PROJECT SRS:

IEEE Software Requirements Specification (SRS) Document) for a Real-Time Credit Card Fraud Detection System

1. Introduction

1.1 Purpose

This document specifies the Software Requirements Specification (SRS) for a Real-Time Credit Card Fraud Detection System. The system aims to identify fraudulent credit card transactions in real-time using machine learning and rule-based detection methods.

1.2 Scope

This SRS covers the functional and non-functional requirements of the Real-Time Credit Card Fraud Detection System. It outlines the system's functionalities, performance expectations, security considerations, and reliability needs.

1.3 Intended Audience

This document is intended for stakeholders including system developers, testers, administrators, and any other personnel involved in the design, development, deployment, and maintenance of the Real-Time Fraud Detection System.

2. Overall Description

2.1 Product Perspective

The Real-Time Credit Card Fraud Detection System is a software application that integrates with existing payment gateway infrastructure. It receives real-time transaction data, analyzes it for anomalies, and flags potentially fraudulent transactions.

2.2 Product Functions

The core functionalities of the system include:

- Receiving real-time transaction data from the payment gateway securely.
- Employing machine learning algorithms to detect anomalies in transaction patterns.
- Implementing predefined rules to identify suspicious transactions based on specific criteria.
- Generating reports on detected fraudulent activities and associated risk levels.

2.3 User Characteristics

The primary users of the system will be fraud analysts and security personnel who monitor the system for suspicious activities. Additionally, the system may integrate with other applications within the organization's fraud management ecosystem.

2.4 General Constraints

- The system should be developed using industry-standard programming languages and development tools.
- The system should be compatible with existing IT infrastructure.
- Security protocols and user access controls should comply with relevant regulations and organizational policies.

2.5 Assumptions and Dependencies

- A secure connection with the payment gateway is established for real-time data transfer.
- Historical transaction data is available for training machine learning models.
- System administrators have the necessary permissions to configure rules and manage user access.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Real-time Transaction Monitoring

- The system shall establish a secure connection with the payment gateway to receive real-time transaction data.
- Data transmission shall adhere to industry-standard encryption protocols (e.g., PCI DSS).
- The system shall process each transaction within 100 milliseconds.
- Transactions shall be processed in chronological order based on their timestamps.

3.1.2 Anomaly Detection

- The system shall employ machine learning algorithms to detect anomalies in transaction patterns.
- Machine learning models shall be trained regularly with up-to-date data to improve accuracy.
- The system shall be capable of adapting to new fraud patterns through continuous learning.
- The system should consider factors such as transaction amount and time when identifying anomalies.
- Users should be able to modify the parameters that define an anomaly.

3.1.3 Rule-based Detection

- The system shall implement predefined rules to flag transactions based on specific criteria (e.g., transaction amount limits, location-based restrictions).
- System activities shall not be interrupted when applying rule modifications.
- Rules should be prioritized and executed in a logical order to prevent conflicts.

3.1.4 Reporting and Alerting

- The system shall generate reports on detected fraudulent activities.
- Reports shall be comprehensive, providing details on flagged transactions, associated risk levels, and supporting evidence.
- Real-time report availability and record-keeping are essential.

3.2 Non-Functional Requirements

3.2.1 Performance

- The system shall handle a minimum of 10,000 transactions per minute.
- The system should be scalable to handle higher transaction volumes during peak times.
- Performance testing shall be conducted regularly to ensure the system meets or exceeds benchmarks.
- Response time for detecting fraudulent activities shall not exceed 2 seconds.
- Monitoring tools should be in place to identify and address any performance bottlenecks.

3.2.2 Security

- Access to system and its data should be restricted to authorized personnel.
- Access to sensitive information, including transaction data and machine learning models, should be restricted based on user roles and permissions.
- The implementation of user authentication techniques, such as multi-factor authentication, is required.

3.2.3 Reliability

- The system shall have a minimum uptime of 99.9%.
- Regular maintenance activities should be planned to minimize downtime.
- In the case of hardware or software failures, redundancy and recovery methods should be in place to guarantee continuous functioning.
- Backup and recovery procedures should be in place to ensure data integrity.
- Regular backups of critical data should be performed and tested for reliability.
- Disaster recovery plans should be documented and periodically tested.

