### **Product Requirements Document (PRD)**

#### **Project Title: Domain Monitoring System**

#### **1. Executive Summary**

* **Objective**: Develop a domain monitoring system that allows users to register, log in, and monitor the status and SSL certificates of their domains. This system will provide an easy-to-use interface for managing multiple domains, giving real-time updates on their status.
* **Background**: Domain monitoring is crucial for users who rely on website uptime and SSL security. This project offers a practical approach to managing domains, ensuring users can monitor domain health and SSL expiration dates easily and efficiently.
* **Target Audience**: Webmasters, small business owners, IT professionals

#### **2. Goals and Success Metrics**

* **Product Goals**:
  + Enable users to register, log in, and access a personalized dashboard.
  + Allow users to add, manage, and monitor multiple domains.
  + Use multithreaded scanning to efficiently monitor domains.
* **Success Metrics**:
  + Efficiency: Domain checks for up to 100 domains complete within 5 seconds per user.
  + Error rate: Less than 2% failure rate for domain checks and SSL validations.
  + System is able to handle 10 users, performing 100 domains scans each concurrently, with no significant performance degradation.

#### **3. Functional Requirements**

* **User Stories**:
  + As a **user**, I want to **register and log in** to have access to a personalized dashboard for my domains.
  + As a **user**, I want to **add single or bulk domains** to efficiently monitor my websites’ health and SSL status.
  + As a **user**, I want to **view my domains' current status** and receive real-time updates on issues like SSL expiration.
* **Features & Specifications**:
  + **User Registration and Login**:
    - Users register with a unique username and password stored in users.json.
    - Users log in with their credentials and maintain a session using Flask’s session object.
  + **Dashboard**:
    - After logging in, users see a dashboard displaying their domains and monitoring results.
  + **Domain Monitoring**:
    - For each domain, the system checks:
      * **Liveness**: Domain returns a status code 200.
      * **SSL Expiration**: Retrieves SSL expiration date and issuing authority.
    - Users can add domains individually or in bulk via a .txt file upload.
  + **Multithreaded Scanning**:
    - Uses concurrent.futures.ThreadPoolExecutor to handle multiple domain checks concurrently.
  + **Data Storage**:
    - User data stored in users.json.
    - Each user has an individual JSON file <username>\_domains.json containing domain data and monitoring results.

#### **4. Non-Functional Requirements**

* **Performance**:
  + The system should support up to 100 domains per user with completion of checks within 5 seconds per user.
* **Security**:
  + For simplicity, store passwords in plain text initially (later to be encrypted).
  + Use secure sessions to manage logins.
* **Reliability**:
  + The system should handle JSON read/write operations without impacting performance.
* **Scalability**:
  + At a later stage, the JSON-based storage will be transitioned to a database.

#### 

#### **5. User Interface and User Experience (UI/UX) Requirements**

* **User Interface Design**: Simple, minimalistic layout using a Nordic color palette for an intuitive, user-friendly experience.
* **User Experience Flow**:
  + **Registration and Login**: Users can create an account and log in to access the dashboard.
  + **Domain Management**: Users can add domains, view the current status of their domains, and remove domains if needed.
  + **Dashboard**: Displays domains, status, SSL expiration date, and SSL issuer in a clear table format.

#### **6. Technical Constraints and Considerations**

* **Technology Stack**: Backend in Python and Flask; Frontend in HTML/CSS/JavaScript.
* **Dependencies**: Flask, concurrent.futures for multithreading, requests library for liveness checks, ssl module for SSL validation.
* **Assumptions**:
  + Each user will have a maximum of 100 domains for monitoring.

#### **7. Acceptance Criteria**

* Users can register, log in, and view a personalized dashboard.
* Users can add domains individually or in bulk, with domains stored in <username>\_domains.json.
* The monitoring system performs concurrent domain checks, providing status and SSL details.
* Monitoring results are displayed in a table on the user’s dashboard, with correct domain statuses and SSL details.

#### **8. Timeline and Milestones**

* **Milestone 1**: Set up user registration and login functionality - *[Date]*
* **Milestone 2**: Implement domain upload with single and bulk upload options - *[Date]*
* **Milestone 3**: Finalize multithreaded domain checking and data storage - *[Date]*
* **Milestone 4**: Complete UI for dashboard with domain status display - *[Date]*

#### **9. Risks and Mitigations**

* **Potential Risks**:
  + High I/O operations with JSON files could slow down the system.
  + Multithreading complexities may introduce bugs during concurrent domain checks.
* **Mitigations**:
  + Optimize JSON read/write operations and consider caching.
  + Conduct thorough testing for concurrency issues, implementing error logging for debugging.