**Credit Fraud Detection Using the Hidden Markov Model**

Team Members:

Sergio Gabriel Jiawei Kun ; s\_kun@mail.fhsu.edu

Lee Joseph Judkins ; ljjudkins@mail.fhsu.edu

Project Github: https://github.com/kunsergio117/CreditFraudDetectionHMM.git

Instructor: Prof. Hieu Vu

**Revision History:**

|  |  |
| --- | --- |
| Part 1 | 07/10/2024 |
| Part 2 | 13/10/2024 |

Table of Contents

[Interaction Diagrams 3](#_Toc179842420)

[Sequence Diagram 3](#_Toc179842421)

[Class Diagram and Interface Specification 5](#_Toc179842422)

[Class Diagram 5](#_Toc179842423)

[Class Descriptions 5](#_Toc179842424)

[Traceability Matrix 7](#_Toc179842425)

[Evolution of domain concepts 7](#_Toc179842426)

[Domain Concepts Evolution 7](#_Toc179842427)

[Project Management 9](#_Toc179842428)

[References 11](#_Toc179842429)

# Interaction Diagrams

## Sequence Diagram

The sequence diagram below illustrates the process flow for use cases 1,2,4,5 and 6, where a user uploads transaction data and the subsequent handling by the system. It encompasses the interactions from user authentication through to data processing and result generation, including potential alerts for fraudulent activity.

A diagram of a computer program

Description automatically generated

Process:

1. User Accesses the Login Page:

- The user navigates to the application's login page via a web browser.

- Objects Involved: User Interface, User

2. User Authentication:

- The user enters their login credentials (username and password).

- The User Interface forwards these credentials to the Authentication Service for verification.

- Upon successful verification, an authentication token is generated, and the user is granted access to the dashboard.

- Design Principle: The Authentication Service employs the Single Responsibility Principle by focusing solely on verifying user credentials.

3. Navigation to the Dashboard:

- Upon login, the user is presented with a dashboard interface, providing options to upload a CSV file or explore other features.

- Objects Involved: User Interface, Dashboard

4. Choosing CSV Upload:

- The user selects the option to upload CSV transaction data.

- The User Interface navigates to the CSV Upload Page.

- Design Principle: Separation of Concerns is utilized, distinguishing user interface tasks (navigation and input) from data processing tasks.

5. CSV Upload Process:

- On the Upload Page, the user selects and uploads a CSV file containing transaction data.

- The file is submitted to the Transaction Data Validator, which checks for correct format and content.

- Objects Involved: User Interface, Transaction Data Validator

6. Data Storage and Preprocessing:

- Once validated, the transaction data is stored in the Database.

- The system preprocesses the data, preparing it for analysis by the Hidden Markov Model (HMM) and Recurrent Neural Network (RNN) models.

- Objects Involved: Database, Preprocessor, HMM, RNN

7. Fraud Detection and Analysis:

- Preprocessed data is analyzed by the HMM and RNN models.

- The models evaluate the transaction data, detect patterns, and assess potential fraud.

- \*\*Design Principle\*\*: \*\*Information Expert\*\* principle guides that the detection logic resides within specialized fraud analysis components.

8. Results Storage and Reporting:

- The fraud analysis results are returned to the Database for storage.

- The results are sent back to the Dashboard, where summarized fraud reports and analyses are displayed to the user.

- Objects Involved: Database, Dashboard

9. Alert Generation:

- If any transaction is flagged as fraudulent, the system generates alerts.

- Alerts are displayed on the dashboard and may be sent via email or SMS, depending on user settings.

- Objects Involved: AlertManager

# Class Diagram and Interface Specification

## Class Diagram

A screenshot of a computer

Description automatically generated

## Class Descriptions

**User**

The User class represents individuals who interact with the system, including customers, analysts, and administrators. It contains essential attributes for authentication purposes, such as username and password. By encapsulating user-related functionalities, this class facilitates user management, including the creation of user accounts, password verification, and the management of user sessions. The User class forms the foundation of the system's security model, ensuring that only authorized personnel can access sensitive data and functionalities.

**Transaction**

The Transaction class encapsulates the details of each financial transaction processed by the system. Key attributes of this class include the transaction amount, date, and status (e.g., pending, completed, flagged). The Transaction class serves a critical role in tracking the flow of financial activities and provides the necessary data for fraud analysis. This class also interacts with various system components that utilize transaction data to identify patterns indicative of fraudulent behavior.

