fraud detection system that ensures secure digital transactions by leveraging advanced technologies such as Machine Learning (ML) and real-time validation. This project aims to:

- **Verify Payment Integrity:** Ensure the authenticity of QR codes and UPI IDs to prevent tampering and spoofing.
- **Identify Fraud Patterns:** Analyze transaction behaviours using ML algorithms to detect anomalies and flag suspicious activities.
- **Provide Real-Time Alerts:** Notify users instantly about potentially fraudulent activities, empowering them to take preventive measures.
- **Facilitate Monitoring and Management:** Offer a dashboard for real-time transaction monitoring and dynamic blacklist management.
- **Enhance User Trust:** Foster confidence in digital payment systems by providing a secure, user-friendly, and reliable solution.

1.4 CHALLENGES OF PROJECT

Despite significant advancements in digital payment infrastructure, fraud remains a critical issue. Common challenges include:

- **QR Code Tampering:** Fraudsters often manipulate QR codes by embedding malicious links or altering payment details, leading unsuspecting users to make payments to fraudulent accounts.

- **Spoofed UPI IDs:** Fraudulent actors create UPI IDs that mimic legitimate accounts to deceive users into transferring funds to unauthorized entities.
- **Lack of Real-Time Monitoring:** Traditional fraud detection systems often fail to operate in real-time, allowing fraudulent transactions to be completed before any preventive measures are taken.
- **Evolving Fraud Tactics:** Cybercriminals continuously develop new methods to exploit vulnerabilities in digital payment systems, making static security mechanisms inadequate.
- **Limited User Awareness:** Many users lack the knowledge to identify fraudulent activities, making them more susceptible to scams.

1.5 ORGANISATION OF DOCUMENTTATION

Session 1:

In chapter 1 we had seen the Introduction, motivation, problem definition, objective of project, challenges of project.

Session 2:

In chapter 2, we see the **Analysis**- Introduction, project requirements, content diagram of project, algorithm and flowcharts, conclusion.

Session 3:

In chapter 3 we will discuss about the **Design**- Introduction, UML diagrams, module design and organization, conclusion.

Session 4:

The chapter 4 is about **Implementation and Results**- Introduction, explanation of key function, method of implementation (forms, output screens, result analysis), conclusion.

Session 5:

In chapter 5 we will see **Testing &Validation**- Introduction, design

CHAPTER-2
LITERATURE SURVEY

2.1 INTRODUCTION

The literature survey is an essential component of research projects, offering a thorough examination of existing knowledge, research findings, and advances in a certain topic or subject area. In the context of the proposed project, which focuses on the development of a Proactive Fraud Detection in Digital Payments using ML, the literature survey is critical for establishing the current state of the art, identifying knowledge gaps, and informing research methodology and design. The increasing prevalence of digital payment fraud has led researchers and organizations to explore various approaches for enhancing security. This literature survey reviews existing fraud detection techniques, their limitations, and the proposed system's improvements over traditional methods.

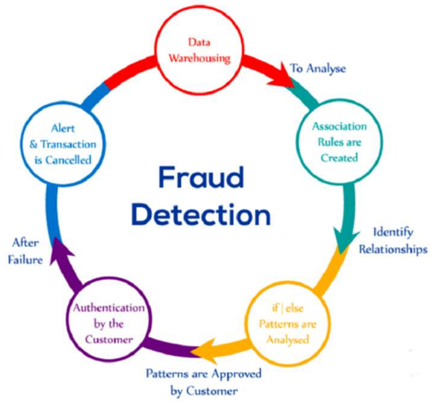

Fraud Detection
Data Warehousing
To Analyse
Association Rules are Created
Identify Relationships
if | else Patterns are Analysed
Patterns are Approved by Customer
Authentication by the Customer
After Failure
Alert & Transaction is Cancelled

Fig 2.1.1 Fraud Detection Cycle

Fig 2.1.2 Fraud Detection Categories

2.2 EXISTING SYSTEM

Traditional fraud detection techniques like rule-based systems, manual transaction monitoring, and heuristic analysis struggle to keep pace with evolving fraud. Rule-based systems, relying on pre-defined thresholds, generate false positives and negatives, requiring constant, costly updates. Manual monitoring, while more nuanced, is slow and resource-intensive, unable to handle the volume of modern transactions, leading to missed fraudulent activity. Heuristic analysis, though incorporating some learning, remains limited by pre-defined indicators, failing to detect emerging fraud schemes. These methods are inherently reactive, detecting fraud after it occurs, and struggle to adapt in real-time to increasingly complex tactics. This necessitates more advanced solutions leveraging AI and machine learning.

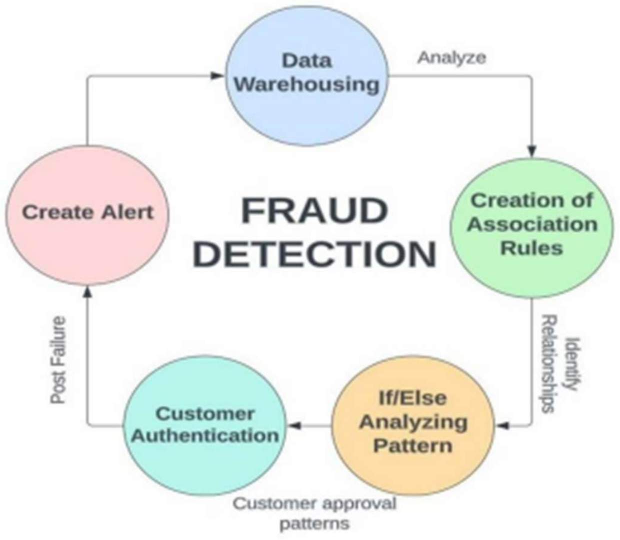


Fig 2.2.1 Rule-based fraud detection systems

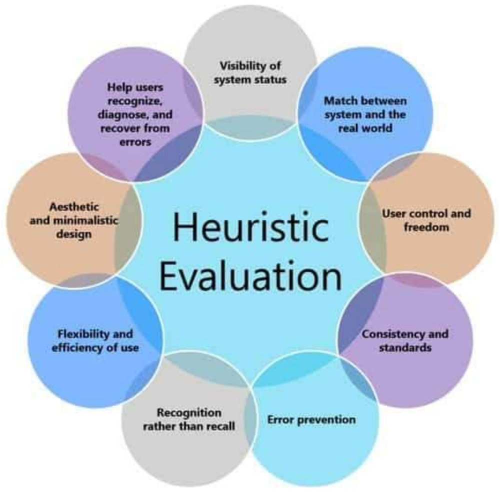

Heuristic Evaluation
Visibility of system status
Match between system and the real world
User control and freedom
Consistency and standards
Error prevention
Recognition rather than recall
Flexibility and efficiency of use
Aesthetic and minimalistic design
Help users recognize, diagnose, and recover from errors

Fig 2.2.2 Heuristic Evaluation

2.3 DISADVANTAGES OF EXISTING SYSTEM

Existing fraud detection systems suffer from several critical disadvantages. Their reliance on static, pre-defined rules makes them inherently lack adaptability. As fraudsters constantly evolve their tactics, these static rules quickly become outdated, leaving the system vulnerable to new and unforeseen attack vectors. This leads to a constant game of catch-up, where systems are only updated after new fraud patterns are identified, rather than proactively preventing them. Furthermore, these systems often suffer from delayed fraud detection. Transactions are frequently flagged only after they have been completed, sometimes even days later. This delay significantly hinders recovery efforts, as the funds may have already been dispersed or the fraudster may have disappeared.

Another significant drawback is the high rate of false positives. Legitimate transactions are often incorrectly flagged as fraudulent, leading to unnecessary account freezes, customer frustration, and increased operational costs for financial institutions. This problem is exacerbated by the scalability issues inherent in many systems. Manual transaction monitoring, a common component of these systems, becomes increasingly inefficient and error-prone as transaction volumes grow. Human analysts simply cannot keep pace with the sheer number of transactions, leading to either overlooked fraudulent activity or an overwhelming number of false positives. Finally, many existing systems offer limited user protection. They often fail to provide real-time alerts or warnings to users, allowing fraudulent transactions to proceed unchecked. This leaves customers vulnerable to financial losses and erodes trust in the financial institution.

2.4 PROPOSED SYSTEM

SafeScan represents a significant advancement in fraud detection by leveraging the power of machine learning. Instead of relying on static rules, SafeScan employs sophisticated machine learning models that continuously learn and adapt from the vast amounts of transaction data they analyze. This dynamic learning process allows SafeScan to identify subtle patterns and anomalies that would be missed by traditional rule-based systems, significantly reducing the number of false positives. By minimizing these false alarms, SafeScan improves the overall efficiency of fraud detection, allowing analysts to focus on genuinely suspicious activity. Critically, SafeScan operates in real-time, providing immediate analysis of every transaction as it occurs. This real-time capability is crucial for proactive fraud prevention, enabling SafeScan to intervene and block potentially fraudulent transactions before they are completed, rather than simply flagging them after the fact.

The core strength of SafeScan lies in its use of machine learning algorithms. These algorithms are designed to automatically adapt to new and evolving fraud tactics. As fraudsters develop new methods, SafeScan's models learn to recognize these changes, ensuring that the system remains effective even against previously unseen attack vectors. This continuous adaptation is a key differentiator from traditional systems that require manual updates to their rules. SafeScan's real-time transaction analysis provides granular insights into each transaction, assessing its risk level based on a multitude of factors. This allows for highly accurate fraud detection and prevention.

These real-time alerts empower users to take immediate action, such as verifying the legitimacy of the transaction or freezing their account, preventing any financial loss from occurring. This proactive approach not only protects users from financial harm but also fosters greater trust in the security of their financial transactions.

2.5 CONCLUSION

SafeScan's strength lies in its strategic application of machine learning (ML) algorithms, which fundamentally enhance fraud detection across three key dimensions: accuracy, adaptability, and real-time monitoring. Traditional systems often struggle with accurately distinguishing between legitimate and fraudulent transactions, leading to high false positive rates. SafeScan's ML models, trained on vast datasets of transaction patterns, learn to identify subtle anomalies and suspicious behaviours that humans might miss, significantly improving the accuracy of fraud detection. This translates to fewer false alarms and a more focused approach to investigating genuinely suspicious activity.

Furthermore, the dynamic nature of fraud requires systems that can adapt to ever-evolving tactics. SafeScan's ML algorithms provide this crucial adaptability. Beyond accuracy and adaptability, SafeScan's real-time monitoring capabilities are essential for proactive fraud prevention. By analyzing transactions as they occur, SafeScan can identify and flag suspicious activity immediately, allowing for timely intervention and preventing fraudulent transactions from completing. This real-time approach is far more effective than traditional methods that often detect fraud only after the damage has been done.

This multi-faceted approach, leveraging ML for enhanced accuracy, adaptability, and real-time monitoring, significantly mitigates the risks associated with digital payment fraud. By proactively identifying and preventing fraudulent transactions, SafeScan fosters a safer and more secure transaction environment for both users and financial institutions. The comprehensive security mechanism offered by SafeScan goes beyond simply relying on ML-based fraud detection. The inclusion of QR code verification and UPI ID validation adds layers of security, ensuring that transactions are initiated and processed securely. This combination of advanced technologies creates a robust defence against fraud, surpassing the capabilities of traditional methods and making digital transactions safer, more reliable, and trustworthy for all stakeholders.

