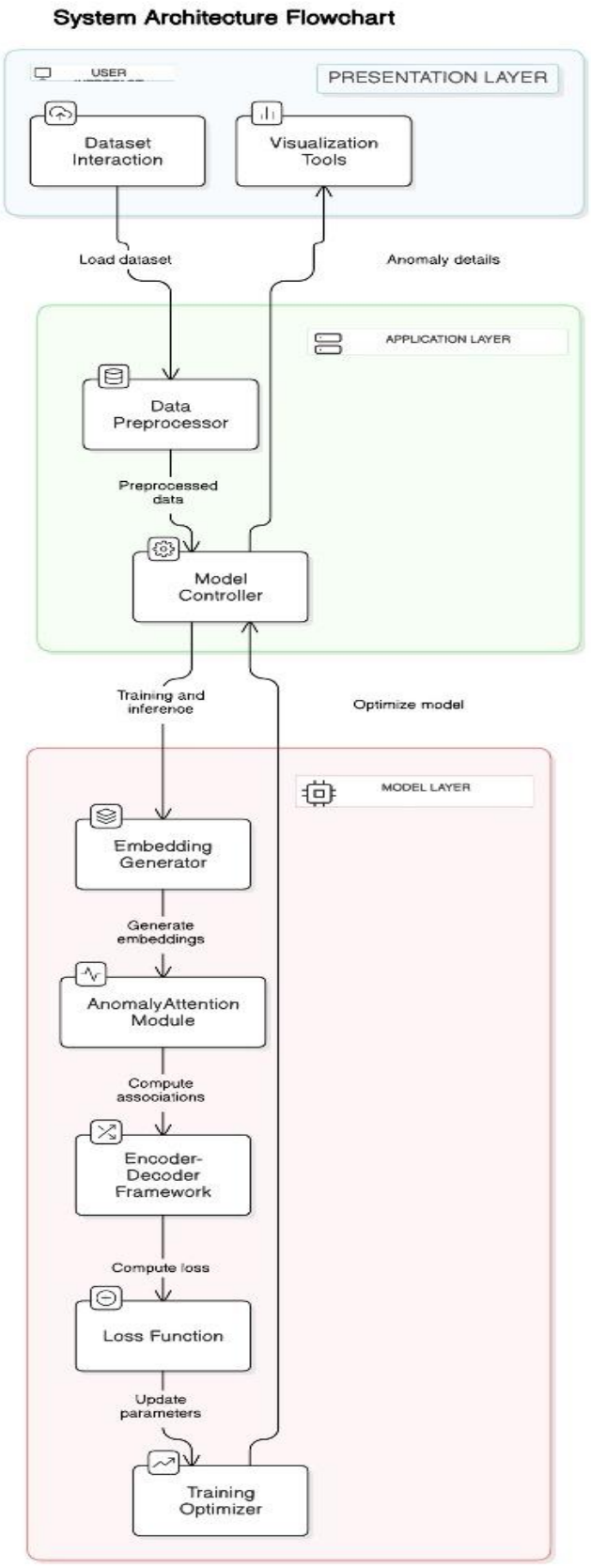
LIST OF FIGURES

ARCHITECTURE DIAGRAM



DESCRIPTION ABOUT ARCHITECTURE DIAGRAM:

→USER INTERFACE

- **Dataset Interaction:**

  Description: This is the entry point where users interact with the system to upload and manage datasets. It supports operations like dataset selection, format validation, and initial inspection of the dataset structure (e.g., headers, columns, etc.).

  Objective: Allow users to select and load datasets into the system for analysis and model training.

- **Visualization Tools**:

  Description: Tools for displaying data and anomaly details using graphs, charts, or other visual representations. These tools allow for detailed inspection of trends, distributions, and detected anomalies.

  Objective: Help users visually understand the dataset and identify anomalies or patterns effectively.

→ APPLICATION LAYER

- **Data Preprocessor:**

  Description: This module ensures the dataset is clean and formatted correctly. Key steps include handling missing values, scaling numerical values, encoding categorical variables, and transforming the data into a consistent format. It may also include techniques like time-series normalization or feature extraction.

  Objective: Ensure the data is clean and structured to improve the performance of downstream models.

- **Model Controller**:

  Description: Acts as a coordinator for the application. It handles communication between the pre-processed data and the model. It manages training, testing, and inference workflows, ensuring that the appropriate model configurations and parameters are used.

