<Project Design and Architecture>

Design Document

Version 1.0

Group Id: F24PROJECT3D9E7

Supervisor Name : Fouzia Jumani

Revision History

Date (dd/mm/yyyy)	Version	Description	Author
27/12/2024 - 10/01/2025	1.0	Completed half of the project	BC200408411
11/01/2025 - 27/02/2025	//	Completed Full Project	BC200408411

Table of Contents

- ## Introduction of Design Document

The design phase marks a pivotal step in the software development **lifecycle**, converting functional and non-functional requirements into a detailed blueprint for implementation. This document is dedicated to laying the groundwork for the successful realization of the Anomaly Detection System in **Blockchain,** a project aimed at enhancing the security of decentralized networks by detecting and mitigating 51% attacks through advanced machine learning techniques.

Purpose of the Design Document

The primary purpose of this design document is to bridge the gap between the theoretical requirements captured in the Software Requirements Specification (SRS) and the practical development of the system. It ensures that the architectural and technical foundations align with the project's goals, enabling efficient development and seamless integration of components.

Contents of the Document

This document encompasses the following key design elements:

- **Entity Relationship Diagram (ERD):** A detailed depiction of the system's data schema, capturing the relationships and dependencies among entities.
- **Sequence Diagrams:** Graphical representations of system interactions for key use cases, showcasing the sequence of operations.
- **Architecture Design**: A tiered architecture outlining the system's structural framework, emphasizing scalability and maintainability.
- **Class Diagram:** A comprehensive blueprint of the system's classes and their interrelationships, ensuring alignment with object-oriented principles.
- **Database Design:** Detailed diagrams of the database schema, optimized for storing and retrieving Blockchain transaction data.
- **Interface Design:** Prototypes and mockups illustrating the graphical user interface (GUI) for primary system functionalities.
- **Test Cases:** A collection of test scenarios to validate the system's performance, accuracy, and robustness under different conditions.

Significance of the Design Phase

The design phase ensures the development process is guided by a well-structured plan, reducing risks, minimizing ambiguities, and promoting coherence among team members. For the Anomaly Detection System in Blockchain, this phase is instrumental in addressing the following:

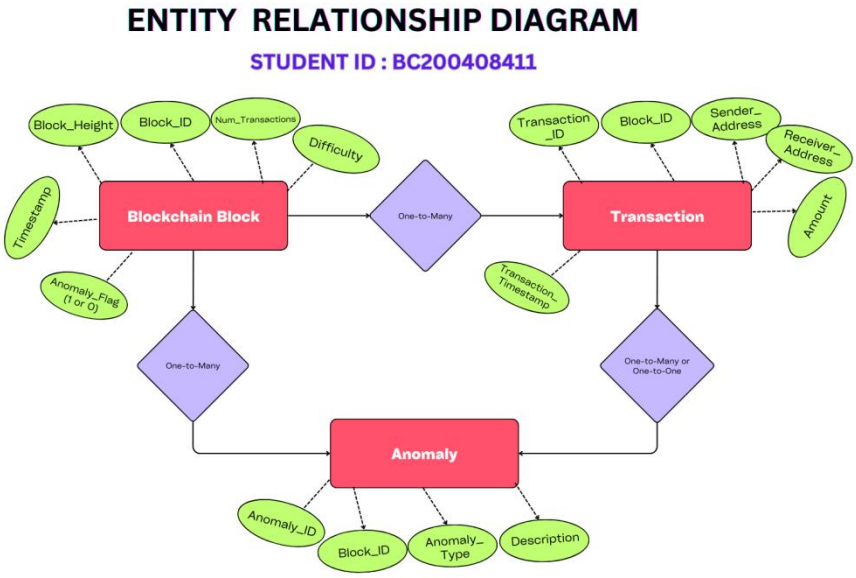
- **Security:** By providing a robust architecture, the system is better equipped to identify and respond to malicious activities such as 51% attacks.
- **Efficiency:** Detailed diagrams and designs streamline the implementation of machine learning models and anomaly detection mechanisms.
- **Scalability:** The modular architecture ensures the system can adapt to varying transaction volumes and evolving Blockchain technologies.

Relevance to the Project Goals

The design document is tailored to the specific needs of the Anomaly Detection System in **Blockchain.** By leveraging design methodologies and best practices, this document facilitates the integration of machine learning algorithms **(SVM, Random Forest, AdaBoost, XGBoost)** to detect anomalies in real-time. Additionally, it supports the efficient **preprocessing of Blockchain** datasets, injection of artificial anomalies, and evaluation of model performance using industry-standard metrics.

