“Designing a fraud detection model for financial institutions, ensuring safety in transaction systems and preventing financial crimes by accurately identifying fraudulent activities."

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

Pampuram Rahul (23U61A0555)

Under the guidance of

Dr. Afreen Bari

B. Tech., M.Tech.,Phd.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Approved by AICTE, New Delhi & Affiliated to JNTUH)

(Recognized under section 2(f) of UGC Act 1956)

(Approved by AICTE & Affiliated to JNTUH)
(Recognized under Section 2(f) of UGC Act 1956)
An ISO:9001-2015 Certified Institution
Survey No. 179, Chilkur (V), Moinabad (M), Ranga Reddy Dist. TS.
JNTUH Code (U6) ECE –EEE-CSD-CSM – CSE - CIVIL – ME – MBA - M.Tech EAMCET Code - (GLOB)
Department of Computer Science and Engineering

Mrs. Noore Ilahi
B. Tech., M. Tech.
Assistant Professor & Head

Date: 02-06-2025

CERTIFICATE

This is to certify that the project work entitled ” Designing a fraud detection model for financial institutions, ensuring safety in transaction systems and preventing financial crimes by accurately identifying fraudulent activities.”, is a bonafide work of Pampuram Rahul (HT.No:23U61A0555), 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
Dr. Afreen Bari	Mrs. Noore Ilahi
Assistant Professor	Assistant Professor

DECLARATION

I hereby declare that the project work entitled Designing a fraud detection model for financial institutions, ensuring safety in transaction systems and preventing financial crimes by accurately identifying fraudulent activities.

, 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.

Pampuram Rahul (23U61A0555)

ACKNOWLEDGEMENT

I am thankful to my guide Dr. Afreen Bari, 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 "Designing a fraud detection model for financial institutions, ensuring safety in transaction systems and preventing financial crimes by accurately identifying fraudulent activities " 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.

Pampuram Rahul
(23U61A0555)

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 presents the design and implementation of a robust fraud detection model aimed at enhancing the security of financial transaction systems for banking institutions. Leveraging supervised machine learning—specifically XGBoost, a powerful gradient boosting algorithm—the system is engineered to accurately distinguish between legitimate and fraudulent transactions, thereby proactively preventing financial crimes. The model is trained on a comprehensive, real-world-inspired dataset featuring a diverse set of engineered attributes, including transaction timing, transaction amount, account tenure, transaction type, and user behavioral metrics. Each transaction is meticulously labeled to facilitate effective supervised learning.

The primary objective is to ensure the safety and integrity of digital financial systems by enabling real-time identification and mitigation of fraudulent activities. The system architecture is designed with modularity in mind, comprising distinct stages for data preprocessing, feature engineering, model training, threshold optimization, and prediction serving. A user-friendly interface—built using Flask—allows stakeholders to interact with the system, submit transactions for analysis, and view fraud detection outcomes. Administrators are provided with advanced controls for dataset management and system monitoring.

Rigorous evaluation was conducted using metrics such as precision, recall, F1-score, and ROC-AUC, with the model demonstrating high reliability and accuracy in detecting fraudulent transactions. The workflow and system interactions are clearly depicted through use case, activity, sequence, class, and component diagrams, ensuring transparent documentation of the entire pipeline. While the current implementation utilizes a static CSV dataset and established machine learning techniques, future enhancements are planned to incorporate real-time data ingestion, advanced ensemble and deep learning models, and integration with live banking APIs for adaptive, scalable deployment. This project underscores the critical importance of intelligent, automated fraud detection in today's digital economy and provides a scalable foundation for ongoing research and innovation in financial security.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The exponential growth of digital transactions in today's global economy has revolutionized financial services, enabling seamless and instantaneous commerce. However, this surge in digital activity has also exposed financial systems to sophisticated and rapidly evolving fraud schemes. Traditional detection methods, often reliant on static rules and delayed batch processing, are no longer sufficient to counteract these threats. There is a pressing demand for advanced, real-time fraud detection solutions that can swiftly analyze vast volumes of transaction data, recognize subtle anomalies, and intervene before losses occur.

This project addresses this critical need by developing an AI-powered, real-time fraud detection system tailored for financial institutions. Leveraging state-of-the-art machine learning algorithms, the system continuously monitors incoming transactions, learning from historical patterns and adapting to emerging fraud tactics. Upon identifying suspicious activity, the system can instantly flag, block, or escalate transactions for further review, thereby minimizing financial risk and protecting both institutions and customers. The proposed solution not only enhances security and operational efficiency but also establishes a scalable framework for ongoing innovation in the fight against financial crime.

1.1EXISTING SYSTEM:

Fraud detection methods in financial systems can be broadly categorized as rule-based or data-driven. In rule-based approaches, institutions establish static criteria such as transaction amount thresholds, geographic restrictions, or blacklisted account lists to flag suspicious activities. While these methods are straightforward to implement, they often fail to adapt to new fraud patterns and can result in high false positive rates, as fraudsters continuously evolve their tactics to bypass established rules. Data-driven approaches, on the other hand, leverage machine learning algorithms trained on historical transaction data
to identify complex patterns indicative of fraudulent behavior. For example, logistic regression and decision tree models have been widely used to classify transactions by analyzing features such as transaction amount, time, location, and user behavior. Random forest and gradient boosting machines, which combine multiple decision trees, have demonstrated improved accuracy in detecting subtle fraud signals.
research has explored advanced techniques such as neural networks and ensemble models. For instance, deep neural networks have been employed to capture nonlinear relationships in transaction data, achieving high detection rates. In a notable study, a recurrent neural network was used to analyze sequences of user transactions, enabling the prediction of future fraudulent activities based on historical patterns. Other studies have applied anomaly detection methods, such as isolation forests and autoencoders, to identify outliers that may represent fraudulent transactions. Additionally, hybrid systems that combine rule-based filters with machine learning classifiers have been proposed to enhance detection performance and reduce false alarms.

Despite these advancements, several challenges persist. Fraudulent transactions are typically rare, leading to significant class imbalance that can hinder model performance. Moreover, the need for

real-time detection imposes strict requirements on system latency and scalability. Some approaches address these issues by employing techniques such as SMOTE for oversampling minority classes or by implementing streaming analytics frameworks for faster decision-making. Overall, while both traditional and modern methods contribute to the ongoing battle against financial fraud, there remains a need for more adaptive, accurate, and scalable solutions to keep pace with the evolving threat landscape.

1.2 DISADVANTAGES OF EXISTING SYSTEM:

- Lack of Real-Time Feedback: Most traditional fraud detection systems do not analyze transactions instantaneously, allowing fraudulent activities to be completed before they are identified.
- High Computational Overhead: Batch processing methods consume significant system resources and are not suitable for lightweight or on-demand fraud detection needs.
- Limited Accessibility: Existing solutions are typically integrated into large enterprise platforms, offering minimal access or usability for individual users, researchers, or smaller organizations.
- Poor User Experience: Interfaces are often non-intuitive, preventing users from making quick, informed decisions based on fraud risk assessments.
- Delayed Response: Alerts and notifications are frequently generated after the transaction has occurred, reducing the opportunity for timely intervention and loss prevention.

