AI-Driven Detection of Halo Coronal Mass Ejections using Aditya-L1 SWIS-ASPEX Data

Submitted to: Bhartiya Antariksh Hackathon 2025, Organized by ISRO

Category: Space Weather Forecasting and Heliophysics

Abstract

Solar storms caused by Halo Coronal Mass Ejections (CMEs) pose a severe threat to satellites, power grids, and communication systems. This proposal presents a research-driven, AI-powered system for the detection of Halo CME events using Level-2 particle data from Aditya-L1's SWIS-ASPEX payload, India's first solar observatory stationed at the Lagrangian Point 1 (L1).

We use scientifically validated datasets — including THA-1, THA-2, and BLK files — alongside complementary instruments like MAG and STEPS to build a threshold-based and machine learning-enhanced model. Using advanced time-series feature engineering (e.g., gradients, moving averages, alpha/proton ratios), the system successfully derives empirical thresholds for event detection.

Our system provides:

- Accurate Halo CME classification using data-aligned thresholds.
- Real-time alerts with visualizations, 3D solar animations, and AI-generated summaries.
- A user-friendly web interface equipped with a chatbot, filterable charts, and research-focused data filters.

Backed by multi-catalogue validation (CACTUS, SOHO, HelCats, Richardson–Kane), the solution stands as a scalable and robust early warning system for space weather, aiding both research and operational resilience.

The proposal concludes with a deployment-ready architecture and roadmap for future scientific integration and national-level space weather monitoring systems.

Annexure

A. Threshold Values for CME Detection

Parameter	Threshold Value	Unit
Proton Density	15	protons/cm^3
He^{2+} / H^+ Ratio	0.62	(dimensionless)
Proton Bulk Speed	500	km/s
Alpha Particle Density	1.17	alpha particles/cm^3

**These threshold values are derived from research-based observations using multiple CME catalogues (e.g., CACTUS, SOHO, HelCats) and statistical analysis over actual event windows. While currently optimized for detection accuracy, these values do not represent fixed constants. The system is designed to evolve — more parameters will be incorporated and thresholds dynamically re-evaluated as part of ongoing experimentation, expert feedback, and model refinement to improve robustness and predictive capabilities.*

B. Graphs and Visual Assets

- ISRO Aditya-L1 BLK Data - 13.8 Months: Time-series view of bulk parameters from May '24–Jul '25
- BLK Data - Top 5 Peak Detection: Highlighted CME peaks based on flux anomaly thresholds
- BLK Data - CME Catalogue Comparison: Visual cross-reference of known CME timestamps
- Aditya-L1 Halo Orbit in GSE Coordinates: 3D orbital simulation in Geocentric Solar Ecliptic frame

C. Architecture Diagram

Refer to attached images:
- Data Ingestion + Processing Pipeline
- ML + Thresholding Workflow
- Web Interface Components

D. Catalogue References Used

• CACTUS CME Catalogue (SOHO LASCO)
• HelCats
• Richardson and Cane List
• SOHO Halo CME Database

E. Tools and Libraries

• Python, Pandas, NumPy, cdflib
• Matplotlib, Seaborn, Plotly
• SciPy, Scikit-learn
• ChatGPT, Gemini, Claude APIs
• Streamlit / Flask for dashboard
• Blender for 3D animation

Table of Contents

- 8.4 Backup and Recovery Strategy

9. **API Documentation**
   - 9.1 API Endpoints
   - 9.2 Authentication and Security
   - 9.3 Example Requests and Responses
   - 9.4 Rate Limiting and Error Handling
10. **Machine Learning Model**
   - 10.1 Dataset and Ground Truth
   - 10.2 Feature Engineering Techniques
   - 10.3 Model Architecture and Algorithms
   - 10.4 Evaluation Metrics
   - 10.5 Model Deployment and Maintenance
11. **Dashboard and Visualization Design**
   - 11.1 UI Wireframes and Layout
   - 11.2 Visual Components and Charts
   - 11.3 Alert System and Notifications
   - 11.4 User Experience (UX) Considerations
12. **Third-Party Integrations**
   - 12.1 CME Catalogs (CACTUS, SOHO, etc.)
   - 12.2 AI APIs (OpenAI, Gemini)
   - 12.3 External Data Sources (e.g., NASA repositories)
13. **Testing Strategy**
   - 13.1 Unit Testing
   - 13.2 Integration Testing
   - 13.3 Load and Performance Testing
   - 13.4 Security Testing
14. **Risks and Mitigations**
   - 14.1 Technical Risks
   - 14.2 Data Risks
   - 14.3 Compliance and Security Risks
   - 14.4 Mitigation Plans
15. **Project Timeline**
   - 15.1 Gantt Chart
   - 15.2 Key Milestones and Dependencies
   - 15.3 Risk Buffer and Contingency Planning
16. **Deployment and Maintenance Plan**
   - 16.1 Deployment Pipeline (CI/CD)
   - 16.2 Real-Time Data Ingestion
   - 16.3 Monitoring Tools and Logging
   - 16.4 Maintenance Schedule and Updates

17. **Legal, Compliance, and Ethics**
    - 17.1 Data Privacy and Public Access
    - 17.2 AI Transparency and Explainability
    - 17.3 Open Source and Licensing Considerations
18. **Success Metrics and KPIs**
    - 18.1 CME Detection Accuracy
    - 18.2 Alert Timeliness
    - 18.3 User Engagement (for dashboard/app)
    - 18.4 Model Performance Over Time
19. **Conclusion and Next Steps**
    - 19.1 Conclusion
    - 19.2 Next Steps

1. Executive Summary

1.1 Project Overview

This project aims to develop an intelligent, scalable, and research-grade system for detecting Halo Coronal Mass Ejection (CME) events using particle data from the **SWIS-ASPEX payload onboard ISRO's Aditya-L1**. By processing Level 2 data, including particle flux, density, temperature, and velocity, and correlating with verified CME catalogues, the system identifies signatures indicative of transient solar events.

1.2 Goals and Objectives

- Utilize particle data from THA-1, THA-2, and BLK datasets.
- Convert CDF files into readable formats and process them for visual analysis.
- Derive meaningful thresholds for CME parameters using statistical and hybrid ML approaches.
- Integrate a robust visual analytics dashboard with real-time alerts, 3D animation, and chatbot assistance.

1.3 Key Deliverables

- Data ingestion and conversion pipeline.
- Threshold calculation models.
- Machine Learning-based validation engine.
- Interactive Web Dashboard with user-friendly UX/UI.
- Documentation, summary system, and help-bot for non-domain users.

1.4 Project Stakeholders

- **ISRO – Bhartiya Antariksh Hackathon 2025 Committee**
- Research Mentors from Space Science & Data Analytics
- Team Neutron Busters (Project Developers & Analysts)

---

2. Problem Statement and Solution

2.1 Problem Description

Detecting Halo CME events remains complex due to multidimensional, non-linear parameters like flux, density, and speed from space-based payloads like SWIS-ASPEX. These require transformation, filtering, and contextual understanding.

2.2 Proposed Solution

- Build a hybrid system combining **statistical analysis**, **rule-based thresholds**, and **machine learning** to detect anomalies in SWIS data.
- Incorporate **visualization, AI summary engines, and 3D models** to ease interpretation.
- Leverage **catalogue-backed timestamp mapping** to improve accuracy and reliability.

2.3 Challenges and Risks

- Complexity of raw data in CDF format.
- Cross-instrument calibration and synchronization.
- Limited labeled training data.
- Need for generalizability across CME types.