**FraudDetector**

The FraudDetector class is the central component responsible for coordinating the fraud detection process and analysis. It leverages advanced algorithms, including Hidden Markov Models (HMM) and Recurrent Neural Networks (RNN), to analyze transaction patterns in real time. By processing input data from multiple transactions, this class generates insights that help identify potential fraudulent activities. The FraudDetector acts as the brain of the system, routing information to and from other components for further action.

**AlertManager**

The AlertManager class is tasked with the creation and management of alerts triggered by potential fraudulent activity detected by the FraudDetector. It monitors transaction evaluations and determines when to initiate alerts to users or system administrators. Alerts may be sent through various channels, such as in-app notifications, emails, or SMS, ensuring that users are promptly informed of any suspicious activities. This class enhances user awareness and response to fraudulent risks.

**Database**

The Database class represents the storage layer for the application, responsible for maintaining persistent data across user sessions. It stores important information related to transactions, user data, fraud detection results, and alerts. This class facilitates data retrieval and updates, ensuring that all relevant information is safely stored and easily accessible when needed. The Database class plays a vital role in supporting the application's integrity and performance.

# Traceability Matrix

|  |  |  |
| --- | --- | --- |
| **Domain Concept** | **Class** | **Rationale** |
| User | User | Represents individuals interacting with the system |
| Transaction | Transaction | Each transaction corresponds to a financial operation |
| Fraud Alert | AlertManager | Alerts generated when suspicious activity is detected |
| Fraud Detection | FraudDetector | Analyzes transactions for suspicious patterns |
| Data Storage | Database | Manages persistent storage of user, transaction, and alert data |

## Evolution of domain concepts

The **User** class directly evolved from the domain concept of "User" who submits transactions.

The **Transaction** class evolved from the concept of financial transactions submitted by the users.

The **AlertManager** class corresponds to the fraud detection concept, which was originally derived from the need to alert suspicious transactions.

**TransactionProcessor** and **FraudDetector** emerged from the need to separate processing and detection responsibilities, adhering to the **Single Responsibility Principle**.

**Database** was introduced to handle persistent storage, evolving from the need to store data long-term.

## Domain Concepts Evolution

- User: Directly maps to User class. Represents anyone interacting with the system for authentication and data handling.

- Transaction: Maps to the Transaction class, capturing the financial details analyzed for fraud.

- Fraud Detection: Leads to classes like FraudDetectionSystem and AlertManager.

- Explanation: These derived classes separate concerns, where FraudDetectionSystem processes data while AlertManager deals with user notifications.

# Project Management

Both team members will be actively engaged in the development process, utilizing the GitHub repository to track progress and contributions clearly.

Responsibilities will be delegated based on each member's strengths and expertise, with a shared accountability structure ensuring that all aspects of the project are covered. This is because our group consists of only a pair and thus realistically we will both need input and validation from one another in all development areas.

Below is our projected milestones, with week 1 representing the week beginning in 23 Sept 2024.

| **Week** | **Task Description** | **Responsible Team Members** |
| --- | --- | --- |
| Week 1 | Project kick-off meeting, define scope and objectives | Sergio Kun, Joseph Judkins |
| Week 2 | Research HMM algorithms and relevant literature | Sergio Kun |
|  | Familiarization with data sources and datasets | Joseph Judkins |
| Week 3 | Develop initial design and architecture of the application | Sergio Kun, Joseph Judkins |
| Week 4 | Implement CSV transaction data upload feature | Sergio Kun |
|  | Initial development of the user authentication process | Joseph Judkins |
| Week 5 | Develop the manual checking function for transaction validity | Sergio Kun |
| Week 6 | Implement the alerting function for suspicious transactions | Joseph Judkins |
| Week 7 | Integrate the simulated fraudulent transaction feature | Sergio Kun |
| Week 8 | Testing functionality and debugging | Sergio Kun, Joseph Judkins |
| Week 9 | Develop and integrate reporting tools for transaction summaries | Joseph Judkins |
|  | User interface refinement and user experience enhancements | Joseph Judkins |
| Week 10 | Conduct user testing and gather feedback | Sergio Kun, Joseph Judkins |
| Week 11 | Finalize features based on user feedback | Sergio Kun, Joseph Judkins |
| Week 12 | Prepare project presentation and documentation | Sergio Kun, Joseph Judkins |
| Week 13 | Project review and adjustments based on tutor feedback | Sergio Kun, Joseph Judkins |
| Week 14 | Final project deployment and presentation to the class | Sergio Kun, Joseph Judkins |

# References

Marsic, I. (n.d.). *Software Engineering Project Report*. Retrieved from http://eceweb1.rutgers.edu/~marsic/Teaching/SE/report2.html