3. Software Requirement Specification (SRS)

3.1 Introduction

The proposed system is a machine learning–driven predictive maintenance and outage management platform developed in collaboration with K-Electric (KE), Pakistan's leading power distribution company. The system aims to enhance operational efficiency by predicting transformer faults, estimating outage recovery durations, and assessing transformer health to support proactive maintenance and replacement decisions.

Currently, K-Electric experiences increased operational costs due to repeated customer complaints arising from inaccurate outage duration estimates and unexpected transformer failures. Each complaint ticket costs approximately PKR 70, and repeated complaints for the same issue further escalate expenses. The proposed solution will help reduce these costs by providing accurate fault predictions and data-driven recovery time estimates, allowing engineers to allocate resources efficiently and maintain customer satisfaction.

The system will leverage historical transformer sensor data, IoT-based real-time readings, and external weather information to generate predictive insights. Through a secure web-based dashboard, KE's operations managers will be able to view grid health status, active outages, transformer risk scores, and maintenance recommendations in real time.

3.1.1 Scope

The project's scope covers three main predictive components:

1. **Fault Prediction:** Identify which transformers are likely to fail within a given time frame using historical and environmental data.

2. **Fault Duration Estimation:** Predict the expected recovery time for an ongoing outage based on fault characteristics and weather conditions.

3. **Transformer Health Assessment:** Determine whether a transformer should be maintained or replaced based on its health score and performance history.

The system will enable K-Electric to move from reactive to proactive maintenance, optimize asset utilization, and minimize customer downtime.

3.2 Functional Requirements

3.2.1 Data Ingestion and Processing

- The system must securely connect to and ingest historical transformer data (fault logs, load data, and sensor readings) from CSV or database sources provided by KE.
- The system must support ingestion of real-time data streams from transformer-mounted IoT sensors (e.g., temperature, current, voltage, oil level).
- The system must integrate external weather data (temperature, humidity, precipitation) from an approved API such as OpenWeatherMap.
- The system must preprocess all incoming data by handling missing values, anomalies, and outliers through standardization and cleaning.
- The system must perform feature engineering to create predictive features, including time-based aggregations, load-weather correlations, and transformer health indicators.
- The processed data must be stored in a structured and queryable format (e.g., PostgreSQL or cloud-based data warehouse) suitable for model training and inference.

3.2.2 Predictive Fault Detection and Transformer Health Assessment

- The system must use an ML model to predict the probability of a fault occurring in each transformer within a future time window (e.g., next 7 days).

- The system must calculate a health score for each transformer based on historical performance, load stress, and temperature trends.
- The system must recommend whether a transformer should be replaced or scheduled for maintenance, based on its predicted failure likelihood and health score.
- The system must provide a secure API endpoint to serve health and fault predictions.
- The system must generate alerts when a transformer's fault probability or health score crosses predefined thresholds.

3.2.3 Fault Duration Prediction

- The system must use a regression-based ML model to estimate the expected duration (in minutes) for fault recovery or repair.
- The model must achieve at least 90% accuracy compared to historical recovery durations.
- The system must provide an API endpoint to return the predicted time to recovery for active faults.
- The prediction must update dynamically if new information (e.g., weather or fault severity) is received.

3.2.4 Visualization and Dashboard

- The system must provide a secure, responsive, web-based dashboard accessible to KE operations managers.
- The dashboard must show overall grid health, including total transformers monitored, active faults, predicted high-risk transformers, and maintenance recommendations.
- Transformers must be color-coded based on risk/health score (Green = Healthy, Yellow = Warning, Red = Critical).
- Each transformer entry must be clickable to view detailed information: ID, location, recent readings, historical trends, risk/health score, and fault prediction.

- The dashboard must display ongoing faults with estimated recovery durations and model confidence levels.
- The system must allow filtering and searching transformers by region, risk level, or maintenance status.

3.2.5 Model Management and Performance

- The system must maintain separate ML models for (a) fault prediction, (b) fault duration prediction, and (c) transformer health classification.
- The system must support model retraining using new data as it becomes available.
- The system must track model performance using metrics such as accuracy, precision, recall, RMSE, and F1 score.
- The system must log model predictions, retraining events, and performance metrics for auditability.