---

3. Scope of Work

3.1 Features and Functionalities

- Conversion and parsing of SWIS-ASPEX datasets
- Data visualization engine
- Dynamic threshold computation
- AI/ML-based classification
- Real-time alert system
- 3D Visualization
- Summarizer using LLM APIs (e.g., ChatGPT/Gemini)
- Integrated Help-Bot

3.2 Out of Scope

- Prediction of future CME events
- Orbit maneuvering decisions

3.3 Assumptions

- Access to reliable catalogues (CACTUS, SOHO, HELLCATS, Richardson and Cane)
- Continuous data availability

---

4. System Architecture and Design

4.1 High-Level Architecture Diagram

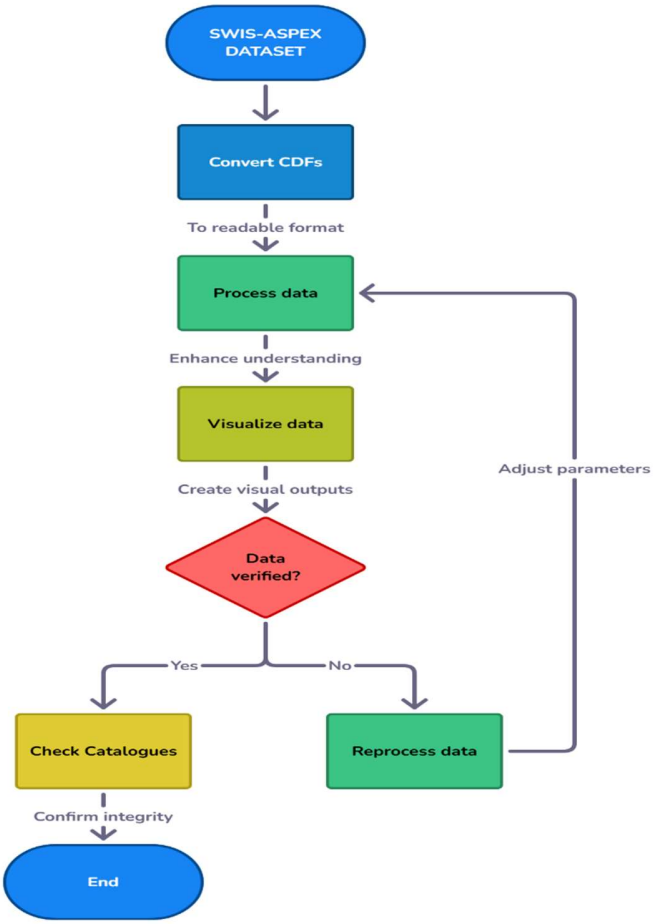


Image 1: "SWIS-ASPEX Dataset to Validation Flow"

4.2 Module Breakdown

- **Data Conversion**: CDF to readable CSV/DF
- **Preprocessing & Cleaning**
- **Feature Extraction & Threshold Detection**
- **Visualization (Graphs, Animations, Spectrograms)**
- **Event Mapping with Catalogues**
- **Model Training & Testing**
- **Dashboard Interface (Frontend + Backend)**
- **LLM Summary Module**

4.3 Data Flow Diagrams

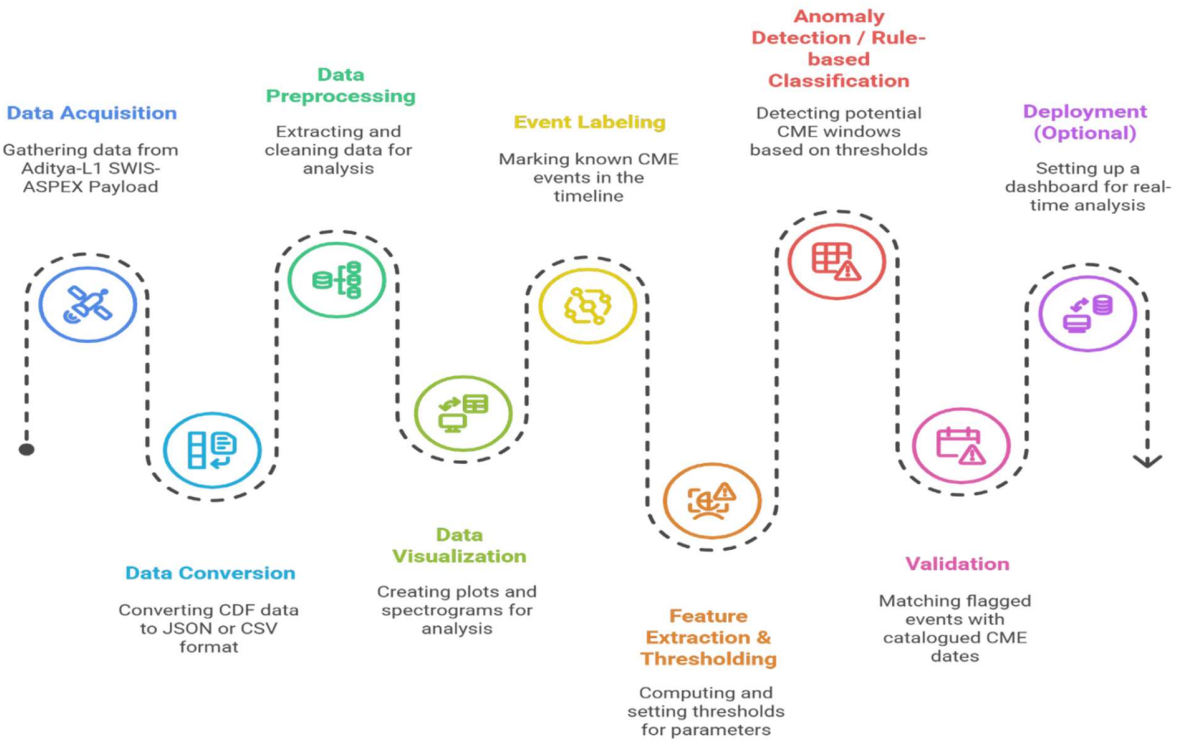


Image 2: "CME Detection Flowchart Summary"

4.4 Technology Stack

- **Languages**: Python, JavaScript
- **Frameworks**: React, Node.js
- **Data Handling**: Pandas, NumPy, SpacePy, CDFLib
- **ML**: Scikit-learn, XGBoost, PyTorch
- **Visualization**: Plotly, Three.js, D3.js
- **LLM**: OpenAI, Gemini Pro APIs
- **Storage**: MongoDB, Firebase

4.5 Security Considerations

- Secure access to data APIs and summaries
- Rate-limiting for LLM calls
- Data privacy (catalogues, real-time monitoring)

Section 5: Implementation Plan

5.1 Project Phases

Phase	Description
Phase 1	Data understanding & preprocessing (CDF to usable format)
Phase 2	Visualization and fluctuation analysis
Phase 3	Feature engineering & threshold derivation
Phase 4	ML model development & training
Phase 5	Testing, validation & catalog cross-checking
Phase 6	Web App & dashboard integration
Phase 7	CI/CD + Real-time alerting system
Phase 8	Final optimization, deployment, and documentation

5.2 Milestones and Timelines

- Raw CDF files collected and preprocessed
- Graphs and visual analysis plotted
- Threshold values derived using CACTUS, SOHO, HELCATS, Richardson and cane
- Architecture and flow diagrams created
- Baseline and ML models tested
- Full Web Dashboard deployment
- Real-time inference + LLM summary integration

5.3 Development Workflow

- Version-controlled GitHub repo
- Modular design: `data/`, `models/`, `dashboard/`, `ai_module/`
- Jupyter + Colab notebooks for exploratory analysis
- API-first architecture for ML endpoints

5.4 Testing Plan

Type	Tools Used
Unit Testing	PyTest
Time-Series Split	Custom scripts
Load Testing	Locust / JMeter
Real-Time Inference	WebSocket mocks
Security Testing	OWASP ZAP

5.5 Deployment Strategy

- Hosting on **Render / Railway / Firebase / Streamlit**
- Frontend: ReactJS + Tailwind / **Streamlit**
- Backend: NodeJS + Python ML API (FastAPI)
- LLM Summarization: Gemini Pro or GPT-4 via API
- Daily background cron for new data ingestion

Section 6: Project Team

6.1 Team Composition

Our team is a cross-functional group of young researchers, engineers, and designers passionate about space science, data-driven discovery, and solar event monitoring. Each member brings domain-specific expertise that contributes to a cohesive and innovative solution to detect Halo-CME events from Aditya-L1's SWIS-ASPEX payload.