In **conclusion,** this design document serves as the cornerstone for translating the project's vision into a functional and effective system. It not only ensures adherence to the defined requirements but also lays the foundation for future enhancements and scalability.

- **Entity Relationship Diagram (ERD) (To be developed using Microsoft Visio or any other drawing software of your choice)**



Entity-Relationship Diagram (Table Format)

Entities and Attributes

Entity Name	Attributes	Primary Key	Foreign Key
Blockchain Block	Block_ID, Block_Height, Timestamp, Num_Transactions, Difficulty, Anomaly_Flag	Block_ID	None
Transaction	Transaction_ID, Block_ID, Sender_Address, Receiver_Address, Amount, Transaction_Timestamp	Transaction_ID	Block_ID
Anomaly	Anomaly_ID, Block_ID, Anomaly_Type, Description	Anomaly_ID	Block_ID

Relationships

Relationship	Entity 1	Entity 2	Type	Description
Blockchain Block ↔ Transaction	Blockchain Block	Transaction	1:N	One Blockchain Block can have many Transactions.

Blockchain Block ↔ Anomaly	Blockchain Block	Anomaly	1:N	One Blockchain Block can have multiple Anomalies.
Transaction ↔ Anomaly	Transaction	Anomaly	1:N (Optional)	One Transaction may cause one or more Anomalies.
Anomaly ↔ Blockchain Block	Anomaly	Blockchain Block	M:N (Optional)	Some Anomalies may span across multiple Blockchain Blocks.
Anomaly ↔ Anomaly	Anomaly	Anomaly	1:N (Optional)	An Anomaly may trigger other related Anomalies, representing a recursive link.

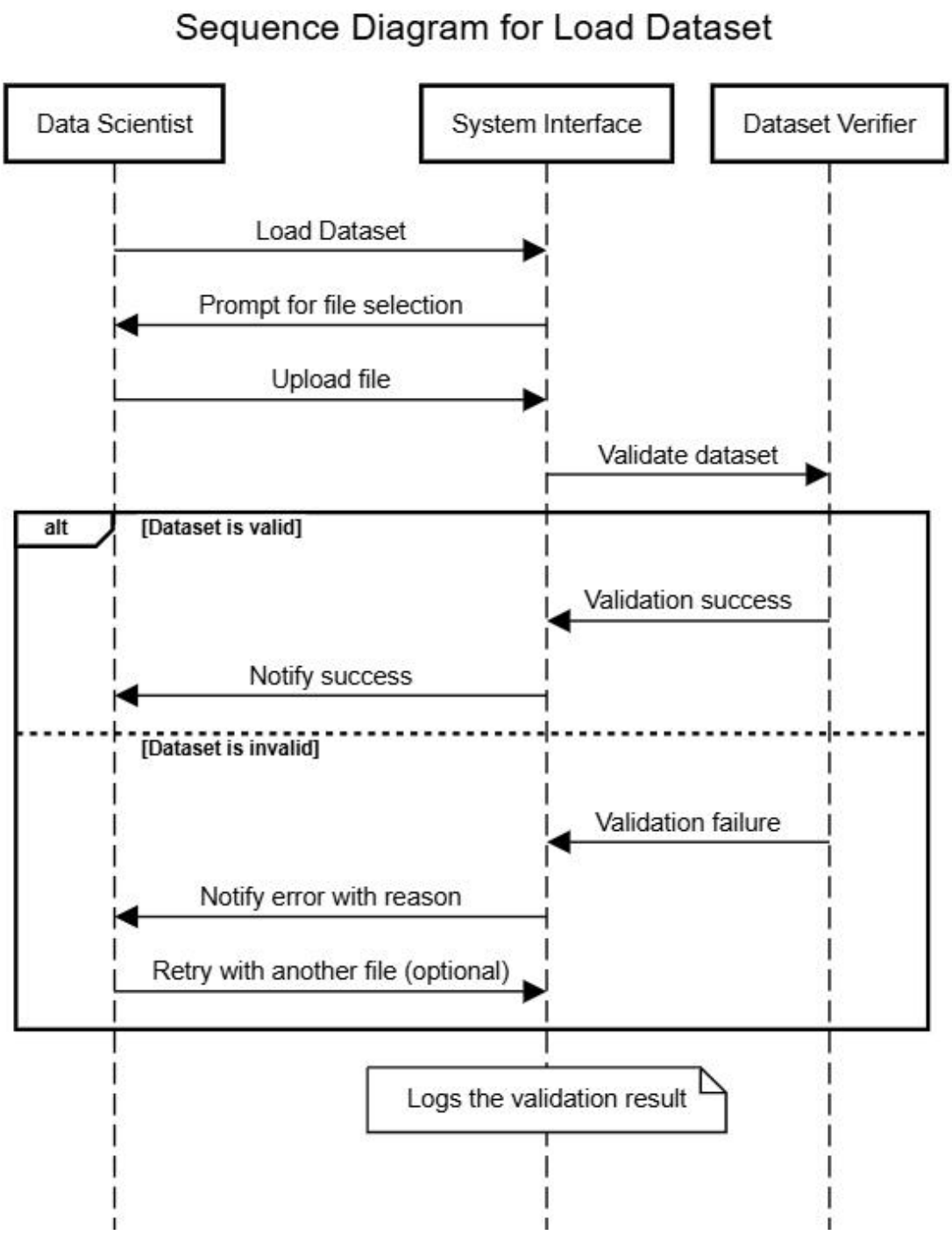
Note:

Optional Relationships: Include these only if they add value to your specific use case and dataset logic.
Recursive Relationship (Anomaly ↔ Anomaly): Useful for detecting causality between anomalies.

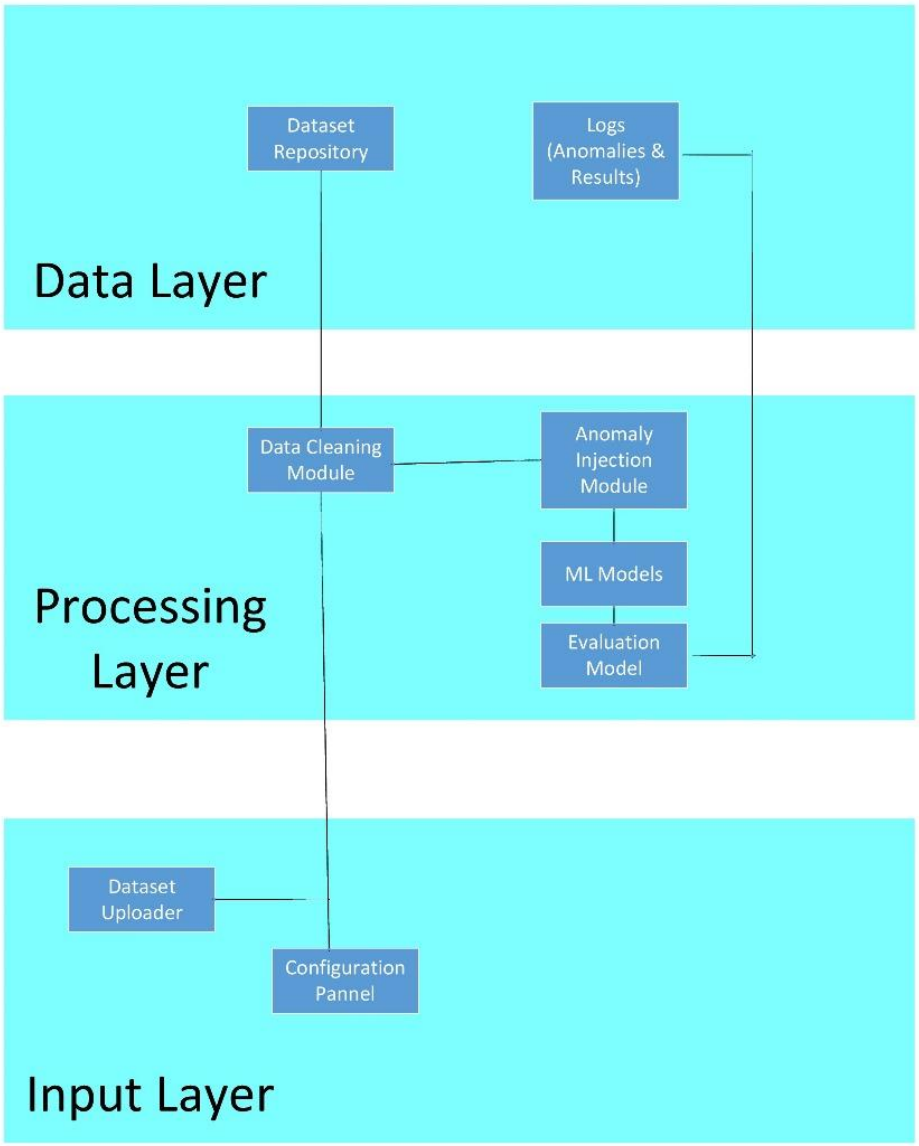
- **Sequence Diagrams (To be developed using Rational Rose or any other drawing software of your choice)**

