### **Product Requirement Document (PRD)**

**Product Name:** Document Search System  
**Version:** 1.0  
**Author:** Deepesh Garg  
**Date:** November 23, 2024

### **1. Objective**

Build a Document Search System that allows users to search documents stored in an Amazon S3 bucket. The system uses Elasticsearch for full-text indexing and searching of file contents and metadata. Users should be able to access the system via a web interface.

### **2. Key Features**

1. **Document Upload and Storage**
   * Upload files (.pdf, .txt) to Amazon S3.
   * Store file metadata such as fileName and updatedAt.
2. **Indexing and Syncing**
   * Sync files from S3 to Elasticsearch, creating searchable indexes for file content and metadata.
   * Update Elasticsearch with new or modified files in S3.
3. **Search Functionality**
   * Full-text search of document content and metadata (e.g., fileName).
   * Search results should include the fileName, S3 URL, and updatedAt.
4. **Responsive Web Interface**
   * A single-page application (SPA) with:
     + A text field and search button.
     + Validations for empty inputs.
     + Search results displayed in the form of cards.
     + Embedded clickable URLs to download files from S3.
     + A spinner overlay during API calls and success/error snackbar notifications.
5. **Backend APIs**
   * **Sync API**: Fetch files from S3 and index them in Elasticsearch.
   * **Search API**: Query Elasticsearch and return relevant documents.

### **3. Scope**

The system is designed for businesses needing a centralized and searchable document repository. It should be scalable to handle thousands of documents while maintaining quick search responses.

### **4. Functional Requirements**

#### **Frontend**

* **Nav Bar:**
  + Displays the title: "Document Search".
* **Search Bar:**
  + A text input field with validation for empty queries.
  + A "Search" button that triggers the Search API.
  + Spinner displayed during the API call.
  + Search results displayed in a styled cards, showing:
    - fileName
    - updatedAt.
    - View Document (Embedded URL)
* **Notifications:**
  + Success or error snackbar notifications in the top-right corner.

#### **Backend**

* **Environment Setup:**
  + Use Environment Variables to securely store credentials for S3 and Elasticsearch.
* **APIs:**
  + **Sync API**:
    - Endpoint: /parseFile/sync
    - Method: POST
    - Fetch all files from the S3 bucket, parse them, and index them in Elasticsearch.
  + **Search API**:
    - Endpoint: /parseFile/search
    - Method: GET
    - Query Elasticsearch based on text input and return matching files.
* **Error Handling:**
  + Handle missing S3 credentials, Elasticsearch connection issues, and invalid inputs gracefully.

#### **Storage**

* **Amazon S3:**
  + Store files and metadata.
* **Elasticsearch:**
  + Store indexed content for fast and scalable search.

### **5. Non-Functional Requirements**

* **Performance:**
  + Search results should be returned within 300ms.
* **Scalability:**
  + Support up to 10,000 documents in Elasticsearch.
* **Security:**
  + Use AWS IAM roles for secure access to S3.
  + Store sensitive credentials (e.g., AWS keys, Elasticsearch URL) in .env files.
  + Enforce CORS policies on backend APIs.

### **6. System Architecture**

1. **Frontend:**
   * React app hosted on Netlify.
2. **Backend:**
   * Express app hosted on Render.
   * Connects to S3 for file storage and Elasticsearch for indexing and searching.
3. **Data Flow:**
   * User searches via the frontend.
   * Frontend calls backend APIs.
   * Backend interacts with S3 and Elasticsearch.

### **7. User Stories**

1. **As a User**, I want to search documents by content or name so that I can quickly find relevant files.
2. **As a User**, I want to view files from the search results so that I can access their full content.
3. **As a User**, I want to see notifications when a search succeeds or fails to know the system status.

### **8. Deployment Steps**

1. **Frontend:**
   * Push the React code to GitHub.
   * Deploy on Netlify with the build command npm run build and build as the publish directory.
2. **Backend:**
   * Push the Express app code to GitHub.
   * Deploy on Render with environment variables:
     + PORT: Port for the backend server.
     + S3\_SECRET\_ACCESSS\_KEY, S3\_ACCESS\_KEY: AWS credentials for S3.
     + ELASTIC\_SEARCH\_USER\_NAME, ELASTIC\_SEARCH\_PASSWORD,ELASTIC\_SEARCH\_HOST,ELASTIC\_SEARCH\_PORT: Elasticsearch Credentials.

### **9. Future Enhancements**

1. Add user authentication and role-based access control.
2. Implement OCR for image-based documents.
3. Allow advanced filters in search (e.g., date range, file type).
4. Add versioning support for files in S3.

### **10. Appendix**

* **APIs Reference:**
  + **Sync API:** /parseFile/sync
  + **Search API:** /parseFile/search?text=<query>