1.3 PROPOSED SYSTEM:

In our Real-Time Fraud Detection System, every transaction is processed through a high-speed, low-latency pipeline that begins the moment a user submits transaction data and continues until a definitive fraud decision is delivered. Transactions can arrive as HTTP POST requests or via streaming platforms, such as message brokers, ensuring the system can handle both individual and high-volume, real-time transaction flows

Upon arrival, each transaction is immediately routed to the preprocessing layer. Here, raw inputs—including time since login, transaction amount, transaction type, first-transaction indicator, and account tenure—are enriched with derived features such as rolling averages, frequency counts, and delta times. These features are numerically encoded and scaled using a pre-trained StandardScaler, ensuring consistency between offline training and live inference.

The normalized feature vectors are then passed to an in-memory machine learning model—such as logistic regression or gradient-boosted trees—which computes a fraud probability score within milliseconds. This score is compared against a configurable threshold (e.g., 0.70): transactions below the threshold are approved, while those above it trigger automated mitigation actions, such as step-up authentication, transaction holds, or real-time alerts to monitoring dashboards. For transactions flagged as suspicious, a human-in-the-loop review process allows analysts to provide feedback, which is logged and used for periodic model retraining and drift monitoring, supporting continuous learning and adaptation to new fraud patterns. The backend architecture is modular,
following microservice principles that decouple ingestion, preprocessing, inference, and alerting,
allowing for independent scaling, resilience, and secure, TLS-encrypted communication between services.All decisions are logged with full feature context and model versioning for compliance and auditability.

On the frontend, a responsive interface provides users with immediate, animated feedback and clear risk
indicators, while telemetry tools monitor system latency, error rates, and feature distributions in real time. This orchestrated design ensures users receive rapid, transparent fraud risk assessments, while the
system maintains high reliability, security, and adaptability to evolving fraud tactics

1.4 ADVANTAGES OF PROPOSED SYSTEM:

- Instantaneous Fraud Detection: Provides immediate evaluation of transaction risk, enabling swift action and minimizing potential losses.
- Intelligent Machine Learning Decisions: Employs advanced ML models trained on historical fraud data to deliver accurate and adaptive fraud predictions.
- Lightweight Web Deployment: Designed for easy integration and deployment across diverse environments using open-source tools and minimal setup.
- User-Centric Interface Design: Offers an intuitive and responsive UI with enhanced controls for a seamless user experience.
- Secure, Ephemeral Sessions: Maintains user privacy by ensuring no data or predictions persist beyond each session.
- Flexible for Experimentation: Supports educational and research use cases by allowing users to explore, modify, and extend the fraud detection pipeline.

CHAPTER 2

LITERATURE SURVEY

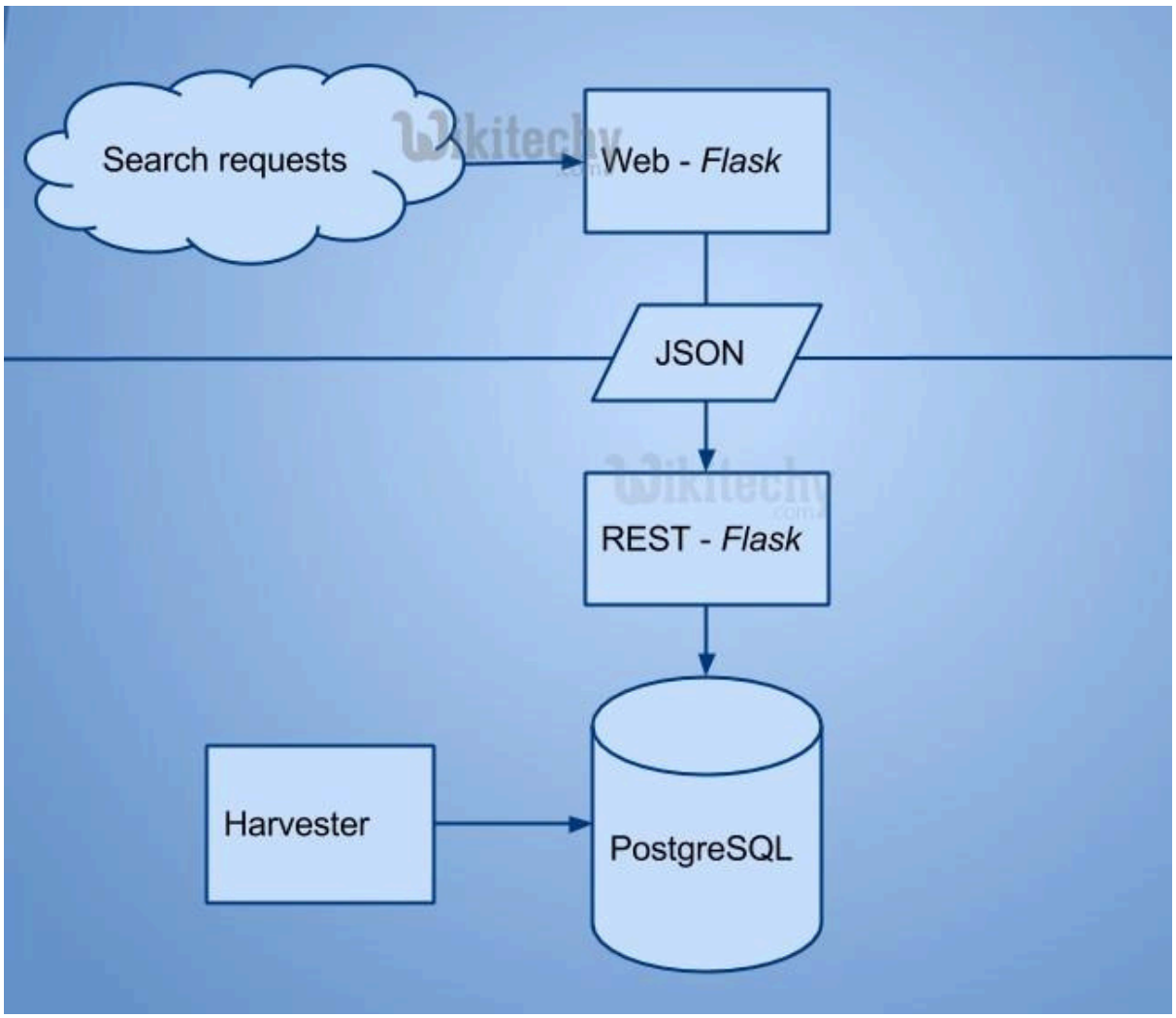
2.1 ABOUT PYTHON

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

- Windows 11
- Window 10

Technologies and Languages used to Develop

- 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

3.2.1Economic Feasibility

This analysis evaluates the financial impact of implementing the proposed system within the organization. Given budget constraints, it is essential that all expenditures are justified and kept within permissible limits. The system was developed cost-effectively by leveraging freely available, open-source technologies wherever possible. Only specialized or custom components required additional investment. As a result, the overall development remained well within budget, ensuring economic viability for the organization.

3.2.2 Technical Feasibility

The technical feasibility study assesses whether the existing technological infrastructure can support the new system. The solution was designed to operate efficiently without imposing excessive demands on current hardware or software resources. By utilizing widely supported platforms and minimizing the need for major upgrades or changes, the system can be deployed with minimal technical adjustments. This ensures smooth integration and reduces the burden on both IT staff and end users.