3.2.6 Continuous Data Integration and Automated Model Retraining

- The system must continuously append new real-time sensor data to the existing historical dataset, maintaining a unified data repository.
- The system must apply identical preprocessing and feature engineering pipelines to both historical and incoming data to ensure compatibility.
- The system must implement a feature store (e.g., Hopsworks, Feast) to store and manage training features.
- The system must implement a CI/CD pipeline that automatically triggers model retraining, which could be based on
  - Time-based schedules (e.g., weekly or monthly)
  - Volume thresholds (e.g., 10% dataset growth or 5,000 new records)
- The system must train new model versions on the merged dataset (historical + new data) and assign unique version identifiers to each trained model.

3.2.7 Use Case Diagram

This section presents detailed use cases of our system.

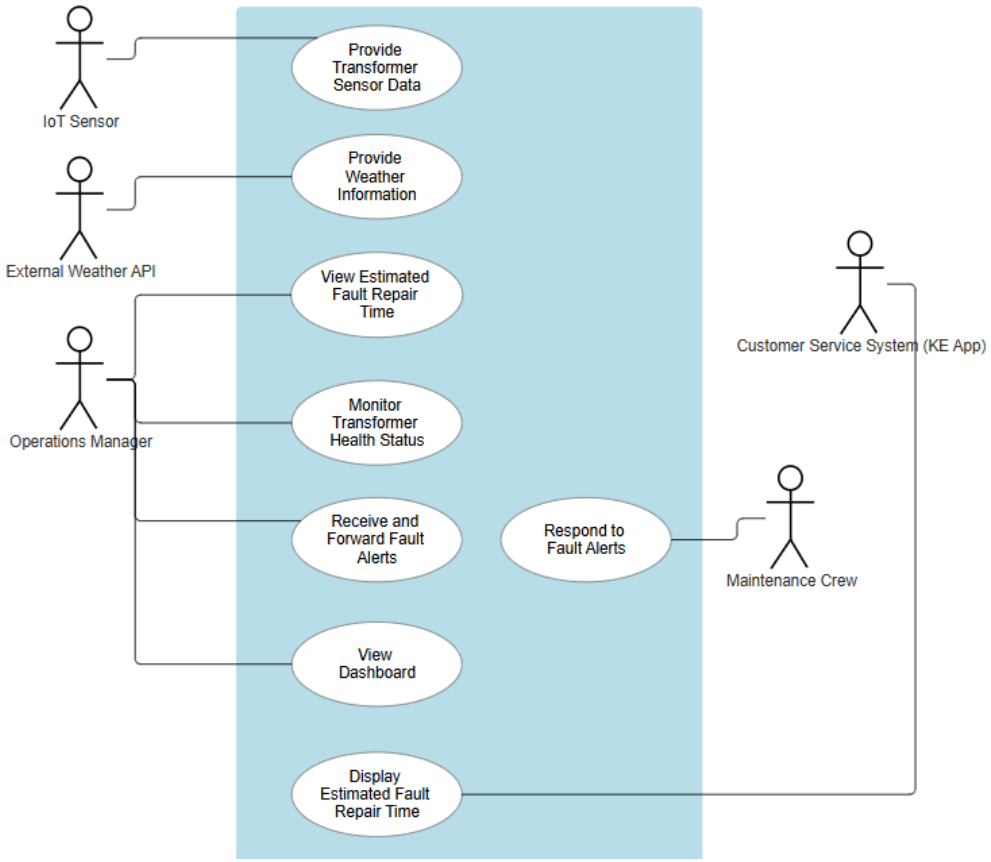

Figure 3.1: Use Case Diagram for the AI-Powered Predictive Fault Management System

3.3 Non-Functional Requirements

This section outlines the non-functional requirements that define the quality attributes of the proposed system developed for K-Electric. These include performance, scalability, reliability, security, usability, and maintainability aspects.

3.3.1 Performance

- The system must process new sensor readings and update transformer risk scores within 5 minutes of data arrival.
- The dashboard must load aggregated grid information within 3 seconds under normal network conditions.
- The ML inference API must respond to prediction requests in under 1 second on average.