PROJECT DIAGRAMS

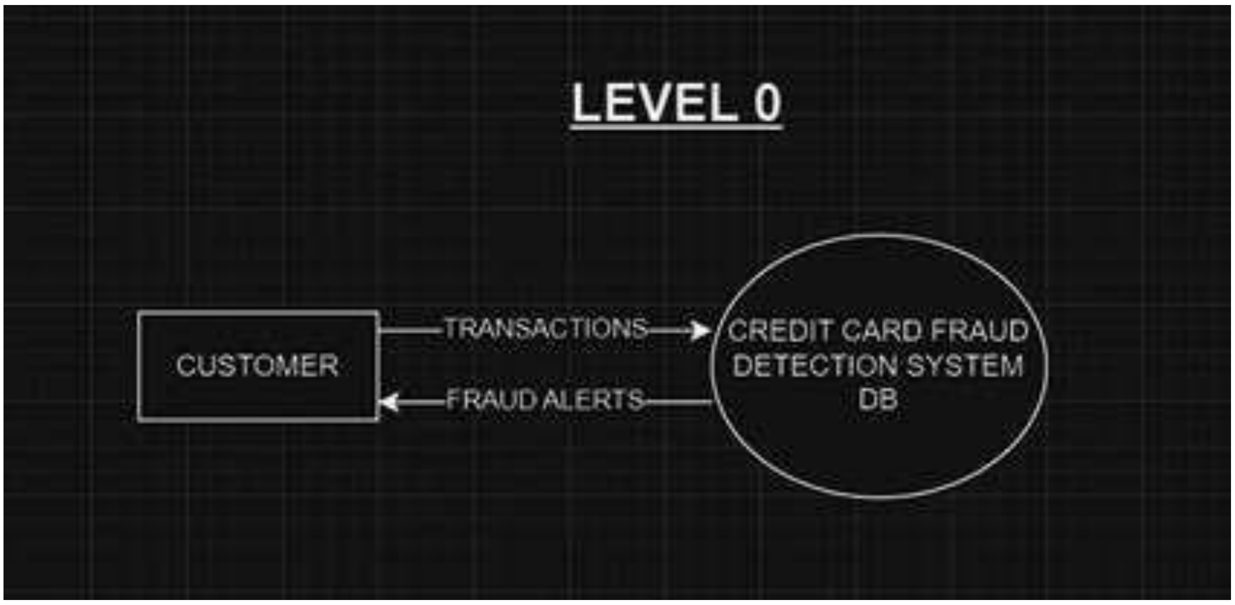
PROJECT DFD:

Data Flow Diagram (DFD):

Focuses on the flow of data through a system. It shows how data moves from inputs to outputs, with processing steps in between. It doesn't depict the specific actions or decisions within the system.