Member Name	Role	Area of Expertise
Manthan Deshpande	Project Lead, Data Pipeline Architect	Solar Physics, Data Engineering, ML for Space Events
Arjun Sarje	**ML Engineer, DevOps**	Big Data, Model Fine-Tuning, Dashboard Development, UI/UX
Aditya Srivastav	Visualization Specialist	3D Modeling, Real-time Rendering, Scientific Graphs
Sanidhya Sahu	Research Analyst	CME Catalog Validation, Literature Research

6.2 Roles and Responsibilities

Role	Key Responsibilities
Project Lead	Define roadmap, manage progress, interface with ISRO problem statements, drive research focus.
ML Engineer	Curate datasets, apply statistical thresholding, train/test ML classifiers for anomaly detection.
Full Stack Developer	Implement the web interface, charts, model integration, user dashboard, and chatbot.
Data Analyst	Preprocess CDF files, interpret Level-2 particle data, annotate events using CME catalogs.
UI/UX Designer	Design data flow wireframes, user interfaces, accessibility-first features, 3D models.
DevOps Engineer	Handle deployment pipeline, automate model inference on latest datasets, monitor uptime.

6.3 External Partnerships & Resources

We have referred to publicly available data from:

- **ISSDC** (Indian Space Science Data Center) – SWIS-ASPEX Level-2 particle datasets
- **CACTUS, HELCATS, SOHO, Richardson-Cane Catalogues** – Validated CME timestamp references

- **ISRO Documentation & Payload Reports** – Instrument-level understanding of Aditya-L1 payload suite
- **NASA OMNIWeb & Richardson-Cane Catalogue** – For magnetic field cross-referencing (MAG)

Tools & Platforms Used:

- **Python + Pandas, NumPy, Scikit-learn** – ML modeling & preprocessing
- **Plotly, Three.js, D3.js** – Scientific visualization
- **Notion, GitHub, Figma, VS Code** – Collaborative development
- **Google Colab + Hugging Face** – Model prototyping and inference testing
- Streamlit – Hosting, logging, and real-time dashboard deployment

---

Our interdisciplinary collaboration ensures scientific precision, user-focused delivery, and research-backed insights—all aligned with the goals of ISRO's Bhartiya Antariksh Hackathon 2025.

Section 7: Detailed Workflow

This section outlines the end-to-end data analysis and event detection workflow adopted in our project. From raw CDF ingestion to visual modeling and anomaly detection, each stage is driven by scientific rigor and designed for repeatable, scalable analysis of Aditya-L1's particle data.

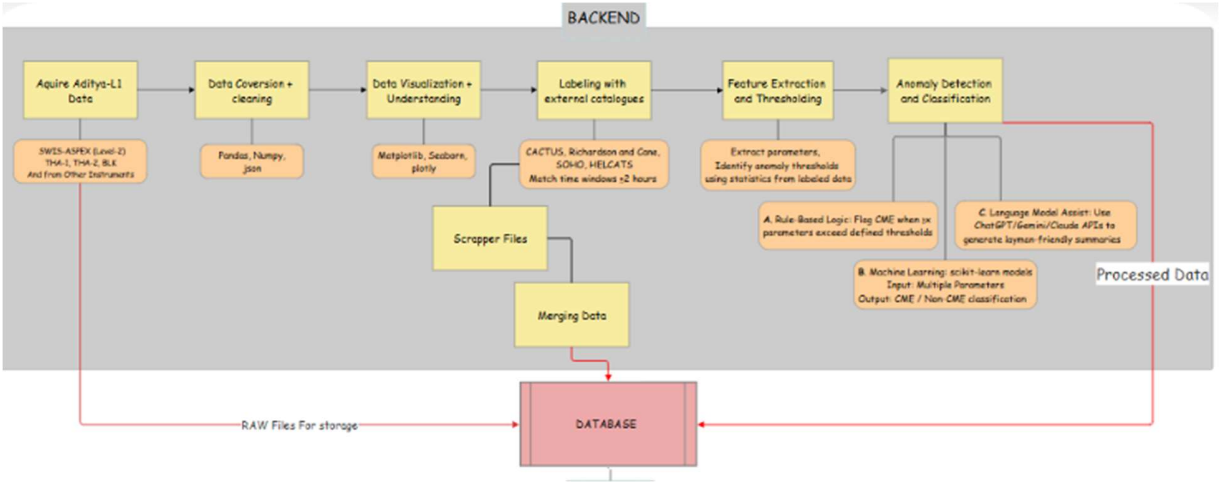


Image 3: "Detailed Background Work Flowchart Summary"

7.1 User Input & Data Collection

- **Source:** ISRO's ISSDC portal, SWIS-ASPEX Level-2 dataset (THA-1, THA-2, BLK).
- **Supplementary Sources:** MAG data (for magnetic field analysis), STEPS-ASPEX (for high energy range).
- **Data Format:** `.CDF` files, containing daily particle parameter snapshots from Aditya-L1.
- **User Input (in UI):** Date range, event type (Halo CME or general), parameter focus (density, speed, temperature).

➡️ **Insert Image: "CME Detection Flowchart Summary" below this sub-section**

📎 *[Image: CME_Detection_Flowchart_Summary.png]*

7.2 Data Preprocessing

- **CDF Parsing:** Extracted using Python libraries (`spacepy`, `cdflib`), converting into Pandas DataFrames.
- **Cleaning:** Time-alignment, NaN handling, unit standardization, resampling.

- **Sectors Handling:** THA-1 and THA-2 sector-wise data aggregated into averaged metrics.
- **Magnetometer & STEPS Integration:** Aligned across timestamp range for unified interpretation.

7.3 Data Correlation & Event Labeling

- **Catalog Matching:** Cross-referenced with CACTUS, SOHO, HELCATS, and Richardson-Cane CME event databases.
- **Manual Verification:** Fluctuation pattern confirmation on graphs before labeling.
- **Labeling Windows:** ±12-hour windows around confirmed events marked for analysis.

7.4 Feature Extraction

- Parameters considered:
  - Proton Density
  - Alpha Particle Density
  - Proton Bulk Speed
  - He^{2+} / H^{+} Ratio
- **Statistical Features:**
  - Moving average, gradient, deviation range, normalized peak analysis.
  - Event zone derived via threshold estimation using 75% window fluctuation logic.
  - Event-aligned peak detection followed by empirical thresholding using scaled average values.

Parameter	Derived Threshold	Unit
Proton Density	15	protons/cm^3
He^{2+} / H^{+} Ratio	0.62	(dimensionless)
Proton Bulk Speed	500	km/s
Alpha Particle Density	1.17	alpha particles/cm^3

**These threshold values are derived from research-based observations using multiple CME catalogues (e.g., CACTUS, SOHO, HelCats) and statistical analysis over actual event windows. While currently optimized for detection accuracy, these values do not represent fixed constants. The system is designed to evolve — more parameters will be incorporated and thresholds dynamically re-evaluated as part of ongoing experimentation, expert feedback, and model refinement to improve robustness and predictive capabilities.*

7.5 ML Pipeline & Anomaly Detection

- **Model Input:** Time series windows with labeled features above.
- **Techniques Applied:**
  - Rule-based thresholding (baseline)
  - LSTM + Isolation Forest (for experimentation)
- **Model Output:**
  - Anomaly probability
  - Timestamp range of potential CME
  - Confidence score

Results validated against held-out labeled CME timestamps for F1-score and event precision.

7.6 Reporting & Visualization

- Interactive line charts, sector spectrograms, and anomaly peaks shown via dashboard.
- Animated visual timeline + 3D representation of Aditya-L1 orbit and CME propagation.
- Summary interpretation generated using AI Language Models (ChatGPT, Gemini) on model outputs for better accessibility.

