CSc4351 CS Capstone 1
Software Design Document
Fall 2024

(Team DS)

Data Pipeline For Space Weather Forecasting

Team Members:

1. Aimen Rehman
2. Rithwik Kamalesh
3. Sadia Akhter
4. Tom Tran
5. Rochelle Daley

Date: (12/12/2024)

Data Pipeline For Space Weather Forecasting

Table of Contents

1. Introduction

1.1 Purpose

The system integrates an ensemble of models that process data from magnetograms, EUV images, X-ray time series, proton flux time series, and physical parameters from active region patches. It is engineered to assess data quality, ensure data availability, and generate accurate predictive analytics. The results are delivered via an interactive web based interface, providing visualizations and insights into potential space weather impacts on key systems such as power grids, aviation, communications, and astronaut safety.

This is important because space weather can cause blackouts, disrupt GPS signals, affect aviation routes, and harm astronauts. Predicting these events allows for proactive measures to mitigate their impact.

1.2 Scope

The Data Pipeline is designed to facilitate the management of data flow, ensuring correct formatting, cleaning and availability of the data. As well as outlining steps to to integrate it with relevant data warehouses and present the information via readable and engaging data visualizations. Our project covers designing the pipeline, outlining necessary steps for its implementation and presenting mockups of the final dashboard/website.

1.3 Definitions, Acronyms, and Abbreviations

- **DP**: Data Pipeline is a systematic series of components designed to automate the extraction, organization, transmission, transformation, and processing of data from one or more sources to a specific destination.
- **UI**: User Interface is the way users interact with the information systems.
- **API**: Application Programming Interface is a set of programming code that allows data transmission from one software product to another.
- **EUV**: Extreme UltraViolet is the region of the electromagnetic spectrum between visible light and X-rays.
- **HMI**: Helioseismic and Magnetic Imager is to study the convection-zone dynamics and the solar dynamo, the creation and evolution of sunspots, active regions and complexes.

1.4 References

- NOAA, "Space Weather Prediction Center," *NOAA Space Weather Prediction Center*, [online]. Available: https://www.swpc.noaa.gov/.
- K. John, “Guide to Data Pipelines: Definition, Components, and Examples,” *Striim*, [online]. Available: https://www.striim.com/blog/guide-to-data-pipelines/.
- E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley, 1994.

1.5 Overview

This document is structured to provide a detailed overview of the DP, including its functional and non-functional requirements, system design, test cases, and an appendix going over specific details. Each section is designed to give users a clear understanding of the system's capabilities and design considerations.