Step	Source	Target	Action	Result	Alternative
1	Data Scientist	System Interface	Request to load dataset	System initiates file selection	-
2	System Interface	Data Scientist	Prompts to select a file	File selection initiated	-
3	Data Scientist	System Interface	Uploads the selected file	File is received	-

4	System Interface	Dataset Verifier	Sends file for validation	Validation process starts	-
5	Dataset Verifier	System Interface	Validation success	"Success" returned	-
6	Dataset Verifier	System Interface	Validation failure	"Failure" returned	Retry option given to Data Scientist
7	System Interface	Data Scientist	Notifies result (success or error)	Completion or Retry	Retry by selecting another file



- Architecture Design Diagram



Details for Architecture Design Diagram:

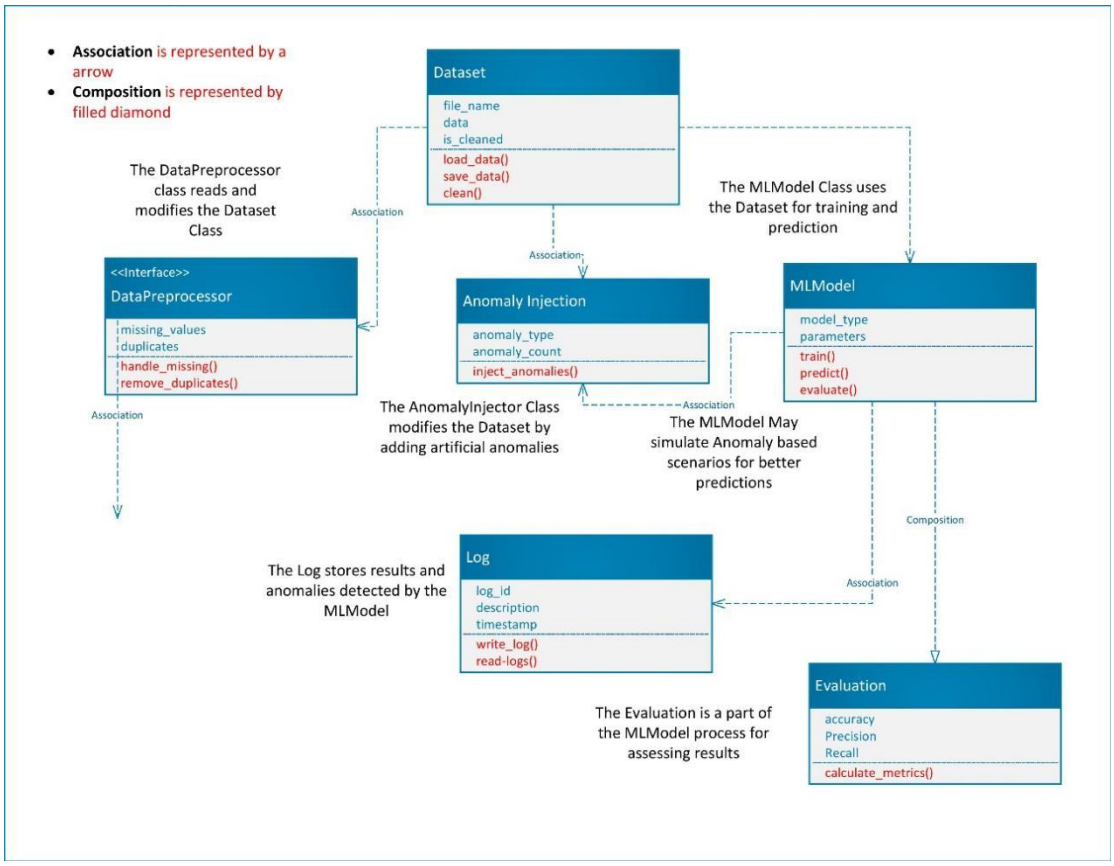
Layer	Component	Responsibility
Input Layer	Dataset Uploader	Allows users to upload raw datasets for analysis.