3.3.2 Scalability

- The system must be scalable to handle data from up to 10,000 transformers without major performance degradation.
- The architecture must support distributed data processing (e.g., using cloud-based data pipelines or batch processing frameworks).

3.3.3 Reliability and Availability

- The system must maintain 99% uptime for its dashboard and API services.
- The system must implement automated data backup and recovery mechanisms to prevent data loss.
- The system must ensure fault-tolerant operation — if one data source or API fails, the system should continue functioning with partial data.
- Backups of processed and raw data must be taken daily and stored securely in a secondary location.

3.3.4 Security

- All communication between the system and KE servers must be encrypted using HTTPS or equivalent.
- User authentication and role-based access control must be implemented for dashboard and API access.
- Sensitive data (e.g., transformer identifiers, KE internal data) must not be stored or transmitted in plain text.

3.3.5 Accuracy and Model Quality

- The fault prediction model must achieve at least 85% recall for identifying potential faults.
- The duration prediction model must achieve at least 90% accuracy based on historical outage recovery data.
- Model retraining must be validated using KE's recent data before deployment to production.

3.3.6 Maintainability

- The system codebase must follow modular design principles for ease of debugging and updates.
- Documentation must be provided for all APIs, data pipelines, and ML workflows.
- Configuration values (e.g., risk thresholds, API keys) must be easily modifiable without code changes.

3.3.7 Usability

- The dashboard UI must be intuitive and require minimal training for KE staff to interpret.
- Alerts and status indicators must use clear color codes and concise textual explanations.
- The interface must be mobile-friendly and accessible via standard browsers.

3.3.8 Cost Efficiency

- The system must aim to reduce the number of repeated customer complaints by providing accurate outage duration estimates.
- The predictive maintenance component must prioritize transformer replacement scheduling to minimize unnecessary maintenance costs.

3.3.9 Data Integrity and Backup

- The system must ensure that all ingested and processed data is validated and free of corruption.
- Regular integrity checks must be performed on data storage to detect inconsistencies.
- Backups of both raw and processed datasets must be automatically maintained and encrypted.

3.4 External Interfaces

We expect every project to have at least of the following subsections. This section must be aligned with your project deliverables. Please consult with your project supervisor regarding which of the following section(s) you should include in your report

3.4.1 User Interfaces

This section includes our mockup screens and briefly explains them.

3.4.2 Application Program Interface (API)

This section describes the library or API interface to our system.

3.4.3 Hardware/Communication Interfaces

This section describes our project's specific hardware/network interfaces.

3.5 Datasets

This section describes the specific dataset(s) used to build our system. An appropriate snapshot of the dataset(s) is also included. Futher details, when needed, are presented in the appendix.

3.6 System Diagram

This diagram gives a high-level view of the different components of our system and the interactions between them. Each component and the particular tools/technologies/libraries used to build it are described.