2. System Overview

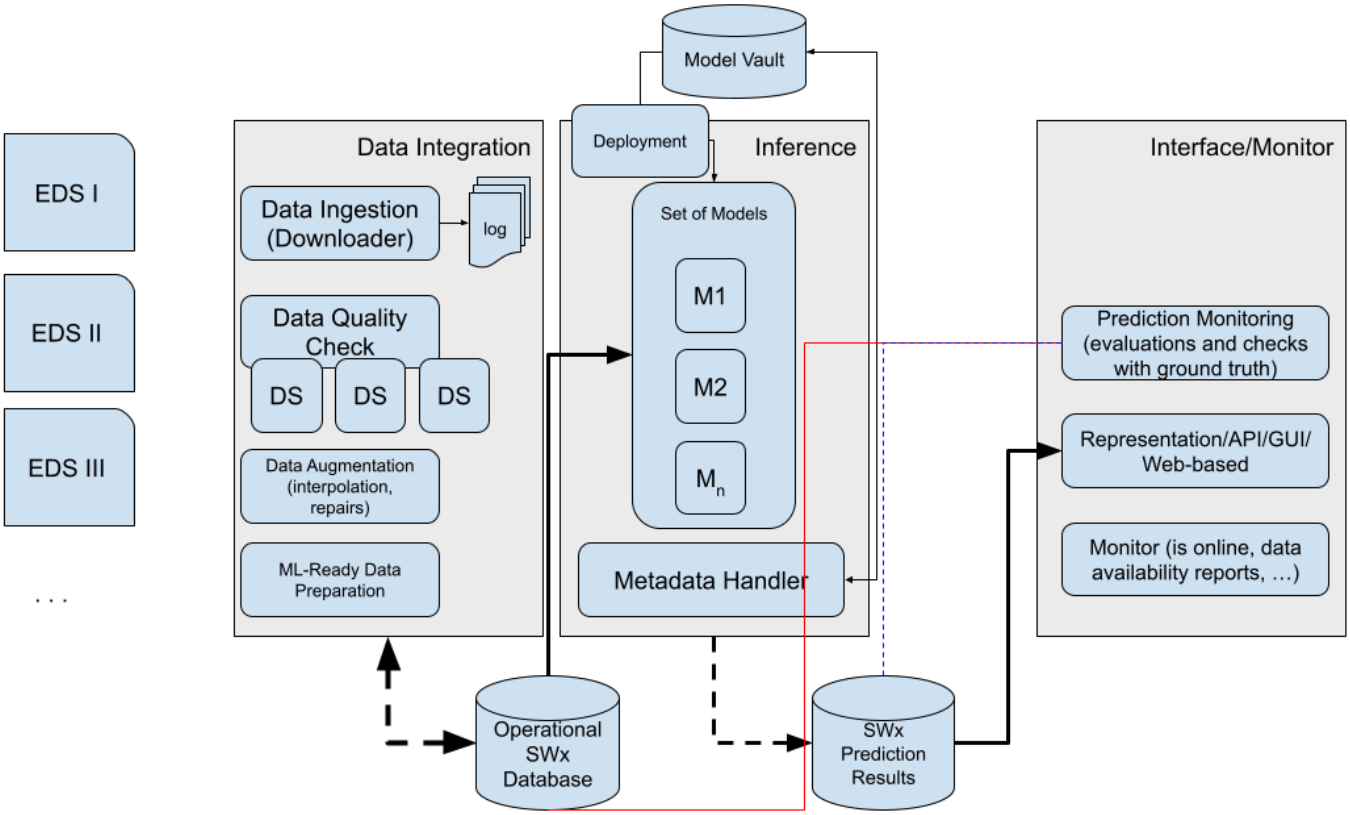
2.1 System Architecture

The DP will be developed using a four-tier architecture consisting of:

- **Presentation Layer**: This is our user interface that interacts with users (students, and research institutions, and space enthusiasts).
- **Destination Layer**: This is where our model outputs will be stored.
- **Data Processing Layer**: This where our data will be processed extensively and formatted to an inference ready state.
- **Source Layer**: This is where our data will be extracted from external sources.

2.2 System Context

The DP website will operate in a web-based environment, accessible via modern web browsers. The pipeline itself will interact with a cloud based data warehouse to store and retrieve model outputs, then display them to users. The following diagram illustrates the system context:



3.0 Functional Requirements

Prep Stage Requirements

• Data must be cleaned continuously and in real time, as the pipeline will be pulling data from external sources at predetermined intervals.

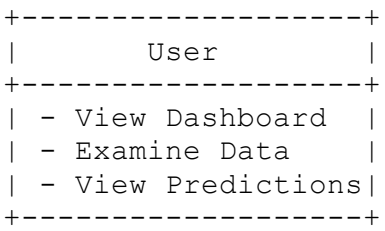
• All data fed to the models must be compatible with all times synced to each other or grouped together to avoid inconsistencies in model output.

• Magnetograms and similar data must be packaged as JSON to ensure they can be read by the model without causing errors and inconsistencies.

• Data downloader must be connected to a data log, so any server connection problems are immediately reported, ensuring the model doesn't run without the appropriate data.