CHAPTER-3

ANALYSIS

3.1 INTRODUCTION

System analysis, a crucial stage in developing new information systems, involves gathering, analyzing, and interpreting data to enhance system performance. Led by systems analysts, its objective is to thoroughly understand proposed systems and their requirements, setting goals and improving effectiveness. This phase explores system objectives, identifies problems, and recommends improvements. User involvement helps pinpoint current activities and needs, guiding system enhancements. Despite inherent limitations, system analysis yields valuable results, ensuring systems effectively fulfil their intended purpose.

This phase includes a comprehensive evaluation of the proposed system. The assessment aims to understand the system's operation and functionalities. It also addresses system issues and end-user requirements for new or updated systems. Upon completion of this phase, analysts should possess a deep understanding of the system's operation and the requirements for the new system.

Users offer valuable insights into their existing activities, objectives, and needs. A system analyst dedicates time to understanding how users interact with the system, identifying challenges, and anticipating their expectations. System analysis is typically a demanding yet rewarding endeavour. This work presents various limitations and requires skilled individuals to execute it successfully.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 User requirements

- PC / Laptop / Mobile
- Webcam Support
- Internet connection
- Power supply

System study

Feasibility Study

This phase involves analyzing the project's feasibility and presenting a business proposal, encompassing a high-level plan and cost estimations. A feasibility study of the proposed system is conducted during system analysis to ensure the planned solution does not become a financial or operational burden for the organization.

A thorough understanding of the system's core requirements is essential for conducting this feasibility analysis. The feasibility analysis includes three main considerations:

- Economic feasibility
- Technical feasibility and
- Social feasibility.

Economic Feasibility

This study examines the financial implications of implementing the system across the organization. Given limited R&D funding, investments require strong justification. The system's design adheres to budgetary constraints, largely through the utilization of open-source or freely available technologies. Only necessary customized components or products are purchased, enabling cost-effective project implementation. Careful financial planning ensures adherence to budget limitations, maximizing resource utilization while minimizing the financial impact on the organization.

Technical Feasibility

The study aims to assess the system's technical feasibility, ensuring compatibility with existing technical resources. Preventing an overload of the client's technological infrastructure is crucial. The developed system should have modest resource requirements to minimize the need for significant infrastructure changes during implementation. This approach facilitates smooth integration without creating substantial technical hurdles. Balancing desired functionality with efficient resource utilization is essential for guaranteeing technical feasibility and successful system implementation.

Social Feasibility

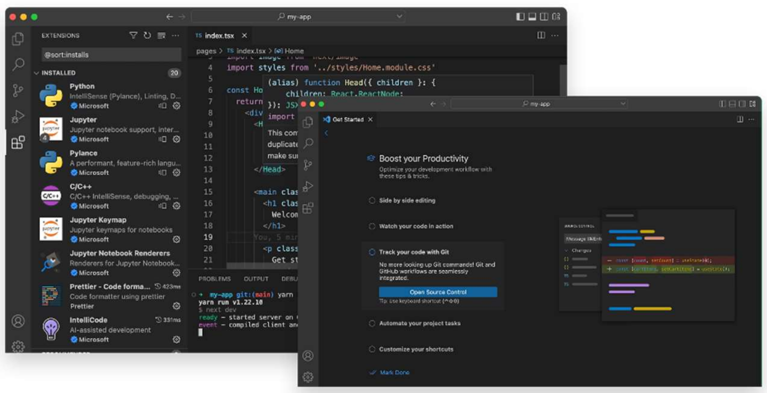
The research focuses on determining user acceptance of the technology, including effective user training techniques. Users should see the system as a necessity rather than a threat, increasing acceptability. User acceptability is determined by the methods used to educate and familiarise users with the system. Building user confidence is essential for encouraging constructive criticism because they are the ultimate end users. Creating a supportive atmosphere for customer feedback encourages continual progress and user pleasure.

3.2.2 SOFTWARE REQUIREMENTS

- Coding language - Python, Machine Learning, HTML, CSS, JavaScript
- Operating System - Windows
- Software - Flask
- Platform - Microsoft Visual Studio Code

Microsoft Visual Studio Code (IDE)

Visual Studio Code (VS Code) is a powerful and versatile code editor developed by Microsoft. It offers a rich set of features, including a text editor for writing and editing code, an integrated terminal for running commands, a robust debugging system, and extensive support for various programming languages through extensions. The editor provides features like syntax highlighting, code completion (IntelliSense), and code folding to enhance developer productivity. The integrated terminal allows developers to execute commands directly within VS Code, streamlining workflows. The debugging capabilities enable developers to identify and fix errors efficiently. VS Code's extensibility allows users to customize the editor with a wide range of extensions, adding support for languages, tools, and other functionalities. This modular design makes VS Code highly adaptable to different development needs and preferences.


my-app
EXTENSIONS
@sort:installs
INSTALLED 20
Python
IntelliSense (Pylance), Linting, D...
Microsoft
Jupyter
Jupyter notebook support, inter...
Microsoft
Pylance
A performant, feature-rich langu...
Microsoft
C/C++
C/C++ IntelliSense, debugging, ...
Microsoft
Jupyter Keymap
Jupyter keymaps for notebooks
Microsoft
Jupyter Notebook Renderers
Renderers for Jupyter Notebook...
Microsoft
Prettier - Code forma...
Code formatter using prettier
Prettier
IntelliCode
AI-assisted development
Microsoft
index.tsx
pages > index.tsx > Home
PROBLEMS
OUTPUT
Get Started
Boost your Productivity
Optimize your development workflow with these tips & tricks.
Side by side editing
Watch your code in action
Track your code with Git
No more looking up Git commands! Git and GitHub workflows are seamlessly integrated.
Open Source Control
Automate your project tasks
Customize your shortcuts
Mark Done

```
import styles from '../styles/Home.module.css'

const Ho
  return
    <div
      <H
      </Head>

      <main clas
        <h1 clas
          Welcom
        </h1>

        <p class
          Get st
```

```
(alias) function Head({ children }: {
    children: React.ReactNode;
}): JS
```

```
→ my-app git:(main) yarn
yarn run v1.22.10
$ next dev
ready - started server on
event - compiled client an
```

Fig 3.2.2.1 Microsoft VS Code IDE Outlook

When you open Visual Studio Code (VS Code), the interface will resemble the figure above. While minor visual differences might exist depending on your operating system (macOS, Linux, or Windows), the core functionality of VS Code remains consistent. The Activity Bar, typically on the left side, provides access to key features like the Explorer (for navigating files and folders), Search, Source Control (for Git integration), Run and Debug, and Extensions.

The Editor area is where you write and edit your code. The Status Bar at the bottom display's information about the current file and project. VS Code also offers a wide range of built-in features and a vast library of extensions, enabling users to customize the editor to suit their specific needs and programming languages.

Windows Software

Microsoft introduced an operating environment named windows on November 20, 1985, as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Apple came to see Windows as an unfair encroachment on their innovation in GUI development as 19 implemented on products such as the Lisa and Macintosh (eventually settled in court in Microsoft's favour in 1993). On PCs, Windows is still the most popular operating system. However, in 2014, Microsoft admitted losing the majority of the overall operating system market to Android, because of the massive growth in sales of Android smartphones. In 2014, the number of Windows devices sold was less than 25% that of Android devices sold. This comparison however may not be fully relevant, as the two operating systems traditionally target different platforms. Still, numbers for server use of Windows (that are comparable to competitors) show one third market share, similar to that for end user use. As of October 2018, the most recent version of Windows for PCs, tablets, smartphones, and embedded devices is Windows 10. The most recent versions for server computers is Windows Server 2019.

Python

Python is a high-level, general-purpose programming language renowned for its readability and versatility. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its use of significant [1] indentation. This focus on clean syntax makes it easier to learn and use, contributing to its popularity among beginners and experienced developers alike. Python is dynamically typed, meaning variable types are checked at runtime, and it supports multiple programming paradigms, including object-oriented, imperative, and functional programming styles. This flexibility allows developers to choose the most appropriate approach for their projects.

Key features of Python include:

- **Interpreted:** Python code is executed line by line by an interpreter, making development and testing faster.

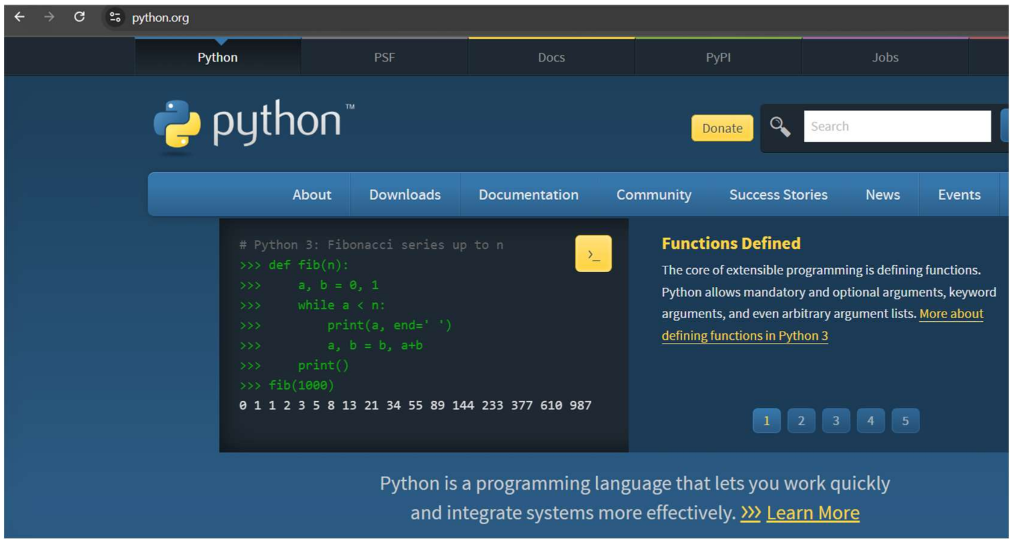
- **Extensive Standard Library:** Python comes with a rich collection of pre-built modules and functions, providing tools for a wide range of tasks, from string manipulation and file I/O to networking and web development.
- **Large and Active Community:** Python boasts a vast and supportive community, contributing to its growth and providing ample resources, tutorials, and third-party libraries.
- **Cross-Platform:** Python code can run on various operating systems, including Windows, macOS, and Linux, promoting code portability.
- **Dynamically Typed:** Python's dynamic typing simplifies development as variable types are automatically inferred.
- **Garbage Collection:** Python automatically manages memory allocation and deallocation, freeing developers from manual memory management.
- **Extensive Third-Party Libraries:** Beyond the standard library, a vast ecosystem of third-party packages (like NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch) extends Python's capabilities for scientific computing, data analysis, machine learning, and more.

