Project Name: **KPI Monitoring and Optimization System**

Project Overview:
The KPI Formula Parser is an essential part of the UTD-5G-Analytics system, designed to compute and parse key performance indicators (KPIs) from raw data collected in 5G network environments. This component ensures the performance data is analyzed and presented in a format that is useful for network monitoring, fault detection, and decision-making.

Purpose:

The KPI Formula Parser's goal is to take raw data from 5G network operations and transform it into meaningful KPIs that can help monitor network performance, detect anomalies, and provide insights for management.

Target Audience:
The main users of this software include:

- Network administrators managing performance metrics.
- Data analysts processing KPIs.
- Engineers responsible for system alerts and fault management.
- Management teams utilizing reports for strategic decision-making.

Scope:
The project will focus on the following areas:

- Parsing and calculating KPI formulas from raw data.
- Processing input from the data pipeline to ensure valid KPI results.
- Generating reports and visualizations for dashboards.
- Multi-tenant support with role-based access control (RBAC).

Excluded:

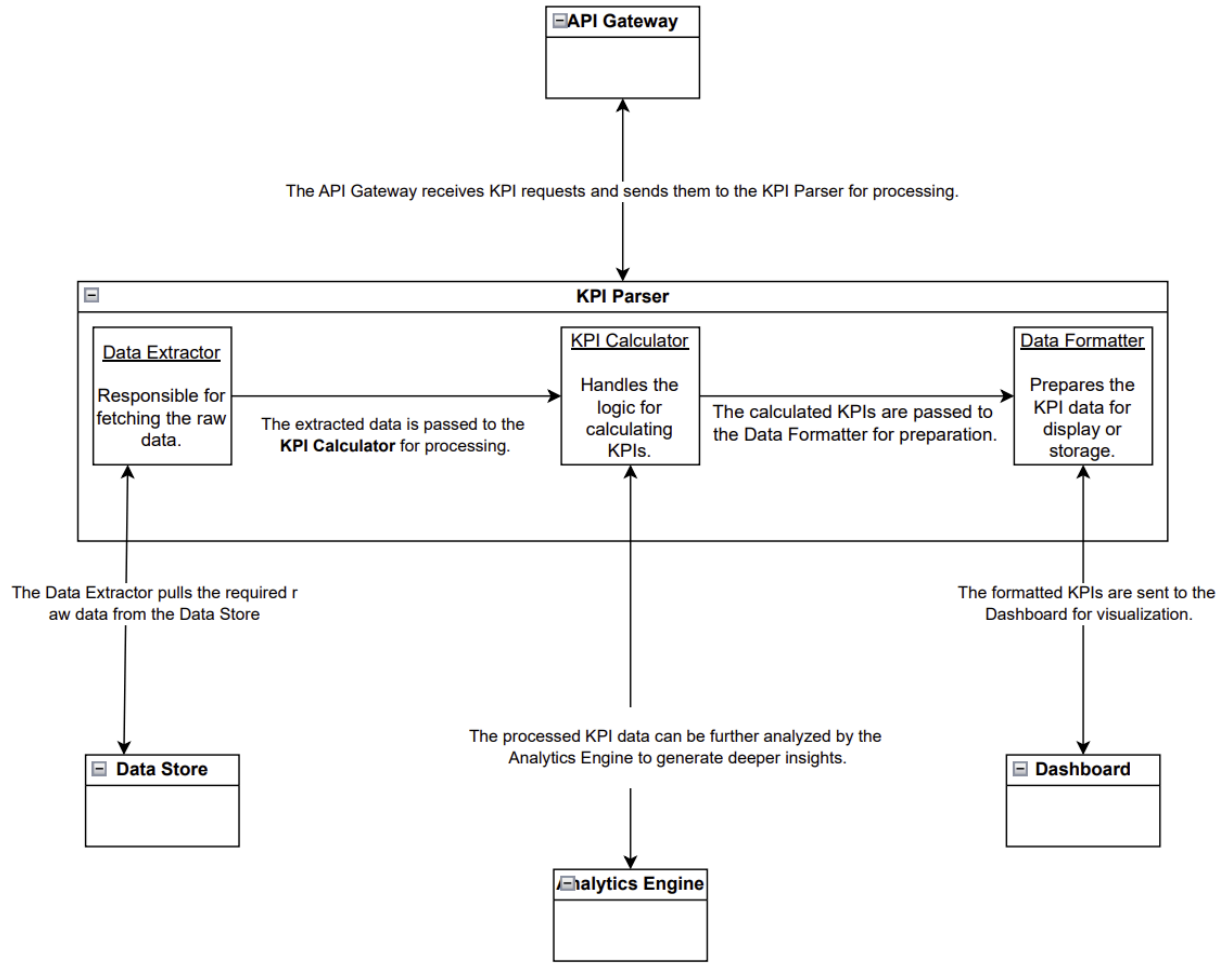
- **Direct data input by users**: The system will not allow direct user input of raw data. Instead, data will be extracted and processed automatically from the Data Store.
- **No API Development**: The focus will solely be on monitoring and optimization of API performance. API development itself is not within the scope of this project.
- **No Third-Party Integrations**: The system will not rely on external services for metrics collection or visualization. Instead, all necessary components for monitoring, data processing, and visualization will be built into the system, as depicted in the Component Diagram and Flow Diagram.

