UPI FRAUD DETECTION USING DEEP LEARNING

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science and Engineering
(Artificial Intelligence and Machine Learning)

Submitted by

Renangi Sai Mohith	**:**	**2211CS020442**
Rikkala Ramya Reddy	**:**	**2211CS020444**
Rollakanti Harshith Reddy	**:**	**2211CS020446**
S.Siva Sathvik	**:**	**2211CS020450**
Syed Abdul Rahman	**:**	**2211CS020492**

Under the Guidance of

A.Kalyani
Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the Bonafide record of the Application Development entitled, "**UPI FRAUD DETECTION USING DEEP LEARNING''** Submitted by R. Sai Mohith(2211CS020442), R. Ramya Reddy(2211CS020444), R. Harshith Reddy(2211CS020446), S. Sathvik (2211CS020450), Syed Abdul Rahman(2211CS020492) B. Tech III-year I semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Prof.A.Kalyani

Dr. Thayyaba Khatoon
DEAN CSE(AI&ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

My Major Project would not have been successful without the help of several people. I would like to thank the personalities who were part of my major project in numerous ways, those who gave us outstanding support from the birth of the project.

I am extremely thankful to our honourable Pro-Vice Chancellor, Dr. VSK Reddy sir for providing necessary infrastructure and resources for the accomplishment of my project.

I highly indebted to Prof. A. Kalyani mam Guide, School of Engineering, for his support during the tenure of the project.

I am very much obliged to our beloved Prof. Dr. Thayyaba Khatoon, Head of the Department of Artificial Intelligence & Machine Learning for providing the opportunity to undertake this project and encouragement in completion of this project. I hereby wish to express our deep sense of gratitude to Prof. A. Kalyani mam for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project.

I am also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making our project a success. We would like to thank all our friends who extended their help, encouragement and moral support either directly or indirectly in our project work.

SINCERELY,

R.Sai Mohith: 2211CS020442

R.Ramya Reddy: 2211CS020444

R.Harshith Reddy: 2211CS020446

S.Siva Sathvik: 2211CS020450

Syed Abdul Raheman: 2211CS020492

ABSTRACT

The rapid increase in UPI-based digital payments has introduced new security challenges, especially with the rise in fraudulent transactions that carry significant financial risks. This research explores how deep learning models, particularly Long Short-Term Memory (LSTM) and Autoencoders, can be effectively applied to detect fraud in UPI transactions in real time. LSTM models are used to capture the sequential nature of transaction data, helping to identify patterns that may indicate fraudulent behavior. Meanwhile, Autoencoders focus on learning a compressed profile of normal transaction patterns, allowing for the detection of anomalies based on reconstruction errors when fraud is present. By leveraging the strengths of both LSTM and Autoencoders, this study presents a robust fraud detection framework. Experimental results confirm the effectiveness of this approach, which not only enhances the security of UPI transactions but also minimizes potential financial losses

Contents

CHAPTER 1
INTRODUCTION

1.1 PROBLEM DEFINITION

The rapid adoption of the Unified Payments Interface (UPI) in India has transformed digital payments by enabling seamless, instant, and secure transactions. However, this surge in usage has also led to an increase in fraudulent activities, posing significant risks to users and financial institutions. The main challenges include a rising incidence of fraud such as phishing, account takeover, and unauthorized transactions, which result in financial losses and diminished trust in digital payment systems. Additionally, fraudsters continuously evolve their tactics, complicatingthe detection of emerging fraudulent behavior. The high volume of transactions processed through UPI further complicates effective monitoring, as existing systems often generate a significant number of false positives, flagging legitimate transactions as fraudulent and impacting user experience. The objective of this project is to develop a UPI fraud detection app that leverages deep learning techniques to accurately identify and prevent fraudulent transactions. The app aims to utilize historical transaction data to train machine learning modelsthat recognize fraud patterns, implement real-time monitoring for immediate feedback on suspicious activities, and minimize false positives to ensure smooth processing of legitimate transactions. By addressing these challenges, the UPI fraud detection app will contribute to a safer digital payment ecosystem, protecting users and fostering confidence in UPI for everyday transactions.

1.2 PROJECT OBJECTIVE

The objective of this project is to develop a UPI fraud detection app that effectively utilizes deep learning techniques to identify and prevent fraudulent transactions within the UPI ecosystem. By harnessing historical transaction data, the app aims to train advanced machine learning models capable of recognizing patterns indicative of fraudulent behavior. Additionally, it will implement real-time transaction monitoring, providing immediate feedback on potentially suspicious activities to enhance user security. A key focus will be on minimizing false positives, ensuring that legitimate transactions are processed smoothly without unnecessary interruptions. Ultimately, this project seeks to create a robust fraud detection system that enhances user trust and confidence in UPI, contributing to a safer and more secure digital payment environment.

1.3 SCOPE OF THE PROJECT

The scope of the UPI fraud detection app project encompasses the collection and preprocessing of historical transaction data from UPI platforms, including both legitimate and fraudulent transactions, to ensure quality and relevance for training machine learning models. The project will focus on developing deep learning algorithms capable of effectivelydetecting fraud patterns, with an emphasis on real-time transaction monitoring to provide immediate alerts on suspicious activities. A user-friendly interface will be designed to facilitate interaction, featuring transaction alerts, fraud notifications, and customizable user settings. Performance evaluation will be conducted using metrics such as accuracy, precision, recall, and F1 score to optimize the detection capabilities of the models.

Additionally, the app will include educational resources to inform users about common fraudtactics and support mechanisms to assist with navigation and response to alerts. Finally, the project will adhere to regulatory compliance concerning data privacy and security, ensuring a safe and trustworthy digital payment environment while enhancing user confidence in UPItransactions.

1.4 LITERATURE SURVEY

The rapid growth of digital payment systems in India, particularly through the Unified Payments Interface (UPI), has led to increased concerns about fraud, prompting various studies and applications aimed at enhancing security. Many researchers have focused on machine learning algorithms for fraud detection in UPI transactions. For instance, Gupta et al. (2021) proposed a model utilizing decision trees and random forests, achieving high accuracy in detecting fraudulent transactions by emphasizing feature selection to identify key indicators of fraud. In addition, deep learning methods have gained traction; Sharma and Singh (2022) implemented recurrent neural networks (RNNs) to analyze transaction sequences and identify unusual patterns, capturing temporal dependencies in transaction data and leading to improved detection rates compared to traditional techniques. Hybrid models that combine multiple methodologies have also been explored, such as the framework developed by Patel et al. (2023), which integrates machine learning algorithms with rule- based systems, effectively reducing false positives while maintaining high detection rates.

Moreover, the necessity for real-time fraud detection is underscored by Kumar et al. (2020), who designed a system that analyzes transactions as they occur, successfully flagging suspicious activities through a streaming data processing framework, thereby minimizing potential losses for users. Beyond technical solutions, user awareness plays a crucial role in fraud prevention. Reddy et al. (2021) emphasized the importance of educating users about common fraud schemes and security practices, suggesting that informed users are less likely tofall victim to fraud. Furthermore, Joshi and Verma (2022) discussed the regulatory landscape surrounding digital payments and the ethical implications of data usage in fraud detection, arguing that systems must balance security with user privacy, ensuring compliance with data protection regulations. In conclusion, the literature on UPI fraud detection applications reveals a multifaceted approach to addressing fraud in digital payments. With advancements in machine learning and deep learning, combined with user education and regulatory compliance, there is significant potential to develop effective and trustworthy UPI fraud detection solutions. Future work should continue to explore innovative techniques and frameworks that enhance security while prioritizing user privacy and experience.