  Objective: Manage the workflow between data preprocessing, model training, and anomaly detection.

→MODEL LAYER:

- **Embedding Generator:**

  Description: Converts raw input data into low-dimensional embeddings. These embeddings are dense, numerical representations that retain the essential characteristics of the data, making them suitable for machine learning tasks.

  Objective: Generate compact and informative representations of the data for anomaly detection.

- **Anomaly Attention Module**:

  Description: Focuses on specific regions or aspects of the data that are more likely to contain anomalies, using attention mechanisms.

  Objective: To improve the precision and robustness of anomaly detection by emphasizing critical data regions or attributes that exhibit irregular behaviour.

- **Encoder-Decoder Framework**:

  Description: This architecture is designed for learning compact data representations and reconstructing the original data. The encoder compresses the input data into a latent space, while the decoder reconstructs it. Reconstruction errors are used as an indicator of anomalies.

  Objective: Measure reconstruction errors to detect anomalies, as anomalies typically have higher errors.

- **Loss Function**:

  Description: A mathematical function that evaluates how well the model is performing. It quantifies the difference between the predicted outputs and actual values or reconstruction errors.

  Objective: Optimize the model by minimizing the error and improving anomaly detection accuracy.

- **Training Optimizer**:

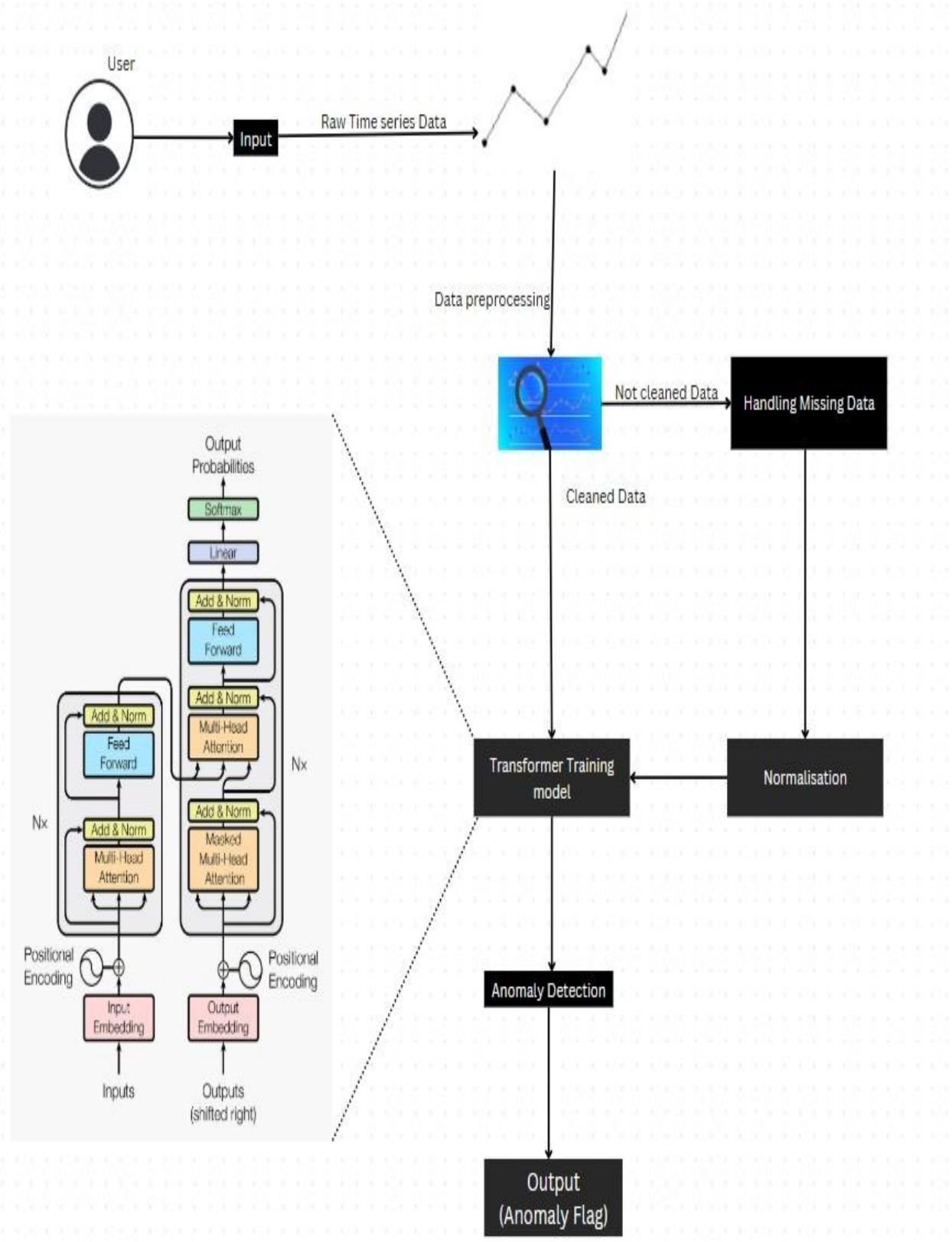
  Description: This component updates the model's weights and parameters based on the loss function using optimization algorithms like stochastic gradient descent (SGD), Adam, or RMSprop. It ensures efficient convergence to an optimal solution.