Functional Requirements:

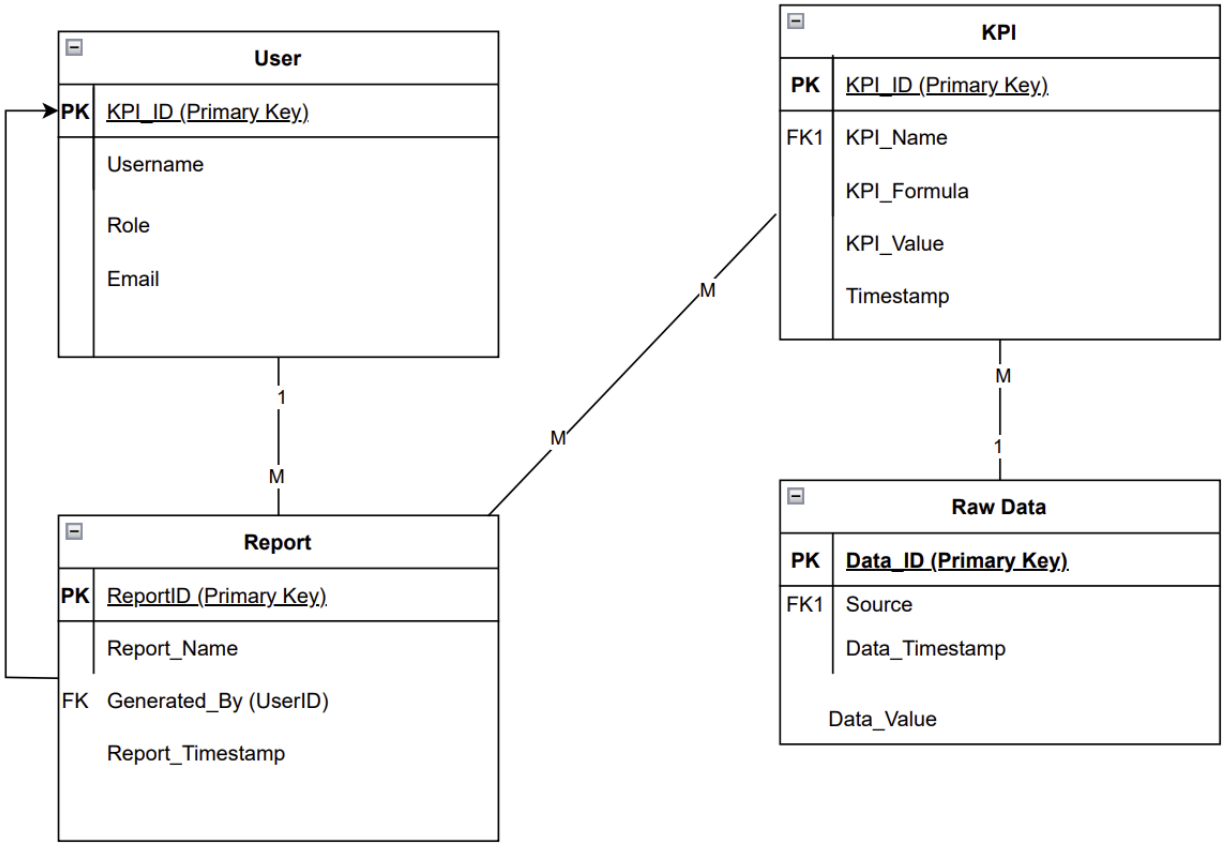
- **Data Extraction and Cleaning**:
  The system will leverage the *Data Extractor* to pull structured and cleaned data from the *Data Store*. This data will undergo a final validation check to ensure that all data points required for KPI calculation are present and consistent, thus minimizing errors during computation.