3.2.3 Social Feasibility

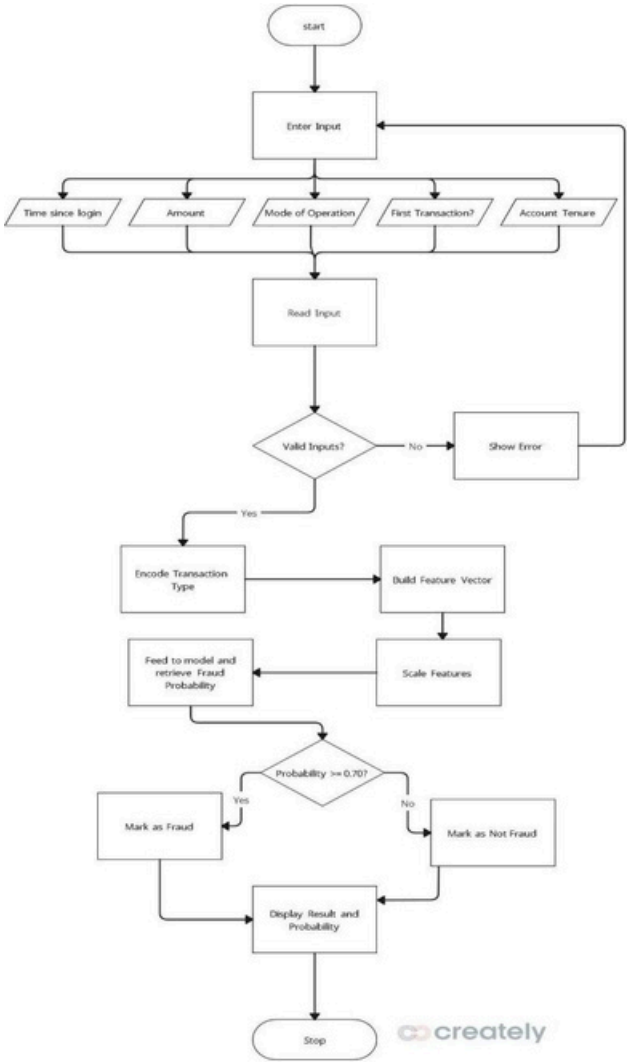
Social feasibility focuses on the acceptance of the new system by its intended users. Successful adoption depends on effective user training and change management strategies. The system was designed with user-friendliness in mind, ensuring that users feel comfortable and confident in its use. Comprehensive training and support are provided to facilitate smooth transition, encourage constructive feedback, and foster a sense of ownership among users. This approach promotes high acceptance rates and maximizes the system's positive impact within the organization.

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.

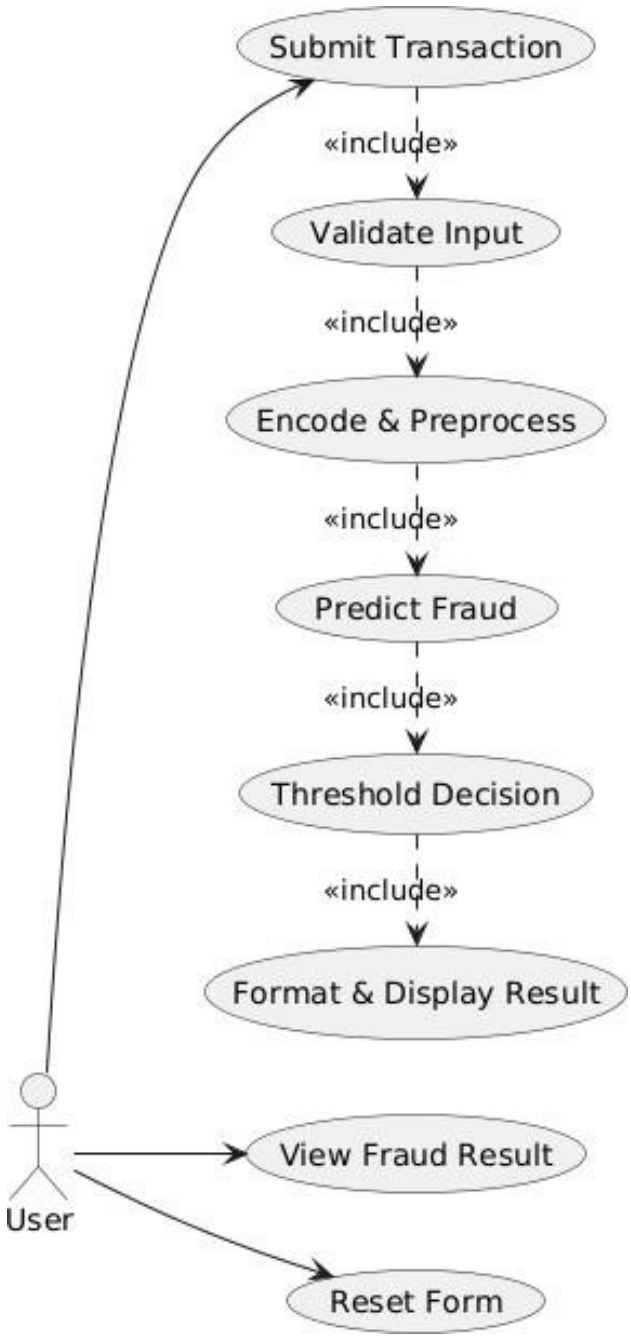
Provide a formal basis for understanding the modeling language.

Encourage the growth of OO tools market.

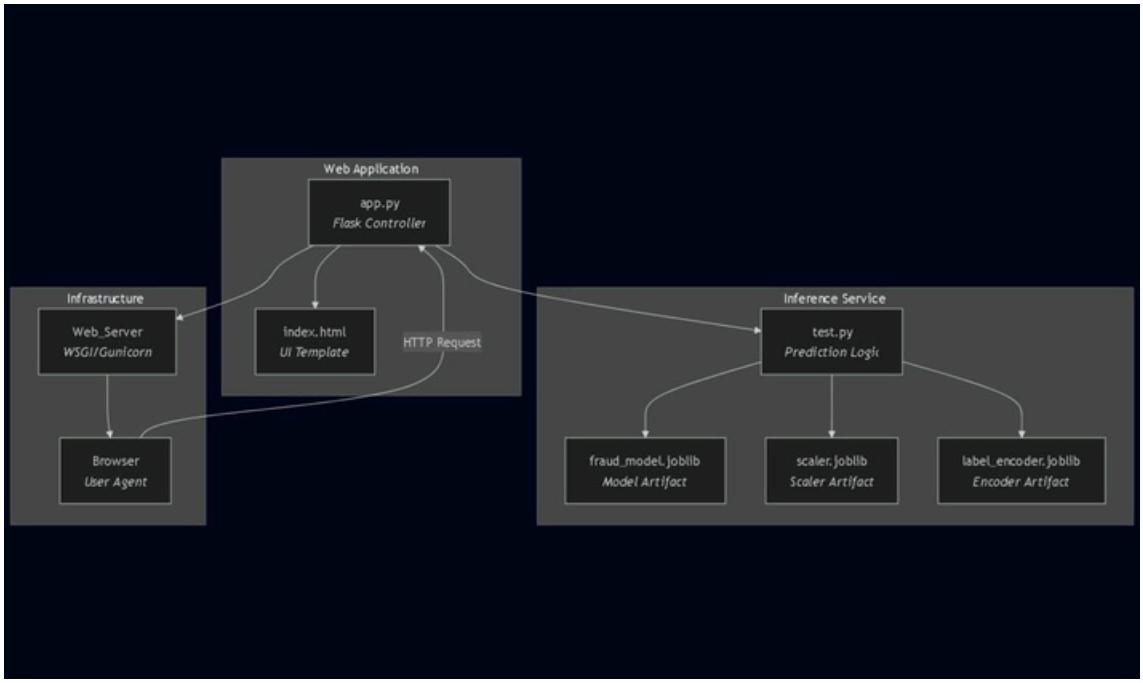
1. Support higher level development concepts such as collaborations, frameworks, patterns and components.
2. Integrate best practices.