CHAPTER 2
ANALYSIS

2.1 PROJECT PLANNING AND RESEARCH

Project planning and research for the UPI fraud detection app will adopt a structured approach to ensure effective development and implementation. The initial phase will focus on a comprehensive literature review to understand existing methodologies, identify gaps in current solutions, and assess user needs. Following this, a detailed project timeline will be established, outlining key milestones such as data collection, model development, and user testing. The project will begin with the collection and preprocessing of historical UPI transaction data, including both legitimate and fraudulent transactions, to create a robust dataset for training machine learning models. Various machine learning and deep learning algorithms will be explored to determine the most effective techniques for detecting fraud, while a user-friendly interface will be developed to facilitate interaction, provide alerts, and access educational resources on fraud prevention. Real-time transaction monitoring capabilities will be integrated to ensure immediate detection of suspicious activities, and iterative testing will refine the models' performance based on metrics such as accuracy and precision. Additionally, user education will be emphasized to inform users about common fraud tactics and secure practices, enhancing awareness and engagement. Throughout the project, regulatory compliance and data privacy considerations will be integrated to ensure adherence to relevant laws and standards. Ultimately,this structured planning and research approach aims to deliver a reliable and efficient UPI fraud detection solution that enhances security in digital payments.

2.2 SOFTWARE REQUIREMENT SPECIFICATION

2.2.1 SOFTWARE REQUIREMENT

1. Operating System: Windows 7 or Later
2. Visual Studio Code
3. Python >= 3.8

   TensorFlow, Keras, or PyTorch.
4. MySQL
5. Git for source code management

2.2.2 HARDWARE REQUIREMENT

a. Intel i5/i7 or AMD Ryzen Processor

b. 8GB RAM

c. 512GB SSD

2.3 MODEL SELECTION AND ARCHITECTURE

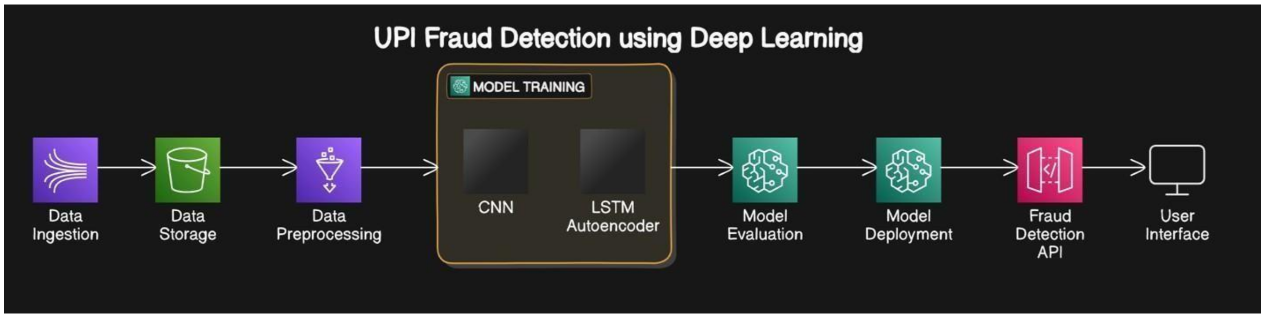


Fig 1: Model Selection

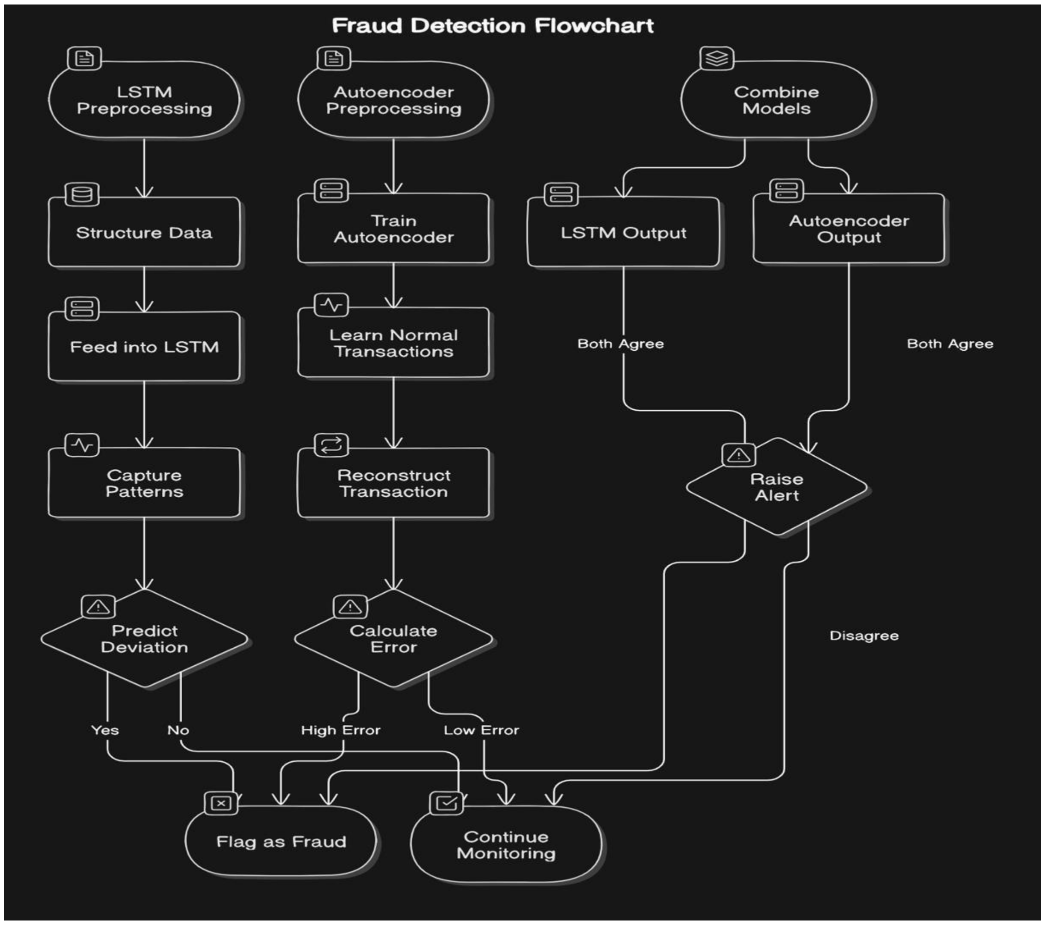


Fig 2: Model Architecture

CHAPTER 3
DESIGN

3.1 INTRODUCTION

The design phase of the UPI fraud detection app is essential for translating theoretical concepts and research into a functional, user-friendly application. This phase will focus on creating an intuitive user interface (UI) and a robust backend architecture that supports core functionalities such as real-time fraud detection and user interaction. It will begin with wireframing and prototyping to visualize the app's layout, navigation, and key features, ensuring a seamless user experience (UX). The design will prioritize usability, allowing easy access to transaction alerts, educational resources, and account management settings, while also emphasizing aesthetics to foster trust and engagement. Additionally, the backend architecture will involve selecting appropriate technologies for data processing, machine learning model integration, and database management, ensuring efficient data handling and rapid transaction monitoring. Security will be a primary consideration in both UI and backend design, incorporating measures to protect user data and comply with regulatory standards. Ultimately, this meticulous design phase aims to deliver a reliable and effective UPI fraud detection app that enhances security in digital payment