- **KPI Calculation**:
  The *KPI Calculator* will be capable of applying both simple and complex formulas to raw data, supporting both static KPI calculations and dynamic adjustments based on network conditions. The system should allow for modular KPI formula changes without affecting the overall structure of the parser. Advanced formula support, such as conditional metrics and weighted averages, will provide flexibility for users.
- **Data Formatting**:
  The *Data Formatter* will prepare the calculated KPIs for integration with both real-time dashboards and long-term reports. Formatting will include options for visual representation (e.g., graphs, tables) and compatibility with third-party analytical tools.
- **Dashboard and Reporting Integration**:
  KPI data must be displayed on a responsive dashboard via the *API Gateway*. The *Dashboard* will provide real-time KPI updates, allowing users to track performance continuously. Additionally, the system should support the generation of automated reports that can be exported for deeper analysis.
- **Interaction with Analytics Engine**:
  The system will interface with the *Analytics Engine*, which performs further analysis on the parsed KPIs. The engine will detect patterns, identify anomalies, and generate insights, providing a robust layer of post-processing.

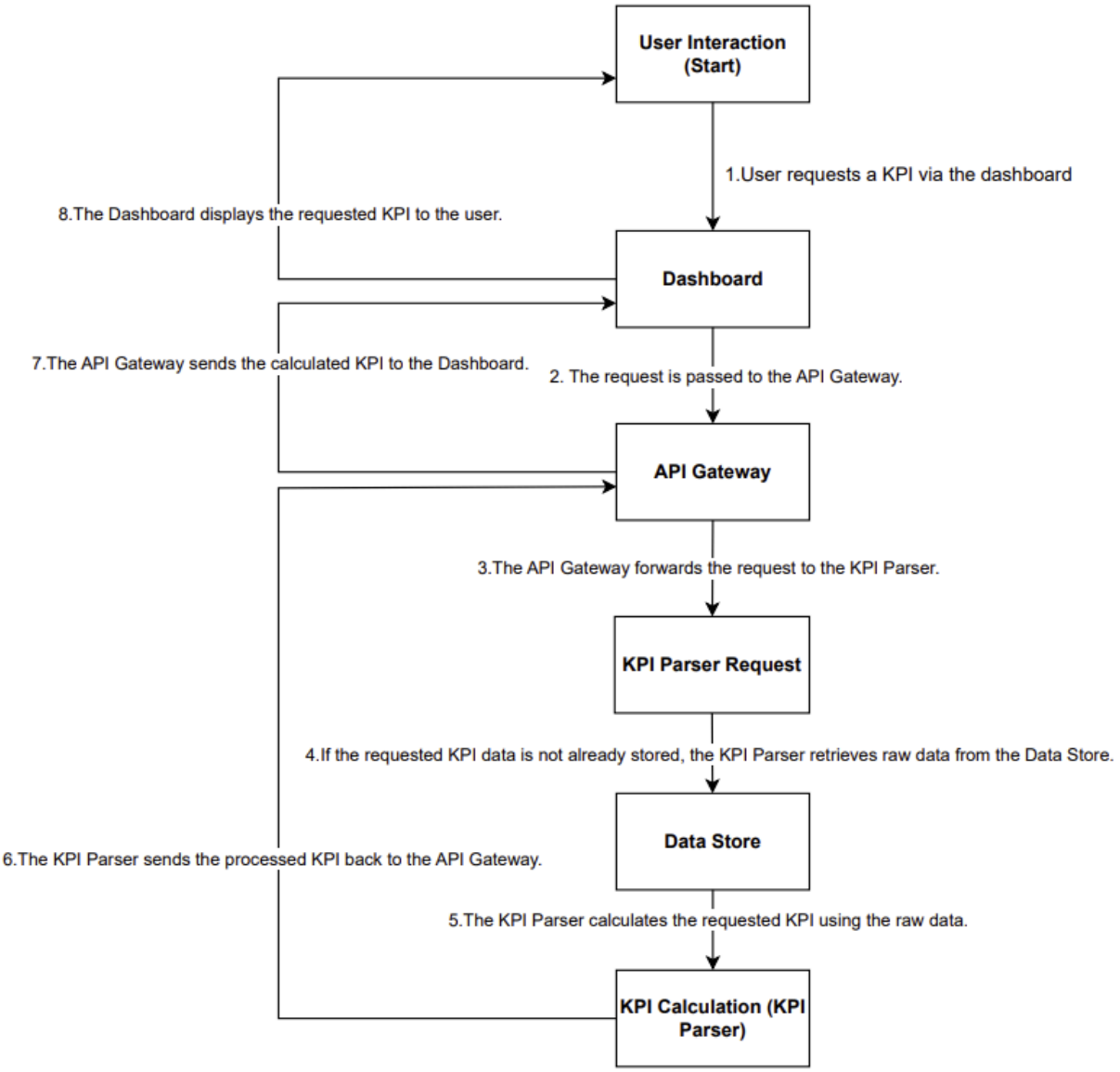
Component Diagram:



ER Diagram:


User
PK
KPI_ID (Primary Key)
Username
Role
Email
KPI
PK
KPI_ID (Primary Key)
FK1
KPI_Name
KPI_Formula
KPI_Value
Timestamp
Report
PK
ReportID (Primary Key)
Report_Name
FK
Generated_By (UserID)
Report_Timestamp
Raw Data
PK
Data_ID (Primary Key)
FK1
Source
Data_Timestamp
Data_Value
1
M
M
M
M
1

Flow Diagram



Components:

- **User**: Initiates the KPI request from the dashboard.
- **Dashboard**: Sends user requests to the API Gateway and receives the final KPI data.