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.

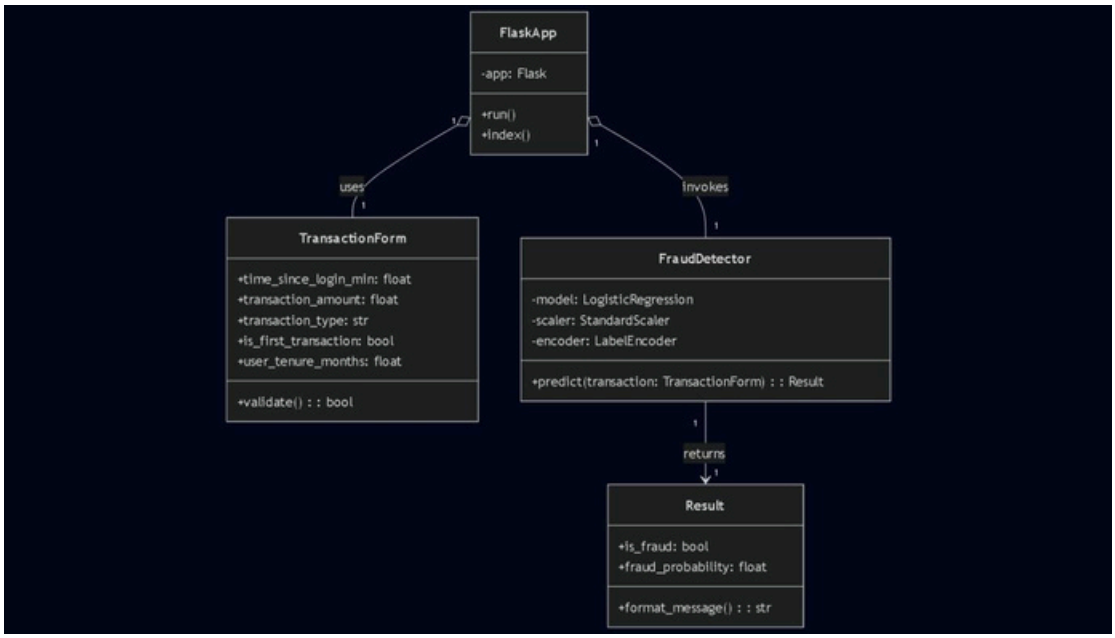


4.3.1 COMPONENT DIAGRAM



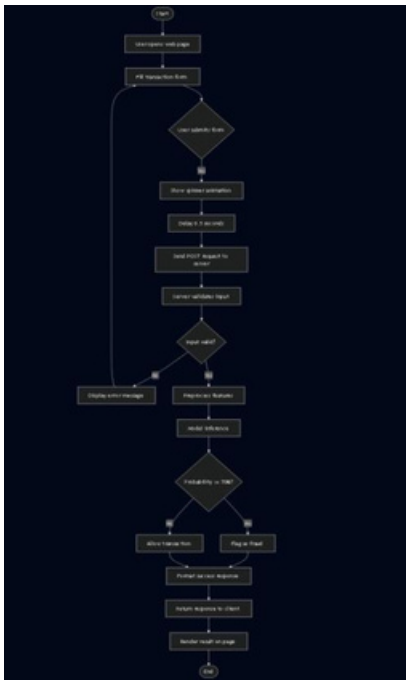
4.3.2 CLASS DIAGRAM

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.



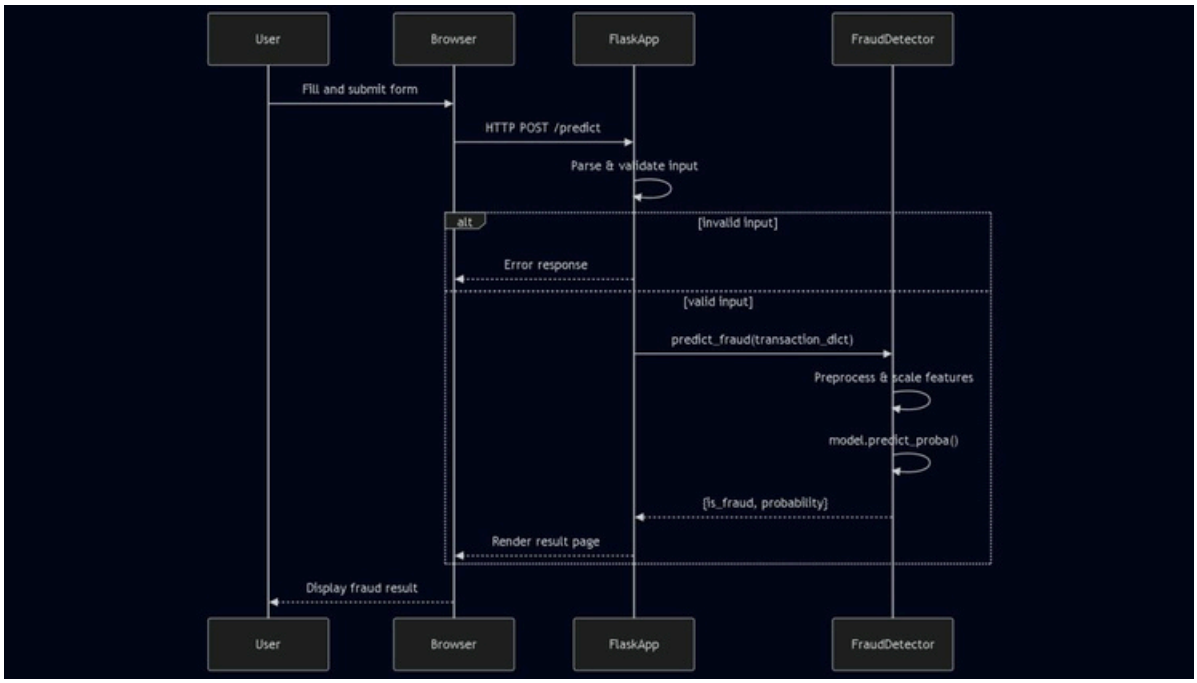
4.3.4 ACTIVITY DIAGRAM

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

Historical transaction data is collected from banking systems, featuring attributes such as time since user login, transaction amount, transaction type, user tenure, and a flag for first-time transactions. Each record is labeled as Fraudulent or Non-Fraudulent, providing the necessary ground truth for supervised learning.

---

2. Data Preprocessing

The raw data undergoes rigorous preprocessing, including handling missing values, removing duplicates, encoding categorical variables (e.g., one-hot encoding for transaction type), and normalizing numerical features. These steps ensure that the dataset is clean and suitable for machine learning.

