Software Design Specifications

Log-Based Testing Through Machine Learning for Hospital Management Systems

Version: 1.0

Project Code	F23-301D
Supervisor	Dr. Atif Tahir
Co-Supervisor	Dr. Nouman Durrani
Project Team	Hafsa Baig 20K-1683 Maryam Raheem 20K-1700 Samia Azeem 20K-1685
Submission Date	2023-12-10

Document History

Version	Name of Person	Date	Description of change
1.1	Samia Azeem	2023-11-19	*Document Created*
1.2	Hafsa Baig	2023-12-01	*Updated Data Dictionary*
1.3	Maryam Raheem	2023-12-03	*Added State diagram 2*
1.4	Dr.Nouman Durrani	2023-12-04	*Reviewed Document and updated System Architecture*
1.5	Samia Azeem	2023-12-07	*Added Data Source*
1.6	Dr. Atif Tahir	2023-12-09	*Reviewed Document*
1.7	Samia Azeem	2023-12-10	*Final Version for Submission*

Distribution List

Name	Role
Dr. Atif Tahir	*Supervisor*
Dr. Nouman Durrani	*Co-Supervisor*

Document Sign-Off

Version	Sign-off Authority	Sign-off Date
1.4	Dr. Nouman Durrani	2023-12-10
1.6	Dr. Atif Tahir	2023-12-10

Document Information

Category	Information
Customer	FAST-NU
Project	Log-Based Testing Through Machine Learning for Hospital Management System
Document	Software Design Specification
Document Version	1.0
Status	Complete
Author(s)	Samia Azeem, Hafsa Baig, Maryam Raheem
Approver(s)	Dr. Atif Tahir and Dr. Nouman Durrani
Issue Date	2023-11-03
Document Location	N/A
Distribution	Advisor

Definition of Terms, Acronyms and Abbreviations

Term	Description
ASP	Active Server Pages
DD	Design Specification

Table of Contents

1 Introduction

1.1 Purpose of Document

Purpose of this document is to specify the design and features of the Hospital Data Anomaly Detection System. It will describe each component and how it will be implemented to comply with the requirements specified in SRS.

1.2 Intended Audience

The intended audience of this document are

- System Developers
- System Architects
- Project Managers
- Quality Assurance Testers
- Maintenance and Support Team
- Project Stakeholders

1.3 Document Convention

This document follows a standard convention:

- Font: Arial
- Font Size: 12 for the main text
- Headings: Bold for easy navigation and readability.

1.4 Project Overview

The project encompasses a development of Hospital Data Anomaly Detection System through Machine learning system which addresses specific challenges related to data anomalies, irregularities, and potential security breaches within the hospital information ecosystem. Supervised machine learning approach will be undertaken for the anomaly detection in the system logs of the hospital.

1.5 Scope

The project includes:

- User authentication and authorization
- Real-time anomaly detection
- Data Visualization/Analysis
- Notification System for critical anomalies

2 Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution for the Anomaly detection System.

2.1 Assumptions and Dependencies

Before the designing of an anomaly detection system of hospital logs, there are some assumptions and dependencies that must be identified. Following are some of the key assumptions that should be considered.

- The hospital logs are assumed to be accurate and reliable. If there are any missing values or inconsistencies in the data then data preprocessing must be done.

- Assuming patient and doctor's privacy is important, access control measures must be implemented in the system for proper authentication of the user.

- Assuming the hospital logs incorporate all major hospital activities and interactions of doctors and patients.

- There are protocols for responding to anomaly detection. Hospital IT departments must be contacted to establish alerts that are to be established for anomaly detection.

2.2 Risks and Volatile Areas

Potential risks and changes can result in new requirements and could impact the system. Following are some of the areas that can be a risk and must be considered when designing the Anomaly Detection System.

- With the advancement in technology, new more efficient algorithms and frameworks may be introduced that can enhance the anomaly detection system's performance. Therefore the system should be made flexible for future integrations of new technologies.

- Hospital's infrastructure may change in the future, therefore the system should be designed to be scalable and compatible with the possible changes.

- Hospital personnel responsible for using the anomaly system can be changed in future, therefore software architecture and functionalities must be documented for better understanding and training of new personnel.

3. System Architecture

This section provides a high-level overview of how the functionality and responsibilities of the System Log Anomaly Detection System are partitioned and then assigned to subsystems or components.

3.1 System Level Architecture

The System Log Anomaly Detection System is divided into following key components that play their respective essential roles.

1. **Data Source**
   Data Source represents the source where a stream of logs exists and provides input to the system. The data source interfaces with the data collection component.

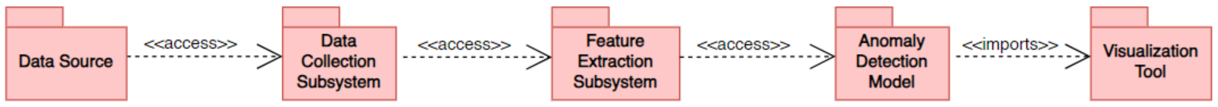
2. **Data Collection Subsystem**
   Data Collection subsystem collects and preprocesses data for further analysis. This system is responsible for cleaning, transforming and organizing data.