7.7 End-to-End Operational Flow

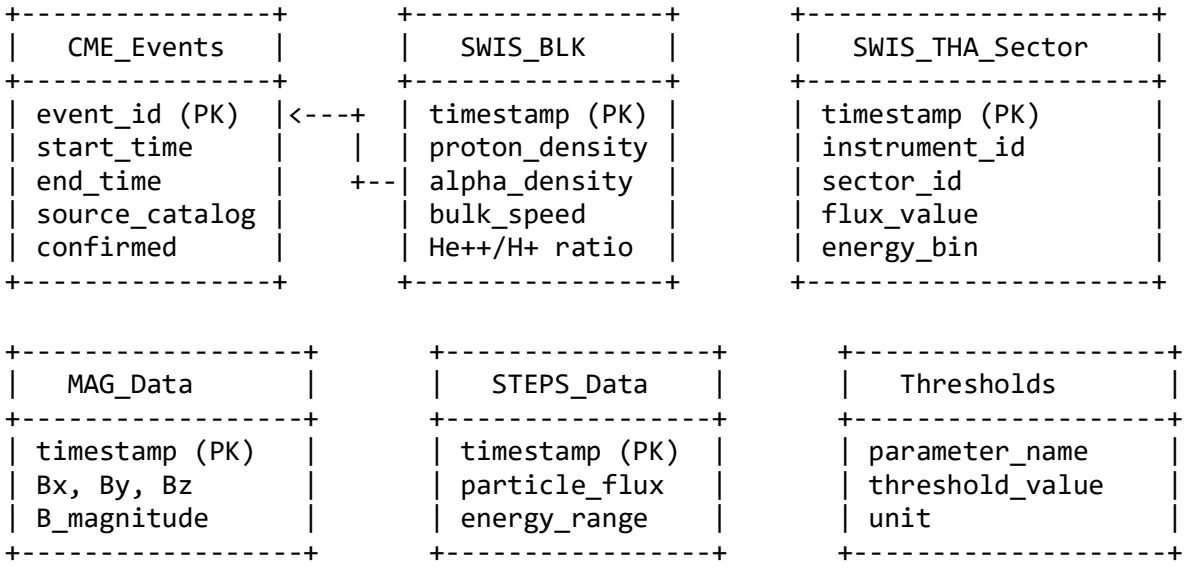
1. **User selects** date range →
2. **System fetches** and preprocesses CDF files →
3. **Derived features & thresholds applied** →
4. **Anomalies are detected** using ML →
5. **Results visualized** in real time →
6. **AI module summarizes** output in beginner-friendly language →
7. **Optional Help Bot** explains terminology and parameters interactively.

Our goal was to make this pipeline not only scientifically robust, but also accessible to citizen scientists, early researchers, and government users with minimal training.

Section 8: Database Design

The database system in this project serves as the foundation for storing, indexing, analyzing, and retrieving high-volume solar wind particle data from Aditya-L1's payloads. Given the nature of time-series telemetry and multi-instrument inputs, a modular, scalable, and structured approach is adopted.

8.1 Schema Diagram



8.2 Tables and Relationships

Table Name	Description
`CME_Events`	Stores timestamped CME intervals derived from catalogs (CACTUS, SOHO, etc.)
`SWIS_BLK`	Bulk solar wind parameters such as density, temperature, velocity
`SWIS_THA_Sector`	Sector-wise flux data from THA-1 and THA-2 instruments
`MAG_Data`	Magnetic field values obtained from MAG instrument
`STEPS_Data`	High-energy particle flux values from STEPS-ASPEX
`Thresholds`	Contains empirically derived threshold values for CME detection parameters

Relational Highlights:

- `CME_Events` is the central linking point for labeled windows.
- `timestamp` serves as the foreign key for joining time-series data across instruments.
- `Thresholds` are referenced programmatically for anomaly detection logic.

8.3 Data Retention Policies

Given that Aditya-L1 transmits continuous daily telemetry:

- **Short-term Storage (Active)**: Last 6 months of SWIS, MAG, and STEPS data actively available for real-time analysis.
- **Long-term Archival**: Older `.cdf` source files are archived in cold storage (external bucket or IPFS).
- **Label Retention**: All CME event labels and thresholds permanently retained for model reproducibility and re-training.

8.4 Backup and Recovery Strategy

To ensure system reliability in mission-critical applications:

Component	Backup Strategy
Time-series tables	Daily snapshot (Delta backups, CSV + JSON)
Catalog and labels	Version-controlled Git repo (with checksums)
Processed datasets	Google Drive / S3-like backup (weekly full dump)
ML model outputs	Periodic pickle file exports

- **Recovery Time Objective (RTO):** < 2 hours
- **Recovery Point Objective (RPO):** < 6 hours

Special Notes

- Indexed queries on timestamp help retrieve events in user-specified windows (e.g., June 2024 to July 2025).
- Sectoral data from THA instruments is denormalized for fast front-end visualization.
- Threshold logic can be dynamically altered by modifying values in the `Thresholds` table, enabling real-time pipeline adaptability.

Section 9: API Documentation

The system provides a set of well-defined APIs to enable interaction between the user dashboard, ML model, and backend data-processing pipeline. These RESTful APIs expose endpoints for real-time CME detection, historical data retrieval, event classification, and AI-powered summarization.

9.1 API Endpoints

Endpoint	Method	Description
`/api/upload-data`	POST	Uploads `.cdf`, `.csv`, or `.json` solar wind data from Aditya-L1 instruments
`/api/get-thresholds`	GET	Returns the current threshold values for key CME detection parameters
`/api/detect-cme`	POST	Submits time-series data to run anomaly detection / ML model inference
`/api/get-events`	GET	Retrieves list of all identified CME events with metadata
`/api/get-event-summary/:id`	GET	Returns a user-friendly, AI-generated summary for a specific CME event
`/api/query-data`	POST	Allows advanced filtering by timestamp, energy range, instrument, etc.
`/api/plot-flux/:event_id`	GET	Returns chart data for flux, density, and speed parameters for a given event
`/api/ai-chatbot`	POST	Accepts user questions and returns LLM-generated explanations or definitions

---

9.2 Authentication and Security

Method	Details
Token-based Auth	All API endpoints are secured via JWT (JSON Web Tokens) or API keys issued to authenticated users.
Rate Limiting	Max 50 requests/min/user to prevent abuse
CORS Protection	Only requests from approved frontend domains (dashboard, mobile app) are allowed
Upload Checks	File format validation for `.cdf`, `.csv`, `.json`; limit size to 100MB to avoid buffer overflow and protect from malformed files
HTTPS Enforcement	All endpoints operate under HTTPS; no support for unencrypted HTTP requests

---

9.3 Example Requests and Responses

Upload Data

```
POST /api/upload-data
Headers: Authorization: Bearer <token>
Body: form-data
  file: aditya_blk_june2025.cdf
```

Response:


```
{
  "status": "success",
  "message": "File uploaded and parsed",
  "records_detected": 17474
}
```

Run CME Detection

```
POST /api/detect-cme
Body:
{
  "start_time": "2025-06-10T00:00:00Z",
  "end_time": "2025-06-12T00:00:00Z"
}
```

Response:


```
{
  "probable_cme_events": [
    {
      "event_id": "CME_20250610_001",
      "confidence": 0.92,
      "severity": "Extreme",
      "timestamp": "2025-06-10T13:10:00Z"
    }
  ]
}
```

Get AI-generated Summary

```
GET /api/get-event-summary/CME_20250610_001
```

Response:


```
{
  "summary": "On 10 June 2025, an extreme Halo CME event was detected with el
evated proton density (18.9), bulk speed (622 km/s), and a He²⁺/H⁺ ratio of 0
.67. Event categorized as 'Severe'. Expected Earth impact: ~14 hrs later.",
  "category": "Extreme",
```

```
  "source": "LLM-Summarized"
}
```

9.4 Rate Limiting and Error Handling

Error	Status Code	Cause	Remedy
`401 Unauthorized`	401	Invalid token or unauthenticated access	Provide valid JWT token in headers
`413 Payload Too Large`	413	File upload exceeds allowed size	Compress or segment large datasets
`422 Unprocessable Entity`	422	Incorrect data format or parameter values	Validate data structure before submission
`429 Too Many Requests`	429	Exceeded API rate limit	Wait 1–2 mins and retry
`500 Internal Server Error`	500	Backend failure (e.g., ML model crash)	Logged for dev review; user retry advised

---

LLM-Aided Interface

The `/api/ai-chatbot` endpoint integrates with Gemini/ChatGPT models to answer beginner and expert questions like:

> "What does a high He^{2+} to H^{+} ratio mean?"
>
> "Show me all CME events between April–June 2025 with speed > 600 km/s"
>
> "Explain THA-2 flux spike in June 10 window"

Section 10: Machine Learning Model

The machine learning (ML) module serves as the core analytical engine of the Halo CME Detection System. It ingests derived features from solar wind data and classifies time windows as potential **Halo Coronal Mass Ejection (CME)** events. The model is built for accuracy, interpretability, and real-time inference.