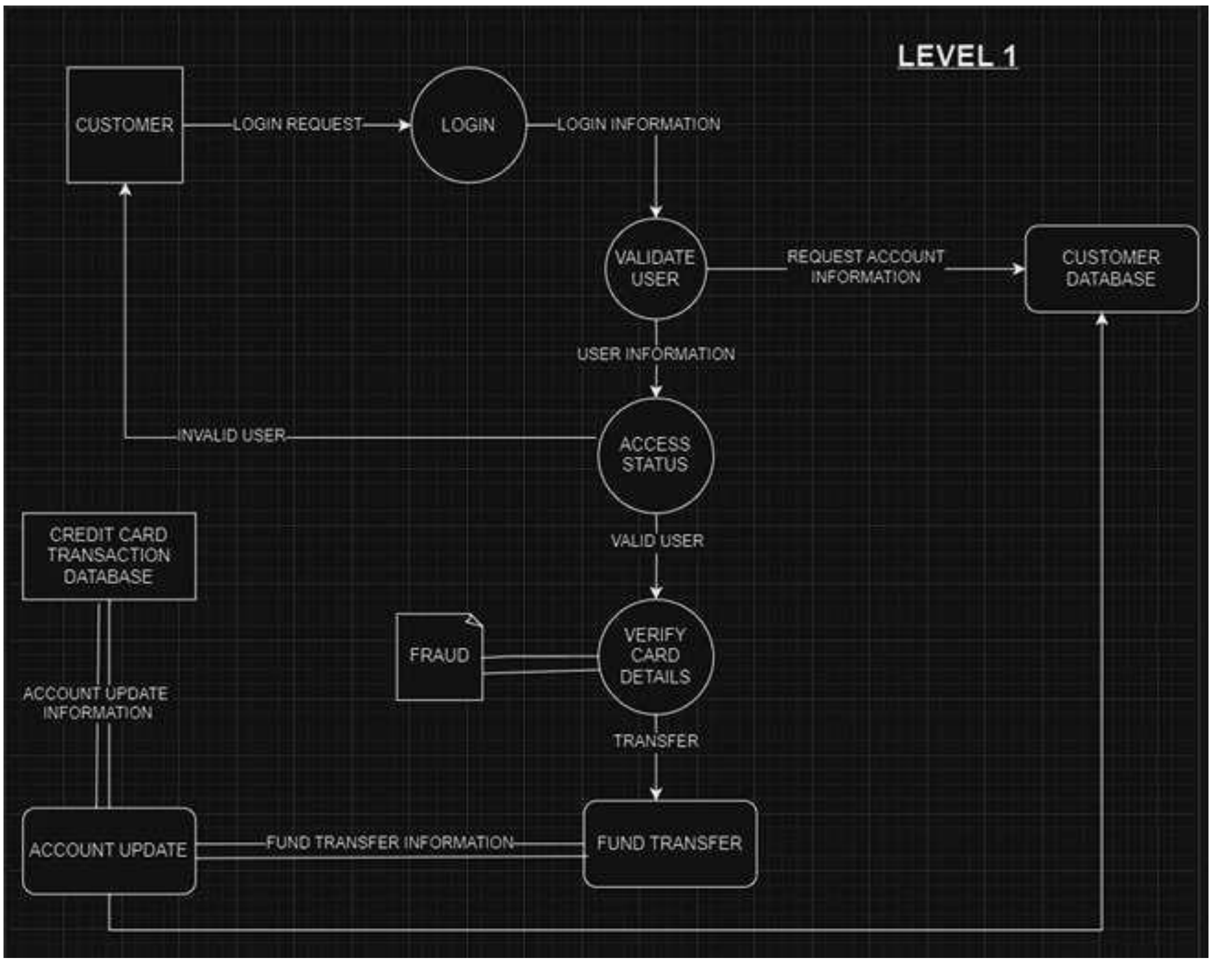
Level 0:

Level 0 DFD provides a high-level overview of the entire system. It shows the system as a single entity interacting with external data sources (e.g., payment gateway) and sinks (e.g., reports). It focuses on what the system does, not how it does it



Level 1:

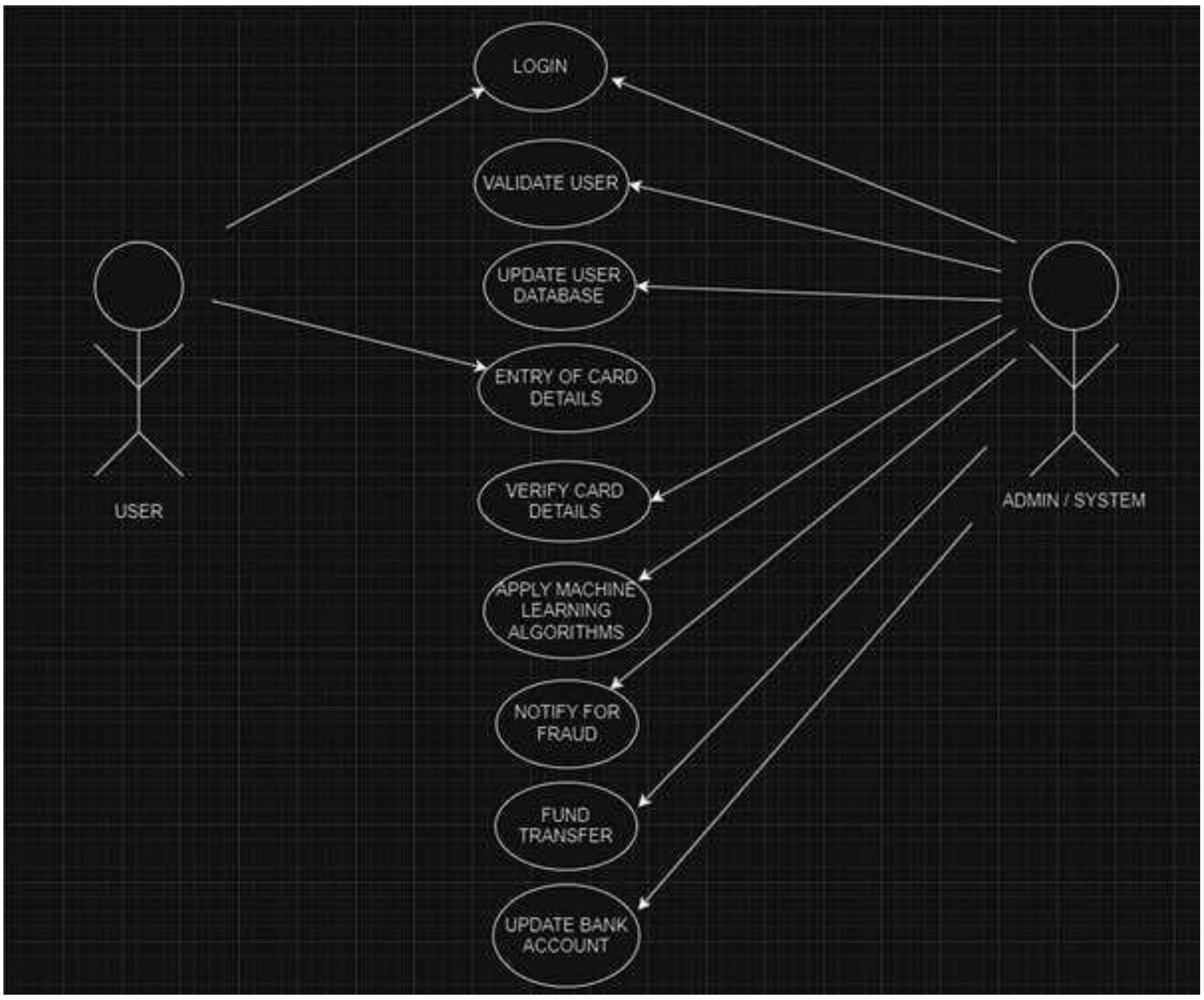
Level 1 DFD provides a high-level overview of the entire system. It shows the system as a single process interacting with external entities (e.g., payment gateway) and the data that flows between them (e.g., transaction data, reports).



PROJECT USE CASE DIAGRAM:

Use Case Diagram:

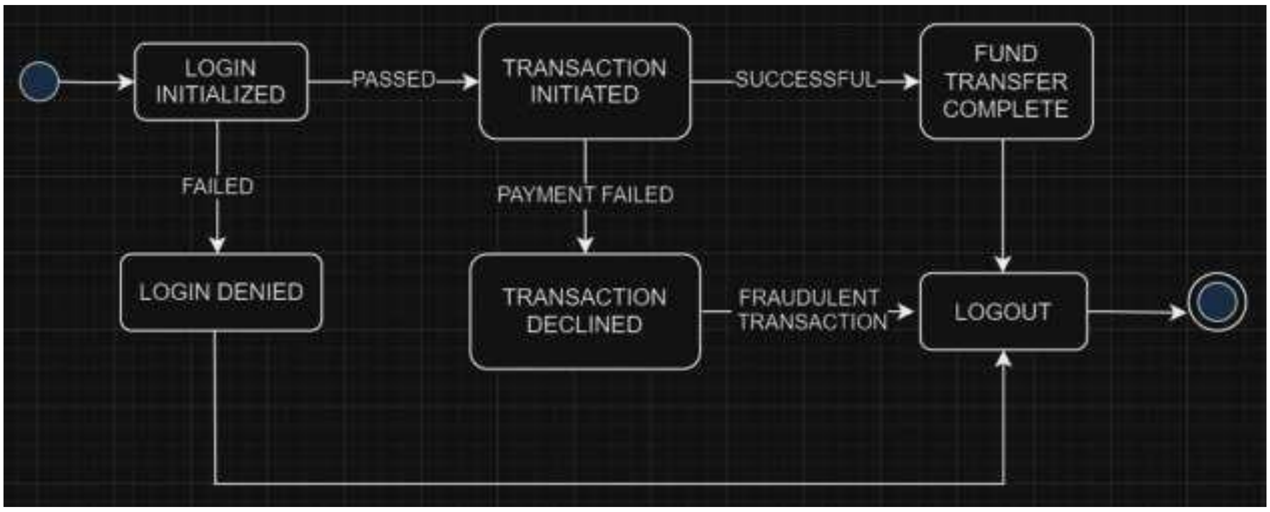
Captures the functionality of a system from the perspective of an actor (user or external system). It shows the interactions between actors and the system, and the use cases (scenarios) that describe how actors achieve goals using the system.