Python finds applications in a wide variety of domains:

- **Web Development:** Frameworks like Django and Flask make Python a popular choice for building web applications and APIs.
- **Data Science and Machine Learning:** Libraries like NumPy, Pandas, and Scikit-learn make Python a powerful tool for data analysis, manipulation, and machine learning.
- **Scientific Computing:** Python is widely used in scientific research and engineering due to its numerical computation libraries.
- **Scripting and Automation:** Python's scripting capabilities make it ideal for automating tasks, system administration, and creating small utility programs.
- **Education:** Python is easy-to-learn syntax makes it a popular language for teaching programming to beginners.
- **Game Development:** Libraries like Pygame allow for game development in Python.

Installing Python

Installing Python involves several key steps, and the process can vary slightly depending on your operating system (Windows, macOS, etc). It is generally recommended to download the latest stable version of Python from the official Python website (python.org)



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Fig 3.2.2.2 Python Home page

Windows:

1. **Download:** Visit python.org and download the Windows installer (usually a .exe file) for the desired Python version. Choose the appropriate installer (32-bit or 64-bit) based on your system architecture

2. **Run the Installer:** Double-click the downloaded .exe file to run the installer.

3. **Customize Installation (Important):** During the installation, **make sure to check the box that says "Add Python to PATH."** This is crucial for being able to run Python commands from your command prompt or PowerShell. You can also customize the installation location if needed.

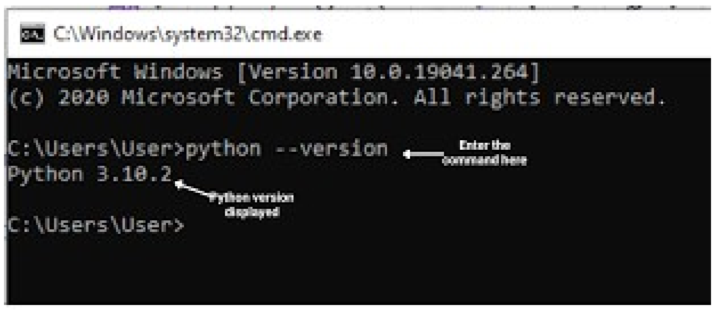
4. **Install:** Click "Install Now" to begin the installation process.

5. **Verify Installation:** Open a command prompt or PowerShell and type python --version (or py --version on some systems). If Python is installed correctly, you should see the Python version number displayed.

macOS:

macOS often comes with a pre-installed version of Python, but it is usually an older version. It is recommended to install a newer version.

1. **Download:** Go to python.org and download the macOS installer (usually a .pkg file).
2. **Run the Installer:** Double-click the downloaded .pkg file and follow the on-screen instructions to install Python.
3. **Verify Installation:** Open Terminal and type python3 --version. This should display the version of Python you just installed. (Older macOS versions might use python for the system Python, so python3 is generally safer to use for your newly installed version.)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\User>python --version
Python 3.10.2

C:\Users\User>
```

Fig 3.2.2.3 Python Installation

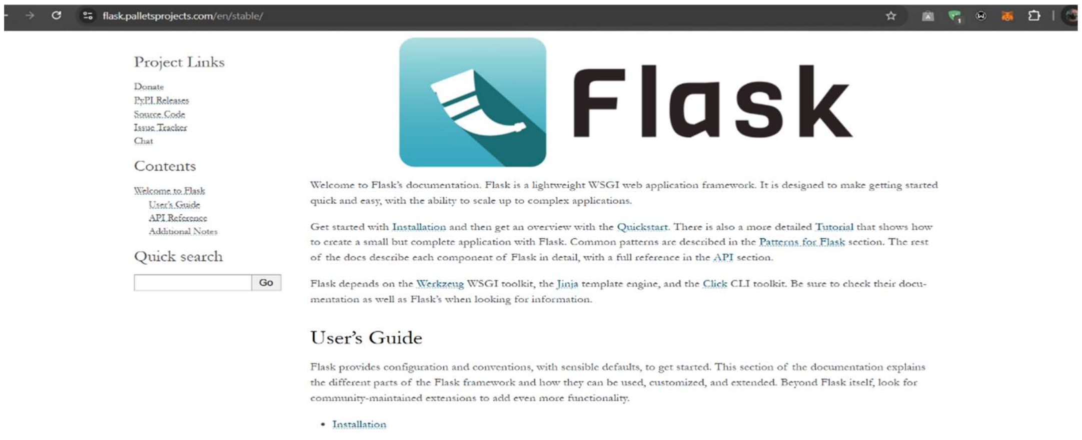
Flask

Flask is a lightweight and flexible web framework for Python. It's classified as a microframework, meaning it provides the essential tools for web development but doesn't enforce a specific project structure or include features like a database abstraction layer or form validation out of the box. This minimalist approach gives developers a high degree of control and allows them to choose the components they need, making Flask ideal for projects of all sizes, from small APIs to complex web applications.

Key features of Flask include:

- **Routing:** Flask makes it easy to define URL endpoints and map them to specific functions. This allows you to create dynamic web pages and APIs that respond to different requests.

- **Templating:** Flask integrates with Jinja2, a powerful templating engine. This allows you to embed Python code within HTML templates, making it easier to generate dynamic HTML content.
- **WSGI Compliance:** Flask is WSGI (Web Server Gateway Interface) compliant, meaning it can work with various web servers like Gunicorn or uWSGI in production environments.
- **Extensions:** Flask's extensibility is a major strength. A wide range of extensions are available to add functionality like database integration (SQLAlchemy, Flask-SQLAlchemy), form handling (Flask-WTF), authentication (Flask-Login), and more. This allows developers to tailor Flask to their exact needs.
- **Lightweight and Simple:** Flask's microframework nature makes it easy to learn and use. Its simplicity reduces boilerplate code and allows developers to focus on the core logic of their applications.
- **Large Community:** Flask boasts a vibrant and active community, providing ample resources, tutorials, and support for developers.



Project Links

Donate
PyPI Releases
Source Code
Issue Tracker
Chat

Contents

Welcome to Flask
User's Guide
API Reference
Additional Notes

Quick search

Welcome to Flask's documentation. Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.

Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.

Flask depends on the Werkzeug WSGI toolkit, the Jinja template engine, and the Click CLI toolkit. Be sure to check their documentation as well as Flask's when looking for information.

User's Guide

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

- Installation

Fig 3.2.2.4 Flask Home page

Because Flask is so lightweight, it is often a good choice for:

- **Small to medium-sized web applications:** Its simplicity makes it quick to develop and deploy smaller projects.
- **APIs:** Flask is excellent for building RESTful APIs due to its straightforward routing and request handling.

- **Prototyping:** Flask's rapid development capabilities make it ideal for quickly prototyping web application ideas.
- **Learning web development:** Flask's relatively small codebase makes it easier to understand the underlying principles of web frameworks.

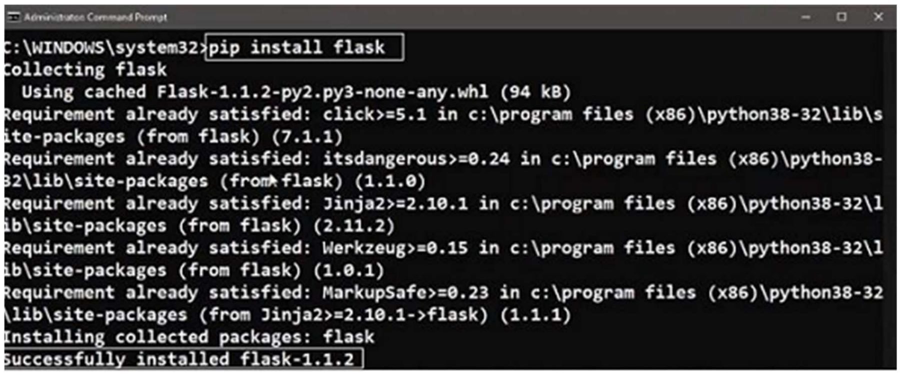
While Flask's minimalist nature is a strength, it can also be a perceived limitation. For very large and complex applications, some developers might prefer a more "batteries-included" framework like Django. However, even in these cases, Flask's flexibility and extensibility can make it a viable choice with the right set of extensions.

Installing Flask

Installing Flask involves several key steps. Flask recommends using the latest Python version (3.9 or newer). Several dependencies, including

- Werkzeug (for WSGI)
- Jinja (for templating)
- MarkupSafe (for security)
- ItsDangerous (for secure data signing)
- Click (for command-line applications)
- Blinker (for Signals)

These are automatically installed with Flask. Optional dependencies like python-dotenv (for environment variables) and Watchdog (for a faster development server reloader) can be installed separately. If using gevent or eventlet, greenlet>=1.0 is required (and PyPy>=7.3.7 for PyPy).