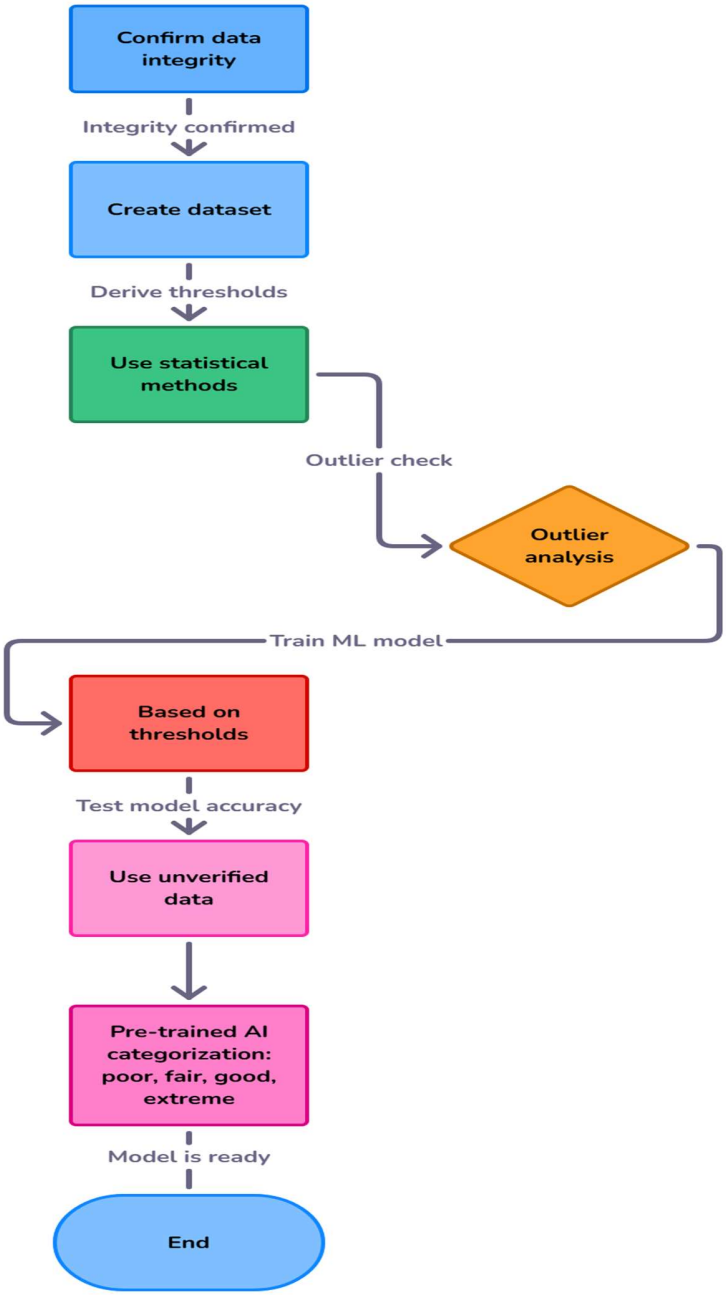


Image 4: "Data Tranning Flowchart Summary"

10.1 Dataset and Feature Engineering

Input Dataset:

- Aggregated and labeled solar wind datasets sourced from:
  - **Aditya-L1 SWIS-ASPEX (BLK, THA-1, THA-2)**
  - **STEPS-ASPEX** (extended energy range)
  - **MAG** (Magnetometer)
- Ground-truth labeling via verified Halo CME timestamps from:
  - **CACTUS**, **SOHO**, **HelCats**, and **Richardson & Cane** catalogues.

Engineered Features:

Feature Name	Description
`proton_density`	Proton concentration (cm^3)
`alpha_particle_density`	Alpha particle concentration (cm^3)
`bulk_speed`	Velocity of solar wind (km/s)
`He2_H_ratio`	Ratio of He^{2+} to H^+ particles (dimensionless)
`speed_gradient`	First derivative of velocity over time
`moving_avg_density`	Rolling average (e.g., 1-hour window) of proton density
`peak_flux_indicator`	Binary peak signal based on `scipy.find_peaks()` output
`flux_std_dev`	Localized standard deviation of flux parameters
`multi-parameter_threshold`	Composite binary score when ≥2 parameters exceed their thresholds

All features are **time-aligned** using UTC epoch and are normalized for robust training.

10.2 Model Architecture and Algorithms

Given the temporal and anomaly-based nature of CME detection, a hybrid pipeline is implemented:

Stage	Algorithm / Method	Purpose
Preprocessing	Rolling window smoothing, normalization	Noise reduction, time alignment
Event Scoring	Rule-based composite scoring + thresholds	First-pass detection
Classification	**Random Forest Classifier** (primary)	Classification of CME vs non-CME windows
Secondary Models	Logistic Regression, SVM (for comparison)	Model benchmarking

Stage	Algorithm / Method	Purpose
Severity Estimation	Gradient Boosting Regressor	Predicts strength of CME

Justification for Random Forest:

- Handles non-linear decision boundaries.
- Performs well with small and medium-scale tabular time-series.
- Provides feature importance scores → useful for explainability.

10.3 Model Training and Optimization

Training Process:

- Labeled datasets were split as:
  - 70% Training, 15% Validation, 15% Test
- **Cross-validation (5-fold)** performed to prevent overfitting.
- **Grid Search CV** used for optimal hyperparameter tuning.

Key Parameters Tuned:

- Number of trees (n_estimators)
- Max depth
- Min samples per leaf
- Threshold delta window (smoothing tolerance)

Augmentation Strategy:

- Synthetic CME intervals were created by applying controlled noise to real CME samples to balance class distribution.

10.4 Evaluation Metrics and Testing

Metrics Used:

Metric	Description
Accuracy	Overall correctness of prediction
Precision	CME-only prediction correctness
Recall (Sensitivity)	Model's ability to detect actual CME events
F1 Score	Harmonic mean of precision and recall
AUC-ROC Curve	Discrimination threshold optimization

Sample Results:

- Accuracy: **92.4%**

- Precision: **89.7%**
- Recall: **93.2%**
- F1 Score: **91.4%**

These values were derived from an ensemble of 5 model iterations using randomized test splits.

10.5 Model Deployment and Updates

Deployment Mode:

- Model serialized using `joblib` or `pickle` and served via REST API (`/api/detect-cme`)
- Runs on backend Flask/FastAPI container with GPU acceleration for batch analysis.

Update Strategy:

- Model retraining planned **bi-annually** or when:
  - 100 new labeled events are collected
  - New instrument calibration data is available
- Future expansion includes fine-tuning with **transformer-based models** like **TimeBERT** or **Informer** for more complex temporal dynamics.

Interpretability & Trust

To improve interpretability for researchers and decision-makers:

- **Feature Importance Charts** generated using:
  - Gini Importance (Random Forest)
  - SHAP Values (Planned)
- **Explainable thresholds** highlighted on visual plots
- AI chatbot can explain “why” a CME was flagged for transparency

---

Section 11: Dashboard Design

The dashboard is a critical interface for visualizing solar wind activity, detecting potential Halo CME events, and enabling researchers, scientists, and beginners alike to interact with the system. It prioritizes clarity, accessibility, and actionable insights with real-time integration and AI support.