3.1 Use Case Diagram

The following use case diagram illustrates the interactions between users and the DP, these use cases are specific to front facing operations:



3.2 Use Cases

1. **View Dashboard**
    - **Actor**: User (students or researchers)
    - **Description**: The user views the main dashboard, which provides analytics about the space weather such as visualizations and predictions.
    - **Preconditions**: The user must be logged in and select their role.
    - **Postconditions**: The dashboard displays the latest visualizations and predictions.
2. **Examine Data (Graph Data)**
    - **Actor**: Researchers
    - **Description**: The user selects specific datasets (e.g., magnetograms) to examine detailed visualizations and data.
    - **Preconditions**: The user must be logged in and have access rights.
    - **Postconditions**: The visualizations are interactive for any selected datasets, such as zooming or filtering.
3. **View Predictions (Any predictions our models will make)**
    - **Actor**: Researchers
    - **Description**: The user accesses model predictions for space weather forecasting.
    - **Preconditions**: The user must be logged in and have authorization.
    - **Postconditions**: Interactive prediction models are displayed.

4. Non-Functional Requirements

With the non-functional characteristics of weather research system, factors which are included are the precision of data, the rate at which updates occur, the dependability of the model, the speed of processing, the ease with which data can be accessed, the system's ability to operate on various platforms, the capacity to manage extensive datasets, the ease of maintaining the research system, and the usability of the user interface. These aspects focus on the efficiency and effectiveness of the research system rather than the specific meteorological elements it examines.

Key considerations regarding non-functional characteristics:

1. Not directly tied to primary functions: Unlike functional requirements that define the tasks a system should accomplish, non-functional characteristics outline the quality of its performance.

2. Influence on user engagement: Aspects such as how quickly data is updated, the time taken to process information, and the simplicity of access significantly affect user interaction with the weather research system.

3. Crucial to system architecture: These characteristics must be integrated into the development phase to ensure the system achieves the necessary performance benchmarks.

Examples of non-functional characteristics in weather research include:

1. Data precision: How accurate is the weather information that was generated by the model?
2. Frequency of updates: The interval at which the weather model revises its forecasts.
3. Model dependability: How reliably the weather model operates under varying conditions.
4. Processing efficiency: The speed at which the system processes and produces weather predictions.
5. Scalability: The system's capability to handle growing volumes of data or complex weather situations.
6. Data accessibility: The ease with which users can access the forecasted weather data.
7. Interoperability: The system's ability to function with other weather data sources or platforms.
8. Usability of user interface: The intuitiveness and simplicity of system navigation for users.

4.1 **Performance Requirements**

1. **Real- Time data ingestion**
    - **Requirement:** The pipeline has to ingest data from multiple reservoirs in real time. This will help the predictions to be continuous and timely.
    - **Specification**
        i. Ingestion Throughout: The system should be able to handle 15 MB/second.
        ii. Concurrent Sources: The system should support up to 5 concurrent data streams.
        iii. Tools: Apache Kafka (better used for data streaming), to maintain high availability and low latency.
    - **Error Handling:**
        i. For error handling, using retry mechanisms for disturbed data streams would work well.

ii. System will also log ingestion failures.

2. **Scaling**
   - **Requirement**: The system should be able to scale properly to handle high amounts of data volumes and concurrent operations.
   - **Specification**
     i. Data volume: Should process up to 1 TB of raw data per day.
     ii. Concurrency: It should handle at least 15 simultaneous model executions.
   - **Infrastructure**
     i. For our infrastructure we will use Amazon S3 for raw data storage and google bigQuery for data warehousing.
     ii. We will also use lightweight environments (Docker) to distribute workloads among several nodes.
   - **High resolution Processing:**
     i. Supports high resolution full disk magnetograms (4096x4096 pixels), less than 2 seconds an image.