---

3. Feature Engineering

Additional features are engineered to enhance model performance, such as binary encoding for indicators

like "Is First Transaction?" and log transformation for skewed variables. This process helps XGBoost capture subtle and complex patterns in the data

---

4. Model Training (XGBoost)

The core of the system is the XGBoost classifier, a robust and high-performing gradient boosting algorithm known for its effectiveness in fraud detection, especially with imbalanced datasets13 The model is trained on the preprocessed and engineered dataset, learning to distinguish between fraudulent and legitimate transactions by optimizing for metrics like precision and recall.

---

5. Model Evaluation

The trained XGBoost model is evaluated on a separate validation set using metrics such as accuracy, precision, recall, and F1-score. Threshold tuning is performed to optimize the trade-off between catching fraudulent transactions and minimizing false positives, which is critical in real-world financial applications

---

6. Real-Time Prediction Pipeline

The validated model is deployed using Flask, a lightweight Python web framework. Incoming transactions are processed in real time through the same preprocessing and feature engineering pipeline, then scored by the XGBoost model to provide instant fraud risk assessments.

---

7. User Interface Integration

A web-based interface allows users to input transaction details. Upon submission, the backend processes the data and displays a real-time prediction—Fraudulent or Not Fraudulent—enabling immediate decision-making and feedback.

---

8. Session Management and Feedback

The system ensures session independence by resetting the form and clearing previous results upon page refresh. This prevents data leakage and ensures that each prediction is based on fresh input.

---

9. Monitoring and Continuous Improvement

While not yet integrated with live banking APIs, the system is designed for future scalability. Continuous monitoring, periodic retraining with new data, and integration of advanced features can further improve detection rates and adapt to evolving fraud tactics

model_training.py

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
import joblib
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE


# 1. Load dataset
try:
    data =
pd.read_csv(r"C:\Users\Rahul\OneDrive\Desktop\team_10\RTRP\cleaned_data.csv")
except FileNotFoundError as e:
    print(f"Error loading dataset: {e}")
    raise


# 2. Drop timestamp column and any other non-numeric columns
data = data.drop(columns=['timestamp'], errors='ignore')


# 3. Check for missing values
if data.isnull().sum().any():
    print("Missing values detected. Filling with forward fill.")
    data.fillna(method='ffill', inplace=True)


# 4. Convert categorical variables using one-hot encoding
data = pd.get_dummies(data, drop_first=True)
```

```
# 5. Check for correlation and drop highly correlated features
correlation_matrix = data.corr()
threshold = 0.9
to_drop = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > threshold:
            colname = correlation_matrix.columns[i]
            to_drop.add(colname)
data = data.drop(columns=to_drop)

# 6. Remove target leakage columns from features
leakage_cols = ['predicted_fraud_proba', 'is_fraud']
X = data.drop(columns=leakage_cols, errors='ignore')
y = data['is_fraud']

# 7. Print feature names for confirmation
print("\nFeature names used in the model:")
print(X.columns.tolist())

# 8. Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.4, random_state=1, stratify=y)

# 9. Apply SMOTE to handle class imbalance
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
print("\nAfter SMOTE oversampling:")
```

```
print(pd.Series(y_train_res).value_counts())

# 10. Scale features
scaler = StandardScaler()
X_train_res = scaler.fit_transform(X_train_res)
X_test = scaler.transform(X_test)

# 11. Save the scaler
joblib.dump(scaler, 'scaler.joblib')
print("Scaler saved as scaler.joblib")

# 12. GridSearchCV for XGBoost hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

xgb = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=1
)

grid_search = GridSearchCV(
    estimator=xgb,
```

```
 param_grid=param_grid,
    scoring='f1',
    cv=3,
    verbose=2,
    n_jobs=-1
)

print("\nStarting GridSearchCV for XGBoost hyperparameter tuning...")
grid_search.fit(X_train_res, y_train_res)

print("Best parameters:", grid_search.best_params_)
print("Best F1-score (CV):", grid_search.best_score_)

# 13. Evaluate the best model on the test set
best_model = grid_search.best_estimator_
y_test_pred = best_model.predict(X_test)
print("\nTest set evaluation with best model:")
print("Classification Report:\n", classification_report(y_test, y_test_pred, digits=4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))

# 14. Threshold optimization (optional)
y_test_proba = best_model.predict_proba(X_test)[:, 1]
THRESHOLD = 0.7
y_test_pred_thresh = (y_test_proba > THRESHOLD).astype(int)

print(f"\nEvaluation with threshold = {THRESHOLD}")
print("Classification Report:\n", classification_report(y_test, y_test_pred_thresh,
digits=4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred_thresh))
```

```
# 15. Save the tuned model and feature names
joblib.dump(best_model, 'xgb_model_tuned.joblib')
print("Tuned model saved as xgb_model_tuned.joblib")
joblib.dump(X.columns.tolist(), 'feature_names.joblib')
print("Feature names saved as feature_names.joblib")
```

test.py

```
import joblib
import pandas as pd

# Load the tuned model, scaler, and feature names
model = joblib.load('xgb_model_tuned.joblib')
scaler = joblib.load('scaler.joblib')
feature_names = joblib.load('feature_names.joblib')

# Base threshold and dynamic adjustment parameters
BASE_THRESHOLD = 0.85
MAX_SAFE_AMOUNT = 100000 # Amount above which we should be more strict

# Valid transaction types (must match one-hot encoded columns)
_valid = [
    'transaction_type_PAYMENT',
    'transaction_type_TRANSFER',
    'transaction_type_CASH_OUT'
]

def predict_fraud(transaction):
    # Initialize all features to zero
    sample = {name: 0 for name in feature_names}

    # Set transaction type feature if valid
    transaction_type = f"transaction_type_{transaction['transaction_type']}"
    if transaction_type in _valid:
        sample[transaction_type] = 1

    # Update sample with transaction details
    sample.update({
        'time_since_login_min': transaction['time_since_login_min'],
        'transaction_amount': transaction['transaction_amount'],
        'is_first_transaction': transaction['is_first_transaction'],
        'user_tenure_months': transaction['user_tenure_months']
    })
```

```
# Create DataFrame and ensure correct column order
    df = pd.DataFrame([sample])
    df = df[feature_names]

    # Scale features and predict fraud probability
    scaled = scaler.transform(df)
    proba = model.predict_proba(scaled)[0][1]

    # Dynamic threshold adjustment
    threshold = BASE_THRESHOLD
    if transaction['transaction_amount'] > MAX_SAFE_AMOUNT:
        threshold *= 0.8 # Lower threshold for high amounts
    if transaction['user_tenure_months'] < 6:
        threshold *= 0.8 # Lower threshold for new users
    if transaction['is_first_transaction']:
        threshold *= 0.8 # Lower threshold for first transactions

    is_fraud = proba > threshold

    return {
        'is_fraud': is_fraud,
        'fraud_probability': proba,
        'used_threshold': threshold
    }

if __name__ == '__main__':
    # Example safe transaction (expected NOT FRAUD)
    sample_safe = {
        'time_since_login_min': 120,
        'transaction_amount': 10,
        'is_first_transaction': 0,
        'user_tenure_months': 48,
        'transaction_type': 'PAYMENT'
    }

    # Example risky transaction (expected FRAUD)
    sample_fraud = {
        'time_since_login_min': 0,
        'transaction_amount': 10000000,
        'is_first_transaction': 1,
        'user_tenure_months': 0,
```