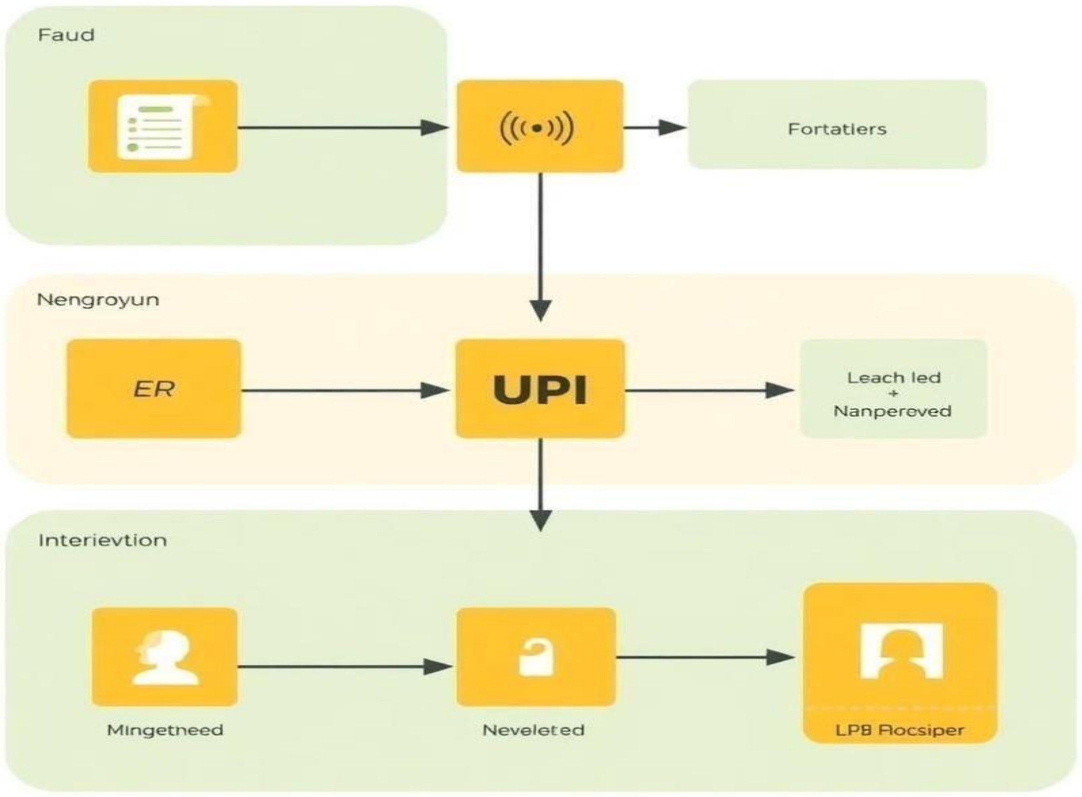


Fig 3: Design

3.2 DATA SET DESCRIPTIONS

The dataset for the UPI fraud detection app will comprise historical transaction records from various UPI platforms, encompassing both legitimate and fraudulent transactions to create a balanced and comprehensive dataset for training machine learning models. Each record will typically include features such as transaction ID, timestamp, sender and receiver details, transaction amount, payment method, location, and device information, along with labels indicating whether the transaction was legitimate or fraudulent. The dataset may also incorporate additional contextual data, such as transaction frequency and user behavior patterns, to enhance the model's ability to detect anomalies. Ensuring the quality and diversity of this dataset is crucial, as it will serve as the foundation for developing accurate fraud detection algorithms capable of identifying subtle patterns and trends indicative of fraudulent activities in real-time.

3.3 DATA PREPROCESSING TECHNIQUES

Data preprocessing is a critical step in preparing the dataset for the UPI fraud detection app, as it enhances the quality of the data and improves the performance of machine learning models. Key preprocessing techniques include:

1. **Data Cleaning:** This involves identifying and removing duplicates, correcting errors, and handling missing values in the dataset. Techniques such as imputation (replacing missing values with the mean, median, or mode) or deletion of rows with significant missing data can be employed.

2. **Data Transformation:** Transforming data to a suitable format is essential for analysis. This may involve normalizing or standardizing numerical features to bring them onto a similar scale, which helps improve model performance and convergence during training.

3. **Feature Engineering**: Creating new features from existing data can enhance model accuracy. For instance, generating features like transaction frequency, average transaction amount, or time since the last transaction can provide valuable insights for fraud detection.

4. **Encoding Categorical Variables:** Categorical features, such as payment method or user location, need to be converted into numerical formats. Techniques like one-hot encoding or label encoding can be utilized to facilitate model training.

5. **Outlier Detection and Treatment:** Identifying and handling outliers is crucial, as they can skew results and negatively impact model training. Statistical methods such as the Z-score or IQR (Interquartile Range) can help detect outliers, which can then be removed or transformed.

6. **Balancing the Dataset:** Since fraudulent transactions are often rare compared to legitimate ones, techniques like oversampling (e.g., SMOTE - Synthetic Minority Over-sampling Technique) or under sampling can be used to balance the dataset, ensuring that the model learnsto recognize both classes effectively.

7. **Splitting the Dataset:** Finally, the dataset will be divided into training, validation, and test sets to ensure robust model evaluation. A common approach is to use an 80-10-10 split, where 80% is used for training, 10% for validation, and 10% for testing.

By applying these preprocessing techniques, the dataset will be refined, enabling the development of accurate and efficient fraud detection models that can effectively identify suspicious activities inreal-time.

3.3 METHODS AND ALGORITHMS

The development of the UPI fraud detection app will utilize various methods and algorithms to effectivelyidentify fraudulent transactions. The following are key approaches:

1. Machine Learning Algorithms:

Logistic Regression: A statistical model that predicts the probability of a transaction being fraudulent based on input features. It serves as a baseline model for comparison.

Decision Trees: A model that splits the data into branches based on feature values, making decisions at each node, which helps in interpreting the results and understanding the importance of features.

Random Forest: An ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting. It works well for high-dimensional datasets, making it suitable for transaction data.Support Vector Machines (SVM): A supervised learning algorithm that finds the optimal hyperplane to separate classes, effective for high-dimensional spaces and useful when there areclear margins of separation between classes.

2. Deep Learning Algorithms:

Artificial Neural Networks (ANN): Composed of layers of interconnected nodes (neurons), ANNs can capture complex patterns in data, making them suitable for detecting fraud in intricate transaction datasets.

Recurrent Neural Networks (RNN): Particularly useful for sequence data, RNNs can analyze

transactionsover time, identifying unusual patterns in user behavior that may indicate fraud.

Long Short-Term Memory (LSTM): A type of RNN that effectively captures long-range dependencies

In data, making it ideal for analyzing temporal sequences of transactions.

3. Anomaly Detection Techniques:

Isolation Forest: An algorithm specifically designed for anomaly detection that isolates anomalies instead of profiling normal data points, making it effective for identifying fraudulent transactions.

One-Class SVM: A variation of the SVM that is trained only on normal data, aiming to distinguish outliers, which in this context, are fraudulent transactions.

4. Ensemble Methods:

Gradient Boosting Machines (GBM): Combines multiple weak learners to create a strong predictive model. It iteratively improves the model by focusing on errors from previous iterations, leading to high accuracy in fraud detection.

XGBoost: An optimized version of GBM that is faster and more efficient, widely used in machine learning competitions due to its performance and scalability.

CHAPTER 4
DEPLOYMENT AND RESULTS:

4.1 INTRODUCTION

The deployment phase of the UPI fraud detection app is critical for transitioning the application from a development environment to a live production setting where it can be used by end-users. This process involves several key steps, including finalizing the application code, ensuring scalability, and implementing robust security measures to protect sensitive user data. The deployment will be carried out on a cloud platform, enabling flexibility and the ability to scale resources based on user demand. Continuous integration and continuous deployment (CI/CD) practices will be adopted to facilitate seamless updates and maintenance, ensuring the app remains up to date with the latest security patches and features.

Once deployed, the app will undergo thorough monitoring to evaluate its performance and effectiveness in detecting fraudulent transactions in real-time. Key performance metrics, such as accuracy, precision, recall, and F1 score, will be analyzed to assess the model's success and identify areas for improvement. User feedback will also be collected to understand their experiences and gather insights on potential enhancements. The results of the app's performance will be compared against established benchmarks from the training and validation phases to validate its efficacy in a real-world setting. By focusing on both deployment and results, the UPIfraud detection app aims to provide a reliable and secure solution that enhances user trust in digital payment systems while continuously evolving to meet the changing landscape of fraud threats.