```
:\WINDOWS\system32>pip install flask
ollecting flask
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
equirement already satisfied: click>=5.1 in c:\program files (x86)\python38-32\lib\s
ite-packages (from flask) (7.1.1)
equirement already satisfied: itsdangerous>=0.24 in c:\program files (x86)\python38-
2\lib\site-packages (from flask) (1.1.0)
equirement already satisfied: Jinja2>=2.10.1 in c:\program files (x86)\python38-32\l
ib\site-packages (from flask) (2.11.2)
equirement already satisfied: Werkzeug>=0.15 in c:\program files (x86)\python38-32\l
ib\site-packages (from flask) (1.0.1)
equirement already satisfied: MarkupSafe>=0.23 in c:\program files (x86)\python38-32
\lib\site-packages (from Jinja2>=2.10.1->flask) (1.1.1)
nstalling collected packages: flask
uccessfully installed flask-1.1.2
```

Fig 3.2.2.5 Flask Installation

It is highly recommended to use virtual environments for managing project dependencies. Virtual environments isolate project dependencies, preventing conflicts between different projects that might require different library versions. Python's venv module is used to create these isolated environments. First, create a project folder (e.g., myproject) and a .venv folder within it using python3 -m venv .venv. Then, activate the environment (e.g., . .venv/bin/activate on macOS/Linux or .venv\Scripts\activate on Windows). The shell prompt will indicate the active environment. Finally, install Flask within the activated environment using pip install Flask. Once installed, you can proceed to the Quickstart or explore the Flask documentation.

3.3 CONTENT DIAGRAM OF PROJECT

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFD's can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data, flows as a unified model.

A DFD has four different symbols:

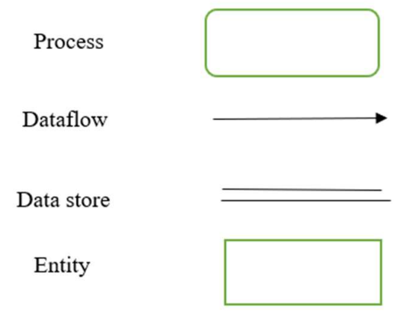


Fig 3.3.1 Content Diagram

Process: It is represented by a circle and it denotes transformations of the input data to produce output data.

Dataflow: It is represented by arrows connecting one data transformation to another. It represents the movement of data that is leaving one process and entering into another process.

Data Store: Data store is data at rest. Incoming arrows indicate the retrieval of data from the data store. It provides temporary storage of data between processes or be a permanent data repository.

Entity: It is the external entity that represents the source. It is represented by a rectangle. It represents people, organization or another system that produces or consumes data. It lies on the boundary of the system but provides information for the system.

Block diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic devise, software design, and process flow diagram. Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagram and layout diagram used in electrical engineering, which show the implementation details of electrical components and physical construction.

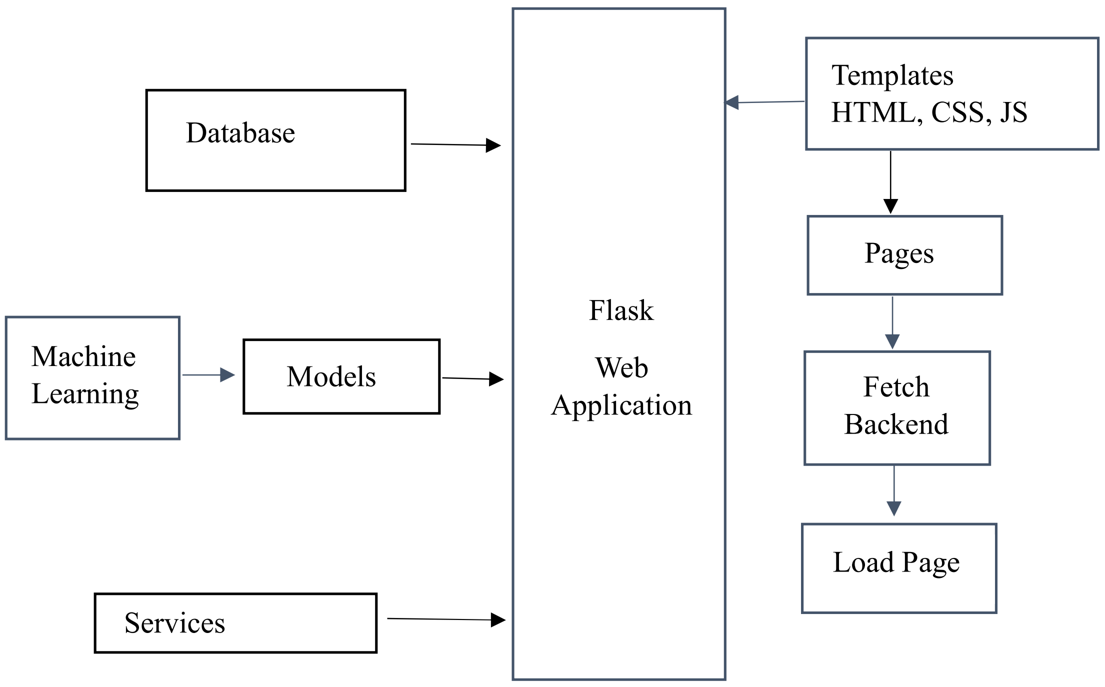


Fig 3.3.2 Block Diagram of Project

3.4 FLOWCHART

A flowchart is a type of diagram that represents an algorithm, workflow, or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem.

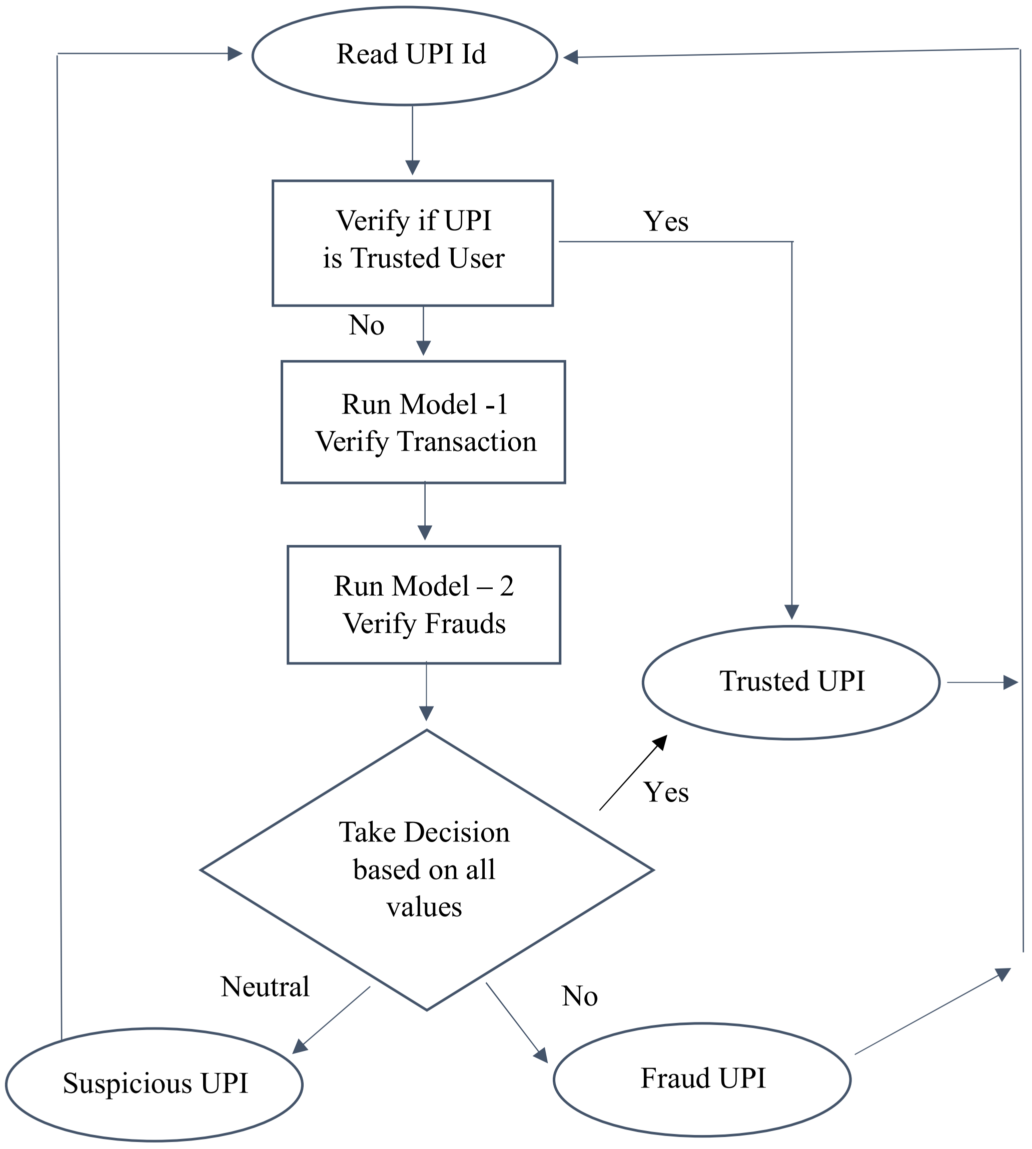
Flowcharts are used in analysing, designing, documenting, or managing a process or program in various fields.

- Flow line
- Terminal
- Process
- Decision

A **flow line** that shows the program's order of operation. A line coming from one symbol and ending at another. Arrow heads are added if the flow is not the standard top to bottom, left to right.
Terminal is beginning or ending of a program or sub-process. Represented as oval or rounded (fillet) rectangle. They usually contain the word "Start" or "End", or another phrase signalling the start or end of a process, such as "submit inquiry" or "receive product".
Process is set of operations that change value, form, or location of data. It is represented as a rectangle.
Decision is a conditional operation determining which of two paths the program will take. The operation is commonly a Yes/ No question or True/ False test. It is represented as a Diamond.

A simple flowchart representing a process for dealing with a non-functioning lamp. A flowchart is a type of diagram that represents an algorithm, workflow, or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. The two most common types of boxes in a flowchart
are:

- A processing step, usually called activity, and denoted as a rectangular box
- A decision usually denoted as a diamond.



Read UPI Id
Verify if UPI is Trusted User
Yes
No
Run Model -1 Verify Transaction
Run Model – 2 Verify Frauds
Trusted UPI
Yes
Take Decision based on all values
Neutral
No
Suspicious UPI
Fraud UPI

Fig 3.4.1 Flowchart

When a user scans a UPI QR code or enters a UPI ID, SafeScan immediately initiates a verification process. The first step involves checking a pre-established list or database of trusted UPI IDs. If the scanned UPI ID is found on this list, SafeScan quickly flags the user as "Trusted," streamlining the transaction process and minimizing any delays. This initial trust check significantly speeds up verification for known and reputable users.

If the UPI ID is not recognized as trusted, SafeScan proceeds to a more in-depth analysis using a machine learning model. This model plays a crucial role in identifying potentially fraudulent activity.

First, the model securely fetches the relevant bank statement associated with the scanned UPI ID. This statement provides a history of transactions, which the model then analyzes. The model performs two key predictions based on this statement: it estimates a typical transaction amount for the user and it predicts the likelihood of the user having been involved in fraudulent transactions in the past. These predictions are crucial because significant deviations from normal transaction amounts or a history of fraudulent activity are strong indicators of potential fraud.

Finally, SafeScan combines the initial trust assessment with the model's predictions to determine the overall risk level associated with the scanned UPI ID. If the model predicts a high likelihood of fraudulent activity or a significant deviation from typical transaction patterns, or if the UPI was not initially trusted, SafeScan flags the transaction as "Fraud" or "Suspicious," triggering appropriate security measures, such as blocking the transaction or requiring additional verification. This multi-layered approach, combining a trust check with sophisticated machine learning analysis, allows SafeScan to effectively identify and prevent fraudulent UPI transactions, ensuring a safer digital payment environment.

3.5 CONCLUSION

The project analysis phase offered a thorough grasp of the project's needs and components, which are critical to its effective implementation. The demarcation of software, hardware, and user requirements has identified the tools, technologies, and functions required to achieve the project's objectives. The block diagram and flowchart visualised the system architecture and operational workflow, providing insight into the system's structure and functionality. By rigorously analysing these elements, the project team has established a firm basis for the next stages of development and implementation. This systematic method assures alignment with user expectations, technical feasibility, and optimal resource utilisation, thus laying the foundation for the project's successful execution and delivery.