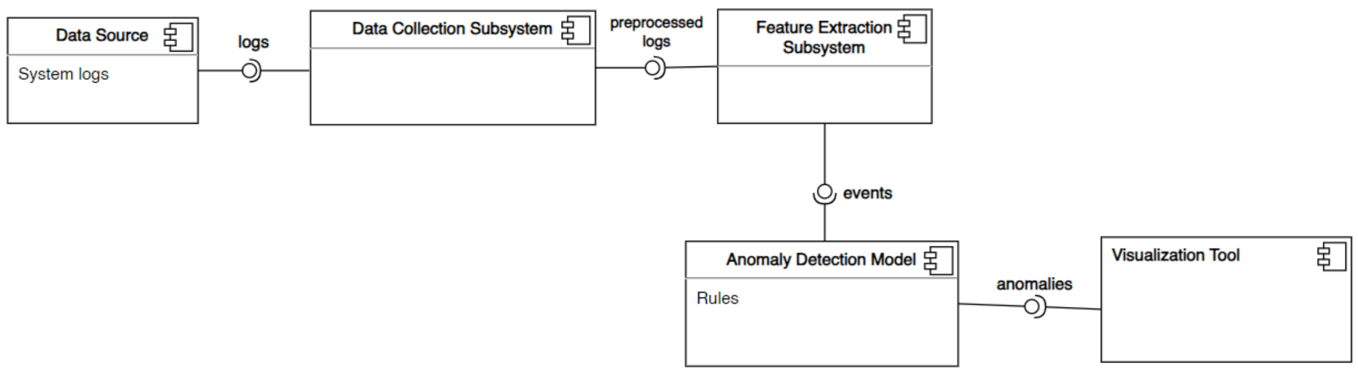
3. **Feature Extraction Subsystem**
   Feature extraction subsystem collects data from data collection subsystem and extracts relevant features and attributes from the preprocessed data to transform each log into an event. This subsystem sends events to anomaly detection model.

4. **Anomaly Detection Model**
   Anomaly detection model uses different machine learning algorithms to identify patterns in the incoming data that deviates from normal behavior. It detects anomalies and generates alerts to the stakeholder. This system connects with the feature extraction subsystem.

5. **Visualization Interface**
   This provides an interface for visualizing and interpreting anomalies. It helps users in understanding and taking decisions based on the anomalies report. It interfaces with the anomaly detection model to get the anomaly results.

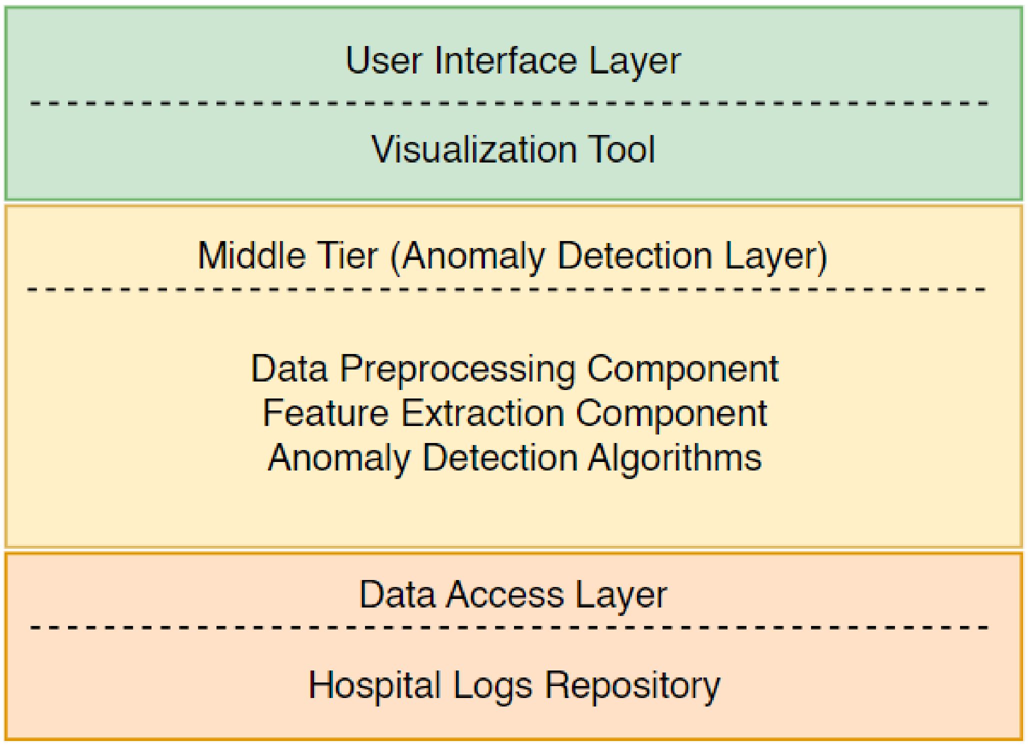
This high-level architecture provides a foundation for more detailed design work, allowing for the development of each subsystem and their interactions in a comprehensive anomaly detection system.





3.2 Software Architecture

System Log Anomaly Detection System's software architecture can be described through the following architecture diagram. It shows the user interface layer, middle tier and data access layer.


