**Product Requirements Document (PRD)**

**Title:** Satellite Metrics Threshold Monitoring WebApp  
**Owner:** Kushal Shah  
**Date:** 2025-05-16  
**Version:** 1.1

**1. Purpose**

To build a scalable and configurable web application that monitors MATLAB-generated metrics for a constellation of 10 satellite payloads. When thresholds are breached, trigger events are recorded and displayed in a web-based dashboard.

**2. Problem Statement**

Satellite operations teams require an automated system to detect and view threshold breaches across payloads, replacing manual tracking for improved efficiency and visibility.

**3. Scope**

* Monitor 3 metrics per payload (e.g., Thermal, Voltage, Latency)
* Detect breaches based on config-defined thresholds
* Store and display trigger events
* Provide a web-based dashboard for real-time visibility
* Support dynamic addition of metrics and payloads

**4. Key Features**

**A. Metric Monitoring & Trigger Detection**

* MATLAB scripts run at regular intervals (e.g., every 10 mins)
* Each script:
  + Reads performance data from archived files
  + Compares values against thresholds from a config file
  + Returns {timestamp, scid, metric\_type, value} if breach is detected
* Python middleware logs breach events in the database

**B. Configuration-Driven Architecture**

* All thresholds and metrics defined in an external config file
* Easily editable without code changes
* Example Config:

json

CopyEdit

{

"metrics": {

"thermal": {"threshold": 75.0},

"voltage": {"threshold": 3.3},

"latency": {"threshold": 250}

}

}

**C. Database (Dynamically Created)**

* SQLite used as local store
* Schema auto-generated at runtime based on config
* **Table:** metric\_triggers
  + id (PK)
  + scid (int)
  + metric\_type (string)
  + timestamp (datetime)
  + value (float)
  + threshold (float)
  + status (enum: NORMAL / BREACH)

**D. WebApp UI**

* Built using Flask + Jinja2
* Styled with a common Python-compatible UI kit (e.g., Bootstrap via Flask extensions)
* **Dashboard View**:
  + Rollup per payload (breach counts per metric)
  + Visual indicators (e.g., red/yellow/green)
* **Trigger Events Table**:
  + Columns: Timestamp, SCID, Metric Type, Value, Threshold
  + Sortable and filterable by scid, metric\_type, date\_range

**5. Non-Functional Requirements**

* **Scalability:** Support dynamic number of payloads and metrics
* **Configurability:** All logic driven by config file, minimal hardcoding
* **Performance:** Fast dashboard load (even with 1000+ events)
* **Portability:** No hosting or CI/CD required; local or internal deployment
* **Security:** Basic access control if needed in future

**6. Tech Stack**

* **Frontend:** Flask + Jinja2, styled using a Python-compatible UI kit (e.g., Flask-Bootstrap or Flask-Admin)
* **Backend:** Python
* **Database:** SQLite (schema created dynamically based on config)
* **Execution:** No hosting; local or internal network execution
* **CI/CD:** Not implemented yet