CHAPTER-4

DESIGN

4.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analysed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

4.2 UML DIAGRAM

Unified Modelling Language is popular for its diagrammatic notations. Any complex system can be easily understandable by making some kind of pictures or diagrams. These diagrams have a better impact on our understanding in a better and simple way.

To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UML's basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout blocks may be put together, and some common mechanisms that apply throughout the UML. Once you have grasped these ideas, you will be able to read UML models and create some basic ones. As you gain more experience in applying the UML. You can build on this conceptual model, using more advanced features of the language.

USE CASE DIAGRAM

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams depict the functionality of a system through actors and use cases. Use cases are a collection of tasks, services, and operations that the system must accomplish. In this context, a "system" refers to something being built or operated, like a website. The "actors" are individuals or things that perform certain responsibilities within the system.

Use case diagrams help visualise a system's functional needs, which inform design decisions and development priorities. They indicate internal and external issues that may affect the system and should be considered.

They provide a good high-level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

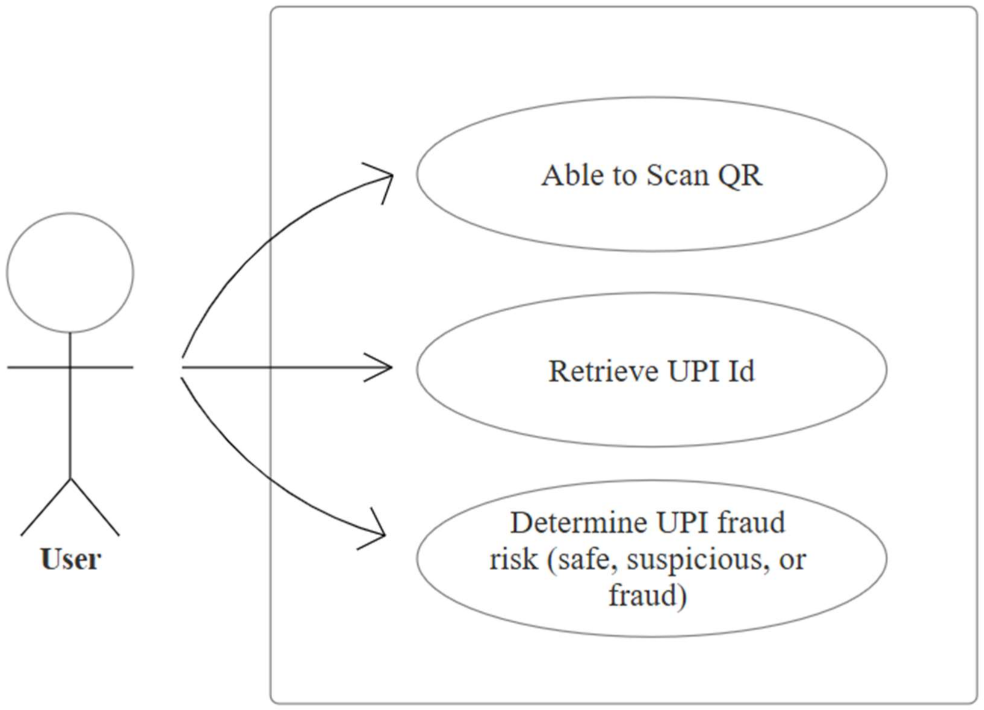


Fig 4.2.1 Use case diagram

SEQUENCE DIAGRAM

Sequence diagram describes the pattern of communication among set of interacting objects. An object interacts with other objects by sending messages. The reception of a message by an object triggers the execution of an operation, which in turn may send messages to other objects. Arguments may be passed along with a message and are bound to the parameters of the executing operation in the receiving object. Sequence diagram represents the behaviour of the system from the perspective of single use case.

A sequence diagram emphasizes the time ordering of messages. A type of interaction diagram, a sequence diagram shows the actor of objects participating in an interaction and the events they generate arranged in a time sequence. Often, a sequence diagram shows the event that result from a particular instance of a use case but a sequence diagram can also exist in amore generic form. Graphically, a sequence diagram is table that shows object arranged along the X-axis and the messages, ordered in increasing time, along the Y-axis. The vertical dimension in a sequence diagram represents time, with time proceeding down the page.

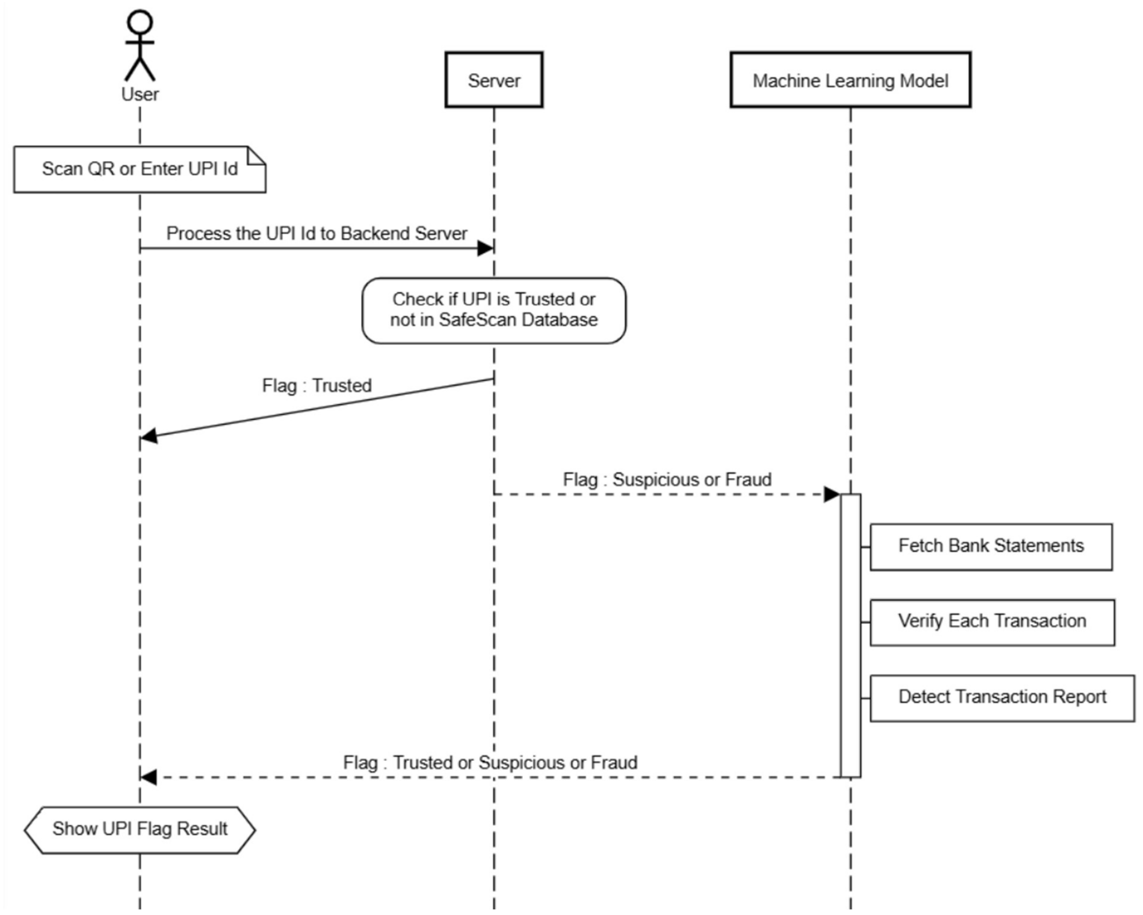

User
Server
Machine Learning Model
Scan QR or Enter UPI Id
Process the UPI Id to Backend Server
Check if UPI is Trusted or not in SafeScan Database
Flag : Trusted
Flag : Suspicious or Fraud
Fetch Bank Statements
Verify Each Transaction
Detect Transaction Report
Flag : Trusted or Suspicious or Fraud
Show UPI Flag Result

Fig 4.2.2 Sequence diagram

CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

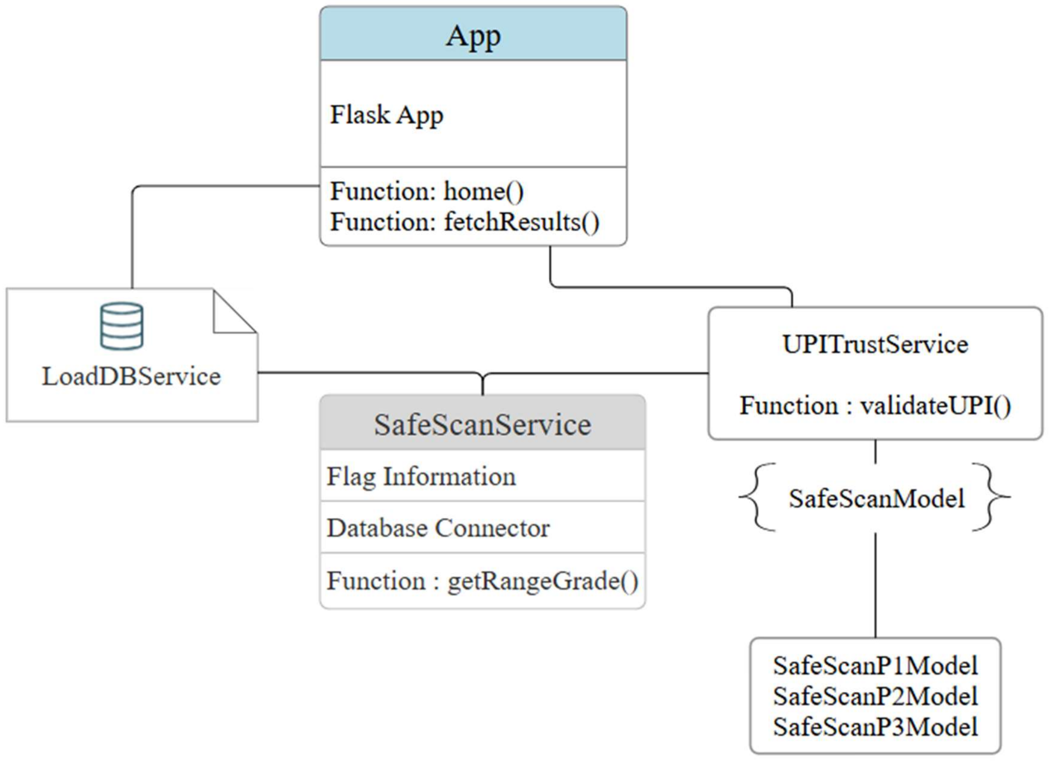


Fig 4.2.3 Class Diagram

4.3 MODULE DESIGN AND ORGANISATION

Regardless of the application and development paradigm, software design is the technical foundation of the software engineering process. Any engineered system or product begins its development phase with design. The aim of the designer is to create a model or representation of a thing that will be constructed in the future. System design is the first of the three technical tasks—design, coding, and testing—necessary to create and validate software, starting after system requirements have been defined and examined. The word "quality" can sum up. In software development, quality is nurtured in the design stage.

We can only effectively translate a customer's perspective into a completed software product or system through design. If the design is weak, we run the risk of creating an unstable system that will be challenging to test and whose quality will not be known until the very end.

CHAPTER-5

IMPLEMENTATION

&

RESULTS

5.1 INTRODUCTION

The implementation phase marks the key transition from theoretical design to functioning system, and it is an important stage in guaranteeing a new project's success and effectiveness. During this phase, conceptual concepts and plans are transformed into operational solutions. This stage requires rigorous preparation, including a full evaluation of the present system and its constraints, in order to properly inform the implementation approach. By carefully considering and strategically preparing these variables, the implementation phase lays the groundwork for the system's performance and success, instilling user confidence in its capabilities and efficacy.

5.2 EXPLANATION OF KEY FUNCTIONS

In Flask, app.route() and Flask() are two critical functions that play different roles in webapp.

app.route():

The @app.route() decorator is fundamental to how Flask handles web requests. It is the mechanism that connects specific URLs to the functions that will process them. Think of it as a traffic controller, directing incoming requests to the appropriate destination within your application. When a user visits a particular URL in their browser, Flask uses the @app.route() decorator to match that URL to a defined route. The function immediately following the decorator is then executed, generating the response that the user sees. For example, @app.route('/home') tells Flask that whenever a user requests the /home URL, the function directly below it should be called. This decorator is what makes Flask dynamic, allowing you to define different behaviours for different URLs and create a rich, interactive web experience. It is the key to mapping user requests to the logic that powers your web application.

Flask():

The line app = Flask(__name__) is the cornerstone of any Flask application. It creates an instance of the Flask class, which is the core object representing your web application. Think of it as the central hub that manages all the different parts of your app. The __name__ argument, a special Python variable, plays a crucial role. It tells Flask the name of the current module or package where your application is defined. This information is essential for Flask to correctly locate templates, static files (like CSS and JavaScript), and other resources that your application might need. Without __name__, Flask would not know the context of your application, leading to errors. So, app = Flask(__name__) not only creates the Flask instance but also establishes the necessary context for it to function properly, effectively laying the groundwork for your entire web application.

5.3 METHOD OF IMPLEMENTATION

5.3.1 PROGRAM

- app.py