User Interface Layer
Visualization Tool
Middle Tier (Anomaly Detection Layer)
Data Preprocessing Component
Feature Extraction Component
Anomaly Detection Algorithms
Data Access Layer
Hospital Logs Repository

1. **User Interface Layer**
   User Interface consists of a visualization tool that provides user interface for the system. It displays the upcoming events along with their description. It shows the alert whenever an anomaly is identified. It provides visualization for proper understanding of the logs and facilitates decision making.

2. **Middle Tier (Anomaly Detection Layer)**
   The middle tier consists of the core functionality of the system. It is responsible for the anomaly detection of system logs and coordinating other processes.

   - **Anomaly Detection Algorithms**
     These algorithms incorporate machine learning algorithms that help in analyzing the patterns to find any abnormal behavior.

   - **Feature Extraction Component**
     This component extracts features and attributes from the system logs and transforms them to events. These events are sent to anomaly detection algorithms.

   - **Data Preprocessing Component**
     This section preprocesses the data logs for further examination. It cleans and organizes the data and sends it to the feature extraction component.

3. **Data Access Layer**
   This layer consists of the hospital system logs that are to be analyzed by the Anomaly Detection Layer. T manages the storage and retrieval of the data.

The proper Interaction flow between the layers is as follows. The user interface layer sends requests for the events to display. The middle tier processes the request and asks the data access layer for the hospital logs. It applies anomaly detection algorithms and sends back alerts to the user interface when an anomaly occurs.

The above architecture is the foundation of a scalable and flexible Hospital Logs Anomaly Detection System.

4 Design Strategy

Following are the design strategies or decisions that impact the overall organization of the Anomaly Detection System for Hospital Logs and its high-level structures. Areas that are considered are the following.

- **Future System Extensions and Enhancements**

  As discussed in risks and volatile areas, there may be system enhancements and extensions in the system. Therefore each functionality of the system i.e data preprocessing, feature extraction, anomaly detection and visualization must be developed as separate modules for modular architecture. But the tradeoff is the overhead of developing the inter module communication.

- **System Reuse**

  As best practice, the modules of the system should be developed for further reuse by system or other systems. In Anomaly Detection System, the anomaly detection algorithm must be designed as a reusable component. But our design has balanced the generality and specificity of the algorithm so it does not compromise any system requirement.

- **User Interface Paradigms**

  User Interface is designed to be user centric and aligns with the user preference. User Interface design contains graphs and tables that help users in proper understanding of the logs. But a balance of flexibility and simplicity must be considered to refrain from clutter in the interface.

- **Data Management (Storage, Distribution, Persistence)**

  Appropriate Data Storage mechanisms should be considered. Data storage mechanisms are chosen on the basis of volume and nature of Hospital Logs.

- **Concurrency and synchronization**

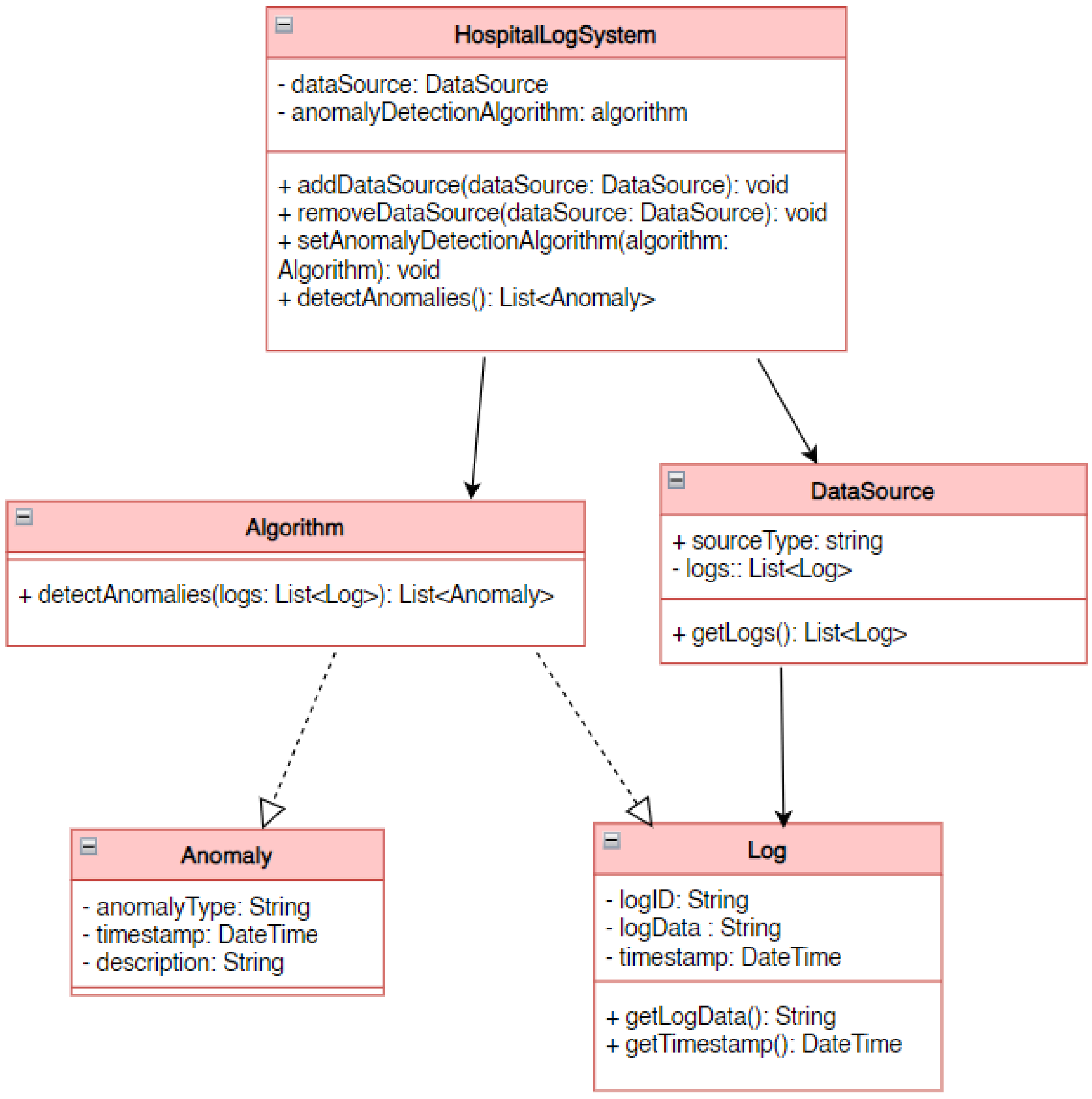
  Introducing asynchronous processing and parallelism in computationally intensive tasks may increase system responsiveness and help in system performance but it has a major tradeoff. It can introduce complexities in error handling and result coordination.