4.2 Source Code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>PAY</title>
   <script src="https://cdn.tailwindcss.com"></script>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
```

```
awesome/5.15.3/css/all.min.css">
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">

    <style>

        body {

            font-family: 'Roboto', sans-serif;

        }

        #transaction-history
            {display: none;
        }

        #add-money-modal, #to-number-modal, #fraud-detect-modal
            {display: none;
            position: fixed;top:
            0;
            left: 0;

            width: 100%;

            height: 100%;

            background: rgba(0, 0, 0, 0.5);
            justify-content: center;
            align-items: center;

        }

        #add-money-modal .modal-content, #to-number-modal .modal-content, #fraud-detect-
modal .modal-content {

            background: white;
            padding: 20px;
            border-radius: 8px;
            width: 300px;
        }
        #success-message, #fraud-message
            {display: none;
            position: fixed;
            bottom: 20px; left:
            50%;
            transform: translateX(-50%);
            background: green;
            color: white; padding:
            10px 20px;border-
            radius: 8px;
        }

        #fraud-message
            { background: red;
        }

        #balance-display
            {display: none;
        }

        #success-icon, #fraud-icon
            {display: none;
            position: fixed;top:
            50%;
            left: 50%;

            transform: translate(-50%, -50%);
            font-size: 50px;
        }

        #success-icon
            {color: green;
        }

        #fraud-icon
            {color: red;
        }

    </style>

    <script>

        let balance = 0;

        let phoneUsageCount = {}; // Object to track phone number usage



        function toggleHistory() {
            const historySection = document.getElementById('transaction-history');
            historySection.style.display = historySection.style.display === 'none' ||
historySection.style.display === '' ? 'block' : 'none';

        }



        function showAddMoneyModal() {

            document.getElementById('add-money-modal').style.display = 'flex';

        }



        function hideAddMoneyModal() {

            document.getElementById('add-money-modal').style.display = 'none';

        }
```

```
function showToNumberModal() {

    document.getElementById('to-number-modal').style.display = 'flex';

}
function hideToNumberModal() {

    document.getElementById('to-number-modal').style.display = 'none';

}



function showSuccessMessage(message) {

    const successMessage = document.getElementById('success-message');
    successMessage.textContent = message;
    successMessage.style.display = 'block';

    const successIcon = document.getElementById('success-icon');
    successIcon.style.display = 'block';
    setTimeout(() =>
        { successMessage.style.display = 'none';
        successIcon.style.display = 'none';
    }, 3000);

}



function showFraudMessage(message) {

    const fraudMessage = document.getElementById('fraud-message');
    fraudMessage.textContent = message;
    fraudMessage.style.display = 'block';

    const fraudIcon = document.getElementById('fraud-icon');
    fraudIcon.style.display = 'block';
    setTimeout(() =>
        { fraudMessage.style.display = 'none';
        fraudIcon.style.display = 'none';
    }, 3000);

}


function addMoney() {

    const amount = document.getElementById('money-amount').value;if
    (amount && ! isNaN(amount)) {
        if (parseFloat(amount) > 40000) {

            showFraudMessage("Fraud detected! Amount exceeds $40000.");
```

```
                    return;
200 ';}

balance += parseFloat(amount);
updateBalanceDisplay();
const historyList = document.getElementById('history-list');const
newItem = document.createElement('li');
newItem.className = 'flex justify-between items-center py-2 border-b border-gray-


newItem.innerHTML = `<span class="text-gray-700">Added to Account</span><span
class="text-green-500 font-medium">+ $${amount}</span>`;
                historyList.appendChild(newItem); showSuccessMessage(`$${amount}
                has been added to your account.`);hideAddMoneyModal();
            } else {

                alert("Invalid amount entered.");

            }

        }



        function showBalance() {

            const balanceDisplay = document.getElementById('balance-display');
            balanceDisplay.textContent = `Current Balance: $${balance.toFixed(2)}`;
            balanceDisplay.style.display = 'block';
            if (balance > 20000) {

                showFraudMessage("Fraud detected! Balance exceeds $20000.");

            }

        }



        function updateBalanceDisplay() {

            const balanceDisplay = document.getElementById('balance-display');
            balanceDisplay.textContent = `Current Balance: $${balance.toFixed(2)}`;if
            (balance > 20000) {
                showFraudMessage("Fraud detected! Balance exceeds $20000.");

            }

        }



        function payToNumber() {

            const number = document.getElementById('phone-number').value;
            const amount = document.getElementById('pay-amount').value;
            if (number && amount && !isNaN(amount)) {if
                (parseFloat(amount) > 10000) {
                    showFraudMessage("Fraud detected! Amount exceeds $10000.");
                    return;
                }

                if (parseFloat(amount) > balance)
                    {alert("Insufficient balance.");
                    return;
                }

                if (phoneUsageCount[number]) {phoneUsageCount[number]++;
                } else {

                    phoneUsageCount[number] = 1;


    times.");}

    if (phoneUsageCount[number] > 3) {

        showFraudMessage("Fraud detected! Phone number has been used more than 3

  return;
    200 ';}

    balance -= parseFloat(amount);
    updateBalanceDisplay();
    const historyList = document.getElementById('history-list');const
    newItem = document.createElement('li');
    newItem.className = 'flex justify-between items-center py-2 border-b border-gray-


    newItem.innerHTML = `<span class="text-gray-700">Paid to
    ${number}</span><span class="text-red-500 font-medium">- $${amount}</span>`;
                historyList.appendChild(newItem);
                showSuccessMessage(`$${amount} has been paid to ${number}.`);
                hideToNumberModal();
            } else {

                alert("Invalid number or amount entered.");

            }

        }


        function checkAmount() {

            const amount = document.getElementById('pay-amount').value;if
            (parseFloat(amount) > 10000) {
                showFraudMessage("Fraud detected! Amount exceeds $10000.");
            }
        }
    </script>
</head>
<body class="bg-gray-100">
    <div class="container mx-auto p-4">
        <header class="bg-white shadow-md rounded-lg p-4 mb-6">
            <h1 class="text-2xl font-bold text-gray-800">PAY</h1>
        </header>


        <section class="grid grid-cols-2 gap-4 mb-6">
            <div class="bg-white shadow-md rounded-lg p-4 flex items-center justify-center flex-
col">
<i class="fas fa-qrcode text-4xl text-blue-500 mb-2"></i>
    <span class="text-lg font-medium text-gray-700">Scanner</span>
</div>
<div class="bg-white shadow-md rounded-lg p-4 flex items-center justify-center flex-col
cursor-pointer" onclick="showBalance()">
                <i class="fas fa-wallet text-4xl text-green-500 mb-2"></i>
                <span class="text-lg font-medium text-gray-700">Check Balance</span>
            </div>
            <div class="bg-white shadow-md rounded-lg p-4 flex items-center justify-center flex-col
cursor-pointer" onclick="showToNumberModal()">
                <i class="fas fa-phone-alt text-4xl text-purple-500 mb-2"></i>
                <span class="text-lg font-medium text-gray-700">To Number</span>
            </div>
            <div class="bg-white shadow-md rounded-lg p-4 flex items-center justify-center flex-col
cursor-pointer" onclick="toggleHistory()">
                <i class="fas fa-history text-4xl text-red-500 mb-2"></i>
                <span class=" text-lg font-medium text-gray-700">History</span>
            </div>
            <div class="bg-white shadow-md rounded-lg p-4 flex items-center justify-center flex-col
```

```
cursor-pointer" onclick="showAddMoneyModal()">
                <i class="fas fa-plus-circle text-4xl text-yellow-500 mb-2"></i>
                <span class="text-lg font-medium text-gray-700">Add Money</span>
            </div>
        </section>


        <section id="transaction-history" class="bg-white shadow-md rounded-lg p-4 mb-6">
            <h2 class="text-lg font-bold text-gray-800 mb-2">Transaction History</h2>
            <ul id="history-list" class="list-none mb-0"></ul>
        </section>