PROJECT TRANSITION DIAGRAM:

Transition Diagram:

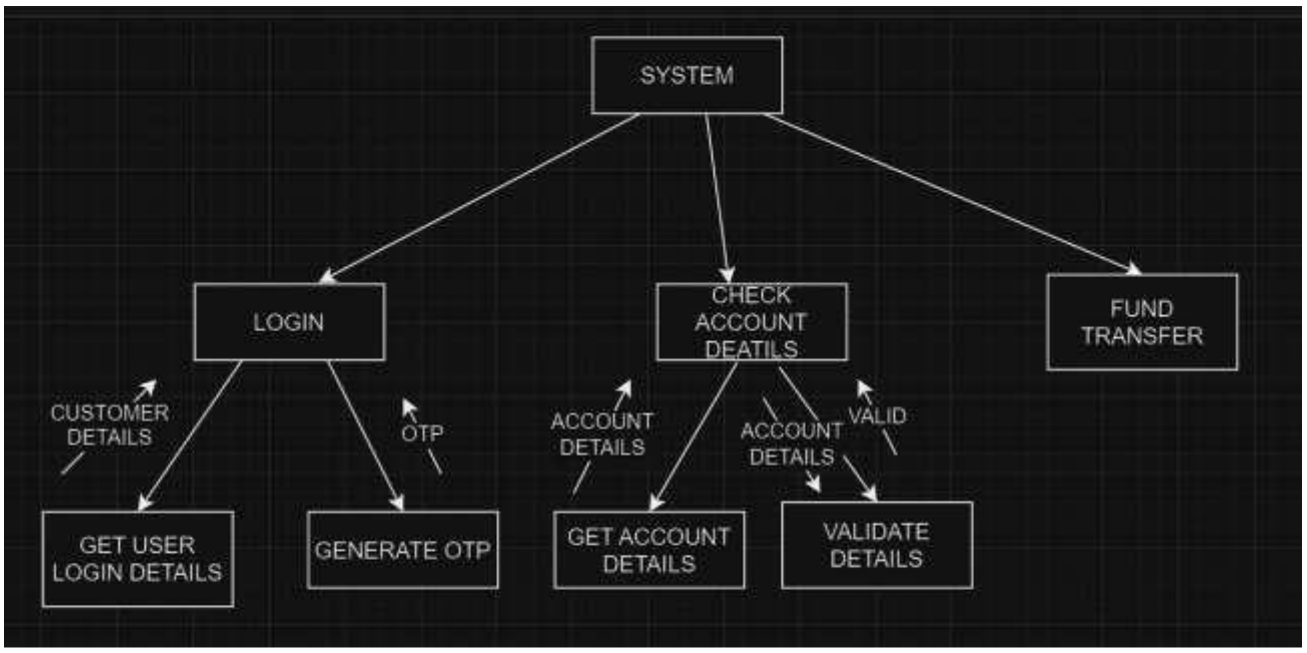
Illustrates the behavior of an object or system in response to events. It shows the different states an object can be in, and the events that cause it to transition between those states. It's helpful for understanding how an object reacts to different stimuli.



