# Project Documentation for Monitoring and Reporting Platform

## Platform Overview and Concept

### Platform Summary

This platform is designed to serve as a centralized monitoring and reporting system for multiple Drupal websites.   
It will track and report on module versions, including security and non-security updates, across all registered sites.   
By facilitating comprehensive version tracking and update reporting, the platform enables site administrators and developers   
to maintain updated, secure environments.

### Core Principles

1. \*\*Centralized Monitoring\*\*: Aggregate module version data from multiple Drupal websites, providing administrators a consolidated view of module versions and pending updates.  
2. \*\*Version History and Reporting\*\*: Retain historical records of module versions and updates, enabling detailed reporting for any time period (e.g., weekly, monthly, year-to-date).  
3. \*\*Prioritized Security Updates\*\*: Emphasize security updates for easy identification, ensuring that site administrators can prioritize critical patches.  
4. \*\*Role-Based Access at Organization Level\*\*: Recognize users at the organization level rather than individual accounts. Each organization can have multiple members with different roles, such as Administrators (full access) and Developers (read-only access).  
5. \*\*Customizable Notifications\*\*: Offer configurable email notifications for module updates, including security and non-security updates, with frequency settings and preferences.

## Technology Stack and Architectural Overview

The following technology stack has been selected for the initial development of the monitoring and reporting platform,   
chosen based on requirements for high concurrency, scalability, and efficient handling of data from multiple Drupal sites.  
  
1. \*\*Back-End Framework\*\*: \*\*FastAPI (Python)\*\*  
 - Provides asynchronous processing for high throughput and efficient handling of REST API requests.  
 - Supports easy integration with Redis and PostgreSQL, enabling smooth data ingestion and processing.  
  
2. \*\*Front-End Framework\*\*: \*\*React\*\*  
 - Allows for dynamic, responsive interfaces that support complex data visualization and user interactions.  
 - Enables integration with role-based access controls, providing distinct views and functionalities for different user roles (Administrator vs. Developer).  
  
3. \*\*Database\*\*: \*\*PostgreSQL\*\*  
 - A robust, relational database with support for JSONB fields to store semi-structured JSON payloads from the Drupal sites.  
 - Ideal for structured storage and relational querying, essential for tracking module versions and maintaining historical records.  
  
4. \*\*Queue and Data Streams\*\*: \*\*Redis Streams\*\*  
 - Redis Streams will handle incoming data bursts from the API, enabling asynchronous processing and buffering during high-traffic periods.  
 - Facilitates real-time updates and high-speed data handling with low latency.  
  
5. \*\*Authentication and Security Layer\*\*:  
 - \*\*API Key-Based Authentication\*\*: For secure communication between Drupal sites and the REST API.  
 - \*\*Rate Limiting\*\*: Implemented using Redis-based counters to manage and control request frequency.  
  
6. \*\*Push Notifications (Stage 2)\*\*:  
 - Notifications for security and non-security updates will be handled via \*\*Firebase Cloud Messaging (FCM)\*\* and \*\*Apple Push Notification Service (APNS)\*\* for mobile alerts.  
  
7. \*\*Mobile Application (Stage 2)\*\*:  
 - Future mobile applications will be developed using \*\*React Native\*\* to allow code reuse and seamless integration with the existing React front-end components.  
  
This stack provides a balanced approach to performance, scalability, and flexibility, accommodating the current functional requirements and facilitating future expansion into mobile and real-time notification capabilities.

## Overall Requirements

### Functional Requirements

1. \*\*Data Ingestion via Drupal Module\*\*:  
 - The Drupal module installed on each monitored website will perform a scheduled (cron-based) POST request in JSON format to the platform’s REST API.  
 - Each POST request includes:  
 - Site identifier (unique to each site).  
 - Module version data, including Drupal core version and module versions.  
 - After updates are applied on a site (e.g., during a patch run), the module will push the latest module versions to the platform, ensuring records remain up-to-date.  
  
2. \*\*Site and Module Management\*\*:  
 - \*\*Site List\*\*: A user-accessible list of all sites registered within the organization. Each entry includes:  
 - Site name (editable in the interface by administrators).  
 - URL (editable in the interface by administrators).  
 - Date added (read-only).  
 - Current core version and counts of modules with available updates and security updates.  
 - \*\*Site Detail View\*\*: Displays:  
 - List of modules installed on the site.  
 - Module name, current version, link to Drupal.org module page.  
 - \*\*Update Prioritization\*\*:  
 - Security updates should be highlighted at the top in a standalone list.  
 - Non-security updates follow.  
 - Modules with no updates will be displayed in a separate list.  
  