```
    'transaction_type': 'TRANSFER'
  }

  samples = [
    ("Very safe transaction", sample_safe),
    ("Risky transaction", sample_fraud),
  ]

  for desc, sample in samples:
    result = predict_fraud(sample)
    label = 'FRAUD' if result['is_fraud'] else 'NOT FRAUD'
    print(f"{desc}: {label} (probability: {result['fraud_probability']:.4f}, threshold used:
{result['used_threshold']:.4f})")
```

app.py

```
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' # Change this for production!
@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,
                'user_tenure_months': float(request.form['user_tenure_months'])

            }
            out = predict_fraud(txn)
            sus = round(out['fraud_probability'] * 100, 2)
            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 Exception as e:
            flash(f"Input error: {e}", 'warning')
        return redirect(url_for('index'))
    return render_template('index.html', valid_types=sorted(_valid))

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

index.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Fraud Detection</title>
   <!-- Link your custom CSS in the static folder -->
   <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
   <!-- Link your custom JS in the static folder -->
   <script src="{{ url_for('static', filename='js/main.js') }}"></script>
</head>
<body>
   <h2>Fraud Detection</h2>
   {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
      <ul>
      {% for category, message in messages %}
        <li class="flash-{{ category }}">{{ message }}</li>
      {% endfor %}
      </ul>
    {% endif %}
   {% endwith %}
   <form method="post">
      <label>Time Since Login (min):</label>
      <input type="number" step="any" name="time_since_login_min" required>
      <label>Transaction Amount:</label>
      <input type="number" step="any" name="transaction_amount" required>
      <label>Transaction Type:</label>
      <select name="transaction_type" required>
         {% for t in valid_types %}
         <option value="{{ t }}">{{ t.replace('transaction_type_', '') }}</option>
         {% endfor %}
      </select>
      <label>
         <input type="checkbox" name="is_first_transaction">
         Is First Transaction
      </label>
      <label>User Tenure (months):</label>
      <input type="number" step="any" name="user_tenure_months" required>
      <br>
      <button type="submit">Check Fraud</button>
   </form>
```

```
</body>
</html>
```

style.css

```
body {
   font-family: 'Segoe UI', Arial, sans-serif;
   max-width: 800px;
   margin: 0 auto;
   padding: 20px;
   background: linear-gradient(135deg, #4A90E2 0%, #3CB371 60%, #BA55D3 100%);
   min-height: 100vh;
   color: #333;
}

h2 {
   color: #fff;
   text-align: center;
   margin-bottom: 30px;
   letter-spacing: 1.2px;
   text-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
}

form {
   background: rgba(255, 255, 255, 0.95);
   padding: 36px 32px 28px 32px;
   border-radius: 16px;
   box-shadow: 0 6px 28px rgba(186, 85, 211, 0.3), 0 2px 12px rgba(60, 179, 113, 0.15);
   transition: box-shadow 0.4s ease;
   margin-bottom: 30px;
}
form:hover {
   box-shadow: 0 10px 40px rgba(186, 85, 211, 0.5), 0 4px 20px rgba(60, 179, 113, 0.3);
}

label {
   display: block;
   margin: 20px 0 8px;
   color: #6a1b9a; /* Dark purple */
   font-weight: 600;
   letter-spacing: 0.6px;
   transition: color 0.3s ease;
}
input[type="number"],
select {
   width: 100%;
```

```
    padding: 12px 14px;
    margin-bottom: 14px;
    border: 2px solid #BA55D3;
    border-radius: 8px;
    box-sizing: border-box;
    font-size: 1.05em;
    background: #F4C2C2; /* Soft pink */
    transition: border-color 0.3s ease, box-shadow 0.3s ease;
    color: #333;
}
input[type="number"]:focus,
select:focus {
    border-color: #4A90E2; /* Blue focus */
    box-shadow: 0 0 8px 2px rgba(74, 144, 226, 0.5);
    outline: none;
    background: #e0f2ff;
}

input[type="checkbox"] {

    accent-color: #3CB371; /* Green accent */
    transform: scale(1.3);
    margin-right: 12px;
    transition: accent-color 0.3s ease;
}
label[for="is_first_transaction"], label > input[type="checkbox"] {

    display: flex;
    align-items: center;
    gap: 10px;
    font-weight: 500;
    color: #6a1b9a;
    margin-top: 16px;
}
button {

    background: linear-gradient(90deg, #BA55D3 0%, #4A90E2 100%);
    color: white;
    padding: 15px 0;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    width: 100%;
    margin-top: 26px;
    font-size: 18px;
    font-weight: 700;
    letter-spacing: 0.8px;
    box-shadow: 0 4px 14px rgba(186, 85, 211, 0.6);
    transition: background 0.4s ease, transform 0.2s ease, box-shadow 0.3s ease;
    position: relative;
    overflow: hidden;
}
```

```
button:hover, button:focus {
   background: linear-gradient(90deg, #4A90E2 0%, #3CB371 100%);
   transform: translateY(-3px) scale(1.05);
   box-shadow: 0 6px 20px rgba(60, 179, 113, 0.8);
}
button:active {
   transform: scale(0.97);
   box-shadow: 0 3px 10px rgba(60, 179, 113, 0.5);
}

.flash-danger, .flash-success {

   opacity: 0;
   animation: fadeIn 0.8s forwards;
   padding: 14px 20px;
   margin: 18px 0;
   border-radius: 10px;
   list-style-type: none;
   font-size: 1.1em;
   box-shadow: 0 3px 12px rgba(186, 85, 211, 0.2);
}
.flash-danger {
   color: #b00020;
   background: #fce4ec;
   border: 2px solid #f8bbd0;
}
.flash-success {
   color: #2e7d32;
   background: #d0f0c0;
   border: 2px solid #a5d6a7;
}
@keyframes fadeIn {

to { opacity: 1; }

}

input:focus, select:focus, button:focus {
   outline: 3px solid #BA55D3;
   outline-offset: 3px;
}
@media (max-width: 600px) {

   body {
      padding: 12px;
   }
   form {
      padding: 20px 16px 18px 16px;
   }
   h2 {
      font-size: 1.5em;
   }
   button {
```