PROJECT STRUCTURE CHART:

Structure Chart:

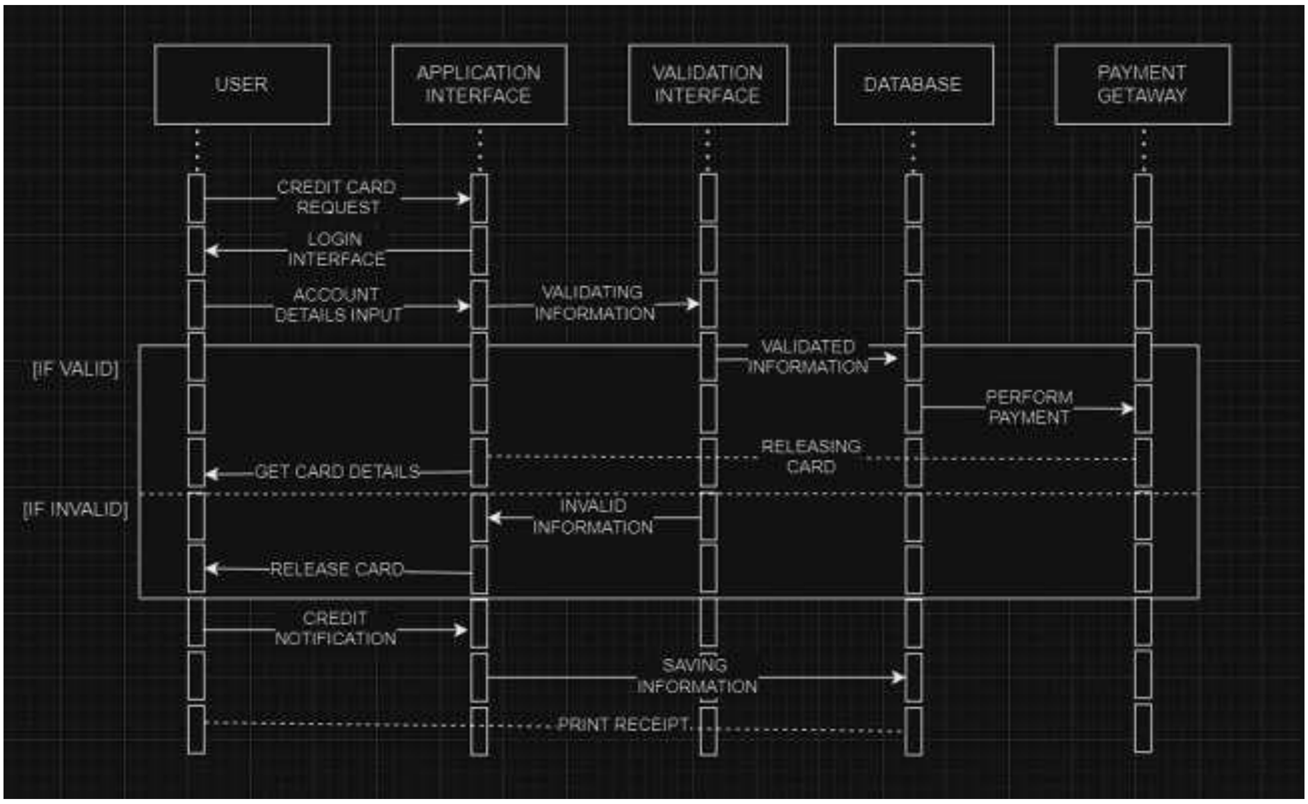
A Visual tool used in software design to show the top-down breakdown of a program into functions and their sub-functions. It depicts the calling hierarchy and relationships between different program modules.



PROJECT SEQUENCE DIAGRAM:

Sequence Diagram:

A sequence diagram shows the interactions between objects in a specific order, like a timeline. It uses vertical lifelines for objects and horizontal arrows to represent messages exchanged between them.


USER
APPLICATION INTERFACE
VALIDATION INTERFACE
DATABASE
PAYMENT GETAWAY
CREDIT CARD REQUEST
LOGIN INTERFACE
ACCOUNT DETAILS INPUT
VALIDATING INFORMATION
[IF VALID]
VALIDATED INFORMATION
PERFORM PAYMENT
RELEASING CARD
GET CARD DETAILS
[IF INVALID]
INVALID INFORMATION
RELEASE CARD
CREDIT NOTIFICATION
SAVING INFORMATION
PRINT RECEIPT