  Objective: To iteratively improve the model's ability to detect anomalies by minimizing the loss function, ensuring better performance with each training epoch.

→FLOW:

Objective: The system ensures seamless interaction between user input, data processing, model training, and anomaly detection to provide accurate and interpretable results.

DATA FLOW DIAGRAM



1. User Input

Raw Time Series Data:

The user provides the input data, typically in the form of a raw time-series dataset. This data contains observations over time and may include anomalies.

2. Data Preprocessing

Not Cleaned Data:

The raw time-series data is passed to the preprocessing stage. This data may contain issues such as missing values, noise, or irregularities.

Handling Missing Data:

Missing or incomplete data points in the dataset are identified and handled. Common strategies include interpolation, imputation, or filling with default values to ensure data integrity.

Cleaned Data:

After handling missing data and other preprocessing tasks (e.g., filtering noise or removing duplicates), the data is transformed into a cleaned and structured format, ready for the next steps.

3. Normalization

Purpose:

The cleaned data is normalized to bring all values into a similar scale or distribution. This helps in improving the performance and stability of the transformer model during training and inference.

4. Transformer Training Model

Process:

The normalized data is fed into a transformer model for training. Key components of the model include:

Positional Encoding: Adds information about the order of data points in the sequence.

Multi-Head Attention: Allows the model to focus on different parts of the sequence simultaneously to identify patterns and relationships.

Feed Forward: Applies transformations to the attended data.

Add & Norm: Combines inputs and applies normalization to stabilize training.

Output Probabilities:

The model outputs probabilities or scores representing how likely each data point is to be anomalous.

5. Anomaly Detection

Purpose:

The trained model detects anomalies in the data based on learned patterns. It flags data points that deviate significantly from normal behaviour.