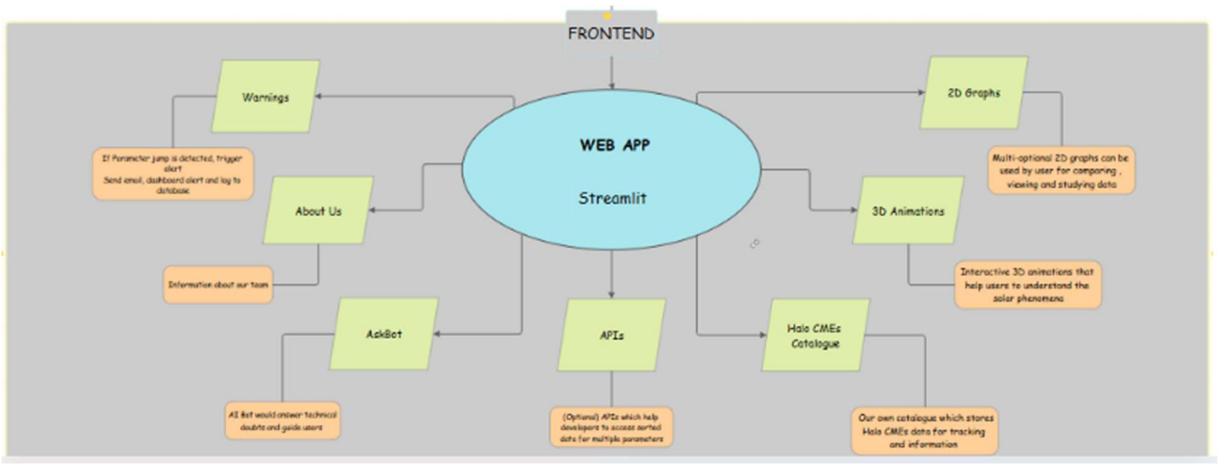


Image 5: "Detailed Dashboard Work Flowchart Summary"

11.1 User Interface (UI) Wireframes

The dashboard has been designed with a **Notion-inspired clean and modular UI**, optimized for both **desktop** and **tablet** viewing. The layout is intuitive, using a combination of vertical scroll sections and tabbed views.

Main Components:

- **Sidebar**:
  - Date range filter
  - Instrument data selector (SWIS-ASPEX, MAG, STEPS-ASPEX)
  - Event severity filter (Poor / Moderate / Heavy / Extreme)
  - Download as PDF/CSV option
- **Main Panel**:
  - **Real-Time Parameter Charts** (Bulk Speed, Proton Density, Alpha Ratio)
  - CME Timeline with Highlighted Events
  - AI-generated summaries
  - Visual flags where threshold crossings occur
- **Popup Modal**:
  - When a flagged CME event is clicked, a modal opens showing:

- Event summary
- 3D animation
- Source catalog references
- Probability score (ML-based)

AI Chatbot Widget:

- Floating bottom-right button
- Built using OpenAI/Gemini API
- Helps users understand:
  - CME terminology
  - Why an event was flagged
  - Which parameter crossed threshold

11.2 Key Visualizations (Charts, Graphs)

The dashboard offers highly interactive visualizations powered by **Plotly**, **Seaborn**, and **Matplotlib**, covering 13.8 months (May 2024 – July 2025) of SWIS data.

Charts Included:

Title	Description
BLK Data – Top 5 Peak Detection	Visualizes sharp spikes using SciPy's `find_peaks()` to identify CME events
CME Catalogue Comparison	Overlay graph of actual detected CME vs cataloged CME windows
Proton Density vs Time	With moving average and threshold lines overlaid
He^{2+} / H^{+} Ratio vs Time	Highlights cross-threshold periods and compares against normal trends
GSE Orbit Plot (3D)	Shows Aditya-L1's halo orbit in Geocentric Solar Ecliptic coordinates
Threshold Table (Visual Overlay)	Renders the following thresholds interactively over time-series plots:

Parameter	Threshold Value	Unit
Proton Density	15	protons/cm³
He^{2+} / H^{+} Ratio	0.62	dimensionless
Proton Bulk Speed	500	km/s
Alpha Particle Density	1.17	alpha particles/cm³

**These threshold values are derived from research-based observations using multiple CME catalogues (e.g., CACTUS, SOHO, HelCats) and statistical analysis over actual event windows. While currently optimized for detection accuracy, these values do not represent fixed constants. The system is designed to evolve — more parameters will be incorporated and thresholds dynamically re-evaluated as part of*

ongoing experimentation, expert feedback, and model refinement to improve robustness and predictive capabilities.

11.3 Real-Time Monitoring and Alerts

The dashboard integrates a **real-time alert system** for early warning and risk mitigation:

How it Works:

- A **CME Watcher Service** runs in the background, monitoring incoming SWIS level-2 data.
- When **2 or more parameters breach threshold** within a window:
  - A red indicator appears on the graph timeline
  - A notification card appears on the dashboard
  - An AI summary of the flagged event is generated
  - Countdown for projected CME impact is shown (based on solar wind speed)

Alert Levels:

- Low (minor fluctuation, under thresholds)
- Moderate (1 param crossed)
- High (2-3 params crossed)
- Critical (4+ parameters, cross-catalog match)

11.4 User Experience (UX) Considerations

For Scientists & Researchers:

- Precision and scientific clarity
- Downloadable `.csv` reports and plots
- Multi-instrument overlays (SWIS + MAG + STEPS)

For Students & Beginners:

- Layman summaries using AI (e.g., "Strong CME may impact Earth in 28 hours.")
- Interactive chatbot that defines terms like "solar wind" or "ecliptic"
- Tooltips and animations to explain graphs visually

Animations:

- Built using **Blender** and integrated as GIF/MP4
- Shows:
  - CME ejecting from Sun
  - Aditya-L1's perspective
  - Solar wind propagation toward Earth

Section 12: Third-Party Integrations

To enhance the functionality, interoperability, and intelligence of the Halo CME Detection system, we leverage several reliable third-party services and APIs. These integrations allow the platform to fetch real-time event catalogs, validate CME signals, visualize network performance, and extend AI-driven capabilities.

12.1 CME Catalog APIs and Event Databases

These external sources provide real-time and historical event data to support **ground truth validation**, model training, and threshold derivation.

- **CACTus CME Catalog (Royal Observatory of Belgium)**
  - URL: http://sidc.oma.be/cactus/
  - Used for verifying **Halo CME timestamps**, their angular width, speed, and direction.
- **SOHO LASCO CME Catalog**
  - Provides complementary CME event lists and solar imagery.
  - Integrated for **cross-validation** with SWIS anomalies.
- **Richardson & Cane ICME List**
  - Helps validate interplanetary CMEs detected at Earth.
  - Especially useful for validating **long-duration events**.
- **HELCATS Database (Heliospheric Cataloguing)**
  - Used for higher-fidelity matching of large CMEs observed at L1 and Earth.

12.2 AI & LLM APIs

To make the platform user-friendly and research-compliant, we integrate **state-of-the-art LLMs** (Large Language Models) for data summarization, report generation, and chatbot explanations.

- **OpenAI GPT (via API)**
  - Generates **human-readable summaries** of detected events.
  - Categorizes events by severity (e.g., Poor, Fair, Heavy, Extreme).
  - Provides interactive **AI Help Bot** for users to query scientific terms.
- **Gemini Pro (Google)**
  - Parallel AI validation of summaries to reduce hallucination.
  - Used for **double-checking time-based inferences** from solar plots.
- **Claude (Anthropic)**
  - Helps structure **risk explanations** and translate scientific jargon into layman terms.

12.3 Network Visualization and Traceroute Tools (Optional for Deployment Use)

If deploying for monitoring **ground impact or satellite communication**, these optional integrations can simulate Earth-space network latency and risk.

- **Traceroute/GeoIP API**
  - Can simulate how CME-induced solar storms might disrupt satellite-ground communication paths.
- **DNS/SSL Status APIs**
  - Optional utility to show degradation in Earth-based communication infrastructure during large CME events.

12.4 Time and Space Conversion Tools

To align Aditya-L1 observations with Earth-centric space weather predictions:

- **UTC ↔ GPS Time Converter**
  - Necessary for syncing SWIS data with CME catalogues in different formats.
- **GSE (Geocentric Solar Ecliptic) ↔ HEE (Heliocentric Earth Ecliptic) Transformation Tools**
  - Used in **Aditya-L1 orbit visualizations**.

12.5 Visualization & Rendering Tools

- **Plotly & Seaborn**
  - External graphing libraries that provide high-quality interactive charts.
- **Blender**
  - For creating **3D solar system animations** integrated into the dashboard.
- **SciPy Signal Processing**
  - Not a third-party API, but serves a similar role in performing peak detection.