3. \*\*Reporting and Historical Records\*\*:  
 - \*\*Historical Report Generation\*\*: Allow users to generate historical reports covering specified timeframes.  
 - \*\*Patch Run Reports\*\*: Track each update cycle as a "patch run," documenting updated versions.  
 - \*\*Update Lag Tracking\*\*: Report on the time taken to apply security updates.  
  
4. \*\*Email Notifications\*\*:  
 - Configurable email alerts for updates, with user-configurable preferences:  
 - Frequency: For security and non-security updates.  
 - Email Address: Specify preferred emails for updates.  
 - Disable Notifications: Option to disable notifications entirely.

## Refactored Database Structure

To accommodate expanded tracking and auditing requirements, the database structure has been revised as follows:  
  
1. \*\*Organizations Table\*\*:  
 - `id` (Primary Key): Unique identifier for the organization.  
 - `name`: Organization name.  
 - `created\_at`: Timestamp of organization creation.  
 - `created\_by`: Foreign key linking to the user who created the organization.  
 - `modified\_at`: Timestamp of last modification.  
  
2. \*\*Organization Revisions Table\*\* (New):  
 - \*\*id\*\* (Primary Key): Unique identifier for each revision.  
 - `organization\_id`: Foreign key linking to the Organizations table.  
 - `created\_by`: User who created this revision.  
 - `created\_at`: Timestamp of when the revision was created.  
 - `modified\_by`: User who last modified this revision.  
 - `modified\_at`: Timestamp of the last modification.  
  
3. \*\*Users Table\*\*:  
 - `id` (Primary Key): Unique identifier for each user.  
 - `email`: Email address of the user.  
 - `organization\_id`: Foreign key linking to the Organizations table.  
 - `role`: Role within the organization (Administrator or Developer).  
 - `password\_hash`: Hash of the user password (stored securely).  
 - `created\_at`: Timestamp of user creation.  
 - `modified\_at`: Timestamp of the last modification.  
 - `notification\_preferences`: JSON field storing user-specific notification settings.  
  
4. \*\*Sites Table\*\*:  
 - `id` (Primary Key): Unique identifier for each site.  
 - `organization\_id`: Foreign key linking to the Organizations table.  
 - `uuid`: Unique identifier for the site (provided to the Drupal module).  
 - `name`: Name of the site.  
 - `url`: URL of the site.  
 - `created\_at`: Timestamp when the site was registered.  
  
5. \*\*Modules Table\*\*:  
 - `id` (Primary Key): Unique identifier for each module.  
 - `module\_name`: Name of the module.  
 - `drupal\_link`: URL to the module’s page on Drupal.org.  
  
6. \*\*Module Versions Table\*\*:  
 - `id` (Primary Key): Unique identifier for each module version.  
 - `module\_id`: Foreign key linking to the Modules table.  
 - `version\_number`: Version number of the module.  
 - `release\_date`: Date when this version was released.  
  
7. \*\*Site Modules Table\*\*:  
 - Tracks the current version of each module installed on each site.  
 - `site\_id`: Foreign key linking to the Sites table.  
 - `module\_id`: Foreign key linking to the Modules table.  
 - `current\_version\_id`: Foreign key linking to the Module Versions table.  
 - `update\_available`: Boolean indicating if an update is available.  
 - `security\_update`: Boolean indicating if it’s a security update.  
  
8. \*\*Patch Runs Table\*\*:  
 - Records each patch run cycle for tracking historical updates.  
 - `id` (Primary Key): Unique identifier for each patch run.  
 - `site\_id`: Foreign key linking to the Sites table.  
 - `run\_date`: Timestamp for when the patch run was recorded.  
  
9. \*\*Audit Logs Table\*\*:  
 - `id` (Primary Key): Unique identifier for each log entry.  
 - `site\_id`: Foreign key linking to the Sites table.  
 - `data`: JSON field capturing raw POST data from the Drupal module.  
 - `status`: Status of the request (e.g., success, validation error).  
 - `created\_at`: Timestamp of the log entry.