6. Output

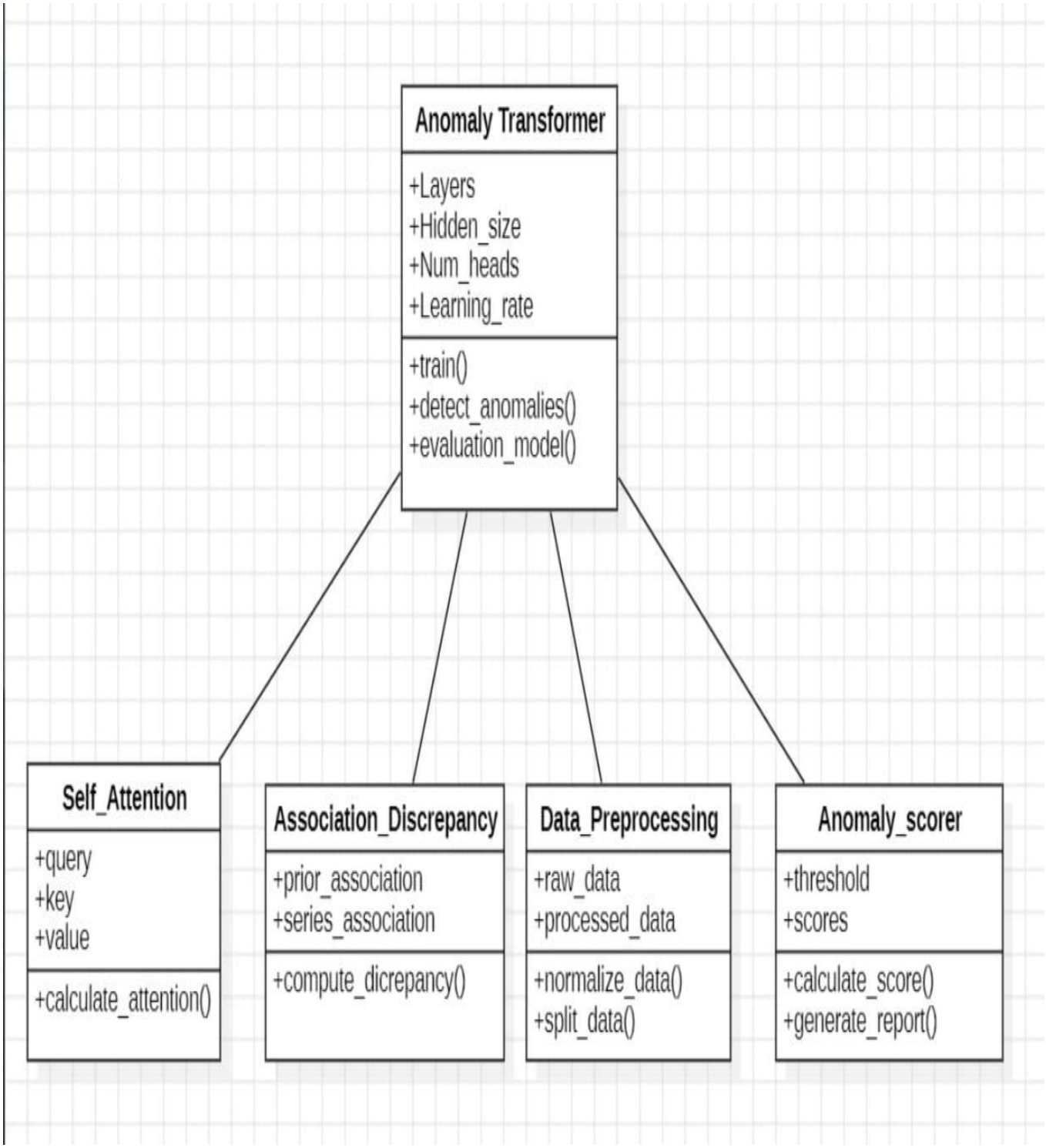
Anomaly Flag:

The final output of the system is a flag or indicator for each data point. Data points identified as anomalies are marked accordingly, providing actionable insights to the user.

Overall Flow:

The data moves through preprocessing, cleaning, normalization, and model inference, leveraging transformer-based architecture to detect and flag anomalies in time-series data. This pipeline ensures robust handling of data and accurate anomaly detection.

CLASS DIAGRAM



This class diagram represents the architecture of an anomaly detection system based on the Anomaly Transformer model. Below is a detailed explanation of each block and its components:

1. Anomaly Transformer (Main Class)

Attributes:

+Layers: Number of transformer layers in the model.

+Hidden_size: Dimension of the hidden layer in the transformer network.

+Num_heads: Number of attention heads used in the self-attention mechanism.

+Learning_rate: Learning rate for training the model.

Methods:

+train(): Method to train the model on input data using a suitable optimization algorithm.

+detect_anomalies(): Core method that utilizes the transformer architecture to identify anomalies in the data.

+evaluation_model(): Method to evaluate the performance of the anomaly detection model using metrics like precision, recall, and F1-score.

2. Self_Attention (Subcomponent)

Attributes:

+query: Represents the query matrix used in the self-attention mechanism.

+key: Represents the key matrix used in the self-attention mechanism.

+value: Represents the value matrix used in the self-attention mechanism.

Methods:

+calculate_attention (): Computes the attention weights using the formula:

\text {Attention} (Q, K, V) = \text{softmax} \left (\ frac {QK^T} {\sqrt{d_k}} \right) V

3. Association_Discrepancy (Subcomponent)

Attributes:

+prior_association: Represents the prior knowledge or patterns learned from the data.

+series_association: Captures the temporal or sequential relationships in the data series.

Methods:

+compute_discrepancy (): Compares the prior and series associations to identify anomalies by computing a discrepancy score.

4. Data_Preprocessing (Subcomponent)

Attributes:

+raw data: Raw, unprocessed input data.