- **Security Measures**

  Hospital logs contain sensitive information about doctor and patients activities that must remain private. Major security measures should be implemented in the system to secure the log data. Proper authentication and authorization must be practiced in the system.

5 Detailed System Design

Hospital Logs Anomaly Detection System is designed considering above stated assumptions and risks. As we have discussed system architecture, now detailed system design is discussed below.


HospitalLogSystem
- dataSource: DataSource
- anomalyDetectionAlgorithm: algorithm
+ addDataSource(dataSource: DataSource): void
+ removeDataSource(dataSource: DataSource): void
+ setAnomalyDetectionAlgorithm(algorithm: Algorithm): void
+ detectAnomalies(): List<Anomaly>
Algorithm
+ detectAnomalies(logs: List<Log>): List<Anomaly>
DataSource
+ sourceType: string
- logs:: List<Log>
+ getLogs(): List<Log>
Anomaly
- anomalyType: String
- timestamp: DateTime
- description: String
Log
- logID: String
- logData : String
- timestamp: DateTime
+ getLogData(): String
+ getTimestamp(): DateTime

The above class diagram shows five classes with their respective attributes and methods.

HospitalLogSystem Class:

- addDataSource: adds a dataSource to the system
- removeDataSource: removes a dataSource from the system.
- setAnomalyDetectionAlgorithm: sets an algorithm for detecting algorithms depending on the specific requirement and use case.

- detectAnomalies: detects anomalies from the set of log through the selected algorithm

DataSource Class:
- getLogs: retrieves the logs for data source

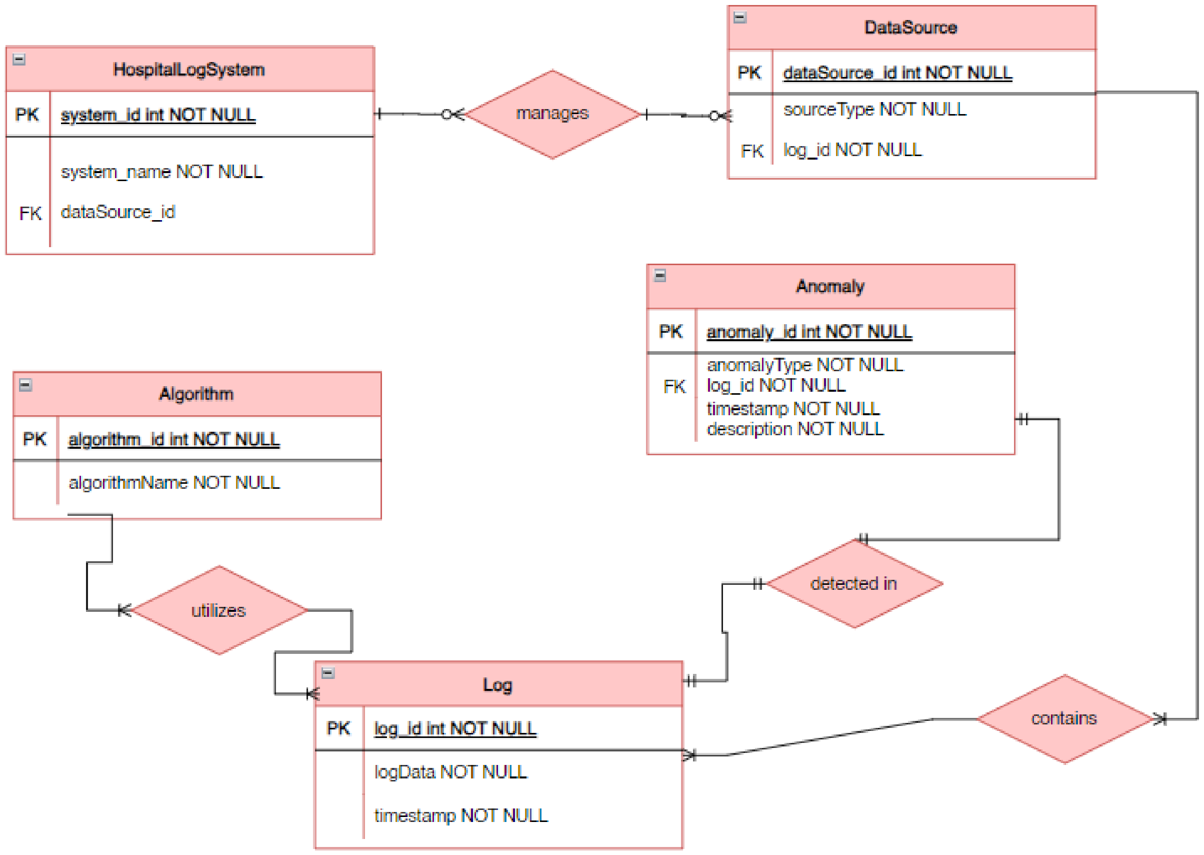
Algorithm Interface:
- detectAnomalies: an interface for different anomaly detection algorithms