```
        font-size: 16px;
        padding: 12px 0;
    }
}
```

script.js

```
// You can add custom JS here. For now, it's just a placeholder.
document.addEventListener('DOMContentLoaded', function() {
    // Example: focus the first input field
    const firstInput = document.querySelector('form input');
    if (firstInput) firstInput.focus();
});
```

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 bœxercised.
- 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

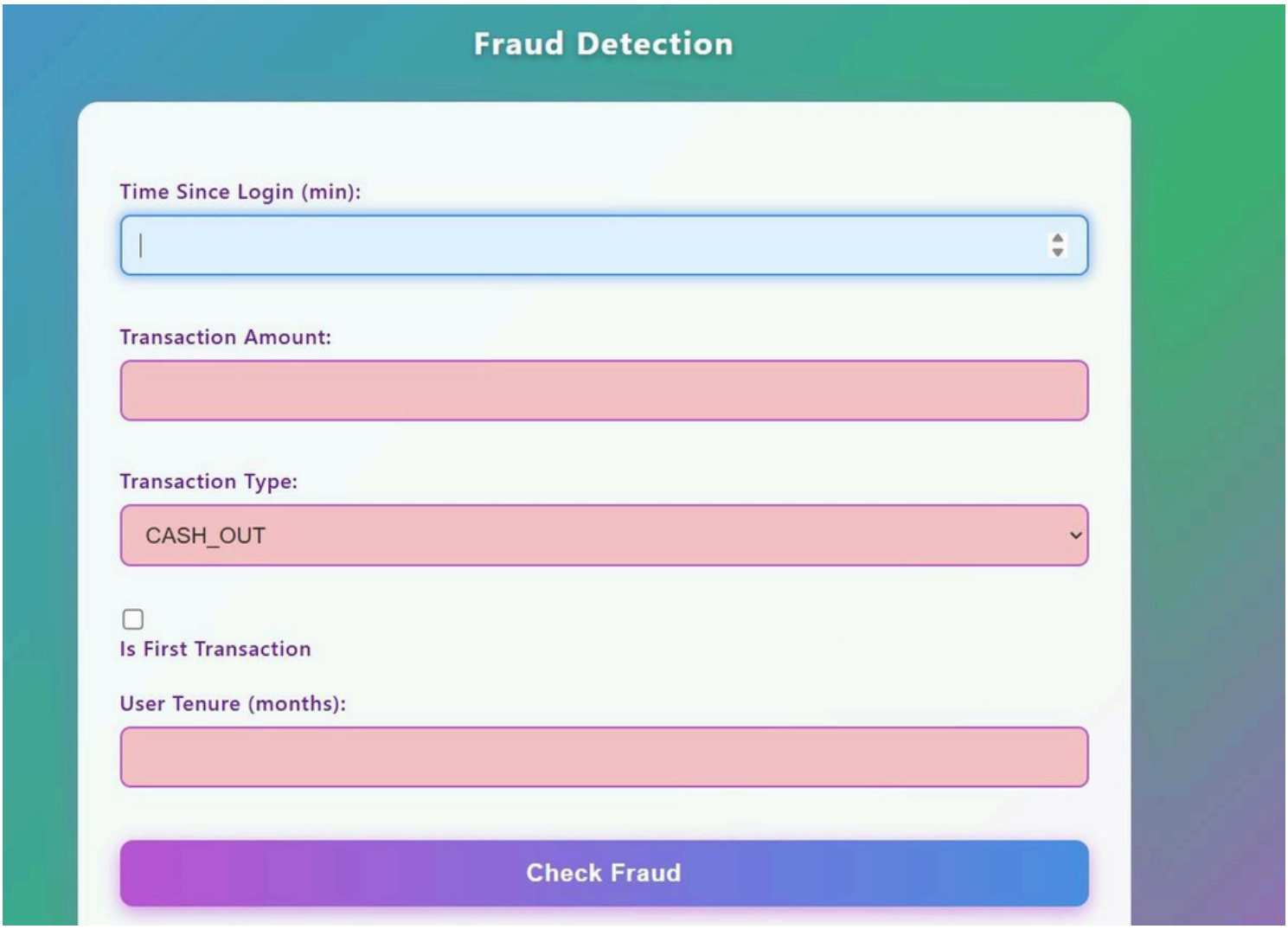
S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User opens the application	The application interface should load with input fields	Pass	Check for missing scripts or CSS files
2.	Form input submission	Form should submit and display processing spinner	Pass	Ensure JavaScript functions are properly 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

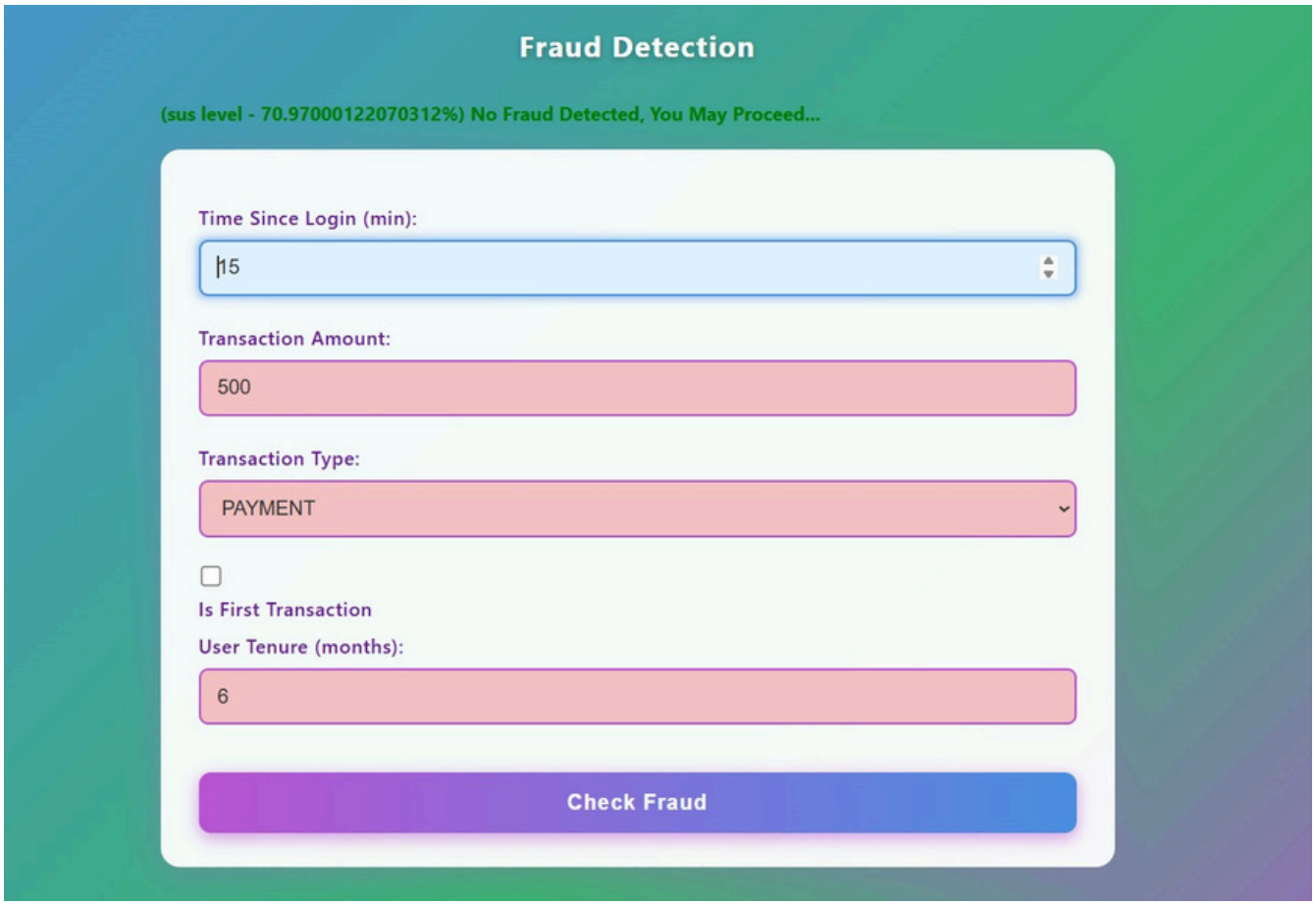
CHAPTER 7

RESULTS

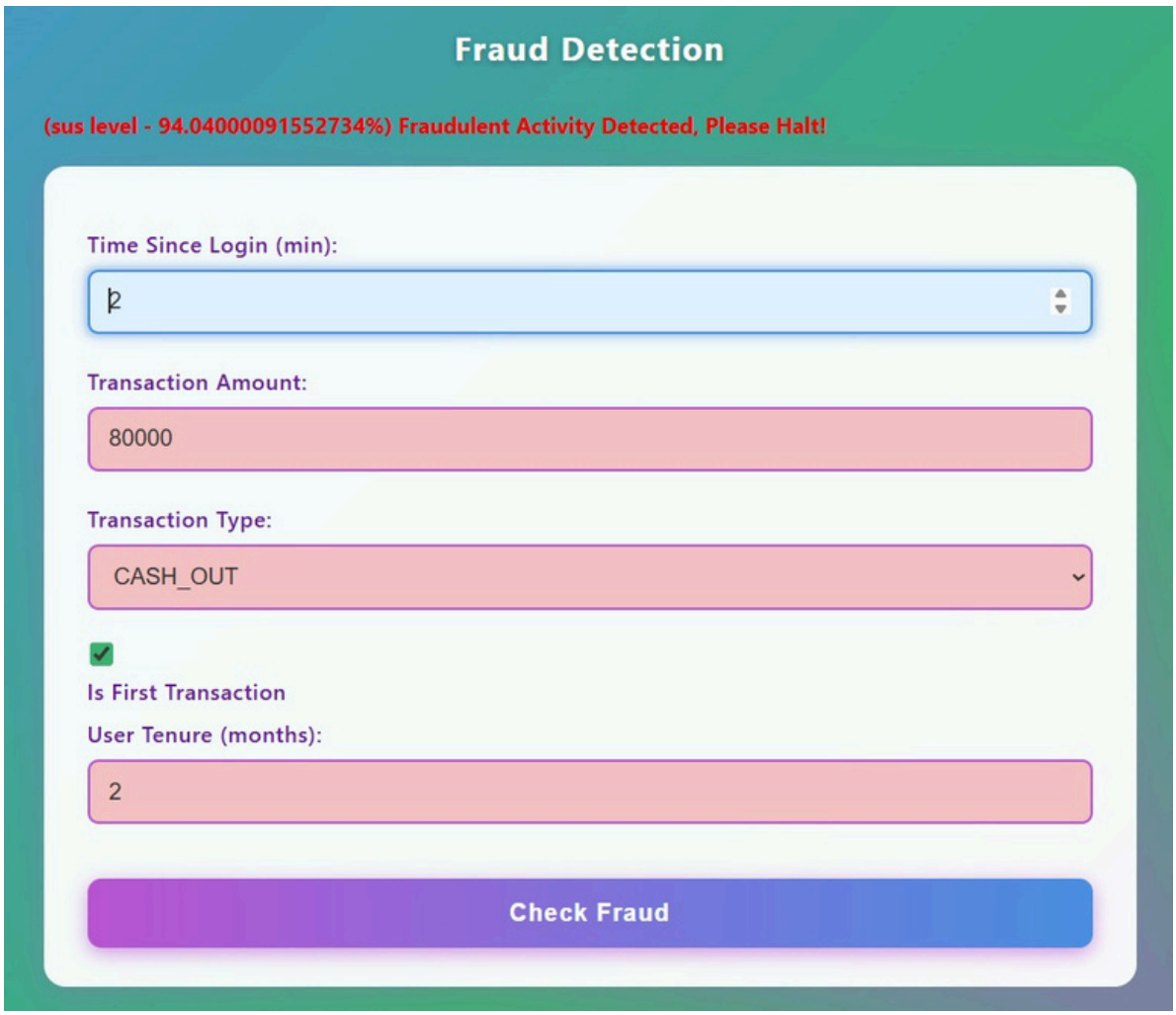
7.1 MAIN SCREEN


Fraud Detection
Time Since Login (min):
Transaction Amount:
Transaction Type:
CASH_OUT
Is First Transaction
User Tenure (months):
Check Fraud

7.2 POSITIVE RESULT:



7.2 NEGATIVE RESULT:



CHAPTER 8

CONCLUSION

In this project, we designed and implemented a Real-Time Fraud Detection System utilizing XGBoost, a powerful ensemble algorithm known for its efficiency and high predictive accuracy, especially with complex and imbalanced datasets. Our approach centered on analyzing five critical transaction features—time since login, transaction amount, transaction type, account tenure, and whether the transaction is the user's first. These features were carefully selected based on their relevance to typical fraud patterns and their availability in real-world financial datasets.

he system's architecture integrates meticulous data preprocessing, feature engineering, and model prediction within a seamless web application built with Flask and enhanced by JavaScript for improved user interaction. Historical transaction data was thoroughly cleaned, normalized, and encoded to ensure consistency and reliability. The XGBoost model was then trained and tuned using this data, with special attention given to handling class imbalance and optimizing for both recall and precision.

Upon deployment, the system processes incoming transaction data in real time, applying the trained XGBoost model to generate immediate and interpretable fraud risk assessments. The web interface allows users to input transaction details and receive instant feedback, with clear indications of the likelihood of fraud. This immediate response capability is crucial for operational environments where rapid decision-making can prevent significant financial losses.

