Software Requirements Specification (SRS)

Project Name: Fraud Detection System for Financial Transactions
Author: SANKET RAVINDRAKUMAR SHRIVASTAV
Date: 03-Feb-2026
Version: 1.0

---

1. Introduction

1.1 Purpose
The purpose of this system is to provide a **real-time fraud detection mechanism** for financial transactions. It uses **machine learning** to identify unusual or suspicious transaction patterns and flag them for review. This helps banks, payment platforms, and financial institutions reduce financial risks.

1.2 Scope

- Web-based application using Python and Flask.
- Users can input transaction details (amount, location, account type, etc.).
- The system predicts whether a transaction is **fraudulent** or **legitimate**.
- The system is **extensible**: it can be connected to a database or real-time transaction feed in the future.

1.3 Intended Audience

- Banks and financial institutions.
- Project reviewers on LMS.
- Developers and stakeholders interested in ML applications in finance.

1.4 Definitions / Acronyms

- **ML:** Machine Learning
- **CSV:** Comma-Separated Values
- **SRS:** Software Requirements Specification
- **API:** Application Programming Interface

---

2. Overall Description

2.1 Product Perspective

- This is a standalone web application.
- Can be extended to connect with **real-time banking APIs**.
- Uses **Python ML libraries** (scikit-learn, pandas, numpy).

2.2 Product Functions

1. Accept user input for transaction details.
2. Apply ML model to predict fraudulent transactions.
3. Display the prediction result instantly on the web page.
4. Optional: Export predictions to CSV (future extension).

2.3 User Classes and Characteristics

- **End User:** Bank employees, financial auditors.
- **Developer:** Anyone maintaining or enhancing the system.
- **Admin:** Can modify ML model or extend system functionality.

2.4 Operating Environment

- Windows 10/11, Linux, or Mac OS.
- Python 3.10+ environment.
- Local CMD or terminal to run Flask app.

2.5 Design and Implementation Constraints

- Must use Python and Flask.
- Must run in a virtual environment (venv).
- ML model must be pre-trained or trained on sample dataset.

---

3. System Features

Feature	Description	Priority
Transaction Input	User enters transaction amount, location, account type, etc.	High
Fraud Prediction	ML model predicts fraud / legitimate transactions	High
Result Display	Shows prediction instantly on the web page	High
Data Export	Save predictions to CSV (optional)	Medium
Security	No sensitive information stored	High

---

4. Functional Requirements

1. User can open a browser and access the web app.
2. User can enter transaction details via a web form.
3. ML model predicts fraud or legitimate status.
4. Prediction result is displayed immediately.
5. System should handle invalid inputs gracefully.

---

5. Non-Functional Requirements

Requirement	Description
Performance	Results displayed in under 2 seconds.
Reliability	System should not crash on normal inputs.
Scalability	Can handle multiple transaction inputs in the future.
Security	No sensitive data is stored; only simulated inputs.
Usability	Easy-to-use web interface.

---

6. Technical Requirements

- **Programming Language:** Python 3.10+
- **Framework:** Flask
- **Libraries:** pandas, scikit-learn, numpy, textblob (optional)
- **Web:** HTML, CSS for frontend
- **IDE:** CMD / VS Code / PyCharm
- **OS:** Windows/Linux

---

7. System Design Approach

1. **Frontend:**
    - HTML form for user input.
    - CSS for basic styling.
2. **Backend:**
    - Flask routes handle / (home page) and /predict.
    - ML model loaded in Python (scikit-learn) predicts the transaction.
3. **Data Flow:**
4. User input → Flask backend → ML model → Prediction → Display on webpage
5. **ML Model:**
    - Pre-trained Random Forest / Logistic Regression model (sample dataset).
    - Can be replaced with real banking data in future.

---

8. Testing Plan

- **Unit Testing:** Verify individual functions like ML prediction, input validation.
- **Integration Testing:** Ensure form input, prediction, and result display work together.
- **User Testing:** Test with sample transactions for correctness.
- **Edge Cases:** Empty input, extremely high values, unusual location.

---

9. Deliverables

1. Fully functional Flask app (app.py)
2. Templates and static files (index.html, style.css)
3. requirements.txt for dependencies
4. SRS document (this document)
5. README.md for GitHub / LinkedIn submission

---

10. References

- Flask Documentation: https://flask.palletsprojects.com/
- scikit-learn Documentation: https://scikit-learn.org/stable/
- pandas Documentation: https://pandas.pydata.org/

✅ Save this as **SRS.docx** or **SRS.pdf** in your project folder:

fraud_detection_system/

│

├── SRS.docx

├── app.py

├── requirements.txt

├── README.md

├── templates/

└── static/