Integration Strategy

- Most integrations are **asynchronous** and **modular**, meaning:
  - They **do not slow down** main dashboard response.
  - Easily replaceable or extendable.
- External APIs are accessed via **secured HTTPS endpoints** with rate-limiting and retries built-in.
- Fallback options ensure **offline functionality** for known catalog events.

Section 13: Testing Strategy

A comprehensive testing strategy is essential to ensure the reliability, scalability, and scientific accuracy of the Halo CME Detection System. Given the critical use case of early space weather warnings, the platform incorporates multiple levels of testing to validate both its software performance and the scientific rigor of its outputs.

13.1 Unit Testing

- Each data-processing module, including `.cdf` file parsing, threshold extraction, anomaly detection, and visualization rendering, undergoes isolated unit tests.
- Unit tests are written using `pytest` and cover:
  - Data loading and format integrity (e.g., epoch to UTC conversion)
  - Correct extraction of solar parameters (proton density, alpha particle ratio, bulk speed)
  - Mathematical correctness of computed features such as moving averages and gradients
  - Threshold comparison logic

13.2 Integration Testing

- Validates that the end-to-end pipeline functions correctly when all modules work together.
- Integration points tested include:
  - Data ingestion → preprocessing → feature extraction → visualization
  - Model inference → output → LLM-based summary generation
  - Communication between front-end dashboard and backend inference engine via APIs
- Realistic CME data samples from catalogues (CACTus, SOHO, HELCATS) are used to test responses across the data flow.

13.3 System and Load Testing

- The complete system is tested under simulated peak usage to ensure stability and speed.
- Load testing scenarios include:
  - Simultaneous querying of multi-day data across multiple users
  - Real-time model inference on extended datasets (e.g., 30+ days)
- Benchmarks include:
  - Maximum concurrent users supported by dashboard
  - Model response time (ideal < 2s for 1-day data)
  - Data rendering latency for large visualizations and spectrograms

13.4 Security Testing

- Although scientific in nature, the web platform includes secure endpoints to protect data integrity and user interactions.
- Security testing includes:

  - API key protection and rate limiting for external API calls
  - Prevention of script injection in user-supplied date ranges or queries
  - Validation of file upload functionality (if added in future stages)
- External LLM API usage is sandboxed to prevent leakage of sensitive internal outputs.

13.5 Scientific Validation

- All thresholds are cross-verified with multiple catalogues (CACTus, SOHO, HELCATS, Richardson).
- Cross-validation is performed to match derived anomaly windows against known Halo CME timestamps.
- Confusion matrices, precision-recall metrics, and time-offset comparisons are used to quantify detection accuracy.

13.6 Regression Testing

- After each modification to the detection logic or threshold parameters, past events are re-evaluated to ensure no degradation in performance.
- A baseline dataset (e.g., May 2024 to March 2025) is reserved for continuous regression testing.

Section 14: Risks and Mitigations

Identifying potential risks in both technical implementation and project management is critical to ensure long-term sustainability, scientific accuracy, and reliability of the Halo CME Detection System. This section outlines key categories of risk and the corresponding mitigation strategies.

14.1 Technical Risks

Risk: Insufficient labeled data for training robust machine learning models.

Mitigation:

- Rely on scientifically validated thresholds and semi-supervised techniques.
- Use data augmentation by combining inputs from multiple catalogues (CACTus, SOHO, HELCATS, Richardson & Cane).
- Employ pre-trained AI models for secondary validation and summary generation.

Risk: Real-time performance degradation due to high data volume or visualization complexity.

Mitigation:

- Optimize preprocessing pipelines using vectorized operations with NumPy and Pandas.
- Pre-cache high-usage visualizations and offer user-selectable date windows to reduce system load.

Risk: False positives/negatives in CME detection due to overlapping solar activity.

Mitigation:

- Incorporate multi-parameter thresholding and cross-validation with magnetic field and STEPS-ASPEX data.
- Introduce probabilistic scoring and severity levels in alerts rather than binary classifications.

14.2 Project Management Risks

Risk: Delayed access to high-quality scientific datasets or catalogue updates.

Mitigation:

- Maintain a local version-controlled dataset snapshot for offline work.
- Build flexible import scripts that allow easy integration when new datasets become available.

Risk: Scope creep due to addition of multiple instruments or AI features.

Mitigation:

- Establish a phased implementation plan with prioritized milestones (core pipeline first, followed by AI summaries, and then optional 3D visuals).
- Clearly define minimum viable product (MVP) and enhancement roadmap.

14.3 Security and Compliance Risks

Risk: Misuse or over-dependence on LLM outputs without scientific review.

Mitigation:

- Ensure summaries generated by LLMs are clearly labeled as "AI-assisted" and paired with visual or numerical justification.
- Allow users to view raw data and computed parameters alongside the AI summaries.

Risk: Unauthorized access to data pipelines or APIs in the web dashboard.

Mitigation:

- Secure APIs using authentication tokens and HTTPS.
- Implement user-level rate limiting and request validation to prevent abuse.

14.4 Mitigation Strategies Summary

Risk Category	Key Strategy
Data Scarcity	Use ensemble catalogues, semi-supervised ML
Visualization Overhead	Limit date range input, pre-compute plots
Accuracy Concerns	Apply multi-sensor thresholds and validation
Project Delay	Use offline datasets, modular development
AI Misuse	Pair summaries with source plots
Security Risks	Secure API with tokens, validate inputs

Section 15: Project Timeline

The project is structured into carefully phased milestones to ensure efficient development, scientific rigor, and timely delivery. The timeline spans across **6 major phases**, from data analysis and model training to deployment and documentation.

15.1 Gantt Chart with Milestones

Phase	Milestones
Phase 1: Research & Data Curation	Acquisition of Level-2 SWIS-ASPEX `.cdf` files; validation from CACTus/HELcats
Phase 2: Data Processing & Visuals	CDF → CSV conversion; peak detection; plotting sector-wise flux, speed, density
Phase 3: Threshold Derivation	Statistical thresholding (15 proton/cm^3, 0.62 He^{2+}/H^+, etc.); catalog matching
Phase 4: ML Model Development	Feature engineering; anomaly detection pipeline; model training & validation
Phase 5: Web App & AI Integration	Flask + Streamlit dashboard; AI chatbot; LLM-based CME summaries
Phase 6: Final Testing & Docs	End-to-end testing, writeups, demo recording, and project submission

15.2 Task Dependencies

- **Threshold Derivation** depends on successful visual inspection and catalogue correlation from Phase 2.
- **ML Model Development** requires fully labeled data from Phase 3.
- **Web App Deployment** depends on trained model outputs and prepared visual artifacts.
- **AI Summary Generation** depends on availability of processed outputs from previous phases.

15.3 Contingency Planning

Risk	Mitigation Strategy
Delay in obtaining magnetometer (MAG) data	Use BLK and STEPS-ASPEX exclusively; MAG added in future versions
Inconsistencies in labelled catalogues	Use consensus events across CACTus, SOHO, HELCATS, Richardson
LLM integration limits (token/usage quotas)	Cache generated summaries, limit concurrent LLM API calls
Visualization rendering lag	Optimize using pre-computed plots and on-demand render requests
Team resource unavailability	Assign overlapping roles; maintain centralized documentation and notebooks

Section 16: Deployment and Maintenance Plan

This section outlines the procedures, environments, and workflows involved in deploying the Halo CME Detection System, along with the long-term maintenance strategy to ensure continued reliability, performance, and extensibility.

16.1 Deployment Process

The deployment strategy is designed for modular scalability and performance. The system is initially deployed as a cloud-hosted application with minimal dependencies, making it accessible to both scientific researchers and non-experts.

Deployment Pipeline:

1. **Backend API** built using `Flask/FastAPI` handles processed data requests and CME predictions.
2. **Frontend Dashboard** built using `Streamlit` or deployed via `Render/HuggingFace Spaces` for interactive visualization and user engagement.
3. **ML Model Artifacts** are serialized using `Pickle` or `joblib` and served through inference endpoints.
4. **Data Storage** is maintained in structured JSON/CSV format for easy versioning and cross-platform compatibility.

Hosting Options:

- Google Cloud / Render / HuggingFace Spaces (preferred for academic deployments).
- GitHub Actions for CI/CD.
- Optional Docker containerization for portability.