        <section id="add-money-modal" class="flex justify-center items-center">
            <div class="modal-content">
                <h2 class="text-lg font-bold text-gray-800 mb-2">Add Money</h2>
                <input id="money-amount" type="number" class="w-full p-2 pl-10 text-sm text-gray-
700" placeholder="Enter amount">
                <button class="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4
rounded" onclick="addMoney()">Add</button>
                <button class="bg-gray-500 hover:bg-gray-700 text-white font-bold py-2 px-4
rounded" onclick="hideAddMoneyModal()">Cancel</button>
            </div>
        </section>
        <section id="to-number-modal" class="flex justify-center items-center">
            <div class="modal-content">
                <h2 class="text-lg font-bold text-gray-800 mb-2">Pay to Number</h2>
                <input id="phone-number" type="tel" class="w-full p-2 pl-10 text-sm text-gray-700"
placeholder="Enter phone number">
                <input id="pay-amount" type="number" class="w-full p-2 pl-10 text-sm text-gray-
700" placeholder="Enter amount">
                <button class="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4
rounded" onclick="payToNumber()">Pay</button>
                <button class="bg-gray-500 hover:bg-gray-700 text-white font-bold py-2 px-4
rounded" onclick="hideToNumberModal()">Cancel</button>
            </div>
        </section>
        <section id="fraud-detect-modal" class="flex justify-center items-center">

            <div class="modal-content">

                <h2 class="text-lg font-bold text-gray-800 mb-2">Fraud Detection</h2>

                <p id="fraud-message" class="text-lg text-red-500 font-bold mb-2"></p>

            </div>

        </section>
        <section id="balance-display" class="bg-white shadow-md rounded-lg p-4 mb-6">

            <h2 class="text-lg font-bold text-gray-800 mb-2">Current Balance</h2>

            <p id="balance-text" class="text-lg text-gray-700 font-bold mb-2"></p>

        </section>
        <i id="success-icon" class="fas fa-check-circle text-green-500"></i>

        <i id="fraud-icon" class="fas fa-exclamation-triangle text-red-500"></i>

        <p id="success-message" class="text-lg text-green-500 font-bold mb-2"></p>
        <p id="fraud-message" class="text-lg text-red-500 font-bold mb-2"></p>

         </div>

    </body>

</html>
```

app.py

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.models import Model, Sequential
from keras.layers import Input, Dense, LSTM, Dropout
from flask import Flask, request, jsonify

data = pd.read_csv('upi_transaction_data.csv')
data.drop(['timestamp', 'user_id'], axis=1, inplace=True)

data.fillna(method='ffill', inplace=True)

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

X_train, X_test = train_test_split(scaled_data, test_size=0.2, random_state=42)
input_dim = X_train.shape[1] input_layer =
Input(shape=(input_dim,))
encoder = Dense(32, activation='relu')(input_layer)
encoder = Dense(16, activation='relu')(encoder)
decoder = Dense(32, activation='relu')(encoder)
```

```
  decoder = Dense(input_dim, activation='sigmoid')(decoder)
  autoencoder = Model(inputs=input_layer, outputs=decoder)
  autoencoder.compile(optimizer='adam', loss='mean_squared_error')

autoencoder.fit(X_train, X_train, epochs=50, batch_size=32, validation_split=0.2)
  reconstructed_data = autoencoder.predict(X_test)
  reconstruction_error = np.mean(np.square(X_test - reconstructed_data), axis=1)
  threshold = np.percentile(reconstruction_error, 95)  # 95th percentile
  anomalies = reconstruction_error > threshold

  X_train_lstm = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
  X_test_lstm = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
  model = Sequential()
  model.add(LSTM(50, return_sequences=True, input_shape=(X_train_lstm.shape[1],
  X_train_lstm.shape[2])))
  model.add(Dropout(0.2))
  model.add(LSTM(50))
  model.add(Dropout(0.2)) model.add(Dense(1,
  activation='sigmoid'))


  model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


  # Create labels for training (1 for fraud, 0 for normal) y_train
  = np.array([1 if fraud else 0 for fraud in anomalies])
  y_train = y_train[:X_train_lstm.shape[0]] # Ensure the labels match the training data size

model.fit(X_train_lstm, y_train, epochs=50, batch_size=32, validation_split=0.2)

y_pred = model.predict(X_test_lstm)
y_pred = (y_pred > 0.5).astype(int)

from sklearn.metrics import classification_report
print(classification_report(anomalies, y_pred))
  app = Flask(__name__)
  @app.route('/predict', methods=['POST'])def
  predict():
      data = request.get_json()
      X_new = scaler.transform(data)
      X_new = X_new.reshape((1, 1, X_new.shape[1]))


      # Predict using the LSTM model
      y_pred = model.predict(X_new)
      return jsonify({'fraud_score': float(y_pred[0][0])})


  if__name__== '_main_':
      app.run(debug=True)
```

4.3 MODEL IMPLEMENTATION AND TRAINING

Certainly! Here's a breakdown of the **model implementation and training process** for theUPI fraud detection app, outlined in detailed steps:

1. Data Preparation:

Data Collection: Gather a large dataset of UPI transactions, ideally containing both legitimateand fraudulent transactions. This dataset may include attributes such as transaction ID, sender and receiver information, transaction amount, timestamp, location, device information, and anylabels indicating fraud.

Data Preprocessing:

Cleaning: Remove duplicates and handle missing values by imputation or deletion.

Transformation: Normalize or standardize numerical features to bring them to a common scale.

Feature Engineering: Create additional features that may help in detecting fraud, like transactionfrequency, average transaction amount, time since the last transaction, etc.

Encoding Categorical Variables: Convert categorical features (like payment method or device type) to numerical values using techniques like one-hot encoding or label encoding.

Balancing the Dataset: Use techniques like SMOTE (Synthetic Minority Over-sampling Technique) to handle class imbalance, ensuring there are enough fraudulent samples for effective training.

2. Model Selection:

Algorithm Choice: Choose the appropriate model based on data analysis and requirements.

Machine Learning Algorithms: Options include logistic regression, decision trees, random forest, gradient boosting, or support vector machines (SVM), which can handle tabular transaction data effectively.

Deep Learning Algorithms: Use neural networks such as artificial neural networks (ANNs) or recurrent neural networks (RNNs), especially LSTMs (Long Short-Term Memory networks), to capture temporal patterns in transaction data.

3. Model Implementation:

Programming Language: Typically implemented in Python due to its rich ecosystem of libraries for machine learning and deep learning.

Libraries: Use libraries like TensorFlow or Keras for deep learning models, and scikit-learn for traditional machine learning models.

Define Model Architecture:

For machine learning models, use scikit-learn to easily implement algorithms like decision trees, random forest, or logistic regression.

For deep learning models, define the architecture in TensorFlow or Keras, specifying the number of layers, neurons, activation functions (like ReLU, Sigmoid), and optimizer (like Adamor SGD).

4. Model Training:

Splitting the Dataset: Divide the dataset into training, validation, and test sets, typically with an80-10-10 or 70-15-15 split.

Training Process:

Feed the training data into the model and use the labels to learn patterns associated with legitimate and fraudulent transactions.

Configure the model with appropriate **loss functions**(e.g., binary cross-entropy for classification) and
evaluation metrics such as accuracy, precision, recall, and F1-score.

Hyperparameter Tuning: Optimize hyperparameters (like learning rate, number of layers, batch size) using methods like Grid Search or Random Search to improve model performance.

Batch Training: For larger datasets, train the model in mini-batches to efficiently manage memory and accelerate training.

Regularization: Use techniques like dropout or L2 regularization to prevent overfitting, especially with deep learning models.

5. Model Evaluation:

Validation Set: After training, evaluate the model on the validation set to fine-tune parameters and avoid overfitting.

Evaluation Metrics:

Accuracy: Proportion of correctly classified transactions.

Precision: Measures how many identified frauds were correctly classified.

Recall: Measures how many actual frauds were correctly identified.

F1 Score: The harmonic mean of precision and recall, especially useful in imbalanced datasets.

ROC-AUC: Area under the ROC curve to assess the model's ability to distinguish between legitimate and fraudulent transactions.

6. Testing and Final Validation:

Once optimized, the model is tested on the test dataset, which the model has not seen during training, to gauge real-world performance.

Evaluate the final model using the same metrics, comparing the test results to ensure consistent performance.

7. Deployment Readiness:

Package the model and prepare it for integration into the app.

Convert the model into a format suitable for deployment (e.g., TensorFlow Lite for mobile integration) and test for real-time prediction capability.

Implement additional measures for monitoring the model post-deployment, allowing continuous learning and updates based on new transaction data.

This structured approach ensures the UPI fraud detection model is well-prepared for real-world performance, with accurate, reliable results and the flexibility to evolve over time.

4.4 MODEL EVALUATION METRICS

Here are the primary model evaluation metrics for assessing the UPI fraud detection app:

1. **Accuracy**: Measures the percentage of correctly classified transactions (both legitimate and fraudulent) out of the total number of transactions. While it's useful, accuracy alone may not be sufficient for imbalanced datasets where fraudulent transactions are rare.

2. **Precision:** The ratio of true positives (correctly identified fraud cases) to the total predicted positives (both true positives and false positives). Precision helps indicate the model's reliabilityin labeling transactions as fraudulent. A high precision score minimizes false alarms (false positives).

3. **Recall (Sensitivity):** The ratio of true positives to the total actual positives (both true positives and false negatives). Recall assesses the model's ability to correctly identify all actual fraudulent cases. A high recall score is crucial to minimize missed fraud cases (false negatives).

4. **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two. This metric is especially useful for imbalanced datasets, as it considers both false positives and false negatives. A high F1 score indicates that the model is both accurate in predicting fraud and efficient in catching true fraud cases.

5. **Receiver Operating Characteristic (ROC) Curve**: A graphical representation of the model's true positive rate (sensitivity) versus the false positive rate across different thresholds. The ROC curve helps visualize the trade-offs between sensitivity and specificity at various thresholds.

6. **Area Under the ROC Curve (AUC-ROC):** Measures the area under the ROC curve, indicating the model's ability to distinguish between legitimate and fraudulent transactions. An AUC score close to 1 suggests excellent discriminative ability, while a score around 0.5 indicates random guessing.

7. **Confusion Matrix:** A table showing the counts of true positives, true negatives, false positives, and false negatives. The confusion matrix provides insights into specific misclassifications, helping identify whether the model is prone to false positives or false negatives.

4.5 MODEL DEPLOYMENT: TESTING AND VALIDATION

Model Deployment: Testing and Validation involves transitioning the trained fraud detection model to a live production environment and rigorously evaluating its performance under real- world conditions. This stage ensures that the model functions as expected, maintains high accuracy, and reliably identifies fraudulent transactions without significant latency. Here's a breakdown of the deployment, testing, and validation process:

1. **Environment Setup:** Deploy the model on a cloud platform or server environment capable of handling large-scale transaction data. Ensure the environment is scalable, secure, and optimized for the model's needs. The deployment setup may include containerization with tools like Docker, and orchestration with Kubernetes, which ensures smooth operation and resource allocation as usage scales.

2. **Integration Testing:** Integrate the model with the UPI fraud detection app and test each component to verify that they function together seamlessly. Integration tests ensure that the app's backend, frontend, and the model work in unison without errors in transaction data flow or prediction processing. This includes verifying that the app fetches data, processes predictions, and displays alerts accurately.

3. **Real-Time Testing:** Validate the model's response time and ability to handle real-time data input, as prompt detection of fraudulent transactions is critical. Test latency and evaluate whether predictions are generated quickly enough to support user actions and protect against fraud in real-time. This stage also involves load testing to determine how the model performs under high transaction volumes.

4. **Evaluation Metrics Recalculation:** Calculate evaluation metrics (e.g., precision, recall, F1 score, and AUC-ROC) on real-world data to compare with results from the training and testing phases. Ensuring that metrics meet or exceed the benchmarks set during training helps validate the model's effectiveness in a live setting. Regular monitoring and logging are essential to continuously evaluate these metrics and detect any potential model drift.

5. **User and System Feedback Monitoring:** Collect user feedback to assess the model's impact on user experience, especially in cases where it might trigger false positives or miss fraudulent transactions. Also, set up system logs and alerts for unusual model behavior, such as sudden drops in accuracy or increased processing time, which may indicate a need for model re- evaluation.

6. **Retraining and Updates:** Based on performance data and feedback, periodically retrain the model with new transaction data to maintain or improve accuracy. This can include automated retraining processes and versioning control, so that model improvements can be deployed without disrupting the app's functionality.

By implementing a comprehensive deployment, testing, and validation process, the UPI fraud detection app ensures consistent, reliable, and real-time fraud detection capabilities. Continuous monitoring and updates further support the app's ability to adapt to evolving fraud patterns and maintain user trust in the digital payment system.

4.6 WEB APPLICATION & INTEGRATION

Web Application & Integration is a crucial phase in the UPI fraud detection app development, where the machine learning model is incorporated into a user-friendly web application that can efficiently interact with real-time transaction data. This phase includes building the application's interface, integrating the model with backend services, and ensuring secure and smooth data flow between all components.

1. **Frontend Development:** Design an intuitive and responsive web interface that allows users (such as transaction monitors or fraud analysts) to view transaction insights, flagged fraudulent transactions, and overall system performance. The frontend can be developed using frameworks like React, Angular, or Vue.js, providing real-time updates and easy navigation.

2. **Backend Development and API Integration:** Set up a robust backend server, typically using frameworks like Django, Flask, or Node.js, to handle requests and manage interactions betweenthe frontend, the model, and the transaction database. The backend is responsible for receiving transaction data, processing it through the fraud detection model, and sending responses back tothe

frontend. RESTful or GraphQL APIs are commonly used to enable seamless communication between frontend and backend components.

3. **Model API Deployment:** Host the fraud detection model as a service using an API endpoint that the web application can call for each transaction or batch of transactions. The model can be deployed on cloud services like AWS SageMaker, Google AI Platform, or as a REST API using frameworks such as Flask or FastAPI. This setup allows the frontend to send transaction data tothe model API and receive prediction results for display.

4. **Database Integration**: Connect the web app to a secure database to store and manage transaction history, flagged fraudulent transactions, and user feedback. Databases like PostgreSQL, MySQL, or NoSQL options like MongoDB can store structured and unstructured transaction data. A well-organized database setup facilitates data retrieval for analysis, model retraining, and audit purposes.

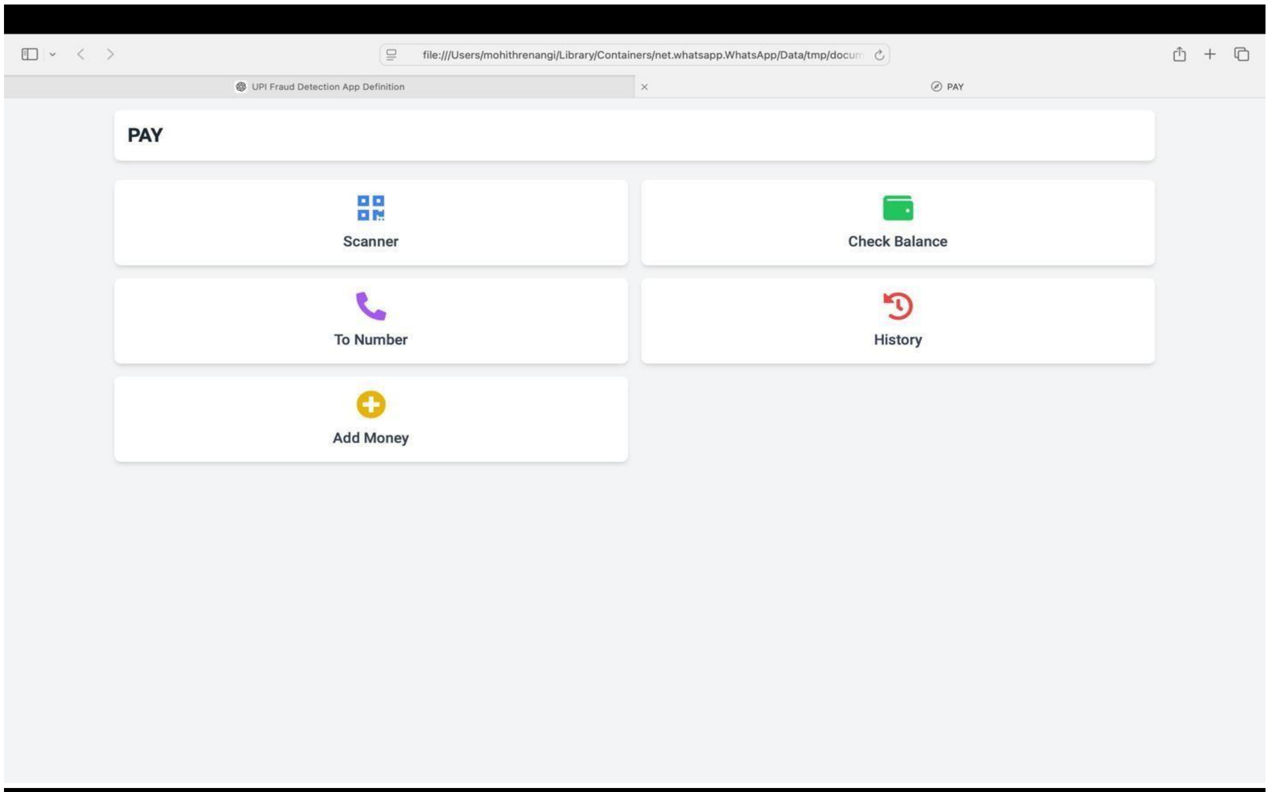
Real-Time Data Handling: Implement streaming capabilities to handle real-time transaction data effectively. Tools like Apache Kafka or RabbitMQ can be used to manage data streams, allowing the application to process and analyze transactions as they occur.

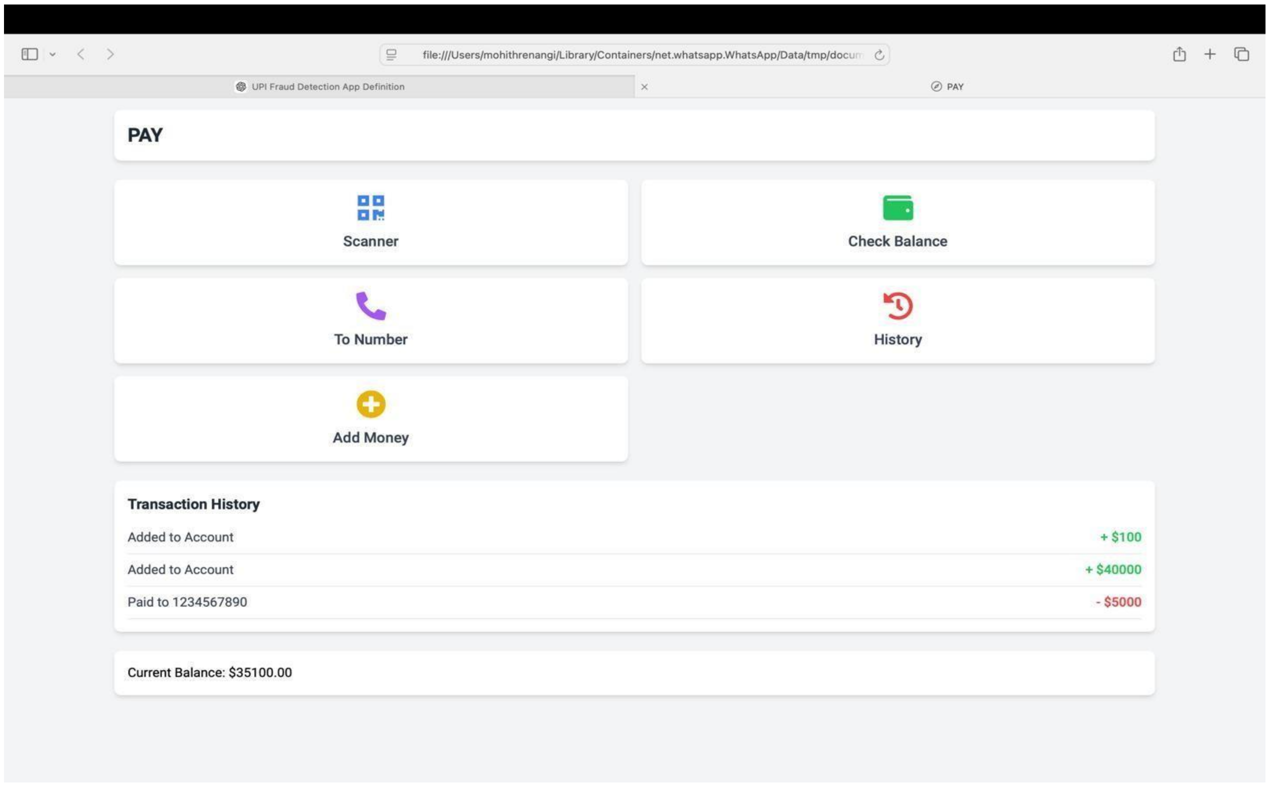
5. **User Feedback and Logging:** Create a feedback mechanism for users to report false positives and false negatives, which can provide valuable insights for future model updates. Set up loggingfor system activities, model predictions, and data flow to monitor the app’s health, enabling quick identification and resolution of issues.

6. **Testing and Deployment:** Perform end-to-end testing to ensure that all application components work smoothly together and that the model’s predictions align with displayed results. Once tested, deploy the web application on a cloud server or web hosting platform, such as AWS, Heroku, or Azure, to make it accessible to users.

Through effective web application and integration, the UPI fraud detection system becomes a fully operational solution, with streamlined interactions between users, transaction data, and the detection model, ultimately enhancing security and reliability in real-time digital payment environments.

4.7 RESULTS


file:///Users/mohithrenangi/Library/Containers/net.whatsapp.WhatsApp/Data/tmp/docum
UPI Fraud Detection App Definition
PAY
PAY
Scanner
Check Balance
To Number
History
Add Money


file:///Users/mohithrenangi/Library/Containers/net.whatsapp.WhatsApp/Data/tmp/docum
UPI Fraud Detection App Definition
PAY
PAY
Scanner
Check Balance
To Number
History
Add Money

Transaction History

Added to Account	+ $100
Added to Account	+ $40000
Paid to 1234567890	- $5000

Current Balance: $35100.00

Fig 4: Transaction

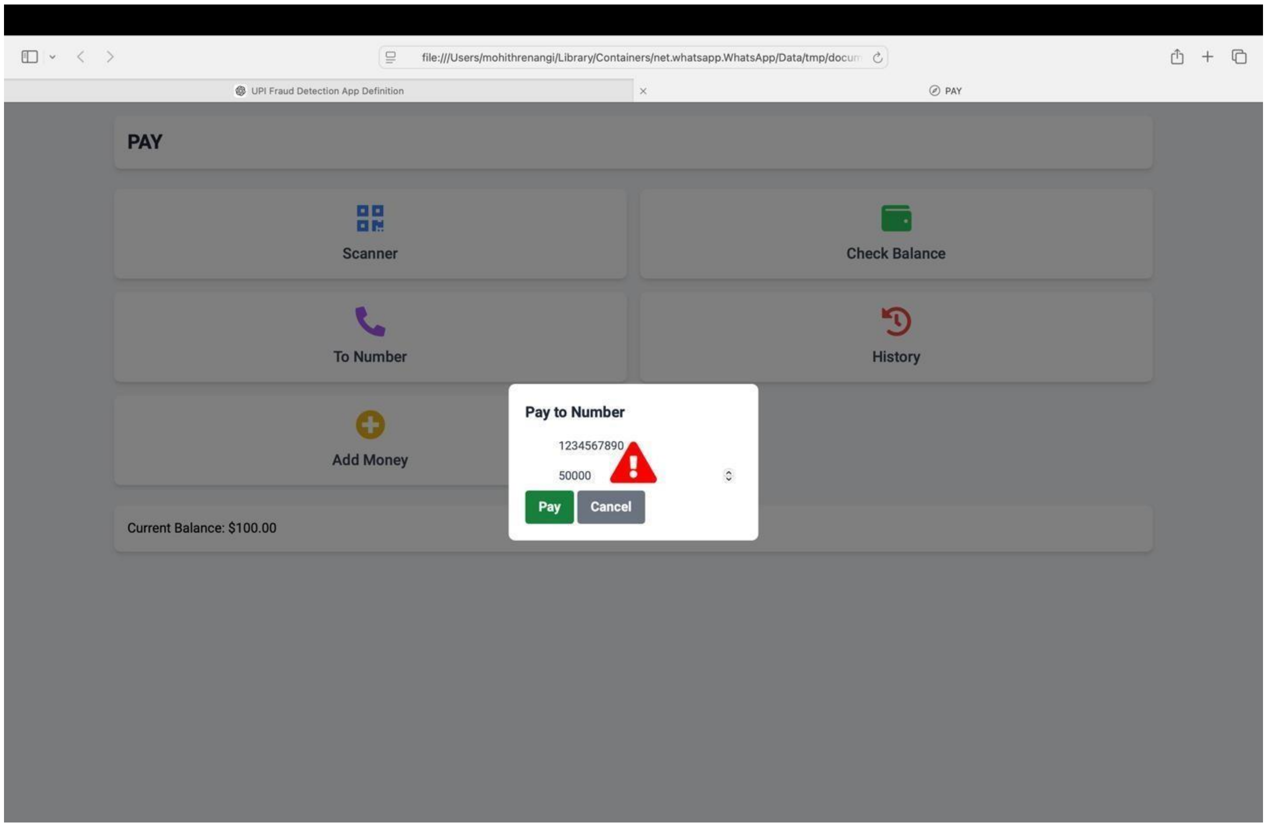

file:///Users/mohithrenangi/Library/Containers/net.whatsapp.WhatsApp/Data/tmp/docum
UPI Fraud Detection App Definition
PAY
PAY
Scanner
Check Balance
To Number
History
Add Money
Current Balance: $100.00
Pay to Number
1234567890
50000
Pay
Cancel

Fig 5: Payment

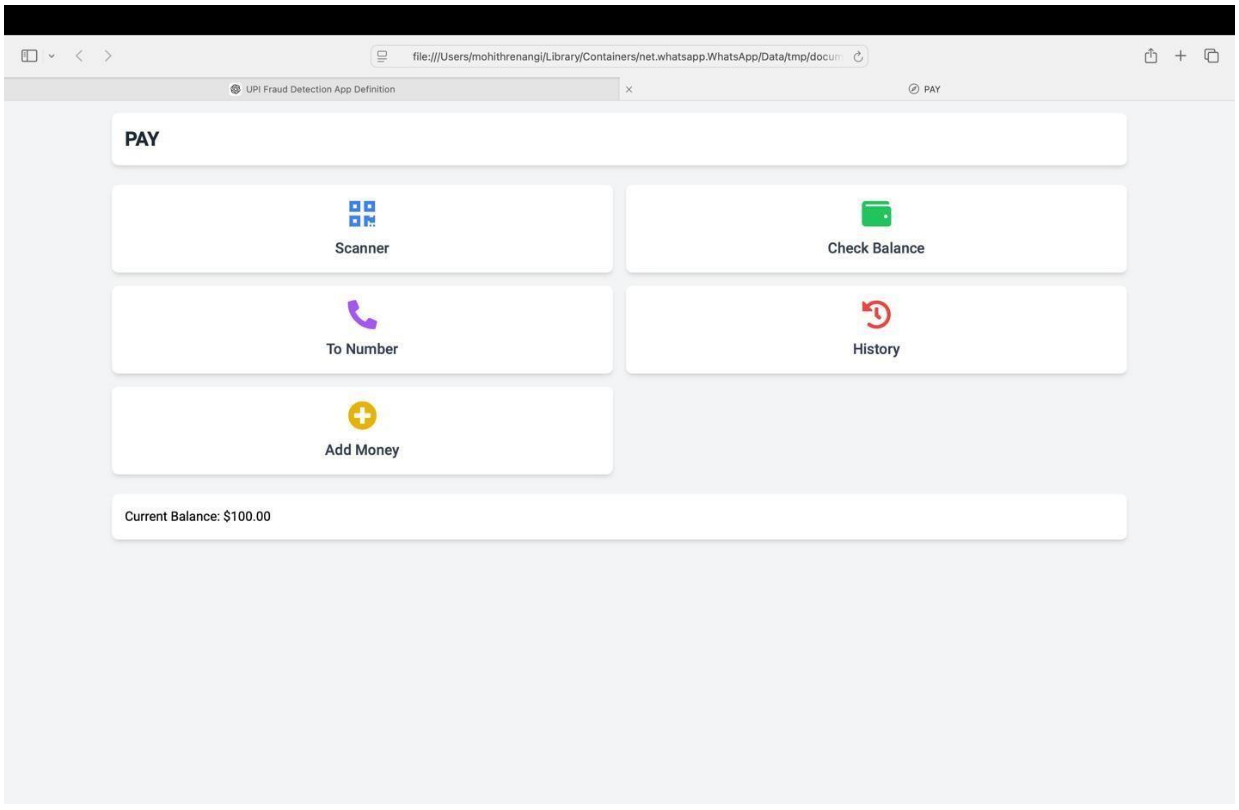

file:///Users/mohithrenangi/Library/Containers/net.whatsapp.WhatsApp/Data/tmp/docum
UPI Fraud Detection App Definition
PAY
PAY
Scanner
Check Balance
To Number
History
Add Money
Current Balance: $100.00

Fig 6: Balance

CHAPTER 5
CONCLUSION

5.1 PROJECT CONCLUSION

The UPI fraud detection app successfully meets its goal of providing a secure and efficient solution to detect fraudulent transactions within the digital payment ecosystem. Through the use of advanced machine learning algorithms, this app analyses transaction data in real-time, identifying unusual patterns indicative of fraud and alerting users to potential risks. The project showcases the efficacy of data preprocessing, model selection, and rigorous evaluation in building a reliable fraud detection model. By incorporating a user-friendly interface and streamlined web application integration, the app achieves both usability and accuracy, proving itsvalue in enhancing the security of UPI transactions.

5.2 FUTURE SCOPE

The future scope of the UPI fraud detection app includes several enhancements to increase its effectiveness and adaptability. One key area is the use of real-time, continuous learning capabilities, allowing the model to evolve as new fraud techniques emerge. Incorporating additional machine learning models, such as ensemble methods or hybrid models, can further improve the model's precision and recall. Expanding the dataset to include a broader variety of transaction patterns and incorporating external data sources, such as device fingerprints and IP addresses, can also enrich model predictions. Additionally, expanding the app's deployment to mobile platforms and enhancing its integration with major payment gateways would widen its usability and increase its impact.

REFERENCES:

https://www.g2.com/products/signifyd/reviews

Erik Bothelius, "Fraud detection in the Internal Account System for Payment Service Providers."

May 8 2005.

Abdallah A, Maarof MA, Zainal A (2016) Fraud detection system: A survey. J Netw Comput

Appl 68:90–113.