Log Class:
- getLogData: retrieves log data
- getTimestamp: retrieves the time log was generated

All the relationships shown in the diagram depict association.

5.1 Database Design

5.1.1 ER Diagram



Cardinalities:

Manages (1:1) DataSource:
- A HospitalLogsSystem manages multiple DataSources.
- A DataSource is managed by one HospitalLogsSystem.

Belongs To (M:1) HospitalLogsSystem:
- A DataSource belongs to one HospitalLogsSystem.
- A HospitalLogsSystem can have multiple DataSources.

Contains (1:N) Log:
- A DataSource contains multiple Logs.
- A Log belongs to one DataSource.

Detected In (1:N) Log:
- An Anomaly is detected in multiple Logs.
- A Log can have zero or one associated Anomaly.

Utilizes (1:N) Log:
- An Algorithm utilizes multiple Logs.
- A Log is utilized by one Algorithm.

Causes (0..1:1) Anomaly:
- A Log may cause zero or one Anomaly.
- An Anomaly is caused by one Log.

Processed By (0..1:1) Algorithm:
- A Log may be processed by zero or one Algorithm.
- An Algorithm processes one Log.

5.1.2 Data Dictionary

Data Dictionary provides a detailed description of the entities along with their attributes and their characteristics. Following are the key entities of our anomaly detection system for hospital logs.

5.1.2.1 HospitalLogSystem

HospitalLogSystem	
Name	HospitalLogSystem
Alias	system
Where-used/how-used	Initialized whenever a anomaly detection system is activated or started
Content description	The content includes hospital log data, which comprises various parameters such as system_id, system_name etc. Utilizing a standardized format (csv or json).

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
system_id	*provides system id*	*int*	*10*	*not nullable*	*0*	*Primary key*
system_name	*provides system name*	*String*	*char[50]*	*not nullable*	*system*	

5.1.2.2 DataSource

DataSource	
Name	DataSource
Alias	
Where-used/how-used	Initialized as an attribute of HospitalLogSystem. Gets added or removed in the HospitalLogSystem. Contains logs.
Content description	

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
dataSource_id	*provides data source id*	*int*	*10*	*not nullable*	*0*	*Primary key*
source_type	*provides the name of the type of the source*	*String*	*char[50]*	*not nullable*	*type*	
system_id	*provides id of the system it is accessed by*	*int*	*10*	*not nullable*	*0*	*Foreign key*

5.1.2.3 Log

Log	
Name	Log
Alias	hospital logs, system logs, data
Where-used/how-used	Data Sources contains logs, anomaly detection algorithm utilizes logs. They are processed by hospitalLogSstem functions.
Content description	

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
log_id	*provides log id*	*int*	*10*	*not nullable*	*0*	*Primary key*
log_data	*provides the events that has generated the log*	*String*	*char[70]*	*not nullable*	*data*	
timestamp	*provides the time at which log was generated*	*DateTime*	*20*	*not nullable*	*00:00:00*	
data_source_id	*provides the id of data source that contains the log*	*int*	*10*	*not nullable*	*0*	*Foreign key*
algorithm_id	*provides the id of the algorithm that uses it*	*int*	*10*	*nullable*		*Foreign key*
anomaly_id	*id of anomaly that is created when anomaly is detected in lo*	*int*	*10*	*nullable*		*Foreign key*

5.1.2.4 Anomaly

Anomaly	
Name	Anomaly
Alias	abnormality
Where-used/how-used	Created when algorithm finds an anomaly in the log. It is detected in the log.
Content description	

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
anomaly_id	*provides id for the anomaly*	*int*	*10*	*not nullable*	*0*	*Primary key*
anomaly_type	*provides type of the anomaly that is created*	*String*	*char[50]*	*not nullable*	*type*	
timestamp	*provides time when anomaly is found*	*DateTime*	*20*	*not nullable*	*00:00:00*	
description	*textual description of the anomaly*	*String*	*char[70]*	*not nullable*	*description*	
log_id	*id of the log where anomaly is detected*	*int*	*10*	*not nullable*	*0*	*Foreign key*