3. **Response time**
   - **Requirement**: The system should be able to display predictions to users with a fast and efficient time frame. This will help real time decision making.
   - **Specifications**
     i. Total response time: Predictions, visualizations and less than 5 second returns.
   - **Time breakdown**
     i. Data Fetching: less than 1.5 seconds
     ii. Data preparation: less than 1 second
     iii. Model inference: less than 2 seconds
     iv. Visualization rendering: less than .5 seconds

   - **Optimizations:**
     i. We will use GPU acceleration for model inference, examples include TensorFlow with NVIDIA GPUs.
     ii. The system will also cache regularly accessed data and results which will help reduce query time.

4. **Error reporting and monitoring**

   - **Requirement**: The system must be able to find errors and report them in real time.
   - **Specification:**
     i. Error Detection:
       1. The system will be able to detect data ingestion failure, corrupted files or model crashes.

- **Logging:**
    i. Track errors
        1. Source connections failures
        2. Inconsistent timestamps in time series data

5. **Data quality and Assurance**

- **Requirement:** The system must be able to maintain high quality data.
- **Specification:**
    i. Validation:
        1. Check time-series consistency, by comparing timestamps across data streams.
    ii. Fall back:
        1. Uses the most recent data, if the data ingestion fails
        2. The system can replace missing data points using interpolation

4.2 **Security Requirements**

When it comes to non-functional attributes for space weather forecasting systems, the security requirements are data protection against unauthorized access, data integrity which deals with the accuracy of the forecasts, being able to provide timely warnings and also being able to control those that are able to view and make changes to critical information. In addition to that, one must also ensure that the system remains operational even during significant space weather events.

Specific security requirements for space weather forecasting systems:

Data Encryption:
Sensitive data needs to be protected, one would be able to do so by encrypting sensitive data which in turn would protect against unauthorized access during transmission and storage.
Access Control:
Important data should not be accessible by everyone. Restricting access to critical forecasting systems only to authorized personnel.

Why are these security requirements crucial for space weather forecasting?

Critical Infrastructure Impact:
It is very crucial to keep in mind that accurate space weather forecasts are vital for protecting critical infrastructure like power grids.

National Security Concerns:
Disruption to critical infrastructure due to inaccurate or manipulated space weather forecasts can pose significant national security risks.

Public Safety:
Timely warnings about potential space weather events can help protect aviation and other sensitive operations that could be affected by disruptions like radio blackouts.

4.3 **Usability Requirements**

With space weather forecasting systems, some of the key aspects include: real-time data updates, clear visualization tools, intuitive interface design, reliable data availability, customizable alerts, and adaptability to different user skill levels; ensuring users can easily access and interpret complex space weather information to make informed decisions, even under time pressure or with varying levels of expertise.

Specific non-functional usability requirements for space weather forecasting could include:

Performance:
- How quickly can the data be loaded?

Reliability:
- Consistent data availability with minimal downtime.
- Redundant data sources to prevent single point of failure.
- Robust error handling mechanisms.

Scalability:
- Ability to handle increasing data volume as forecasting capabilities improve.
- Adaptability to new space weather data sources.

Security:
- Are sensitive data being secured?
- Is data being encrypted to protect privacy?

Accessibility:
- Support for different screen readers and assistive technologies.
- Customizable color schemes and font sizes.

User Interface (UI) Design:
- Intuitive layout with clear navigation.
- Consistent visual cues and icons.
- Customizable dashboards for specific user needs.

Alerting Mechanisms:
- Configurable alert thresholds for different space weather events.
- Timely notifications via multiple channels (email, SMS, push alerts).

Important considerations for space weather forecasting usability:

Target User Groups:
Understanding the needs of different user groups, including space operations personnel, power grid operators, satellite operators, and researchers, to tailor the interface accordingly.

Data Visualization:
Using clear and effective visualizations which could include maps, graphs, and time series plots to allow users to better understand complex space weather data intuitively.

Training and Support:
Providing training to assist users when it comes to understanding the system effectively.

