<Anomaly Detection System in Blockchain >

Software Requirements Specification

Version 1.0

Group Id: F24PROJECT3D9E7

Supervisor Name : Fouzia Jumani

Revision History

Date (dd/mm/yyyy)	Version	Description	Author
12/11/2024	1.0	Introduction of the project	BC200408411

Note by Student:

If any improvements are required, then please inform me through comments before grading.

Thanks in Advance

Table of Contents

SRS Document

SCOPE OF PROJECT:

Project Overview:

The Anomaly Detection System in Blockchain project aims to enhance the security of blockchain networks by detecting 51% attacks in real-time. Using machine learning, this project will identify malicious behaviors by analyzing anomalies within a Bitcoin dataset. This approach enhances the security and reliability of decentralized networks, addressing the critical need for integrity in blockchain transactions.

Objectives:

- To develop an anomaly detection mechanism to detect potential 51% attacks.
- To inject artificial anomalies mimicking such attacks into a Bitcoin dataset.
- To evaluate the performance of machine learning models (SVM, Random Forest, AdaBoost, and XGBoost) in detecting anomalies.

Functional and non Functional Requirements:

Functional Requirements:

The system's key functions are as follows:

- Data Loading and Preprocessing

- Load the Bitcoin dataset containing transaction and block details.
- Clean the dataset by transforming and handling missing or inconsistent data.

- Anomaly Injection

  - Inject artificial anomalies to simulate a 51% attack:
  - Confirmations: Set anomalous blocks to zero or unusually low confirmations.
  - Height: Duplicate or irregular block heights.
  - Transactions: Anomalous transaction volume in certain blocks.
  - Difficulty: Reduced difficulty under attack conditions.
  - Timestamps: Overlapping or irregular block timestamps.

- Anomaly Labeling

  - Label the dataset using a binary indicator, where:
  - 1 = Anomalous block
  - 0 = Normal block

- Model Training

  - Implement and train SVM, Random Forest, AdaBoost, and XGBoost models.
  - Train each model on labeled data to recognize patterns indicative of 51% attacks.

- Model Evaluation

  - Use performance metrics (accuracy, precision, recall, and F1-score) to assess each model's effectiveness.
  - Generate confusion matrices for visualizing true/false positives and negatives.

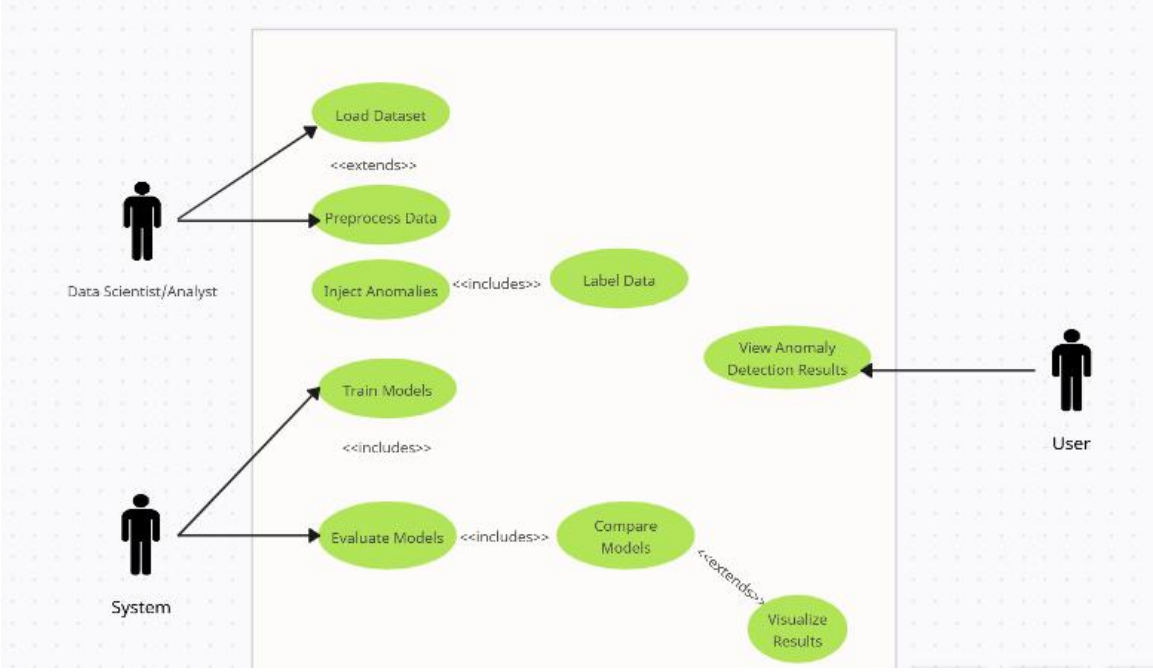
- Model Comparison and Visualization

  - Compare the models to determine the best-performing algorithm.
  - Present results with visual metrics for each model.