```
from flask import Flask, request, render_template
from services import UPITrustService, LoadDBService as db

app = Flask(__name__)

@app.route('/')
def home():
    db.load()
    return render_template('index.html');

@app.route('/safescan-result', methods=["POST"])
def fetchResults():
    upiId = request.form.get('upi-id')
    response = UPITrustService.validateUPI(upiId)
    return render_template('results.html', resp = response)

if __name__ == "__main__":
    app.run(debug=True)
```

- LoadDBService.py

```
import pandas as pd
from . import SafeScanService

upiList = {}

def load():
    data = pd.read_excel(SafeScanService.DB_URI_PATH)

    global upiList
    upiList = data.to_dict(orient='records')
```

- SafeScanService.py

```
# SafeScan Output Rules

FRAUD = {"state": True, "icon": "fraud-icon", "message": "Fraud"}
WARNING = {"state": True, "icon": "warning-icon", "message": "Suspicious"}
TRUSTED = {"state": True, "icon": "trusted-icon", "message": "Trusted"}

# SafeScan Internal Technical Issues

UNKNOWN = {"error": "SafeScan : UPI Unavailable", "icon": "unknown-icon", "message": "Sorry
for the inconvenience — SafeScan is in beta with limited data, and the provided UPI ID isn't available.
We're evolving quickly and will adapt to deliver results for this UPI ID soon. Thank you for your
patience!"}
UNAUTH = {"error": "SafeScan Error", "icon": "restricted-icon", "message": "Unauthorized User :
Access Restricted"}
ERROR = {"error": "SafeScan Error", "icon": "error-icon", "message": "Sorry, We are currently
experiencing technical difficulties."}
```

```
# SafeScan DataBase Rules

DB_BASE = "./database/"
DB_UPI_BASE = DB_BASE + "global/"
DB_BASE_EXTENSION = ".xlsx"
DB_CSV_EXTENSION = ".csv"

DB_URI_PATH = DB_BASE + "/UPIDetails" + DB_BASE_EXTENSION
DB_UPI_PATH = lambda eid : DB_UPI_BASE + str(eid) + DB_CSV_EXTENSION

# SafeScan Result Generator

def getRangeGrade(fraud_count, non_fraud_count):
    if fraud_count < 1 and non_fraud_count < 1 : return None

    total_count = 100 / (fraud_count + non_fraud_count)
    score = fraud_count * total_count

    if score < 46 : return "G"
    elif score > 45 and score < 64 : return "Y"
    else : return "R"
```

- SafeScanModel.py

```
import numpy as np
import joblib
from sklearn.preprocessing import LabelEncoder
from . import SafeScanP1Model, SafeScanP2Model

def predict(path, safeScanP3):
    safeScanP1 = SafeScanP1Model.processModel(path)
    safeScanP2 = SafeScanP2Model.processModel(path)
    if safeScanP1 == None or safeScanP2 == None : return None

    mainModel = joblib.load("./models/libs/SafeScanMainModel.pkl")
    encoder = LabelEncoder()
    encoder.fit(["R", "Y", "G"])

    enc1 = encoder.transform([safeScanP1.upper()])[0]
    enc2 = encoder.transform([safeScanP2.upper()])[0]
    enc3 = encoder.transform([safeScanP3.upper()])[0]
    X_input = np.array([[enc1, enc2, enc3]])

    result = mainModel.predict(X_input)
    label_map = {0: "R", 1: "Y", 2: "G"}

    return label_map[result[0]]
```

- SafeScanP1Model.py

```
import pandas as pd
import numpy as np
import joblib
from sklearn.impute import SimpleImputer
from services import SafeScanService

loaded_model = joblib.load("./models/libs/SafeScanP1_Model.pkl")

def processModel(path):
    data = pd.read_csv(path)
    features = ["type", "amount", "oldbalanceOrg", "newbalanceOrig"]

    if not all(col in data.columns for col in features):
        return None

    data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2, "CASH_IN": 3,
"TRANSFER": 4, "DEBIT": 5})

    imputer = SimpleImputer(strategy='mean')
    predict = imputer.fit_transform(data[features].values)
    predictions = loaded_model.predict(predict)

    fraud_count = np.sum(predictions == "Fraud")
    non_fraud_count = np.sum(predictions == "No Fraud")

    return SafeScanService.getRangeGrade(fraud_count, non_fraud_count)
```

- SafeScanP2Model.py

```
import pandas as pd
from services import SafeScanService

def processModel(path):
    data = pd.read_csv(path)
    counts = data['Report'].value_counts().to_dict()

    return SafeScanService.getRangeGrade(counts[True], counts[False])
```

- index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home | SafeScan</title>
    <link rel="icon" type="image/x-icon" href="{{url_for('static',filename='images/shield.png')}}">
    <link rel="stylesheet" href="{{url_for('static',filename='css/styles.css')}}">
    <link rel="stylesheet" href="{{url_for('static',filename='css/icons.css')}}">
    <script src="{{url_for('static',filename='js/html5-qrcode.min.js')}}"></script>
    <script src="{{url_for('static', filename='js/index.js')}}"></script>
</head>
<body>
    <img class="left-panel" src="{{url_for('static',filename='images/bg-image.png')}}" alt="Logo">
    <div class="right-panel">
        <h2>Welcome to SafeScan</h2>
        <p>This tool verifies UPI IDs & QR codes to determine<br>If a transaction is safe.</p>
        <div class="container">
            <form id="upi-form" action="{{url_for('fetchResults')}}" method="POST">
                <input type="text" name="upi-id" id="upi-id" placeholder="Enter UPI ID">
                <button class="button" type="submit">Scan <span class="arrow-right"></span></button>
            </form>
            <p class="error d-none" id="error-msg">Please Enter Proper ID. Eg: name@bank</p>
            <div class="qr-scanner" id="qr-scanner"></div>
        </div>
    </div>
</body>
</html>
```

- results.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Results | SafeScan</title>
    <link rel="icon" type="image/x-icon" href="{{url_for('static',filename='images/shield.png')}}">
    <link rel="stylesheet" href="{{url_for('static',filename='css/styles.css')}}">
    <link rel="stylesheet" href="{{url_for('static',filename='css/icons.css')}}">
</head>
<body>
    <img class="left-panel" src="{{url_for('static',filename='images/bg-image.png')}}" alt="Logo">
    <div class="right-panel">
        <span class="results-icon {{resp.icon}}"></span>
        {% if resp.state %}
            <h3>SafeScan marks this UPI as {{resp.message}}<br><br></h3>
            <p>Make sure you verify and ensure safety before any Transaction <br> Thank you for trusting 
SafeScan!</p>
            <a class="a-button" href="{{url_for('home')}}">Go to Home <span class="arrow-right"></span></a>
            <p class="note">SafeScan is in beta, still learning and evolving — verify wisely.</p>
        {% else %}
            <h2 class="h2-mb">{{resp.error}}</h2>
            <p>{{resp.message}}</p>
            <a class="a-button" href="{{url_for('home')}}">Go to Home <span class="arrow-right"></span></a>
        {% endif %}
    </div>
</body>
</html>
```

- index.js

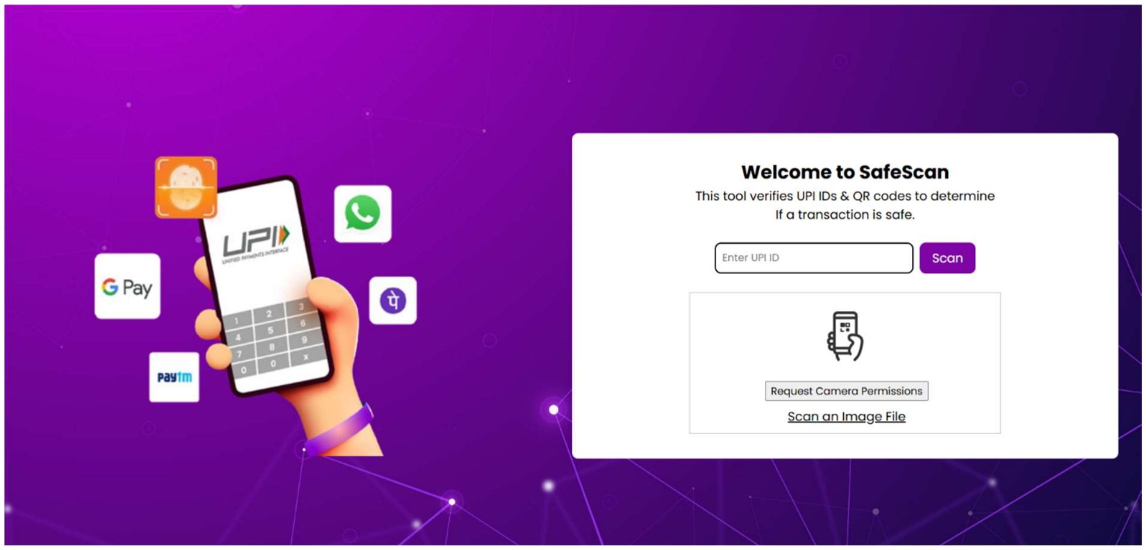
```
let upiField;
let errorElement;

let invalidUPI = () => alert("Invalid UPI");

window.onload = () => {
    let html5QrcodeScanner = new Html5QrcodeScanner("qr-scanner", { fps: 15, qrbox: 250 });
    html5QrcodeScanner.render(onScanSuccess);
    let upiForm = document.querySelector("#upi-form");
    upiField = document.querySelector("#upi-id");
    errorElement = document.querySelector("#error-msg");
    upiField.addEventListener("blur", validateUPI);
    upiField.addEventListener("focus", removeValidation);
    upiForm.addEventListener("submit", (e) => validateForm(e));
};

function onScanSuccess(decodedText) {
    let queryString = decodedText.split('?')[1];
    if (queryString) {
        let params = new URLSearchParams(queryString);
        let upiId = params.get("pa");
        if (upiId) {
            let upiField = document.getElementById("upi-id");
            upiField.value = upiId;
            validateUPI();
        } else invalidUPI();
    } else invalidUPI();
}

function validateUPI() {
    let upiReg = /^[a-zA-Z0-9.\-_]+@[a-zA-Z0-9.\-_]+$/;
    let isValid = upiReg.test(upiField.value.trim());
    upiField.classList.toggle("error", !isValid);
    errorElement.classList.toggle("d-none", isValid);
    return isValid;
}

function removeValidation() {
    upiField.classList.remove("error");
    errorElement.classList.add("d-none");
}