---

5. System Design

5.1 High-Level Design

The high-level design of the DP includes the following components:

- **User Interface**: Developed using HTML, CSS, and JavaScript frameworks (e.g., Angular).
- **Interface Visualizations**: Created using data visualization tools. (e.g, Matplotlib, Seaborn and Tableau).
- **Data Processing**: Implemented using a server-side language (Python), and its various modules. (e.g, Sunpy, Astropy, Numpy)
- **Data Warehouse**: A cloud based data warehouse (e.g., Amazon Redshift, Google BigQuery, Snowflake) to store model outputs, cleaned data, and model information.

5.2 Detailed Design

- **User Interface Components**:
    - Dashboard for Users
    - Graphs/Visualizations
    - Indication of Predictive Data
    - Breakdown Of Data Used
- **Data Pipeline Components**:
    - Data Warehouses
    - Data Preprocessing Algorithms
    - Data Monitoring Tool

5.3 Data Warehouse Design

The data warehouse schema will include the following tables:

- **Time_dim**: time based queries and analytics (timestamp, year, month, day, hour).
- **Raw data**: Stores ingested raw data (raw_data_id, source_name, data_type, ingestion_time, file_path).
- **Processed_Data**: Stores clean and formatted data (processed_data_id, raw_data_id,source_name, data_type, processing_time).
- **Forecast_Fact**: stores models outputs and predictions (forecast_id, processed_data_id, timestamp, prediction_type, prediction_value, confidence_score).
- **Event_Log:** Pipeline activities and errors for monitoring (log_id, event_type, event_time, details)

6. Implementation Plan

6.1 Development Environment

- **IDE**: Visual Studio Code
- **Version Control**: Git and GitHub for source code management
- **Data Warehouse Management**: Google BigQuery for data warehouse management

6.2 Tools and Technologies

- Frontend: Angular
- Backend: Python with Flask/Django
- Data warehouse: Google BigQuery

6.3 Coding Standards

- Follow the PEP 8 style guide for Python.
- Use meaningful variable and function names.
- Write comments and documentation for all major functions and classes.

7. Testing Plan

7.1 Testing Strategy

- **Unit Testing**: Test individual components and functions for expected behavior.
- **Integration Testing**: Test the interaction between different modules.

7.2 Test Cases

1. **Test Case for Data Validity**
    - **Input:** Upload a magnetogram containing unexpected artifacts (e.g., noise or missing regions).
    - **Expected Output:** The pipeline detects anomalies, logs the issue, and either flags the data for review or proceeds with adjusted analysis.
2. **Test Case for Pipeline Functionality**
    - **Input:** Simulate a system failure and restart the pipeline.
    - **Expected Output:** The pipeline resumes processing from the last successful step without duplicating or skipping data.
3. **Test Case for Handling Corrupt Files**
    - **Input:** Upload a corrupted FITS file
    - **Expected Output:** The pipeline logs an error, skips the file and continues processing other data.
4. **Test Case for Time Series Misalignment**
    - **Input:** Upload multiple time series datasets with time misalignments.
    - **Expected Output:** Time series aligned during preprocessing.
5. **Test Case For Missing Data**
    - **Input:** Upload a dataset with missing time steps.
    - **Expected Output:** Flagged by system and uses alternative time steps or performs a prediction.