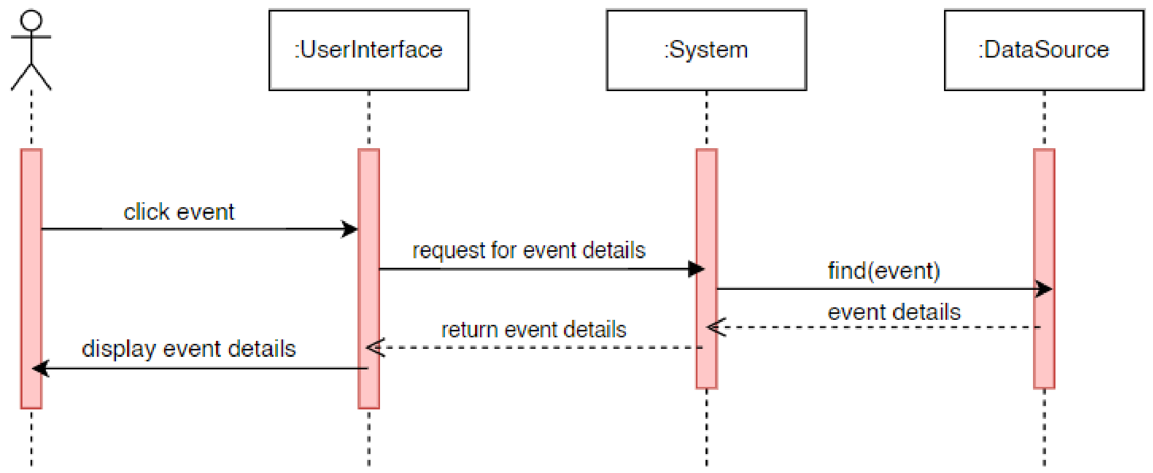
5.2 Application Design

This section describes the working of the system. It represents the chronological interactions of the objects in a Hospital Logs Anomaly Detection System at various events.

5.2.1 Sequence Diagram

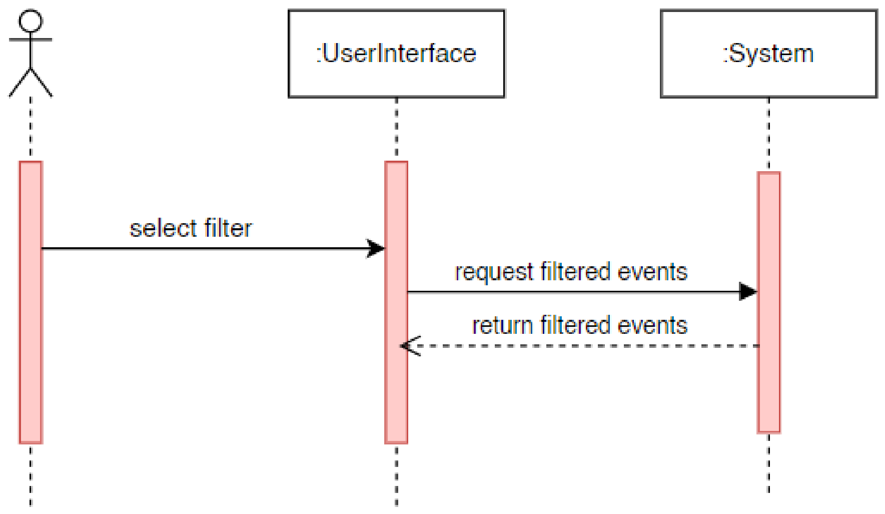
5.2.1.1 Sequence Diagram 1

Diagram shows the sequence of interactions between user interface, system and data source when the user clicks at an event and system sends back the event details.



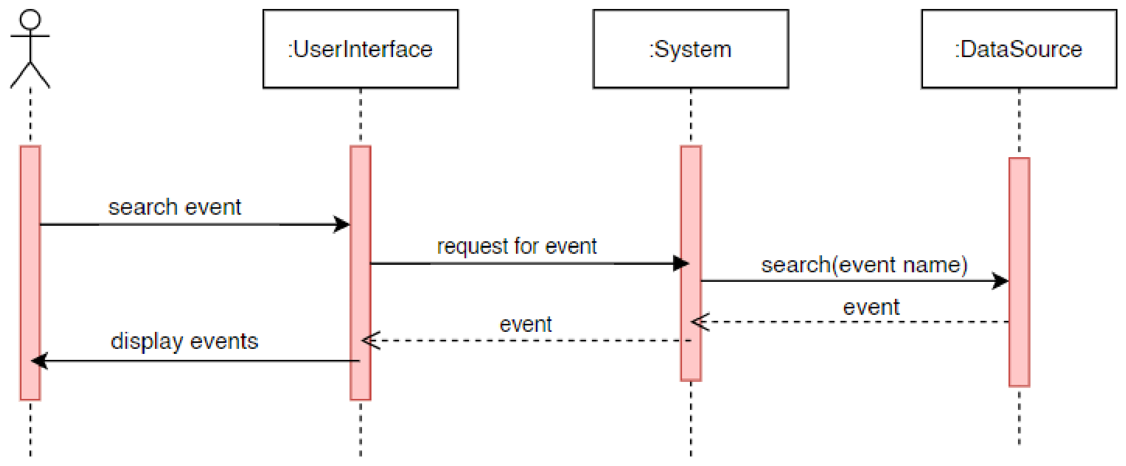
5.2.1.2 Sequence Diagram 2

Diagram shows the sequence of interactions between user interface and system when the user chooses a filter from the drop down menu for the events and system returns filtered events.