+processed_data: Data after normalization and preprocessing.

Methods:

+normalize_data (): Scales the raw data to a standard range (e.g., [0,1] or standardized Z-scores).

+split_data (): Splits the data into training and testing sets for model development.

5. Anomaly_scorer (Subcomponent)

Attributes:

+threshold: The threshold value for determining whether a data point is an anomaly.

+scores: Anomaly scores assigned to each data point.

Methods:

+calculate_score (): Computes the anomaly score based on the model's output.

+generate_report (): Produces a summary or visualization (e.g., graphs) of the detected anomalies.

Relationships in the Diagram

1. Anomaly Transformer acts as the main class that coordinates the functionality of its subcomponents.

2. Self_Attention is responsible for computing attention mechanisms for anomaly detection.

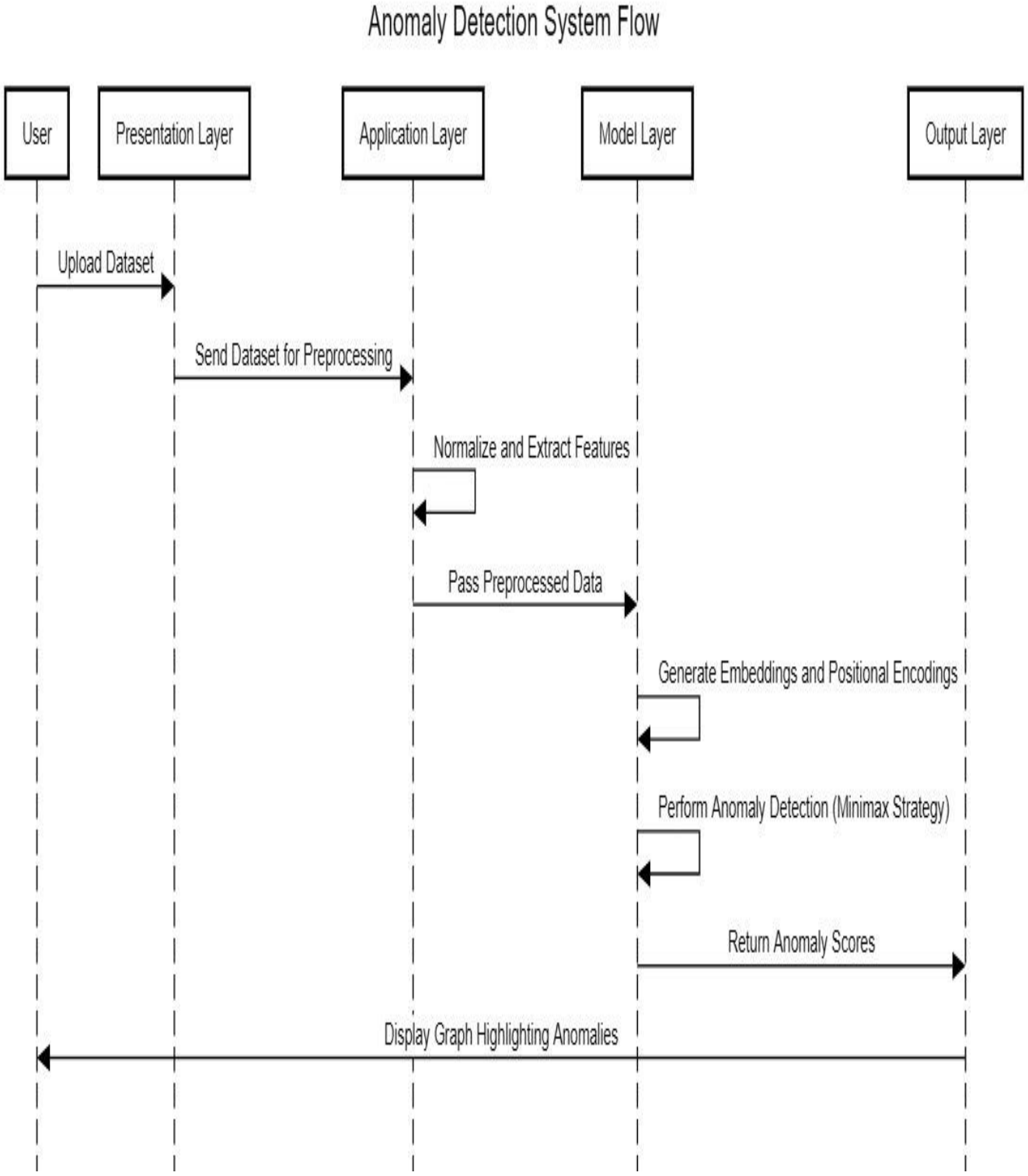
3. Association_Discrepancy measures deviations or unusual patterns in the data.