function validateForm(form) {
    let isValid = validateUPI();
    if (!isValid) {
        form.preventDefault();
    }
}
```

- styles.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,
800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');

:root {
    --black: #000000;
    --white: #ffffff;
    --violet: #7e04a4;
    --red: #ed4337;
}

*{
    margin: 0;
    scroll-behavior: smooth;
    font-family: "Poppins", serif;

    .error{
        color: var(--red) !important;
        border-color: var(--red) !important;
    }

    .d-none{
        display: none !important;
    }
}

body{
    background: url("../images/bg-color.jpg");
    background-size: cover;
    display: flex;
    align-items: center;
    height: 100vh;
}

.left-panel{
    width: 50%;
    height: 100%;
    object-fit: contain;
}

.right-panel{
    background-color: var(--white);
    width: 100%;
    margin-right: 4rem;
    border-radius: .5rem;
    padding: 2rem;

    h2,h3,p{
        text-align: center;
    }

    .h2-mb{
        margin-bottom: .25rem;
    }

    .note{
        font-style: italic;
        margin-top: 2rem;
        color: var(--red);
    }

    .button,.a-button{
        display: flex;
        align-items: center;
        padding: .5rem 1rem;
        background-color: var(--violet);
        color: var(--white);
        width: fit-content;
        border: none;
        outline: none;
        border-radius: .5rem;
        font-size: 1rem;
        transition: all 0.1s ease;

        &:hover{
            cursor: pointer;
            outline: 2px solid var(--violet);
            background-color: var(--white);
            color: var(--violet);
            box-shadow: 0 8px 16px 0 rgba(0,0,0,0.24);

            .arrow-right{
                display: block;
                width: 1rem;
                height: 1rem;
                margin-left: .5rem;
            }
        }
    }

    .a-button{
        text-decoration: none;
        margin: 1.5rem auto 0;
    }
}

.container{
    display: flex;
    flex-direction: column;
    margin-top: 1.5rem;
    align-items: center;

    .qr-scanner{
        width: 400px;
        margin-top: 1.5rem;

        img[alt="Info icon"] {
            display: none;
        }

        .html5-qrcode-element {
            margin-left: .25rem;
            margin-bottom: .5rem;
            margin-top: .25rem;
        }
    }

    form{
        display: flex;
        gap: .5rem;
        width: 100%;
        justify-content: center;

        input{
            padding: .5rem;
            border-radius: .5rem;
            border: 2px solid var(--black);
            width: 40%;
        }
    }

    #error-msg{
        margin-top: .5rem;
        font-size: 14px;
    }
}

@media screen and (max-width : 1023px) {
    body{
        justify-content: center;
    }

    .left-panel{
        width: 40%;
    }

    .right-panel{
        margin-right: 2rem;
    }

    .container{
        form{
            input{
                width: 100%;
            }
        }
    }
}
```

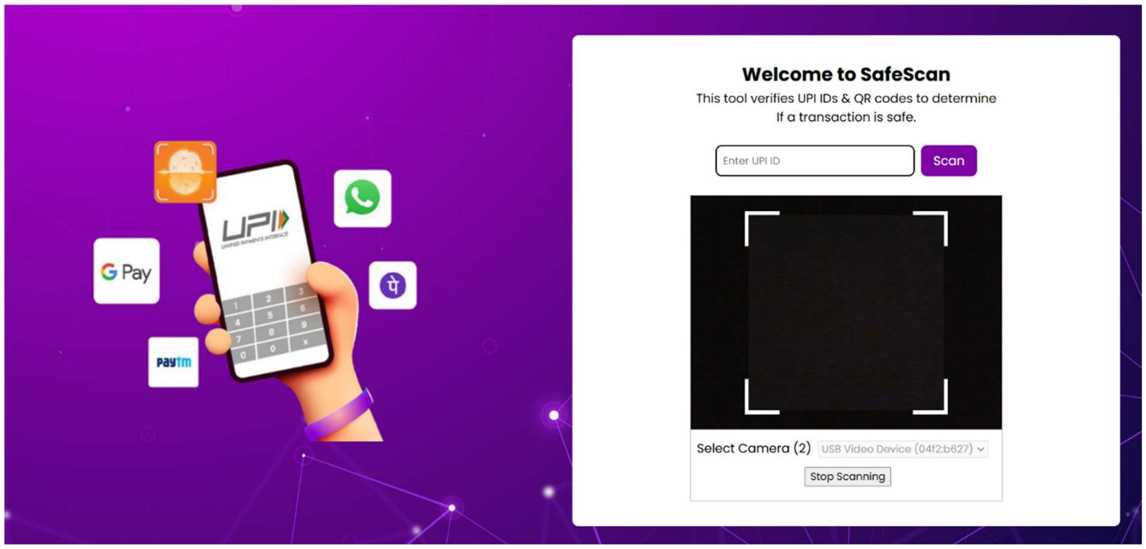
```
@media screen and (max-width: 767px) {
  body{
    background: url("../images/bg-color-mob.jpg");
    background-size: cover;
    background-repeat: no-repeat;
    height: 100vh;
  }

  .left-panel{
    display: none;
  }

  .right-panel{
    width: 75%;
    margin: 0;
  }

  .container{
    .qr-scanner{
      width: 100%;
      height: 100%;
    }
  }
}
```

5.3.2 Output Screens

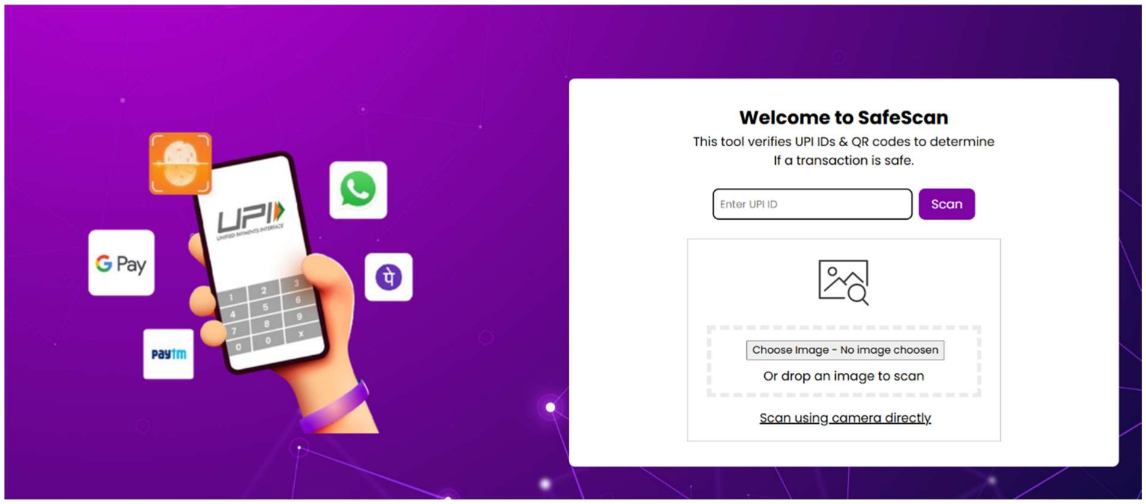


Welcome to SafeScan

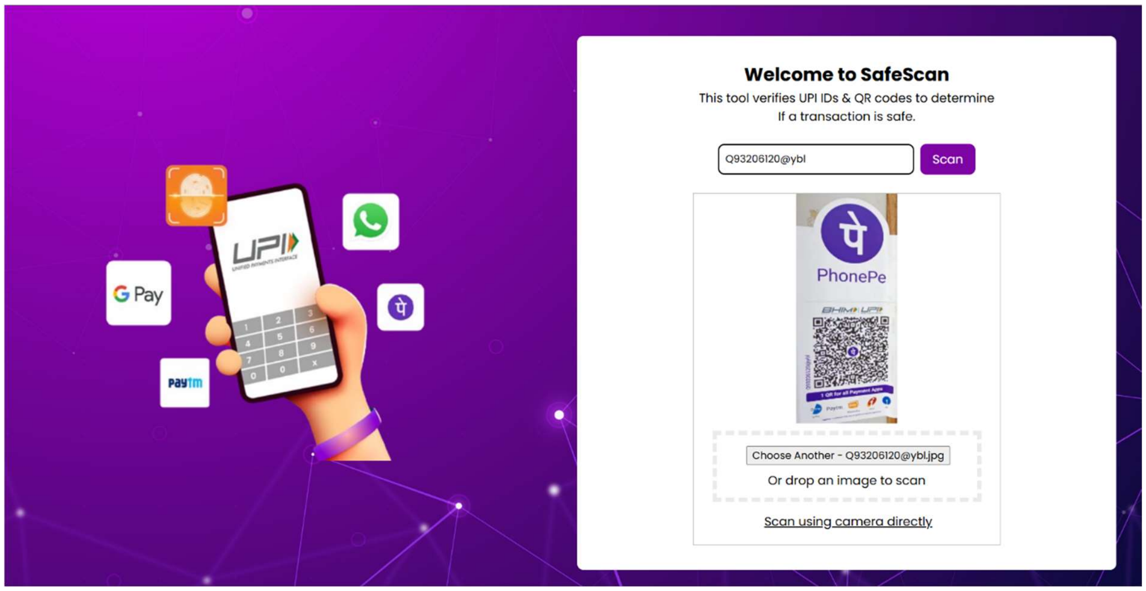
This tool verifies UPI IDs & QR codes to determine If a transaction is safe.



G Pay
Paytm
UPI
Welcome to SafeScan
This tool verifies UPI IDs & QR codes to determine
If a transaction is safe.
Enter UPI ID
Scan
Select Camera (2)
USB Video Device (04f2:b627)
Stop Scanning

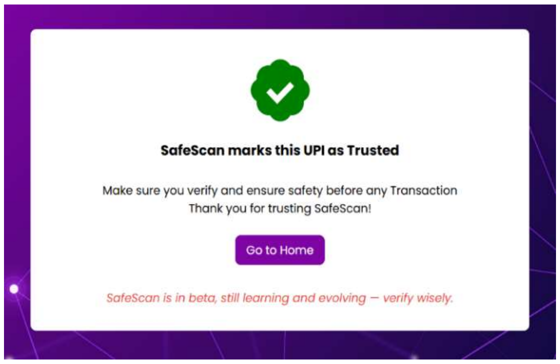


G Pay
Paytm
UPI
Welcome to SafeScan
This tool verifies UPI IDs & QR codes to determine
If a transaction is safe.
Enter UPI ID
Scan
Choose Image - No image choosen
Or drop an image to scan
Scan using camera directly



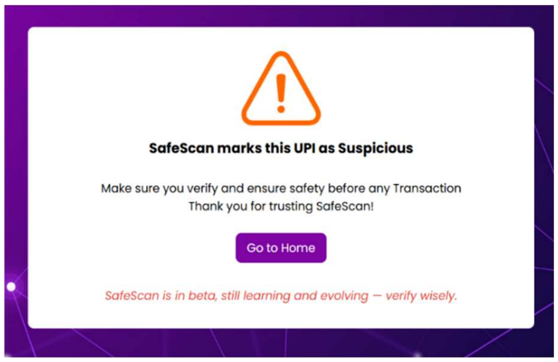
G Pay
Paytm
UPI
Welcome to SafeScan
This tool verifies UPI IDs & QR codes to determine
If a transaction is safe.
Q93206120@ybl
Scan
PhonePe
Choose Another - Q93206120@ybl.jpg
Or drop an image to scan
Scan using camera directly

5.3.3 Result Analysis


SafeScan marks this UPI as Trusted

Make sure you verify and ensure safety before any Transaction
Thank you for trusting SafeScan!