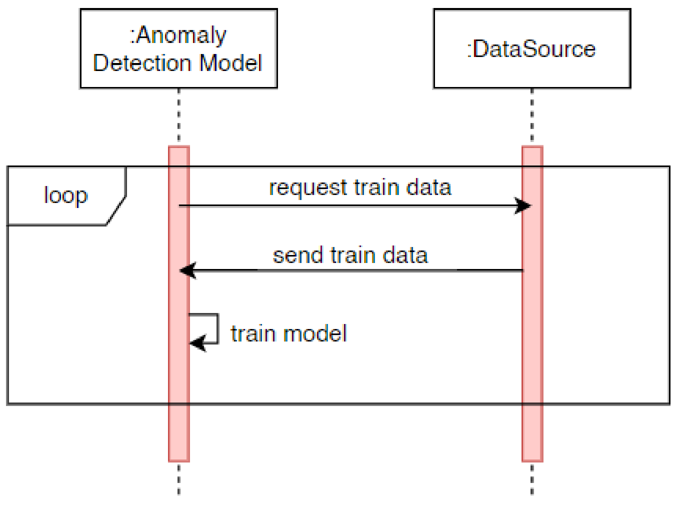
5.2.1.3 Sequence diagram 3

Diagram shows the sequence of interactions between user interface, system and data source when the user searches for an event by its name. System searches it in the data source and returns back the events that are named equal to the name searched.



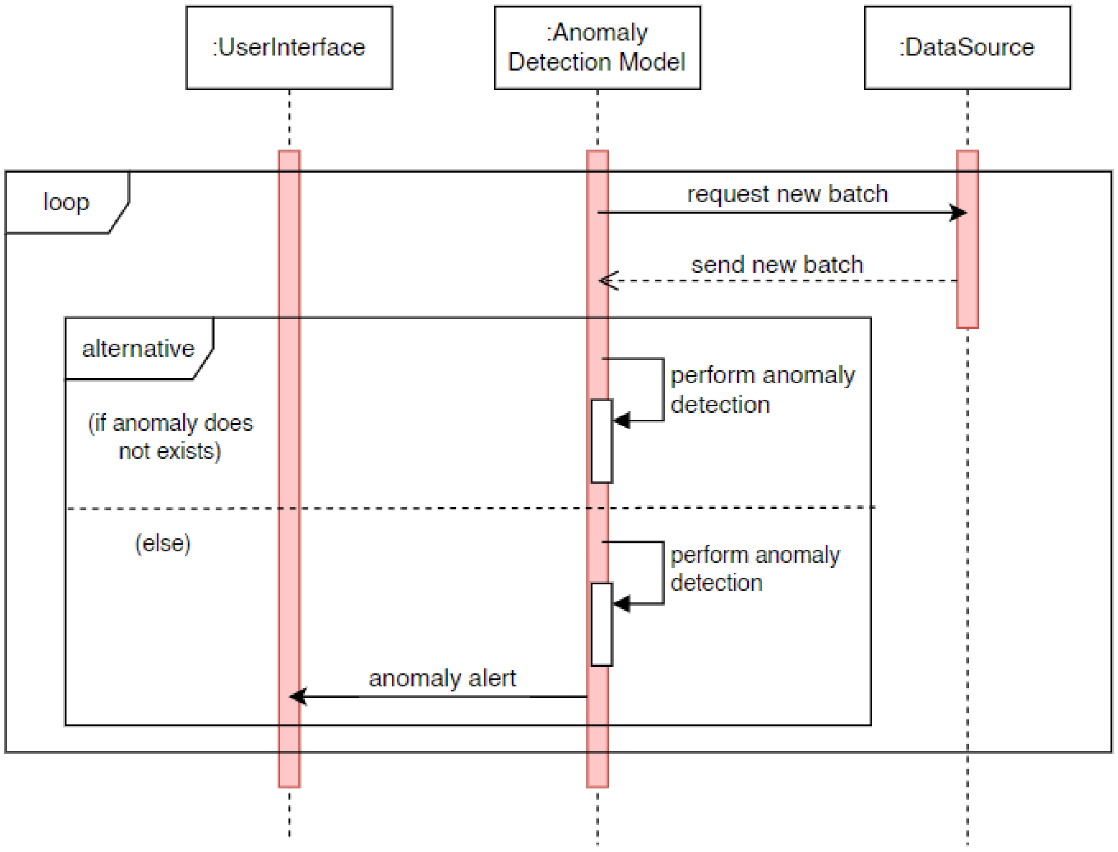
5.2.1.4 Sequence Diagram 4

Diagram shows the sequence of interactions between anomaly detection model and data source during model training phase. Data source is requested for train data for the model and it is sent back to the model. This interaction occurs in loop.