Non Functional Requirements:

- **Performance:** Models should be optimized to deliver real-time anomaly detection, ensuring minimal latency.
- **Scalability:** The system should handle large datasets as blockchain data grows.
- **Usability:** An intuitive interface for non-experts to monitor anomaly detection and performance metrics.
- **Reliability:** Accurate anomaly detection to prevent false positives that could raise unnecessary alarms.
- **Security:** Secure handling of the blockchain dataset, ensuring no unauthorized data modification or breaches.

Use Case Diagram(s):



Use Case Diagram Explained:

Actors and Actions:-

- Data Scientist/Analyst:
  - **Load Dataset:** Loads the blockchain data for analysis.
  - **Preprocess Data (extends Load Dataset):** Cleans and prepares the data if needed.
  - **Inject Anomalies (includes Label Data):** Adds artificial anomalies to simulate attacks.
- System:
  - **Train Models:** Trains machine learning models on the prepared data.
  - **Evaluate Models (includes Train Models):** Assesses model performance based on metrics.
  - **Compare Models (includes Evaluate Models):** Compare different models and choose the best one.
  - **Visualize Results (Extends Compare Models):** After comparing models, system should be able to give insights or visualize/show results.
- User/End User:
  - **View Anomaly Detection Results:** Views the results of the anomaly detection analysis, potentially with visual insights.

Usage Scenarios:

Use Case 1: Load Dataset

Use Case Title	**Load Dataset**
Use Case ID	UC-01

Description	Loads the blockchain data into the system for preprocessing and analysis.
Actors	Data Scientist/Analyst
Preconditions	1. Blockchain dataset is available in a compatible format (CSV, JSON, etc.). 2. System is operational, and the actor has access.
Basic Flow	1. Actor opens the system interface. 2. Actor selects the "Load Dataset" option. 3. Actor browses and selects the dataset file. 4. System verifies the file's format and compatibility. 5. Dataset is successfully loaded.
Alternative Flow	**File Format Error:** 1. Actor uploads an incompatible dataset. 2. System shows an error and prompts the actor to upload a compatible file.
Postconditions	Dataset is successfully loaded into the system for preprocessing.
Exceptions	1. Missing, corrupted, or unsupported dataset. 2. System error while reading the file.
Author	Shameer Hussain
Date	15-11-2024

Use Case 2: Preprocess Data

Use Case Title	**Preprocess Data**
Use Case ID	UC-02

Description	Cleans and prepares the dataset by handling missing values, removing duplicates, and transforming data.
Actors	Data Scientist/Analyst
Preconditions	1. Dataset is loaded into the system. 2. Actor has initiated preprocessing.
Basic Flow	1. Actor selects "Preprocess Data" option. 2. System identifies missing values, duplicates, and anomalies. 3. Actor chooses actions (e.g., filling missing values). 4. System applies preprocessing steps. 5. Dataset is cleaned and prepared.
Alternative Flow	**Preprocessing Skipped:** 1. Actor decides to proceed with the raw dataset. 2. System warns about potential issues.
Postconditions	Dataset is prepared and ready for further analysis or model training.
Exceptions	1. Insufficient memory or system resources for preprocessing. 2. Unexpected data structures cause preprocessing failure.
Author	Shameer Hussain
Date	15-11-2024

Use Case 3: Inject Anomalies

Use Case Title	**Inject Anomalies**
Use Case ID	UC-03
Description	Adds artificial anomalies to simulate attacks or unusual patterns for model training and evaluation.

Actors	Data Scientist/Analyst
Preconditions	1. Preprocessed dataset is available. 2. Actor has access to the anomaly injection tools.
Basic Flow	1. Actor selects "Inject Anomalies" option. 2. Actor specifies anomaly types and quantities. 3. System injects specified anomalies. 4. Actor reviews the updated dataset.
Alternative Flow	**Anomaly Injection Canceled:** 1. Actor decides not to inject anomalies. 2. System retains the clean dataset.
Postconditions	Dataset contains specified anomalies, ready for training and evaluation.
Exceptions	1. Unsupported anomaly types. 2. Dataset corruption after anomaly injection.
Author	Shameer Hussain
Date	15-11-2024

Use Case 4: Label Data

Use Case Title	**Label Data**
Use Case ID	UC-04
Description	Assigns labels to dataset records (e.g., Normal, Anomalous) to enable supervised learning.
Actors	Data Scientist/Analyst
Preconditions	1. Preprocessed dataset (with or without anomalies) is available. 2. Actor has access to labeling tools.