SafeScan is in beta, still learning and evolving — verify wisely.

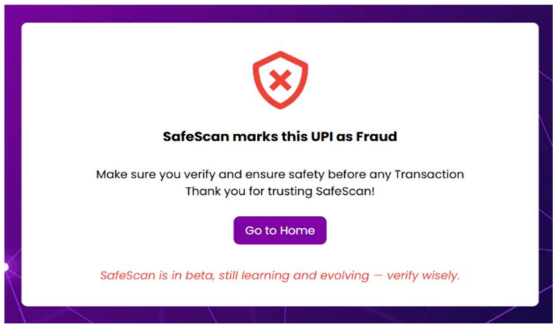

SafeScan marks this UPI as Suspicious

Make sure you verify and ensure safety before any Transaction
Thank you for trusting SafeScan!

SafeScan is in beta, still learning and evolving — verify wisely.

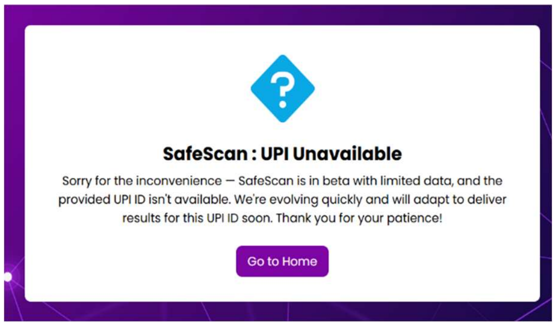
Fig 5.3.3.1 Trusted UPI

Fig 5.3.3.2 Suspicious UPI


SafeScan marks this UPI as Fraud

Make sure you verify and ensure safety before any Transaction
Thank you for trusting SafeScan!

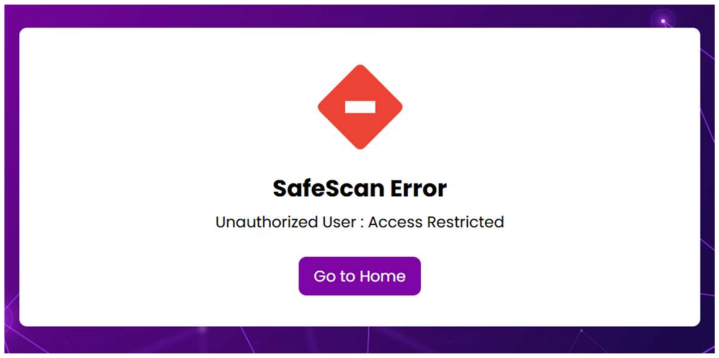
SafeScan is in beta, still learning and evolving — verify wisely.


SafeScan : UPI Unavailable

Sorry for the inconvenience — SafeScan is in beta with limited data, and the provided UPI ID isn't available. We're evolving quickly and will adapt to deliver results for this UPI ID soon. Thank you for your patience!

Fig 5.3.3.3 Fraud UPI

Fig 5.3.3.4 Unknown UPI


SafeScan Error

Unauthorized User : Access Restricted

Fig 5.3.3.5 Unauthorized User

CHAPTER-6

TESTING

6.1 INTRODUCTION

Testing is the final assessment of specification design and coding and is a crucial component of software quality assurance. Testing is the process of preparing test data, which is then utilised to test each module separately and subsequently the field validations provided. Next comes testing, which involves actively trying to make the system fail in order to ensure that all of its parts work together as a whole. The ideal way to do testing is to ask employee departments to help identify every scenario that might come up.

Every test ought to be scheduled well in advance. Testing ought to start "in the small" and work its way up to "in the large." Typically, the initial tests that are designed and carried out concentrate on specific programme components. Testing becomes more focused as it goes along in an effort to identify flaws in integrated cluster modules and, eventually, in the system as a whole. The goal of testing is to identify mistakes. A successful test is therefore one that has minimal errors. One of the two methods can be used to test any engineered product.

Unit Testing: Each core component of SafeScan, including UPI ID verification, transaction data processing, fraud detection algorithms, and alert mechanisms, undergoes unit testing to ensure individual functionality and identify any errors or bugs.

Integration Testing: Integration testing verifies seamless interaction between system components. It ensures that transaction data is correctly gathered, machine learning models analyze patterns effectively, alerts trigger when fraud is detected, and automated verification mechanisms function as expected.

System Testing: System testing evaluates the entire system's performance under real-world conditions. It includes scenarios like detecting fraudulent transactions across various data sources, validating alert mechanisms, and ensuring smooth integration between UPI verification, ML-based fraud detection, and alert triggers.

Acceptance Testing: Acceptance testing ensures SafeScan meets end-user expectations. Real-world testing validates whether the system accurately detects fraud, minimizes false positives, and provides timely alerts, ensuring reliability and usability for fraud prevention teams.

Performance Testing: Performance testing assesses system responsiveness, scalability, and reliability under varying transaction loads. It ensures that the fraud detection model efficiently processes large volumes of data, flags fraudulent transactions in real time, and performs well under high traffic.

Security Testing: Security testing safeguards SafeScan against potential vulnerabilities. It includes testing data encryption during transmission, authentication mechanisms for system access, protection against injection attacks, and ensuring compliance with data privacy standards to prevent unauthorized access.

Usability Testing: Usability testing evaluates the system's user interface, ease of navigation, and overall user experience. Feedback from end-users ensures that fraud reports are easily interpretable, alerts are actionable, and the system provides an intuitive platform for monitoring transactions and analyzing fraud patterns.

By implementing rigorous software testing, developers ensure SafeScan functions reliably and securely. Unit testing guarantees the accuracy of fraud detection algorithms, while integration testing ensures smooth data flow. Functional testing validates adherence to user requirements, enhancing the user experience. Simultaneously, security testing protects against vulnerabilities, maintaining high security standards. Performance testing evaluates system efficiency under real-time conditions, ensuring SafeScan operates with precision, speed, and reliability while delivering a seamless fraud detection experience.

6.2 DESIGN OF TESTCASES AND SCENARIOS

TEST CASE 1: Test Case for Database

Test objective: To verify if webpage shows error when database is not loaded.

Test description: Check if webpage is loaded with SafeScan Database. If not loaded, webpage must throw error when user submits UPI Id.

Requirements verified: yes

Test environment: Flask Application

Expected result: When database is not loaded, when user hits UPI Id then page should show SafeScan Error - Unauthorized / Access Restricted.

Actual result: Webpage shows SafeScan Error - Unauthorized / Access Restricted.

TEST CASE 2: Test Case for Unknown UPI Id

Test Objective: To validate if UPI Id is present in SafeScan Database.

Test Description: When a user provides a UPI ID, the application checks if that UPI ID exists in the SafeScan Database. If the UPI ID is not found in the database, the webpage should display a "Unknown UPI ID" message.

Test Environment: Flask Application.

Expected Result: Application should detect UPI Id in Database. If application does not detect it then page must show SafeScan - UPI Unavailable.

Actual Result: Application successfully detects that UPI Id is not present in Database and page shows SafeScan - UPI Unavailable.

TEST CASE 3: Test Case for Fraud Detection

Test Objective: Verify application's ability to classify UPI ID as Fraud, Suspicious, or Trusted.

Test Description: Upon receiving a UPI ID from a user, the application should search for it within the SafeScan Database. If found, the corresponding data should be sent to a machine learning model, which will then determine and return a grade for that UPI ID.

Test Environment: Flask Application, Machine Learning.

Expected Result: The webpage displays the correct classification (Fraud, Suspicious, or Trusted) for the provided UPI ID.

Actual Result: The webpage successfully displays the correct classification (Fraudulent, Suspicious, or Trusted) for the provided UPI ID.

CHAPTER-7
CONCLUSION

7.1 CONCLUSION

SafeScan represents a transformative approach to fraud detection in digital payments by harnessing the power of machine learning to provide real-time analysis and proactive security measures. Unlike conventional rule-based detection systems, SafeScan continuously learns and evolves to counteract emerging fraud tactics, ensuring long-term effectiveness. By integrating QR code verification, UPI ID validation, and AI-driven anomaly detection, it creates a multi-layered security framework that significantly reduces financial risks for users and institutions. The ability to detect suspicious transactions in real-time and notify users instantly enhances transparency and trust in digital payments. Moreover, the scalability of SafeScan ensures that it can be implemented across various payment platforms, making digital transactions safer, more secure, and resilient against future fraud threats. Ultimately, SafeScan is poised to redefine fraud prevention in digital payments, paving the way for a more secure and reliable financial ecosystem.

7.2 FUTURE SCOPE

The future scope of SafeScan is vast, with significant improvements possible through advancements in technology and evolving fraud detection strategies. As fraud techniques become more sophisticated, our system will integrate enhanced security mechanisms and machine learning models to detect fraudulent activities more effectively. By leveraging AI-driven anomaly detection and behavioural analysis, SafeScan will proactively identify unusual transaction patterns and adapt to emerging fraud trends in real-time. Automation will play a crucial role in reducing human intervention and improving efficiency. Currently, the initial verification of UPI IDs involves manual effort, but future iterations will introduce automated validation, significantly reducing processing time and human workload. Additionally, our system will evolve to process transaction statements in real-time, using AI to flag suspicious activities instantly and make fraud detection more responsive. To support large-scale deployment, SafeScan will focus on improving its speed and scalability. Future developments will optimize machine learning algorithms for high-speed fraud detection, ensuring seamless analysis of vast amounts of financial data. By leveraging cloud-based infrastructure, the system will be capable of handling city-wide or even nationwide implementations. Furthermore, predictive analytics will be integrated to anticipate potential fraud patterns and mitigate risks before they occur. By continuously evolving with new security enhancements, automation, and high-performance computing, SafeScan aims to revolutionize digital fraud detection, making financial transactions safer, faster, and more secure.

7.3 REFERENCES:

1. A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective - Samaneh Sorournejad, Zojah, Atani et.al - November 2016.

2. Ruttala Sailusha ; V. Gnaneswar ; R. Ramesh ; G. Ramakoteswara Rao ,"UPI Fraud Detection Using Machine Learning ",2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) ,19 June 2020.

3. Solving the False positives problem in fraud prediction using automated feature engineering - Wedge, Canter, Rubio et.al - October 2017.

4. PayPal Inc. Quarterly results - https://www.paypal.com/stories/us/paypalreports-third-quarter-2018-results.

5. A Model for Rule Based Fraud Detection in Telecommunications - Rajani, Padmavathamma - IJERT - 2012.

6. Pratyush Sharma, Souradeep Banerjee, Devyanshi Tiwari, and Jagdish Chandra Patni, "ML Model for UPI Fraud Detection - A Comparative Analysis", The International Arab Journal of Information Technology, Vol. 18, No. 6, November 2021 , pp 789-790.

7. Scikit learn - machine learning library - http://scikit-learn.org.

8. Dataset For Fraud Detection - https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset.