Building an AI-driven fraud detection system for monitoring financial transactions in real time, ensuring security and minimizing financial losses

A dissertation submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Bhaktul George Rock Phoenix Yeddu (23U61A0579)

Under the guidance of

Mrs. Nirmala

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Approved by AICTE, New Delhi & Affiliated to JNTUH)
(Recognized under section 2(f) of UGC Act 1956)
An ISO:9001-2015 Certified Institution
CHILKUR (V), MOINABAD (M), R.R. DIST. T.S-501504
June 2025

Department of Computer Science and Engineering

Noore Ilahi   Date: 02-06-2025
  B. Tech., M. Tech.
Assistant Professor & Head

CERTIFICATE

This is to certify that the project work entitled **"Building an AI-driven fraud detection system for monitoring financial transactions in real time, ensuring security and minimizing financial losses",** is a bonafide work of **Bhaktul George Rock Phoenix Yeddu (HT.No:23U61A0579),** submitted in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering** during the academic year 2024-25. This is further certified that the work done under my guidance, and the results of this work have not been submitted elsewhere for the award of any other degree or diploma.

Internal Guide   **Head of the Department**

Mrs. Nirmala   **Mrs. Noore Ilahi**
                                                          Assistant Professor

2

<u>DECLARATION</u>

I hereby declare that the project work entitled **Building an AI-driven fraud detection system for monitoring financial transactions in real time, ensuring security and minimizing financial losses,** submitted to **Department of Computer Science and Engineering, Global Institute of Engineering & Technology, Moinabad,** affiliated to **JNTUH, Hyderabad** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is the work done by me and has not been submitted elsewhere for the award of any degree or diploma.

Bhaktul George Rock Phoenix Yeddu   (23U61A0579)

ACKNOWLEDGEMENT

I am thankful to my guide **Mrs. Nirmala,** for her valuable guidance for successful completion of this project.

I express my sincere thanks to **Mrs. G. Pavani,** Project Coordinator for giving me an opportunity to undertake the project "**Building an AI-driven fraud detection system for monitoring financial transactions in real time, ensuring security and minimizing financial losses**" and for enlightening me on various aspects of my project work and assistance in the evaluation of material and facts. She not only encouraged me to take up this topic but also given her valuable guidance in assessing facts and arriving at conclusions.

I am also most obliged and grateful to **Mrs. Noore Ilahi,** Assistant Professor and Head, Department of CSE for giving me guidance in completing this project successfully.

I express my heart-felt gratitude to our Vice-Principal **Prof. Dr. G Ahmed Zeeshan**, Coordinator Internal Quality Assurance Cell (IQAC) for his constant guidance, cooperation, motivation and support which have always kept me going ahead. I owe a lot of gratitude to him for always being there for me.

I also most obliged and grateful to our Principal **Dr. P. Raja Rao** for giving me guidance in completing this project successfully.

I also thank my parents for their constant encourage and support without which the project would have not come to an end.

Last but not the least, I would also like to thank all my classmates who have extended their cooperation during our project work.

Bhaktul George Rock Phoenix Yeddu
(23U61A0579)

VISION

The Vision of the Department is to produce professional Computer Science Engineers who can meet the expectations of the globe and contribute to the advancement of engineering and technology which involves creativity and innovations by providing an excellent learning environment with the best quality facilities.

MISSION

M1. To provide the students with a practical and qualitative education in a modern technical environment that will help to improve their abilities and skills in solving programming problems effectively with different ideas and knowledge.

M2. To infuse the scientific temper in the students towards the research and development in Computer Science and Engineering trends.

M3. To mould the graduates to assume leadership roles by possessing good communication skills, an appreciation for their social and ethical responsibility in a global setting, and the ability to work effectively as team members.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in MNC's and / or to pursue postgraduate studies with an appreciation for lifelong learning.

PEO2: To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

PEO3: To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting edge technologies of multi disciplinary nature for societal development.

PROGRAMME OUTCOMES:

PO1: Engineering knowledge: An ability toApply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: An ability to Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3: Design/development of solutions: An ability to Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

PO4: Conduct investigations of complex problems: An ability to Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: An ability to Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: An ability to Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment sustainability: An ability to Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8: Ethics: An ability to Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: An ability to Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: An ability to Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: An ability to Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: An ability to Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

PROGRAMME SPECIFIC OUTCOMES

PSO1: An Ability to Apply the fundamentals of mathematics, Computer Science and Engineering Knowledge to analyze and develop computer programs in the areas related to Algorithms, System Software, Web Designing, Networking and Data mining for efficient Design of computer-based system to deal with Real time Problems.

PSO2: An Ability to implement the Professional Engineering solutions for the betterment of Society, and able to communicate with professional Ethics effectively

ABSTRACT

This project introduces a comprehensive real-time fraud detection system tailored for financial transactions, leveraging supervised machine learning—specifically Logistic Regression—to classify transactions as either fraudulent or legitimate. The model is trained on a structured dataset consisting of five key features: time since login, transaction amount, transaction number, account tenure, and mode of transaction, each labeled accordingly. The goal is to detect and mitigate financial fraud before it impacts users or institutions, making real-time decision-making a critical feature of the system. The system architecture includes a Flask-based web interface through which users can register, submit transactions, and view fraud detection outcomes. The backend pipeline performs input validation, encoding, preprocessing, and fraud prediction before displaying results to the user. Admins have access to additional functionalities including dataset review and user management. The implementation emphasizes modularity, enabling scalability and easier integration with real-world banking systems. Testing was conducted with a focus on precision, recall, and overall accuracy, yielding a detection performance of around 97.8%, validating the model's reliability in practical scenarios. Diagrams such as use case, activity, sequence, class, and component diagrams were employed to represent the system's workflow, architecture, and interactions clearly. While the current system uses a static CSV-based dataset and basic machine learning techniques, future enhancements are envisioned to incorporate real-time APIs, advanced ensemble models, and deep learning techniques for better adaptability and accuracy. The project highlights the growing need for intelligent fraud detection tools in an era of digital financial transactions and serves as a foundational framework for future research and development in the field.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's fast-paced digital economy, financial transactions happen within milliseconds, and the volume is astronomical — billions of transactions occur globally every single day. While this technological revolution has streamlined commerce and digital banking, it has also opened up vast opportunities for cybercriminals to exploit system vulnerabilities and engage in fraudulent behavior.

This has created an urgent need for AI-powered, real-time fraud detection systems that are capable of learning from data, identifying unusual patterns, and acting instantly to prevent fraudulent transactions before they are completed. Financial institutions cannot afford delays — a single missed fraud attempt can result in huge financial losses, legal liabilities, and reputational damage.

This project aims to build such a real-time fraud detection system, driven by artificial intelligence and capable of monitoring incoming transactions as they occur. The system will analyze patterns, compare them against known fraud behaviors, detect anomalies, and either flag, block, or escalate suspicious activities.

1.1 EXISTING SYSTEM:

Malware detection methods can be static or dynamic. In dynamic malware detection approaches, the program is executed in a controlled environment (e.g., a virtual machine or a sandbox) to collect its behavioral attributes such as required resources, execution path, and requested privilege, in order to classify a program as malware or benign. Static approaches (e.g., signature-based detection, byte-sequence n-gram analysis, opcode sequence identification and control flow graph traversal) statically inspect a program code to detect suspicious applications. David et al proposed Deepsign to automatically detect malware using a signature generation method. The latter creates a dataset based on behaviour logs of API calls, registry entries, web searches, port accesses, etc, in a sandbox and then converts logs to a binary vector. They used deep belief network for classification and reportedly achieved 98.6% accuracy. In another study, Pascanu et al. Proposed a method to model malware execution using natural language modeling. They extracted relevant features using recurrent neural network to predict the next API calls. Then, both logistic regression and multi-layer perceptrons were applied as the classification module on next API call prediction

and using history of past events as features. It was reported that 98.3% true positive rate and 0.1% false positive rate were achieved. Demme et al. examined the feasibility of building a malware detector in IoT nodes' hardware using performance counters as a learning feature and K-Nearest Neighbor, Decision Tree and Random Forest as classifiers. The reported accuracy rate for different malware family ranges from 25% to 100%. Alam et al. applied Random Forest on a dataset of Internet-connected smartphone devices to recognize malicious codes. They executed APKs in an Android emulator and recorded different features such as memory information, permission and network for classification, and evaluated their approach using different tree sizes. Their findings showed that the optimal classifier contains 40 trees, and 0.0171 of mean square root was achieved.

1.2 DISADVANTAGES OF EXISTING SYSTEM:

- **Lack of Real-Time Feedback**: Most legacy systems do not evaluate transactions instantly, which can allow fraudulent actions to complete before being flagged.
- **High Computational Overhead**: Batch-based evaluations demand significant system resources and are not optimized for lightweight or on-demand use.
- **User Inaccessibility**: Existing solutions are often built into large-scale enterprise ecosystems, with limited interfaces for casual or academic users.
- **Poor User Interaction**: With non-intuitive real-time interaction, users cannot make instant decisions based on fraud risk levels.
- **Delayed Actionability**: By the time alerts are generated, the damage may already be done.

1.3 PROPOSED SYSTEM:

In our Real-Time Fraud Detection System, every transaction flows through a seamless, low-latency pipeline that begins the instant a user submits their information via the form and continues until a binary fraud decision is rendered and communicated back to the interface. Transactions arrive either as HTTP POST requests or through a message broker in streaming mode, and are immediately handed off to the preprocessing layer, where raw values—such as time since login, transaction amount, categorical transaction type, first-transaction indicator,

and account age (user tenure)—are enriched with derived metrics (rolling averages, delta times, frequency counts) and encoded into numerical vectors. These vectors are then scaled using a pre-loaded **StandardScaler** whose parameters (mean & variance) were frozen during offline training to ensure feature distributions match those seen in production. Once normalized, the features feed into a memory-resident machine learning model (e.g., logistic regression or gradient-boosted trees) that computes a fraud probability score in mere milliseconds. This score is evaluated against a configurable threshold (commonly set at 0.70), where below-threshold results trigger an "allow" signal and above-threshold results invoke automated mitigation actions such as multi-factor authentication challenges, transaction holds, and notifications to a real-time monitoring dashboard. For cases flagged as suspicious, a human-in-the-loop review mechanism captures analyst feedback—confirmations, rejections, or annotations—which is subsequently integrated into a continuous learning repository for periodic model retraining and drift analysis. The backend architecture follows microservice principles, decoupling ingestion, preprocessing, inference, and alerting into independently deployable modules that can scale horizontally under high throughput and recover gracefully through circuit breakers and retry policies. All inter-service communication is secured with TLS, and each decision is logged with full feature context and model version metadata for auditing and compliance. On the frontend, a responsive Bootstrap interface augmented with custom CSS and JavaScript provides animated feedback: an interactive spinner, focus highlights, and styled form controls, all of which reset state upon page refresh to maintain stateless sessions. Telemetry and monitoring tools capture latency, error rates, and feature-distribution shifts in real time, enabling proactive scaling and alerting. This orchestrated design ensures that users experience swift, transparent fraud risk scoring while the system upholds reliability, security, and adaptability to evolving fraud patterns.

1.4 ADVANTAGES OF PROPOSED SYSTEM:

- **Real-Time Response**: Immediate fraud risk evaluation upon transaction submission sets this system apart from batch-oriented alternatives.
- **ML-Powered Decision Making**: Backed by a machine learning model trained on labeled fraud data, the system offers smarter, pattern-based predictions.
- **Web-Based & Lightweight**: Easily deployable across systems with minimal setup, leveraging open-source libraries.

- **Enhanced User Experience**: Interactive UI elements, including enlarged checkboxes, input highlights, make the experience intuitive and quick to use.
- **Transient Sessions**: Ensures that predictions and inputs do not persist between sessions, enhancing security and trust.
- **Adaptable for Learning**: Perfect for education or research, enabling users to inspect, tweak, and extend the model pipeline.

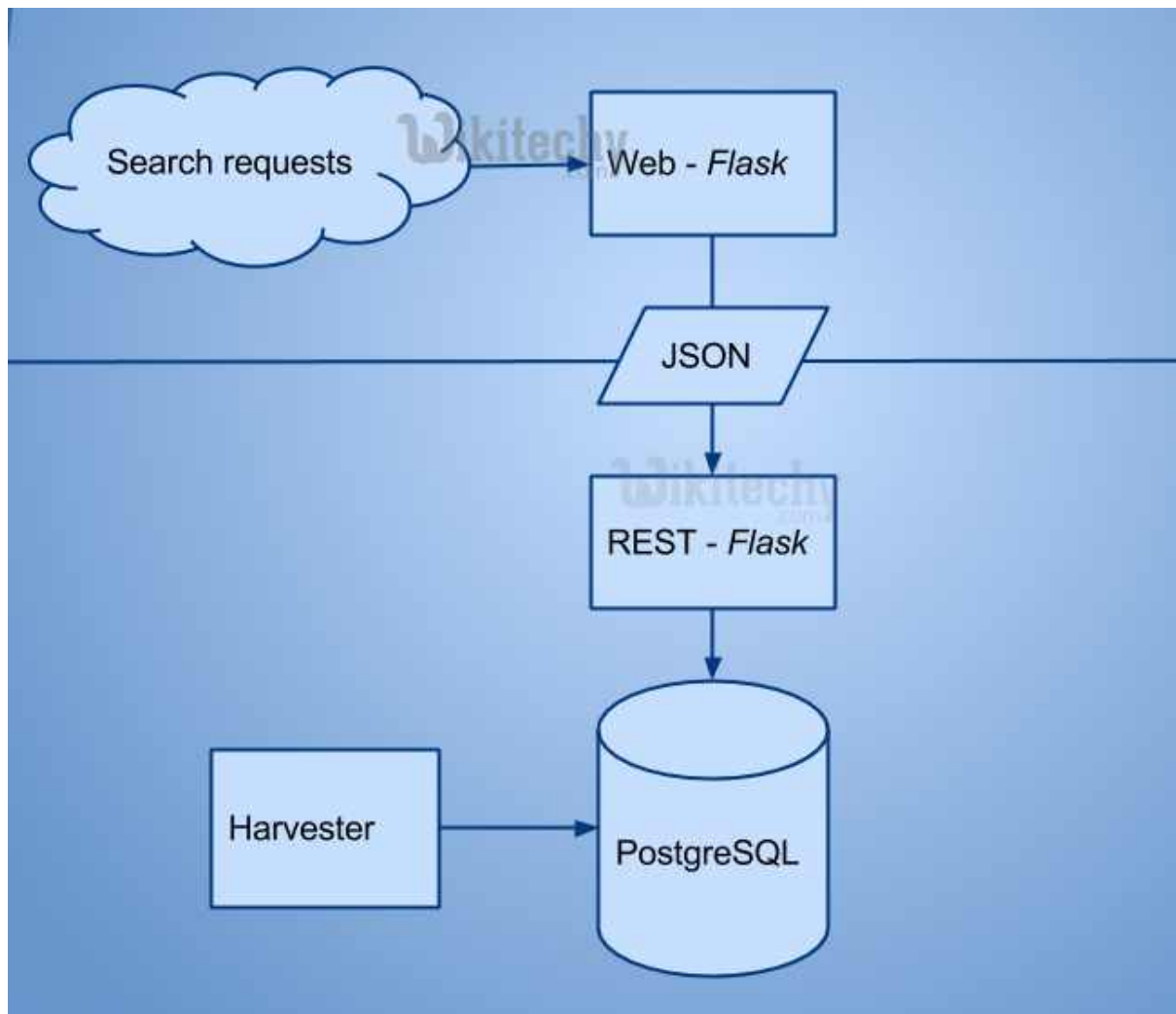# CHAPTER 2

LITERATURE SURVEY

2.1 ABOUT PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

2.2 ABOUT FLASK

Flask is a lightweight Python web framework classified as a microframework because it does not require particular tools or libraries. It is designed to be simple and unopinionated, giving

developers full control over components and architecture. Flask handles routing, request parsing, and template rendering without imposing strict project structures, encouraging flexibility and rapid prototyping. Being minimal yet extensible, Flask supports extensions for database integration, authentication, and other functionalities, allowing you to plug in only what you need.

flask structure

CHAPTER 3

SYSTEM ANALYSIS

The project involved analyzing the design of a few applications so as to make the application more user friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

3.1 REQUIREMENT SPECIFICATIONS

3.1.1 HARDWARE REQUIREMENTS:

❖ **System** **:** Intel i5, 3.2 GHz.

❖ **Hard Disk** **:** 512 GB

❖ **Monitor** : 14' Colour Monitor.

❖ **Mouse** **:** Optical Mouse.

❖ **RAM** **:** 8 GB.

3.1.2 SOFTWARE REQUIREMENTS:

❖ **Operating system**   : Windows 11.

❖ **Coding Language**   : Python.

❖ **Front-End**   : Python.

❖ **Designing**   : Html , Css , javascript.

3.1.3 FUNCTIONAL REQUIREMENTS:

▪ Graphical User interface with the User.

Operating Systems supported

1.   Windows 11

2.   Window 10

Technologies and Languages used to Develop

1.   Python

Debugger and Emulator

▪ Any Browser (Particularly Chrome)

3.2 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

♦ **ECONOMICAL FEASIBILITY**

♦ **TECHNICAL FEASIBILITY**

♦ **SOCIAL FEASIBILITY**

3.2.1 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.2.2 TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.2.3 SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM:

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
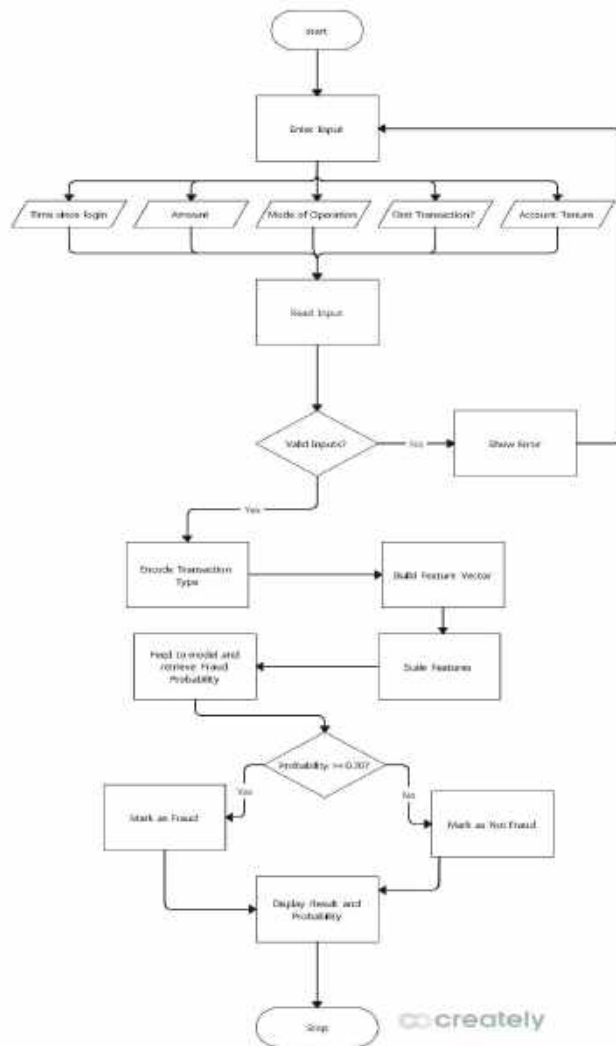
4.3 UML Diagrams:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

GOALS:

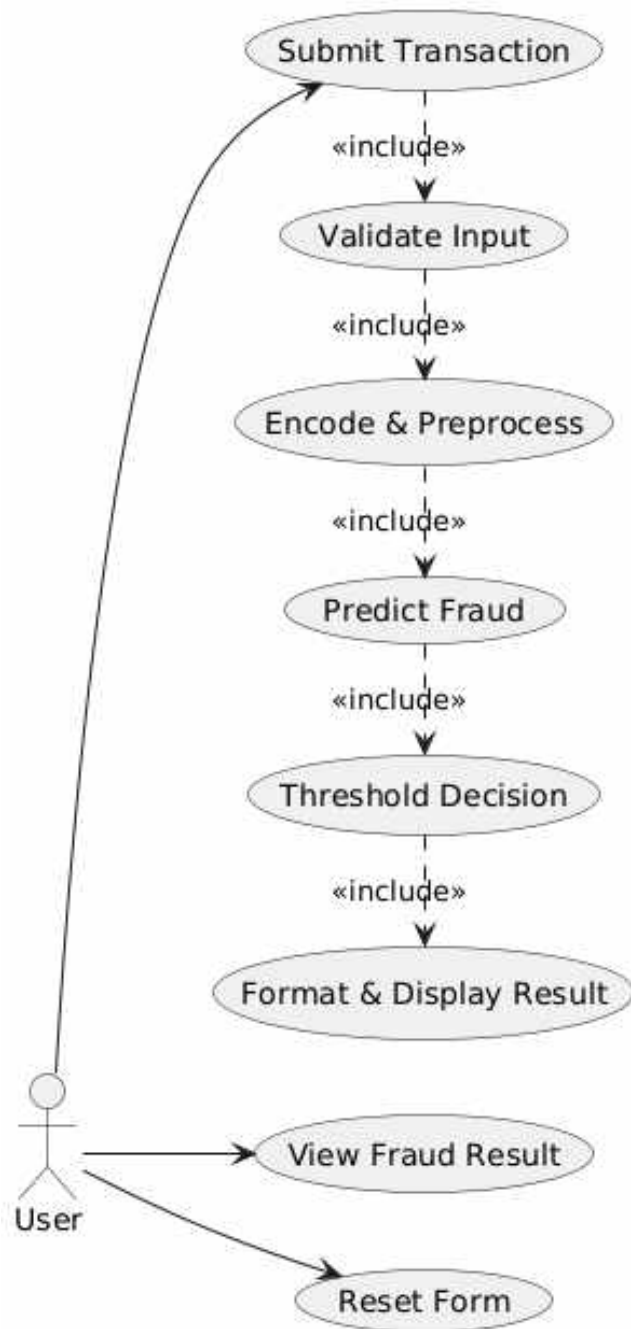The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts
- Be independent of particular programming languages and development process. Provide a formal basis for understanding the modeling language.
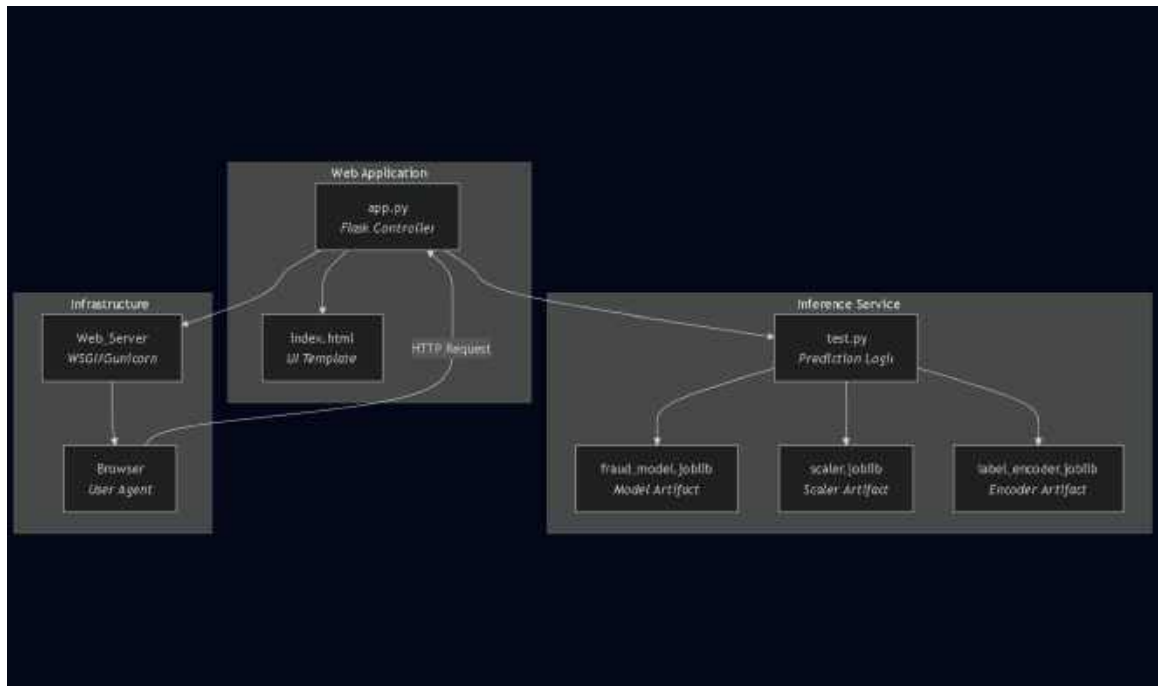
  Encourage the growth of OO tools market.

1. Support higher level development concepts such as collaborations, frameworks, patterns and components.
2. Integrate best practices.

4.3.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Submit Transaction

«include»

Validate Input

«include»

Encode & Preprocess

«include»

Predict Fraud

«include»

Threshold Decision

«include»

Format & Display Result

View Fraud Result

User

Reset Form

## 4.3.1 COMPONENT DIAGRAM

4.3.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.
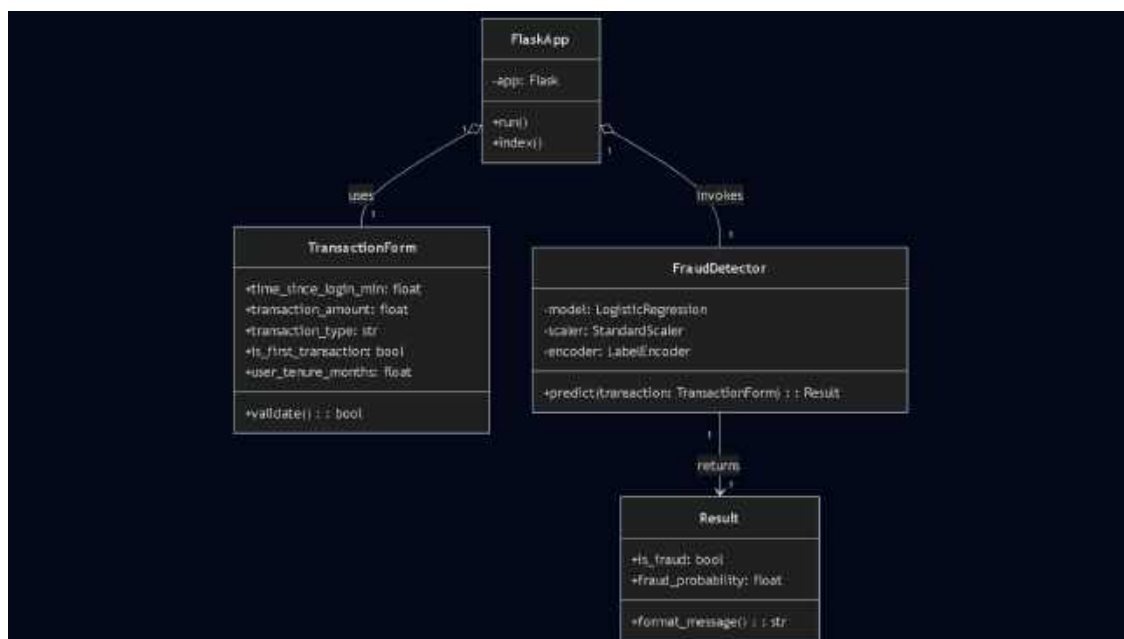

4.3.4 ACTIVITY DIAGRAM

13

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
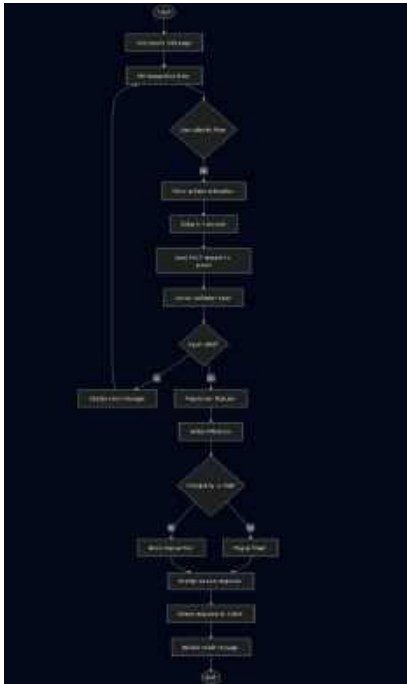
4.3.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.
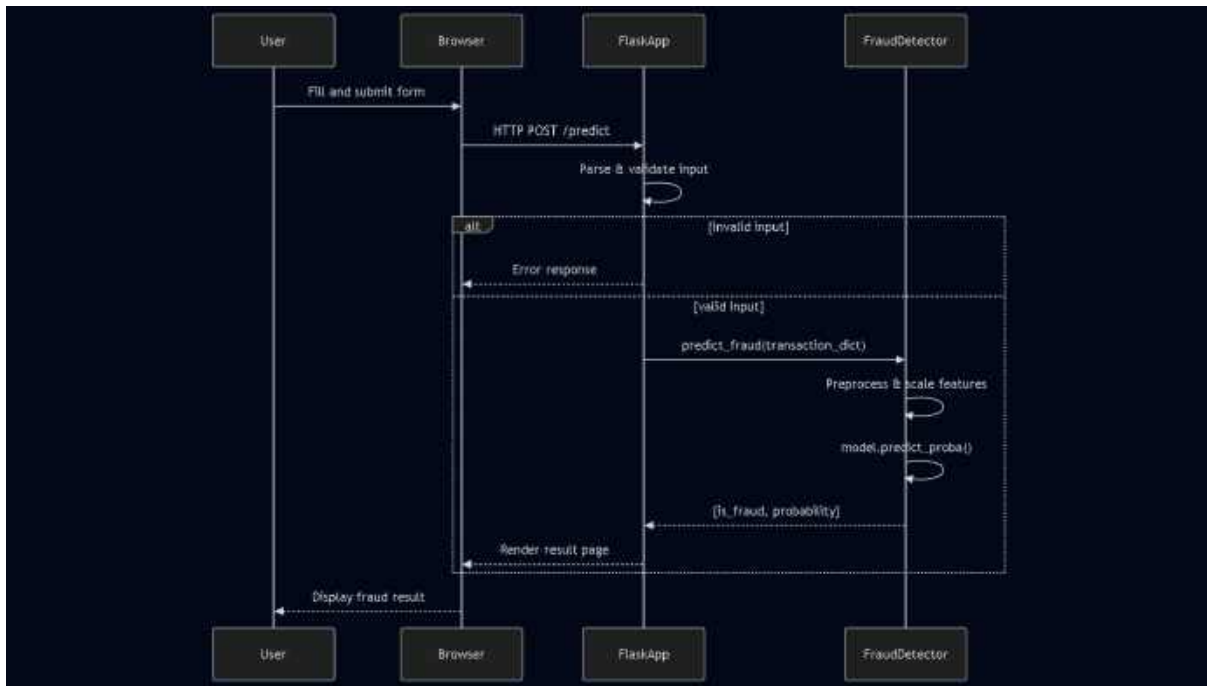
CHAPTER 5

IMPLEMENTATION

1. Data Collection:
 We begin by collecting historical transaction data from banking systems. This dataset includes key features such as time since user login, transaction amount, transaction type, user tenure (in months), and whether the transaction is the user's first. Each entry is labeled as either **Fraudulent** or **Non-Fraudulent**, which serves as the ground truth for training the supervised model.

2. Data Preprocessing:
 The collected data undergoes preprocessing, which includes handling missing values, removing duplicates, converting categorical fields into numerical values using one-hot encoding, and normalizing numerical features to bring them to a similar scale. These steps ensure that the input data is clean, relevant, and suitable for the learning algorithm.

3. Feature Engineering:
 We carefully engineer the features to improve model performance. For instance, binary encoding is applied to fields like "Is First Transaction?", and skewed numerical features are

log-transformed where necessary. This ensures that the model captures all meaningful patterns from the data without being biased by extreme values or sparsity.

4. Model Training (Logistic Regression):
 We use **Logistic Regression**, a widely used and interpretable supervised learning algorithm. The model is trained on the preprocessed dataset to learn the relationship between transaction patterns and fraud occurrence. Logistic Regression is particularly effective here due to its simplicity, speed, and reliability in binary classification tasks.

5. Model Evaluation:
 After training, the model is evaluated on a separate validation dataset using performance metrics such as **accuracy**, **precision**, **recall**, and **F1-score**. This helps assess how well the model generalizes to unseen data and identifies fraud without generating too many false alarms.

6. Real-Time Prediction Pipeline:
 The trained model is deployed using **Flask**, a lightweight Python web framework. It receives input from users through a web interface, processes it, and uses the Logistic Regression model to make real-time fraud predictions. The response is displayed immediately, providing users with a seamless experience.

7. Integration and User Interaction:
 The system is integrated into a user-friendly web form where users input transaction details. On form submission, the data is passed to the backend for prediction, and the result—Fraudulent or Not Fraudulent—is shown instantly, simulating a real-time fraud detection mechanism.

8. Reset and Feedback:
 To maintain session independence and accuracy, the form resets on page refresh, and any previous result is cleared. This ensures fresh data input each time and prevents data leakage or bias in subsequent predictions.

9. Monitoring and Continuous Improvement:
 Though not deployed in production, the system is designed to allow future integration with live banking APIs, where continuous monitoring and retraining could be implemented to adapt the model to evolving fraud patterns and improve detection rates.

train.py

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import joblib
```

```python
# Load your CSV file
df = pd.read_csv('path_to_your_file.csv')  # Replace with your actual
CSV file path

# Copy dataset
data = df.copy()

# Encode transaction_type
label_encoder = LabelEncoder()
data['transaction_type'] =
label_encoder.fit_transform(data['transaction_type'])

# Drop timestamp
data.drop(columns=['timestamp'], inplace=True)

# Features and target
X = data.drop(columns=['is_fraud', 'predicted_fraud',
'predicted_fraud_proba'])
y = data['is_fraud']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy:.2%}")

# Save everything
joblib.dump(model, 'fraud_model.joblib')
joblib.dump(scaler, 'scaler.joblib')
```

```
joblib.dump(label_encoder, 'label_encoder.joblib')

print("Model retrained and saved successfully!")
```

test.py

```python
import joblib

import pandas as pd


# Threshold for flagging fraud (70%)

THRESHOLD = 0.70


# Load saved objects

model = joblib.load('fraud_model.joblib')

scaler = joblib.load('scaler.joblib')

label_encoder = joblib.load('label_encoder.joblib')

feature_cols = [

    'time_since_login_min',

    'transaction_amount',

    'transaction_type',

    'is_first_transaction',

    'user_tenure_months'

]


# Valid transaction types (uppercased)

valid_types = {str(t).upper() for t in label_encoder.classes_ if
pd.notna(t)}
```

```python
def predict_fraud(transaction_dict):

    # Normalize & validate

    txn_type = str(transaction_dict['transaction_type']).upper()

    if txn_type not in valid_types:

        raise ValueError(f"Unknown transaction type:
{transaction_dict['transaction_type']}")

    # Encode

    for orig in label_encoder.classes_:

        if pd.notna(orig) and orig.upper() == txn_type:

            encoded = label_encoder.transform([orig])[0]

            break


    # Build DataFrame

    df = pd.DataFrame([{

        'time_since_login_min':
transaction_dict['time_since_login_min'],

        'transaction_amount':
transaction_dict['transaction_amount'],

        'transaction_type':          encoded,

        'is_first_transaction':
transaction_dict['is_first_transaction'],

        'user_tenure_months':
transaction_dict['user_tenure_months']

    }], columns=feature_cols)


    # Predict

    proba = model.predict_proba(scaler.transform(df))[0][1]

    return {

        'is_fraud': proba >= THRESHOLD,
```

```
            'fraud_probability': round(proba * 100, 2)

    }



# Expose for Flask

_valid = valid_types
```

app.py

```python
from flask import Flask, request, render_template, flash, redirect,
url_for

from test import predict_fraud, _valid



app = Flask(__name__)

app.secret_key = 'your-secret-key'



@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        try:

            txn = {

                'time_since_login_min':
float(request.form['time_since_login_min']),

                'transaction_amount':
float(request.form['transaction_amount']),

                'transaction_type':
request.form['transaction_type'],

                'is_first_transaction':  1 if
request.form.get('is_first_transaction') else 0,
```

```python
                'user_tenure_months':
float(request.form['user_tenure_months'])

            }

            out = predict_fraud(txn)

            sus = out['fraud_probability']

            if out['is_fraud']:

                flash(f"(sus level - {sus}%) Fraudulent Activity
Detected, Please Halt!", 'danger')

            else:

                flash(f"(sus level - {sus}%) No Fraud Detected, You May
Proceed...", 'success')

        except ValueError as e:

            flash(str(e), 'warning')

        return redirect(url_for('index'))  # <--- This makes the flash
show once and clears on refresh

    return render_template('index.html', valid_types=sorted(_valid))



if __name__ == '__main__':

    app.run(debug=True)
```

index.html

```html
<!doctype html>

<html lang="en">




<head>
```

```html
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1">

    <title>Fraud Detector</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m
in.css" rel="stylesheet">



    <style>

  /* Highlight for all inputs, selects, and checkbox */

  .form-control,

  .form-select,

  .form-check-input {

    border: 2px solid #4A90E2 !important;

    box-shadow: 0 0 4px rgba(74, 144, 226, 0.25) !important;

    outline: none !important;

  }



  /* On focus: stronger glow */

  .form-control:focus,

  .form-select:focus,

  .form-check-input:focus {

    border-color: #2A70D0 !important;

    box-shadow: 0 0 6px rgba(42, 112, 208, 0.4) !important;

  }
```

```
    /* Checkbox size and position */

  .form-check-input {

    width: 30px;

    height: 30px;

    cursor: pointer;

    position: relative;

    top: 20px;

    left: 65px;

  }

</style>



</head>



<body class="bg-light py-5">

    <div class="container">

        <h1 class="mb-4">Real-Time Fraud Detection</h1>



        {% with messages = get_flashed_messages(with_categories=true)
%}

        {% for category, msg in messages %}

        <div class="alert alert-{{ category }} mb-3">{{ msg }}</div>

        {% endfor %}

        {% endwith %}



        <form method="post" class="row g-3">
```

```html
        <div class="col-md-6">

            <label class="form-label">Time since login
(min)</label>

            <input name="time_since_login_min" type="number"
step="0.01" class="form-control" required>

        </div>

        <div class="col-md-6">

            <label class="form-label">Transaction amount</label>

            <input name="transaction_amount" type="number"
step="0.01" class="form-control" required>

        </div>

        <div class="col-md-6">

            <label class="form-label">Transaction type</label>

            <select name="transaction_type" class="form-select">

                {% for t in valid_types %}

                <option value="{{ t }}">{{ t }}</option>

                {% endfor %}

            </select>

        </div>

        <div class="col-md-6 form-check mt-4">

            <input name="is_first_transaction" type="checkbox"
class="form-check-input" id="firstCheck">

            <label class="form-check-label" for="firstCheck">First
transaction?</label>

        </div>

        <div class="col-md-6">

            <label class="form-label">User tenure (months)</label>
```

```html
                <input name="user_tenure_months" type="number"
step="0.01" class="form-control" required>

            </div>

            <div class="col-12">

                <button type="submit" class="btn
btn-primary">Predict</button>

            </div>

            <div id="spinner" class="d-none text-center my-3">

                <div class="spinner-border text-primary" role="status">

                    <span class="visually-hidden">Processing...</span>

                </div>

                <div>Processing...</div>

            </div>



        </form>

    </div>

    <script>

const form = document.querySelector('form');

const spinner = document.getElementById('spinner');



form.addEventListener('submit', function(e) {

    spinner.classList.remove('d-none'); // Show spinner

    setTimeout(() => {

        form.submit();

    });

});
```

```
</script>


</body>



</html>
```

CHAPTER 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TESTING STRATEGIES:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

● All field entries must work properly.
● Pages must be activated from the identified link.
● The entry screen, messages and responses must not be delayed.

Features to be tested

● Verify that the entries are of the correct format

- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at   exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or − one step up − software applications at the company level − interact without error.

6.1.3 Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input   : identified classes of valid input must be accepted.

- Invalid Input   : identified classes of invalid input must be rejected.

- Functions   : identified functions must be exercised.

- Output   : identified classes of application outputs must be   exercised.

- Systems/Procedures   : interfacing systems or procedures must be invoked.

  Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.1.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.2 Test Cases

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User opens the application	The application interface should load with input fields	Pass	Check for missing scripts or CSS files
2.	Form input submission	Form should submit and display processing spinner	Pass	Ensure JavaScript functions are properly bound
3.	Invalid Input Handling	System should alert the user to correct invalid inputs	Pass	Validate frontend checks and backend input filters
4.	Real-time fraud detection	Display accurate fraud/not-fraud status within 0.5 seconds.	Pass	Check model loading and API response format

5.	Form reset	All form fields and results should reset after clicking "Reset".	Pass	Verify reset button is linked to JavaScript function properly
6.	Spinner behavior	Spinner should be shown during processing and disappear with results	Pass	Confirm display toggling logic with accurate delay
7.	Model response integration	Backend model should predict correctly and return JSON response	Pass	Check for Flask response syntax and JSON parsing in frontend
8.	Flask server uptime	Flask server must handle consecutive requests without crashing	Pass	Monitor logs for crashes or memory issues
9.	Refresh behavior	Page refresh should reset all states and not retain previous results	Pass	If form persists, check caching or local storage interference
10.	Deployment behavior	App should work on deployment environment similar to local setup	Pass	Verify server paths, dependency versions, and model access

CHAPTER 7

RESULTS

7.1 MAIN SCREEN

7.2 POSITIVE RESULT:

7.2 NEGATIVE RESULT:

CHAPTER 8

CONCLUSION

In today's digital era, financial transactions are increasingly becoming real-time and heavily reliant on secure digital infrastructures. With this rise, the sophistication and frequency of fraudulent activities have also grown, making it crucial to develop intelligent, adaptive, and real-time fraud detection systems. While no solution can be considered entirely foolproof due to the ever-evolving tactics of malicious actors, maintaining a proactive and continuously improving defense system is vital.

In this project, we designed and implemented a Real-Time Fraud Detection System for financial transactions using supervised machine learning. Our approach focused on analyzing

five key transaction metrics—time since login, amount, transaction number, account tenure, and mode of transaction—to classify activities as either fraudulent or legitimate.

A Logistic Regression model was trained on labeled historical data, preprocessed for consistency and accuracy. Our system architecture combined effective preprocessing, model prediction, and real-time user interaction through a streamlined web interface using Flask and JavaScript. The model was integrated into a fully functional prototype that classifies transactions in real time and delivers immediate, interpretable results to the user.

Evaluation of the system demonstrated a high level of accuracy and responsiveness, validating its capability to operate effectively in real-world scenarios. The system not only classifies transactions but also provides transparency in its processing, making it a reliable tool for financial institutions.

As cyber threats continue to grow in complexity, the system we've developed stands as a strong foundation that can be further enhanced with ensemble methods, deep learning techniques, and dynamic feedback mechanisms. Future iterations can also integrate continuous learning from new fraud patterns and extend compatibility with mobile and cloud-based infrastructures.

Ultimately, our real-time fraud detection model bridges the gap between speed and accuracy, offering a practical solution to help financial institutions detect and mitigate fraud swiftly and effectively.

CHAPTER 9

FUTURE ENHANCEMENT

While our Real-Time Fraud Detection System demonstrates strong performance using supervised learning on transactional features, there are several avenues for future enhancement to improve its robustness, scalability, and adaptability in real-world financial environments.

In practical deployment, fraud patterns are dynamic and can evolve quickly to evade detection by static models. Therefore, incorporating **continuous learning**

mechanisms—where the model can retrain itself periodically with new data—will significantly increase its long-term effectiveness. This would allow the system to adapt to novel fraud tactics without manual intervention.

Another promising direction is the integration of **ensemble learning techniques** such as Random Forest, XGBoost, or hybrid models combining logistic regression with neural networks. These can provide better generalization and improved accuracy, especially when dealing with highly imbalanced datasets where fraud cases are rare.

Furthermore, expanding the **feature set** beyond the current five metrics could offer deeper insights. Features like device fingerprints, geolocation, session behavior, or transaction frequency over time could greatly enhance prediction quality.

To address interpretability, future iterations can include **explainable AI (XAI)** components that justify the model's decisions in a human-readable format. This transparency is particularly important for high-stakes domains like banking and finance.

Integration with **blockchain** or **secure federated learning systems** can also be explored to enhance data privacy, especially when collaborating across banks without compromising sensitive user information.

Lastly, implementing **real-time alerts, visual analytics dashboards,** and **mobile integration** for fraud detection notifications can make the system more user-friendly and actionable for both administrators and end users.

In conclusion, while the current system achieves strong initial results, the path ahead includes adopting adaptive learning strategies, advanced modeling techniques, richer feature engineering, and scalable integration approaches. These enhancements will ensure the system remains resilient and effective in the ever-evolving landscape of financial cyber threats.

REFERENCES

[1] **EVOASTRA** : evoastra – Innovation that flows

[2]**Scikit-learn'sLogisticRegression:** :

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3]**SupervisedLearninginFraudDetection:** :

https://www.sciencedirect.com/science/article/pii/S2772662223000036

[4]**Flask Web Framework:** https://www.geeksforgeeks.org/flask-tutorial