4. Data_Preprocessing ensures that the input data is prepared for analysis.

5. Anomaly_scorer evaluates and scores the anomalies detected by the model.

This architecture ensures modularity, making it easier to implement, debug, and extend the system for anomaly detection tasks.

SEQUENCE DIAGRAM



1.User

Action: Upload Dataset

The user interacts with the Presentation Layer by uploading a dataset containing the data to be analysed for anomalies.

2. Presentation Layer

Action: Send Dataset for Preprocessing

The dataset uploaded by the user is passed to the Application Layer for preprocessing, which includes tasks such as normalization, feature extraction, or data cleaning.

Action: Display Graph Highlighting Anomalies

Once the anomaly detection is complete, the output (typically a visualization, such as a graph highlighting anomalies) is displayed back to the user.

3. Application Layer

Action: Normalize and Extract Features

In this step, the dataset is pre-processed:

Normalization: Rescaling data to a standard range or distribution.

Feature Extraction: Identifying and extracting key attributes that are useful for anomaly detection.

Action: Pass Pre-processed Data

The pre-processed data is forwarded to the Model Layer for further processing and anomaly detection.

4. Model Layer

Action: Generate Embeddings and Positional Encodings

Embeddings: Represent the data in a vectorized form for easier analysis.

Positional Encodings: Capture the temporal or spatial relationship in sequential data (e.g., time-series or spatial data).

Action: Perform Anomaly Detection (Minimax Strategy)

Minimax Strategy: A statistical or machine-learning-based technique that identifies anomalies by minimizing the impact of false positives and false negatives.

This step uses the transformed data (embeddings) to detect patterns that deviate from normal behaviour.

Action: Return Anomaly Scores

The model assigns anomaly scores to each data point, indicating how anomalous it is. These scores are sent to the Output Layer.