Basic Flow	1. Actor selects "Label Data" option. 2. System displays the dataset for review. 3. Actor assigns labels to records based on features or patterns. 4. System updates the dataset with assigned labels. 5. Labeled dataset is saved.
Alternative Flow	**Automated Labeling:** 1. Actor opts for automated labeling. 2. System applies predefined rules or thresholds to label the data.
Postconditions	Dataset is fully labeled and ready for model training.
Exceptions	1. System fails to save the labeled dataset. 2. Mislabeling due to incorrect manual or automated rules.
Author	Shameer Hussain
Date	15-11-2024

Use Case 5: Train Models

Use Case Title	**Train Models**
Use Case ID	UC-05
Description	Trains machine learning models using the labeled dataset to detect anomalies.
Actors	System
Preconditions	1. Labeled dataset is ready. 2. Training algorithms are available.

Basic Flow	1. Actor selects "Train Models" option. 2. Actor specifies algorithms (e.g., Random Forest, Neural Networks). 3. System splits the dataset into training and validation sets. 4. System trains models using the training set. 5. System evaluates model performance on the validation set.
Alternative Flow	**Default Algorithms:** 1. Actor does not specify algorithms. 2. System trains models using default algorithms.
Postconditions	Trained models are stored for evaluation.
Exceptions	1. Insufficient computational resources. 2. Dataset errors cause training failure.
Author	Shameer Hussain
Date	16-11-2024

Use Case 6: Evaluate Models

Use Case Title	**Evaluate Models**
Use Case ID	UC-06
Description	Assesses the performance of trained models using various evaluation metrics.
Actors	System
Preconditions	1. Trained models are available. 2. Validation/test dataset is ready.
Basic Flow	1. System retrieves trained models. 2. Actor specifies evaluation metrics (e.g., precision, recall). 3. System evaluates models on the test dataset. 4. System displays performance metrics for each model.

Alternative Flow	**Default Metrics:** 1. Actor does not specify metrics. 2. System evaluates models using default metrics (e.g., F1-score).
Postconditions	Performance metrics are stored and ready for comparison.
Exceptions	1. Errors in the test dataset (e.g., missing labels). 2. System fails to calculate specific metrics.
Author	Shameer Hussain
Date	16-11-2024

Use Case 7: Compare Models

Use Case Title	**Compare Models**
Use Case ID	UC-07
Description	Compares the performance of trained models to identify the best-performing one.
Actors	System
Preconditions	1. Evaluation metrics for multiple models are available. 2. Actor has access to comparison tools.
Basic Flow	1. Actor selects "Compare Models" option. 2. System retrieves evaluation results. 3. Actor specifies comparison criteria (e.g., highest accuracy). 4. System ranks models based on criteria. 5. System displays the comparison results.
Alternative Flow	**Default Comparison:** 1. Actor does not specify criteria. 2. System uses default criteria to rank models.

Postconditions	Comparison results are ready for review, and the best model is selected.
Exceptions	1. Missing or inconsistent evaluation results. 2. System fails to rank models.
Author	Shameer Hussain
Date	16-11-2024

Use Case 8: Visualize Results

Use Case Title	**Visualize Results**
Use Case ID	UC-08
Description	Generates visual insights from model comparison and evaluation results.
Actors	System
Preconditions	1. Model comparison and evaluation results are available. 2. Visualization tools are integrated into the system.
Basic Flow	1. Actor selects "Visualize Results" option. 2. System generates visualizations (e.g., charts, graphs). 3. System displays visual insights for review.
Alternative Flow	**Text-Based Summary:** 1. Actor opts for textual results instead of visualizations. 2. System provides a textual summary of the results.
Postconditions	Results are visualized and ready for interpretation by the actor.
Exceptions	1. Errors in visualization tools (e.g., unsupported graph types). 2. Large datasets causing visualization delays.
Author	Shameer Hussain

Date 17-11-2024

Adopted Methodology

For this project, the **VU Process Model**, an innovative blend of the Waterfall and Spiral models, has been utilized. This hybrid approach offers the structured, step-by-step clarity of the Waterfall model while embracing the iterative, risk-focused flexibility of the Spiral model. This ensures the methodology aligns with the project's objectives and enables the development of a robust and scalable anomaly detection system for blockchain networks.