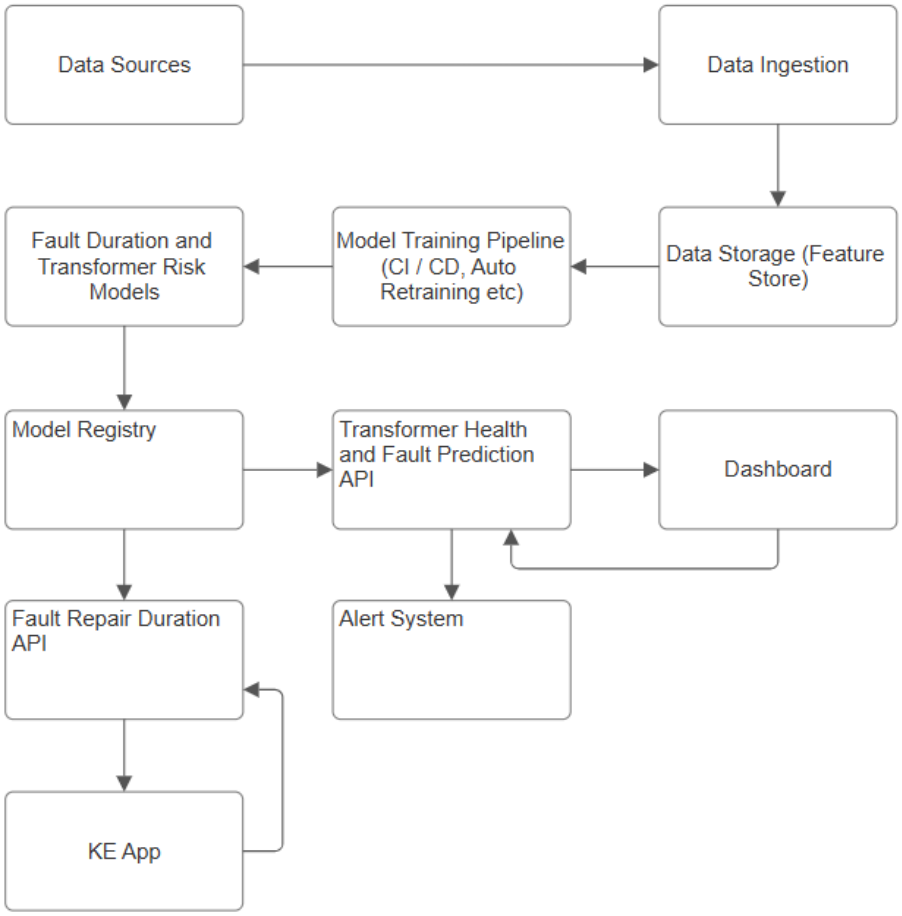

Figure 3.2: System Block Diagram for the AI-Powered Predictive Fault Management System

3.6.1 The Model Training Pipeline (Data-to-Model)

This flow describes how raw data is converted into a trained, production-ready machine learning model.

- **Data Sources:** This is the starting point, representing the external entities that provide raw data. This includes:
  1. CSV files provided by KE.
  2. Live IoT sensor data streams (voltage, current, load, temperature).
  3. The External Weather API.
- **Data Ingestion:** This service is the "front door" for all data. It collects and receives data from all data sources, processes it and forwards it for storage.
- **Data Storage (Feature Store):** This is the central repository for all clean, processed, and feature-engineered data. A logical `Data Processing` step occurs between `Data Ingestion` and `Data Storage`, where raw data is cleaned, normalized, and transformed into predictive features (e.g., rolling average, lag features etc).
- **Model Training Pipeline (CI/CD, Auto Retraining etc):** This is the automated engine that builds the models. It can be triggered on time-based events, like weekly or monthly basis, when the volume of dataset grows by a certain size or manually.
  - **Input:** It takes data from the feature store.
  - **Output:** It produces the trained model files.
- **Fault Duration and Transformer Risk Models:** This block represents the output of the training pipeline. These are the actual model files that contain the learned patterns.
- **Model Registry:** This is the secure, version-controlled repository where the final, trained model files are stored. The model registry receives the models from the training pipeline and holds them, making them available for the live APIs.

3.6.2 The Prediction Pipeline (Model-to-User)

This flow describes how the trained models are used to provide results to end-users.

- **Transformer Health and Fault Prediction API:** This service provides prediction of whether a transformer will go wrong within a time window (eg., 7 days) and the health scores of transformers.
- **Dashboard:** This is the frontend UI for the KE Operations Manager.
  - **Request:** The `Dashboard` sends a request to the API to fetch the risk scores for all transformers.
  - **Response:** The API sends the scores back to the dashboard, which then displays them as a color-coded map.
- **Alert System:** This is a background service.
  - **Input:** It constantly receives the latest risk scores from the `Transformer Health and Fault Prediction API`.
  - **Logic:** It checks if any score is above the configured threshold (e.g., > 90).
  - **Output:** If a high-risk transformer is found, the Alert System sends a notification to the dashboard for the manager to see.
- **Fault Repair Duration API:** This service predicts the expected time required to repair a transformer fault. It leverages historical maintenance data and external factors, such as weather conditions from sources like OpenWeatherMap, to generate accurate repair time estimates. The objective is to achieve a high prediction accuracy (>= 90%) to reduce inconsistencies and errors caused by manual or subjective estimations.
- **KE App:** This is the external K-Electric customer service application.
  - **Request:** When a fault occurs, the KE App sends a request to the API requesting for fault repair time estimate.
  - **Response:** The API runs its model and returns the predicted duration back to the KE App.