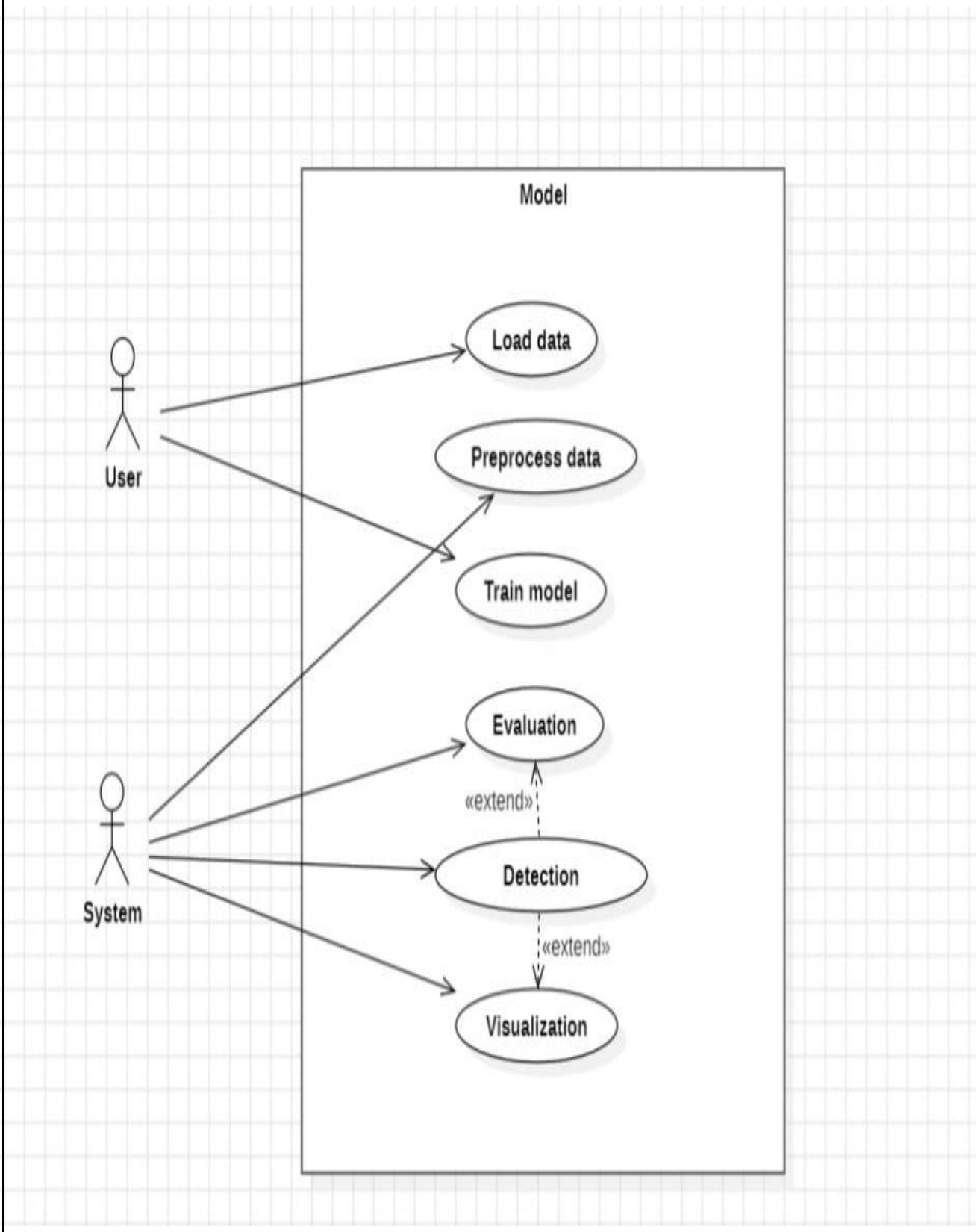
5. Output Layer

Action: Display Graph Highlighting Anomalies

The anomaly scores are visualized, typically as a graph, where anomalies are clearly highlighted for easier interpretation by the user.

This diagram shows how various system components collaborate to process the data, detect anomalies, and present results effectively. Each layer has a specific role to ensure modularity and ease of understandi

USE CASE DIAGRAM



→Actors:

1. **User:**

   Represents a human operator or stakeholder interacting with the system. Responsibilities:

   - Initiates the loading and preprocessing of data.
   - Triggers model training.
   - Utilizes the system for evaluation, detection, and visualization tasks.

2. **System:**

   Represents an automated process or module interacting with the model. Responsibilities:

   - Works with detection and evaluation processes.
   - Handles extended functionalities within the system.

→Use Cases (Functionalities within the Model):

1. **Load Data**:
   - Purpose: To import or fetch the dataset that the model will use for analysis.
   - Interaction: Triggered by the User.
   - Input: Raw data (e.g., CSV, database records, etc.).
   - Output: Data loaded into the system for further processing.
2. **Preprocess Data**:
   - Purpose: To clean, normalize, and transform raw data into a format suitable for model training.
   - Interaction: Triggered by the User.
   - Steps Involved: Handling missing values, encoding categorical data, scaling features, etc.

3. **Train Model**:
    - Purpose: To fit the model using the pre-processed data.
    - Interaction: Triggered by the User.
    - Processes: Involves selecting the algorithm, training on the dataset, and saving the trained model for later use.
4. **Evaluation**:
    - Purpose: To assess the model's performance using appropriate metrics (e.g., accuracy, precision, recall, etc.).
    - Interaction: Triggered by both User and System.
    - Extends: Detection functionalities if performance criteria need anomaly analysis.
5. **Detection**:
    - Purpose: To identify specific patterns, anomalies, or key features in the data based on the trained model.
    - Interaction: Triggered by the System.
    - Extends: Evaluation for deeper insights.
6. **Visualization**:
    - Purpose: To present data, results, or detected patterns through graphical means (e.g., charts, plots, etc.).
    - Interaction: Triggered by the User.
    - Extends: Could include both evaluation and detection outputs for enhanced analysis.

Relationships:

1. **Associations**:
    - Connect actors (User/System) to the use cases they interact with directly.
2. **Extend Relationships**:
    - Between Evaluation and Detection: Indicates that evaluation can optionally include detection tasks for additional insights.
    - Between Detection and Visualization: Suggests that detection results can optionally be visualized for better interpretability.

This diagram effectively outlines the flow and interaction for building, assessing, and analysing models.