Key Features of the Adopted Methodology

1. **Phase-Wise Progression (Waterfall)**
   - Each stage, from requirements gathering to implementation, was completed sequentially, ensuring deliverables were well-documented and reviewed before progressing.
   - This systematic progression reduced ambiguity and ensured clarity in project milestones.

2. Iterative Risk Mitigation (Spiral)

- Each phase involved risk identification, alternative solution exploration, and prototyping where needed.
- Critical risks, such as handling large blockchain datasets or ensuring real-time anomaly detection, were addressed early to minimize potential setbacks.

Implementation Steps

1. **Requirement Analysis and Risk Assessment**

- Detailed requirements, both functional and non-functional, were collected and analyzed.

- Risks such as false positives in anomaly detection and computational inefficiencies were identified and mitigated iteratively.

2. **System Design**

- A modular system architecture was developed to ensure scalability and flexibility, focusing on blockchain data

- preprocessing, anomaly injection, and model training.

3. **Prototype Development and Evaluation**

- Small prototypes of anomaly detection mechanisms were built and tested on subsets of the Bitcoin dataset to validate the feasibility of machine learning models.

4. **Incremental Development and Testing**

- Modules like anomaly injection and model evaluation were developed in increments.

- Testing and evaluations were conducted after each increment to ensure compliance with performance metrics.

5. **Final Integration and Deployment**

- All modules were integrated into a single cohesive system.

- Rigorous system-level testing was performed to ensure reliability, scalability, and security.

Advantages of the VU Process Model in this Project

- **Clarity and Structure:** The Waterfall model ensured that each phase was thoroughly documented, making it easier to track progress and adhere to deadlines.
- **Risk Sensitivity:** The Spiral model's focus on risk management enabled proactive mitigation of potential project challenges.

- **Flexibility:** The iterative cycles provided the flexibility to adapt to new findings during model evaluation phases.

By leveraging the strengths of both methodologies, this project achieved a balanced approach, meeting academic standards while delivering a practical, effective solution for detecting blockchain anomalies.

Work Plan (Use MS Project to create Schedule/Work Plan)

Work Plan

The **work plan** for this project has been meticulously designed using Microsoft Project to ensure timely delivery and alignment with project milestones. The schedule encompasses all critical phases, from initial requirement gathering to the final deployment and documentation.

Key Milestones and Timeline

1. **Requirement Analysis and Planning (Week 1–2)**
   a) Identify and document functional and non-functional requirements.
   b) Conduct risk analysis and define success metrics.
2. **System Design (Week 3–4)**
   a) Develop modular architecture for the anomaly detection system.
   b) Design use case diagrams and finalize design specifications
3. **Dataset Preparation (Week 5–6)**
   a) Load, clean, and preprocess the blockchain dataset.
   b) Inject anomalies and label data for supervised learning.
4. **Model Training and Evaluation (Week 7–10)**
   a) Train machine learning models (SVM, Random Forest, AdaBoost, XGBoost).
   b) Evaluate performance metrics, including accuracy, precision, and recall.
5. **System Integration and Testing (Week 11–12)**
   a) Integrate all modules into a cohesive system.
   b) Perform rigorous testing for scalability, reliability, and usability.
6. **Documentation and Final Review (Week 13)**
   a) Prepare the final project report and user manual.
   b) Conduct reviews with the supervisor and make necessary revisions.

Gantt Chart Overview

A Gantt chart has been created using Microsoft Project to visualize the schedule, dependencies, and resource allocation. This chart includes the following details:

- **Tasks:** Clearly defined project activities, including milestones.
- **Duration:** Estimated time for each task.
- **Dependencies:** Logical relationships between tasks to ensure a smooth workflow.
- **Resource Allocation:** Assigned roles and responsibilities for each team member.

The Gantt chart serves as a comprehensive tool for tracking project progress and managing deadlines. It ensures that each phase transitions seamlessly to the next, maintaining adherence to the timeline.

Task	Start	Duration	End
Requirement Analysis and Planning	11/20/2024	14	12/4/2024
System Design	12/4/2024	14	12/18/2024
Dataset Preparation	12/18/2024	14	1/1/2025
Model Training and Evaluation	1/1/2025	28	1/29/2025
System Integration and Testing	1/29/2025	14	2/12/2025
Documentation and Final Review	2/12/2025	7	2/19/2025