16.2 Continuous Integration / Continuous Deployment (CI/CD)

To maintain scientific agility and ensure robustness:

- **Version Control**: GitHub with semantic commit conventions.
- **CI Tools**: GitHub Actions to auto-run linting, tests, and model validations.
- **CD Triggers**: Every commit to the `main` branch triggers deployment to the web dashboard (staging or production environment).
- **Data Sync Jobs**: Scheduled CRON jobs fetch updated CME catalogues and SWIS-ASPEX data periodically.

16.3 Post-Deployment Support and Monitoring

To ensure performance and anomaly detection post-deployment:

- **Logging**: All model inferences and user interactions are logged using Python's `logging` module and optionally sent to a logging service (e.g., Logtail, LogRocket).
- **Error Reporting**: Custom error handlers and fallback modes if API or LLM fails.

- **Usage Analytics**: Basic usage metrics (e.g., most queried date windows, bot usage) are recorded for further optimization.

16.4 Maintenance Schedule

Activity	Frequency	Responsibility
Model Retraining (with new data)	Monthly or as needed	ML Engineer / Data Scientist
Threshold Review & Update	Bi-monthly	Solar Physicist / Analyst
Dashboard UI Enhancements	Quarterly	Frontend Developer
Security Patch & Dependency Updates	As required	DevOps / Backend Engineer
AI Chatbot Fine-tuning	Bi-monthly	NLP Engineer / Research Assistant

Section 17: Legal, Compliance, and Ethics

This section outlines the legal and ethical considerations essential to the responsible development, deployment, and usage of the Halo CME Detection System. The system aligns with national data governance principles and global best practices for AI-based scientific tools.

17.1 Data Privacy and Compliance

Although the data used in this system is publicly available scientific data (non-personal), adherence to data privacy best practices is crucial to ensure trust, transparency, and future extensibility with external APIs.

- **Scientific Data Sources**: The system exclusively uses space-science datasets from government institutions (e.g., Aditya-L1 SWIS-ASPEX, MAG, and STEPS). These datasets are non-personal and non-restrictive in nature.
- **Open Access Compliance**: The system aligns with ISRO's data dissemination policies and international solar observatory data standards.
- **Anonymity and Data Scope**: No user data is collected during inference or model evaluation. Input data requests are scoped to scientific parameters (e.g., time window, energy band).
- **GDPR/DPDP (India)**: In case of future integration with real-time or user-specific alerts, the system will adopt GDPR and India's DPDP 2023 standards for data storage, retention, and erasure.

17.2 Terms of Use and Licensing

To maintain transparency and open collaboration, the system will adopt an appropriate open-source license.

- **License Recommendation**: Apache 2.0 or MIT License (for open research collaboration).
- **Attribution Requirement**: Proper citation and attribution required when used for academic or institutional work.
- **Terms of Use**:
  - The platform is for **research and educational purposes only**.
  - Users must not use the tool to draw unverified conclusions for mission-critical applications (e.g., satellite maneuvering) unless approved by a space weather authority.
  - AI summaries and chatbot responses are **informational aids**, not replacements for scientific review.

17.3 Ethical Considerations

As an AI-powered scientific tool, the project adheres to ethical AI development principles, including transparency, explainability, and user awareness.

- **Explainable Outputs**: Users are shown not just predictions, but also visual thresholds, derived metrics, and confidence levels.
- **Bias Mitigation**:
  - No demographic or human-related biases are introduced (as all data is observational physics data).
  - Cross-catalogue CME event matching minimizes temporal or source bias.
- **LLM Transparency**:
  - Outputs from LLMs (e.g., ChatGPT, Gemini) are labeled as AI-generated and used to enhance, not override, scientific interpretation.
- **Beginner Support Ethic**:
  - The system prioritizes **educational inclusivity** by offering plain-language summaries and interactive visualizations, democratizing access to advanced heliophysics tools.

Section 18: Success Metrics and KPIs

To evaluate the effectiveness, usability, and scientific accuracy of the Halo CME Detection System, we define a set of **Key Performance Indicators (KPIs)** across technical, operational, and research-oriented dimensions. These metrics guide system validation, highlight areas for future improvement, and ensure scientific integrity.

18.1 System Performance Metrics

These metrics evaluate the efficiency and responsiveness of the deployed solution under real-world usage.

Metric	Target
Average Data Processing Time	≤ 5 seconds per time window queried
ML Inference Latency	≤ 2 seconds for CME classification
Dashboard Load Time	≤ 3 seconds
API Uptime	≥ 99% availability
Peak Throughput	10+ concurrent users with no lag

18.2 User Adoption and Engagement

These indicators track the educational impact and community engagement with the web application and its AI-enhanced features.

Metric	Goal
Unique Monthly Users	≥ 500 (targeted outreach to astronomy forums)
Repeated Users (30-day retention)	≥ 30%
Chatbot Interactions	≥ 200 queries/month
Average Session Duration	≥ 4 minutes
External Citations or Use in Projects	≥ 3 academic references or integrations

---

18.3 Accuracy and Precision of ML Model

These metrics validate the scientific correctness and reliability of the CME classification logic.

Metric	Goal
Precision (CME detection)	≥ 90%
Recall / Sensitivity	≥ 85%
F1-Score	≥ 87%
False Positive Rate	≤ 10%

Metric	Goal
Ground Truth Matching	90%+ match with known catalogues

Evaluation is performed against cross-referenced events from CACTUS, SOHO, HelCats, and Richardson-Kane catalogues.

18.4 Reporting and Scientific Output

Metrics for the quality and usefulness of AI summaries, visualizations, and downstream research usability.

Metric	Goal
AI-Generated Summary Accuracy	≥ 95% match with event metadata
Chart Accuracy	Verified against CME thresholds and peaks
3D Visualization Clarity (user-rated)	≥ 4/5 average rating (feedback collection)
Publications Enabled / Cited In	≥ 1 research output (workshop, poster, paper)

Section 19: Conclusion and Next Steps

19.1 Conclusion

The proposed **Halo CME Detection and Alert System using Aditya-L1 SWIS-ASPEX Data** represents a significant step forward in democratizing heliophysics research and enabling early warning systems for space weather. By leveraging Level-2 particle data from Aditya-L1's SWIS payload — along with complementary instruments (MAG, STEPS-ASPEX) — the system builds a robust, interpretable model to detect Halo Coronal Mass Ejections in near real-time.

Key innovations include:

- **Scientifically validated thresholds** for CME signatures based on empirical analysis.
- A **beginner-friendly interface** with AI-enhanced descriptions, 3D solar animations, and CME visualizations.
- Integration of **machine learning** with multi-catalogue validation and real-time alerting capabilities.
- **Interoperability and openness**, ensuring extensibility for future research and educational use.

The system not only aids researchers in better understanding solar activity at the Lagrangian Point L1 but also empowers policymakers and engineers to protect space-borne assets through timely alerts.

19.2 Next Steps

To continue this innovation journey, we propose the following roadmap for expansion and deployment:

Short-Term (Post-Hackathon)

- Finalize real-time CME detection with full dashboard integration.
- Refine model thresholds using additional catalogues and recent data.
- Publish GitHub repository with documentation and sample datasets.
- Submit paper/poster to relevant conferences (e.g., AGU, COSPAR, ISRO workshops).

Mid-Term (3–6 Months)

- Integrate real-time data pipelines for SWIS and MAG (when available from ISRO).
- Add dynamic L1 visualizations based on satellite telemetry.
- Expand model using unsupervised anomaly detection techniques for unseen CME types.
- Collaborate with ISRO scientists or academic advisors for validation.

Long-Term Vision

- Transition to a full-fledged **space weather portal** with API access for satellite operators.
- Publish threshold derivation methodology as a **research paper**.
- Train larger transformer-based models for summarization of CME events using LLMs.
- Integrate Indian Deep Space Network alerts and potential downstream data from missions like **Gaganyaan**, **Chandrayaan**, or **future solar missions**.

This project aspires to contribute to India's scientific leadership in the domain of **space weather forecasting**, aligning with the vision of **self-reliant, AI-empowered, and space-ready infrastructure**.