- **API Gateway**: Handles requests from the dashboard, forwarding them to the appropriate service (KPI Parser).
- **KPI Parser**: Processes data and calculates the requested KPI.
- **Data Store**: Provides raw data to the KPI Parser and stores the calculated KPIs.

Non-Functional Requirements:

- **Performance**:
  The KPI Formula Parser must maintain low-latency processing, particularly during peak operational periods. The system should support at least thousands KPI calculations per minute without noticeable degradation in performance. Data caching mechanisms should be considered for frequently accessed KPIs to reduce load on the *KPI Calculator*.
- **Security**:
  RBAC must be strictly enforced to ensure that different tenants only access their respective data. All interactions with the *API Gateway* should be encrypted, and OAuth 2.0 must be implemented for authentication. In multi-tenant environments, data isolation and protection are critical.
- **Usability**:
  The dashboard interface should be user-friendly and adaptable to different user roles. The KPI visualizations should be interactive, enabling users to drill down into specific metrics and dynamically adjust their view to different timeframes or conditions.
- **Reliability**:
  The system must ensure high availability and fault tolerance, designed to recover quickly in case of system failure. Data integrity must be maintained even in the event of network or system disruptions, ensuring no loss of critical KPI information.
- **Scalability and Maintainability**:
  The KPI Formula Parser should be designed to scale both vertically and horizontally, capable of handling increasing data loads and growing numbers of users. Modular code structure will allow individual components (e.g., *KPI Calculator*, *Data Formatter*, *Data Extractor*) to be maintained or updated independently without affecting the overall functionality of the system.