---

8. Appendices

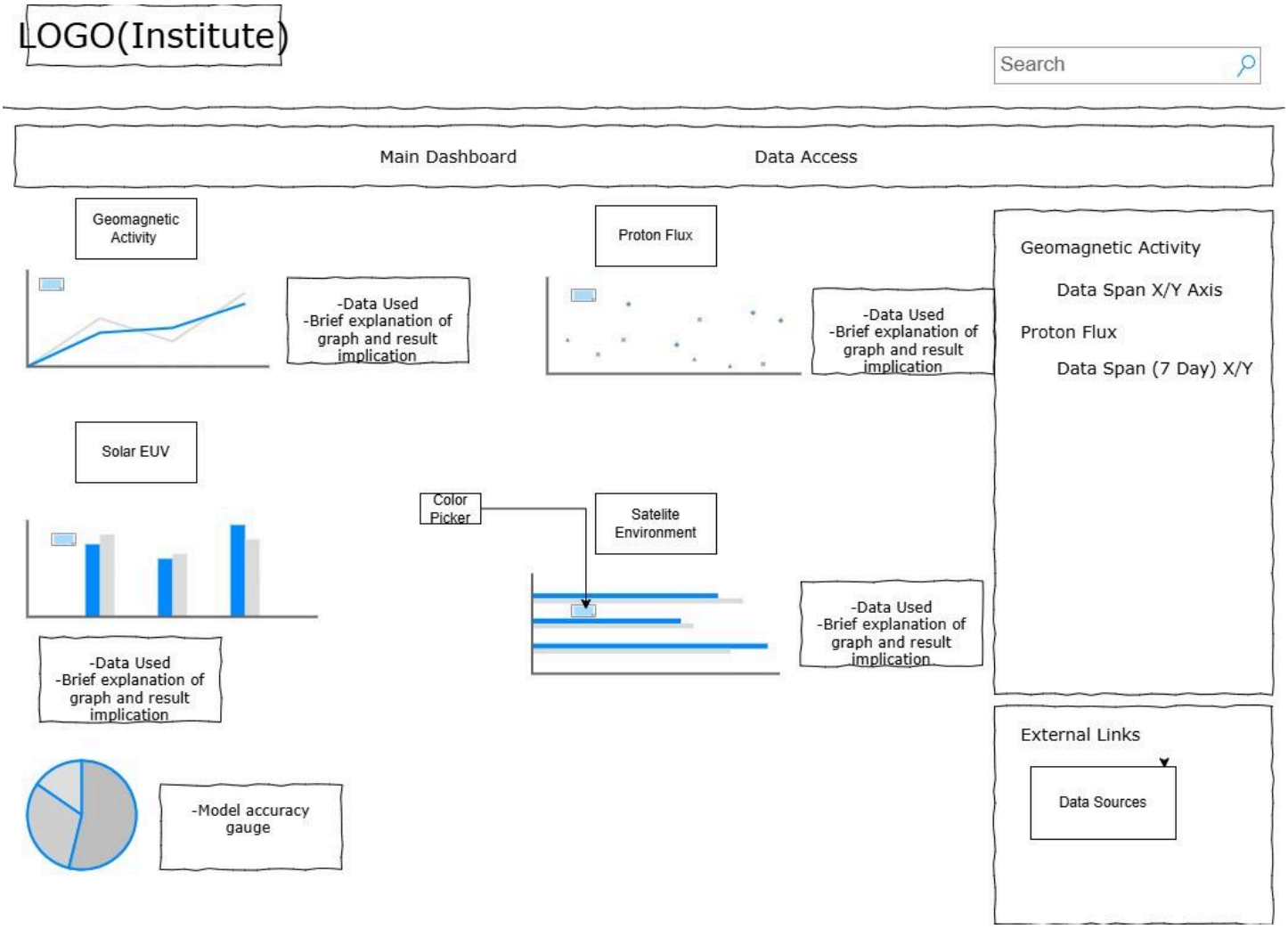
- **10.1 Detailed Data Processing**
    - **Strategy**
        - We will utilize ETL due to the diversity of our data and the need for cleaning prior to database storage.
        - ETL is optimal given the availability of tools like SunPy and Astropy, which will simplify data extraction and cleaning.
        - Data Formats: FITS, CSV, JSON
    - **Data Validation**
        - Validate header information from FITS files to ensure compliance with standard conventions.
        - For FITS files, check for missing or corrupted metadata.
        - For CSV check for empty cells, for JSON check for nulls
        - Flatten nested structures into a tabular format using pandas for JSON
        - Timezone and formats must be checked to ensure consistency. (Can easily be verified using datetime or astropy.time)
        - Verify consistency of variables such as temperature, time, and units.