5.2.1.5 Sequence diagram 5

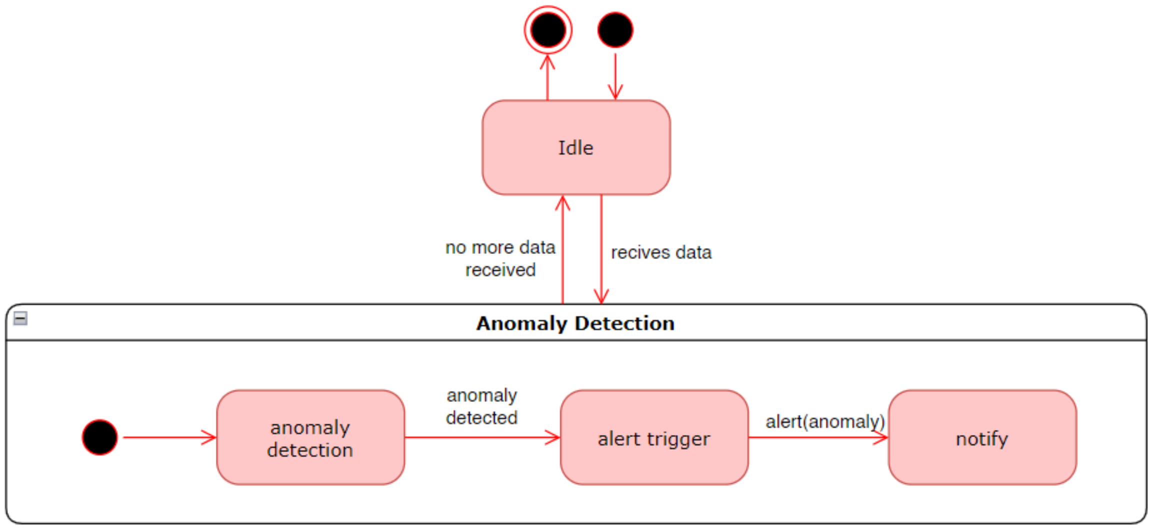
Diagram shows the sequence of interactions between the user interface, anomaly detection model and data source during the anomaly detection phase. Anomaly Detection model requests a batch of logs from data source to detect anomalies. Performs anomaly detection on the logs from the data source. This process occurs in a loop. If anomaly is detected, then an alert is sent to user interface.



5.2.2 State Diagram

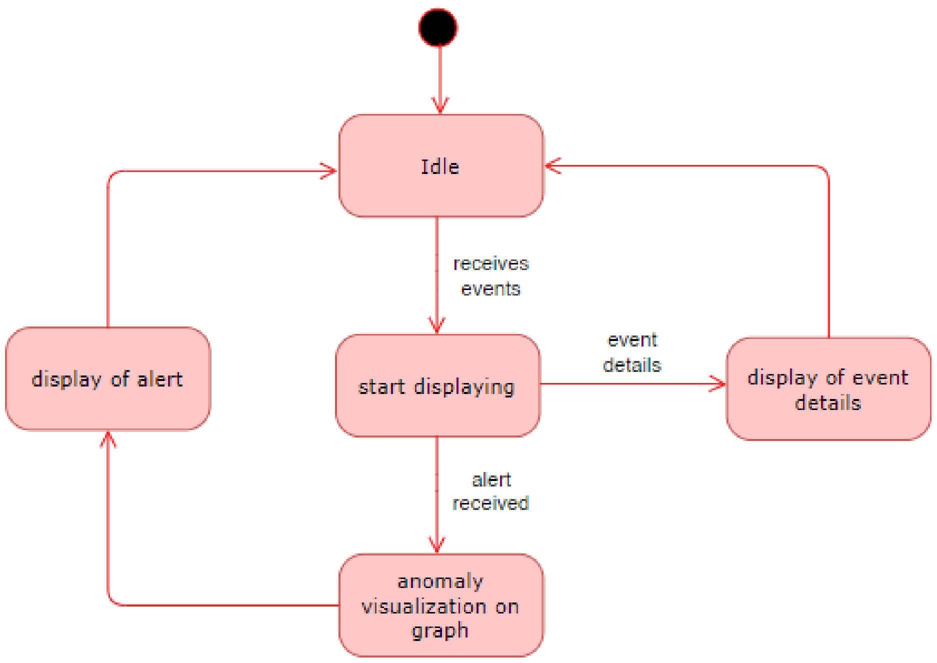
5.2.2.1 State Diagram 1

Diagram shows different states of the system during the anomaly detection process. System is idle when there is no data to analyze. Once the data is received, it goes under anomaly detection state. When an anomaly is detected, an alert trigger state is started.



5.2.2.2 State Diagram 2

Diagram below shows the states of the visualization tool during anomaly detection.



6. References

[1] Chen, M., Zheng, A. X., Lloyd, J., Jordan, M. I., & Brewer, E. (2021). Failure diagnosis using decision trees. In International Conference on Autonomic Computing, 2021. Proceedings (pp.36-43). IEEE.

[2] He, S., Zhu, J., He, P., & Lyu, M. R. (2020). Experience Report: System Log Analysis for Anomaly Detection. 2020 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). doi:10.1109/issre.2016.21

[3] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and Benchmarks for Automated Log Parsing. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)

Appendices

[Include supporting detail that would be too distracting to include in the main body of the document.]