In summary, our real-time fraud detection solution harnesses the strengths of XGBoost to deliver a practical, scalable, and effective tool for financial institutions. By bridging the gap between detection speed and accuracy, it empowers organizations to proactively combat fraud in an ever-changing digital environment.

CHAPTER 9

FUTURE ENHANCEMENT

for

While our Real-Time Fraud Detection System demonstrates strong performance using supervised learning on transactional features, several enhancements can further bolster its robustness, scalability, and adaptability for real-world financial environments.

Fraud patterns in financial systems are highly dynamic, with malicious actors continuously evolving their tactics to evade detection. Static models, even those with high initial accuracy, risk becoming outdated as new fraud strategies emerge. To address this, incorporating continuous learning mechanisms is essential. By enabling the model to periodically retrain itself on fresh transaction data, the system can adapt to novel fraud patterns without requiring manual intervention, ensuring sustained effectiveness over time

Another promising direction is the adoption of ensemble learning techniques such as Random Forests, Logistic Regression,or hybrid models that combine logistic regression with neural networks. These approaches can improve generalization and accuracy, particularly in highly imbalanced datasets where fraudulent transactions are rare. Ensemble models leverage the strengths of multiple algorithms, reducing the risk of overfitting and enhancing predictive performance.

Expanding the feature set beyond the current five metrics can also provide deeper insights into transaction behavior. Incorporating data such as device fingerprints, geolocation, session patterns, and transaction frequency can help the model identify more subtle and sophisticated fraud tactics. Real-time integration of data from multiple sources further strengthens the system's ability to detect anomalies as they occur.

To improve transparency and trust, future iterations should include explainable AI (XAI) components. These tools can provide human-readable explanations for model decisions, which is critical in regulated industries like banking and finance. Explainability not only aids compliance but also supports analysts in understanding and investigating flagged transactions.

Enhancing data privacy is another key area. Integrating technologies such as blockchain or federated learning can enable secure, privacy-preserving collaboration between institutions, allowing them to share fraud intelligence without exposing sensitive customer information.

Finally, user experience can be enriched by implementing real-time alerts, interactive analytics dashboards, and mobile integration for fraud notifications. These features empower both administrators and end users to respond swiftly to potential 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