- **Data Processing Steps**
  - Coordinate Transformations can be performed using WCS from FITS header information to align full disk data.
  - **Interpolation** must also be performed to ensure data is of highest quality and consistency.
    - Resample datasets with higher resolutions to match those with lower resolutions.
    - Time-series data will be upsampled in general for higher quality.
    - Image data will be downsampled from 4K to 512*512 format.
  - **Data Flagging**
    - For FITS files: Check for missing pixels in the header and explicitly mark them.
    - Detect anomalies using z-score or IQR for outlier detection.
    - Apply masks to bad data for tracking and later use in prediction software.
  - **Final Data Output**
    - The model will output results in JSON or a flat file convertible to JSON and include metadata such as; model name and version, time taken for computation, and a link to the model used.

- **10.2 Data Consumption Layer**
  - **Data Sources/Formats**
    - Full disk magnetograms (from SDO/HMI)
    - Full disk EUV images (from SDO/AIA)
    - Hard and soft X-ray time series (GOES)
    - Proton flux time series (GOES)
    - Physical parameters derived from HMI (from Stanford/JSOC)
  - **Data Formats**
    - Magnetograms: FITS
    - Proton Flux: JSON
    - X Ray: JSON
    - EUV Images: FITS
    - Physical Parameter: FITS/CSV
  - **Data Visualizations**
    - **GOES X-Ray Flux Visualization:**
      - After the data is preprocessed, we will plot the X-ray intensity data in a dual-line chart using APIs
      - Each line will represent a different value of energy range and thresholds for the intensities

        - Real-time updates will be provided by connecting the graph to live data streams through API interfaces.
    - **GOES Proton Flux Visualization:**
        - Proton flux time-series data will be shown as a line graph, with a horizontal colored line marking the threshold.
    - **Solar Imagery (EUV):**
        - Display the latest images of the Sun from SDO/AIA
        - A time slider will be included so that user can browse through images that is captured at different times
    - **Forecast Results:**
        - Map projections onto solar images, showing where solar storms might originate
    - **Real-Time Dashboards**
        - Combine multiple visualizations into an interactive dashboard to provide a holistic view of space weather conditions.
- Appendix A: Glossary of Terms
    - Masks: Used to flag or identify regions in datasets or images where data is invalid, unreliable, or irrelevant using arrays.
    - Coordinate Transformations: A coordinate transformation in data science is a mathematical process that changes the representation of data points by converting them from one coordinate system to another, essentially "mapping" the data to a different space while preserving the underlying relationships, often used to align datasets with different scales or orientations, or to prepare data for specific analysis techniques.
    - FITS files: FITS (Flexible Image Transport System) files are a standard file format for storing astronomical data. They support image, table, and metadata storage in a structured and self-describing way.
    - ETL (Extract, Transform, Load): is a data integration process that merges, cleans and organizes data from several sources into a single, consistent data set for storage in a data warehouse.