Assumptions and Dependencies:

Assumptions:

- The system assumes that the APIs being monitored are already optimized to some extent, and the monitoring solution is used to track improvements or detect performance issues.
- It is assumed that all external APIs and services integrated with the system, such as Prometheus, OpenTelemetry, and Grafana, will remain available and functional throughout the system's lifecycle.
- Users of the system are familiar with the basic concepts of KPI monitoring and API performance, and require minimal training to interpret the dashboards and metrics.
- The security model (OAuth 2.0) is assumed to be compatible with the APIs and services in use, and no major compatibility issues will arise.
- It is assumed that the system will primarily be used in a stable network environment with sufficient bandwidth to handle real-time data transmission and monitoring.

Dependencies:

- The system relies on ELK Stack (ElasticSearch, Logstash, Kibana) for logging and log analysis. The continued availability and performance of these tools are critical for system functionality.
- OpenTelemetry is used for tracing and is a key dependency for visualizing how requests move through the system. Any issues or outages with OpenTelemetry would affect tracing capabilities.
- Prometheus is required for pulling metrics data from the system and setting up alerting rules. It also depends on time-series databases like InfluxDB for storing and aggregating data.
- Grafana is used for visualizing all collected metrics and logs. The system depends on Grafana's dashboard for user-friendly access to KPI data.
- The system assumes access to a stable and secure network for both internal and external data exchanges, as network instability could lead to incomplete data or delayed alerts.

Acceptance Criteria:

- The KPI Formula Parser accurately computes KPIs based on predefined and dynamic formulas.
- Calculated KPIs are displayed in real-time on the dashboard and are available for further analysis via the *Analytics Engine*.
- Security protocols, including RBAC and OAuth 2.0, are implemented, and data is securely handled.
- The system can handle thousands of KPI calculations per minute with minimal latency and no significant performance drops.

System Reliability and Scalability

- The system should be highly available and able to run continuously without frequent restarts or manual interventions, provided the network conditions are stable.
- It should support multiple services and APIs, scaling up to handle large data volumes and request traffic while maintaining stable response times, especially in a 5G environment.
- Monitoring and analysis tools like Prometheus and Grafana should seamlessly scale to handle the addition of more services and data in the future.

Security

- The system must implement OAuth 2.0 for authentication and access control, ensuring secure data transmission. All external API interactions should occur over secure channels like HTTPS.
- It should meet the security standards defined in the system design, preventing unauthorized access to or modification of KPI data.
- The system should be capable of defending against common network attacks, such as SQL injection or DDoS attacks, when handling sensitive data.

User Experience

- The system interface should adhere to usability standards, with clear and intuitive dashboards displaying all KPI, logging, and metric information for easy user access.
- Alerts and notifications within the interface should be easily understandable, helping users identify and resolve API performance issues.
- Users should have the flexibility to define KPI thresholds, set custom alert rules, and view historical performance data at any time.

Maintainability

- The system codebase should be easy to maintain and update, allowing new features or fixes to be integrated without disrupting existing functionality.
- All external tools (e.g., Prometheus, OpenTelemetry, ELK Stack) should be properly documented to facilitate future team maintenance and system expansion.
- The system should include comprehensive logging and error tracking, allowing issues to be easily identified and resolved through logs and tracing tools.

Additional Considerations:

- Integration with External Sources: The parser should be adaptable to work with various data sources, including external APIs.
- Compliance: The system should consider basic security and privacy regulations for handling multi-tenant data.