	Configuration Panel	Lets users set parameters like thresholds or model preferences.
Processing Layer	Data Cleaning Module	Cleans the dataset by handling missing values and inconsistencies.
	Anomaly Injection Module	Simulates anomalies for model training and testing.
	ML Models	Detects anomalies using algorithms like SVM, Random Forest, etc.
	Evaluation Module	Assesses model performance with metrics like precision and recall.
Data Layer	Dataset Repository	Stores original and cleaned datasets.
	Logs (Anomalies, Results)	Saves detected anomalies and model outputs.

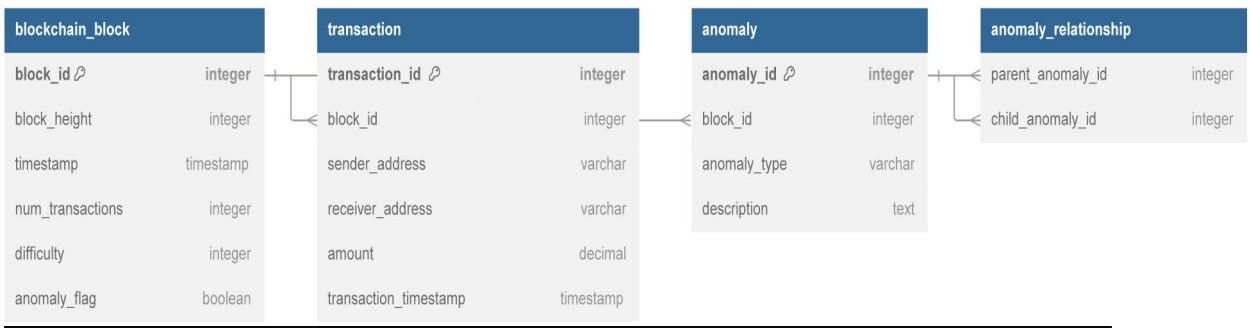
Data Flow Summary:

- **Input Layer** sends raw data and parameters to the **Processing Layer**.
- **Processing Layer** cleans, processes, and evaluates the data.
- **Data Layer** stores the cleaned data, results, and anomaly logs for future use.

- Class Diagram



- **Database Design**



Relationships:

- **Blockchain Block ↔ Transaction (1:N)**: A block can have multiple transactions.
- **Blockchain Block ↔ Anomaly (1:N)**: A block can have multiple anomalies.
- **Anomaly ↔ Anomaly (Recursive 1:N)**: An anomaly can cause other anomalies.

Breif Overview of the Database Design Diagram:

Table Name	Description
Blockchain Block	Stores metadata about each block in the Blockchain
Transaction	Records individual transactions within a block
Anomaly	Logs detected anomalies, including type and description

- Interface Design

The Anomaly Detection System in Blockchain is designed to operate within Python environments such as Jupyter Notebook or Google Colab, leveraging tools like Scikit-learn, Matplotlib, and NumPy. Since the project outputs are directly displayed as visualizations, tables, and logs within the notebook interface, it does not include a separate graphical user interface (GUI).
All interactions, including data upload, anomaly detection, and results visualization, are handled through code cells and outputs in the notebook environment. Therefore, the interface design phase has been skipped as it is not applicable to this project.

- Test Cases

Here are all test cases that are needed for a perfectly working anomally detection project.

Test Case ID	**UC01-TC01**
Test Title	Validate Dataset Loading Functionality
Objective	To ensure that the system successfully loads a valid blockchain dataset.
Preconditions	- A blockchain dataset in a compatible format (e.g., CSV or JSON) must be available. - The system must be operational, and the user must have access.
Test Data	A valid dataset file named blockchain_data.csv containing 100 blockchain blocks and 500 transactions.
Steps	1. Open the anomaly detection system interface. 2. Select the **Load Dataset** option. 3. Browse and select the file blockchain_data.csv. 4. Click the **Submit** button.

	5. Wait for the system to validate the dataset format and compatibility.
Expected Result	- The system should validate the dataset successfully. - A success message should appear: **"Dataset loaded successfully."** - The dataset should be ready for preprocessing.
Actual Result	(To be filled after execution)
Pass/Fail	(To be filled after execution)
Remarks	This test verifies the core functionality of loading datasets, which is a critical step for anomaly detection.