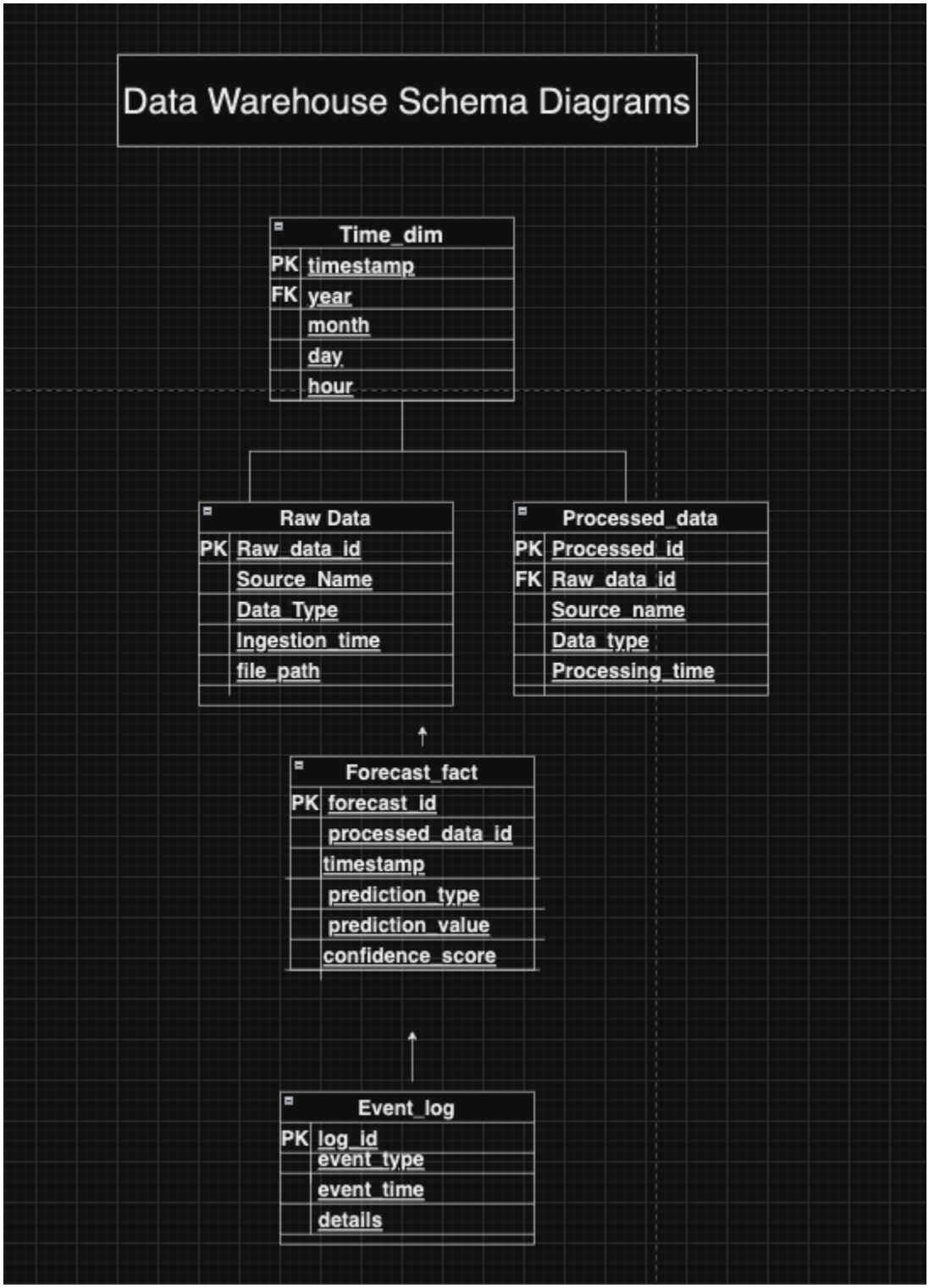
- Appendix B: User Interface Wireframes



Notes:

- When hovering over data points the graph will provide more detailed information
- The diagram will include filters on each graph that allow users to filter out specific times or days
- There will also be tabs that allow users to change graphs from three day to seven day etc

- Appendix C: Data Warehouse Schema Diagrams



---

9. Contribution Paragraph

Aimen Rehman: SDD compilation, system architecture diagram, functional requirements, interface requirements, testing plan, wireframes, detailed data processing, glossary, system overview, implementation plan

Sadia Akhter: Project overview, project objective, data consumption layer, references

Rithwik Kamalesh: Data warehouse schema, data warehouse diagram, performance requirements

Rochelle Daley: Security requirements, usability requirements, non functional attributes

Tom Tran: Use cases

This Software Design Document serves as a comprehensive guide for the development of the Space Weather Pipeline, detailing every aspect from requirements to maintenance. It is intended to ensure that all users are aligned and that the development team has a clear roadmap for building the system.