**Somali Digital Library System Documentation**

**Overview**

The Somali Digital Library is a comprehensive platform designed to provide access to educational and cultural resources for students, researchers, and the general public. The system consists of:

* A **public website** for users to explore and access resources.
* A **university portal** for managing student access to premium content.
* An **admin panel** for managing resources, users, and subscriptions.

This document outlines the **system architecture, features, tech stack, database design, and implementation details**.

**Features**

**Public Website**

1. **Landing Page**:
   * Displays general information about the library.
   * Sections: About, Contact, Impressions, Trusted by Universities/Institutions.
2. **Resource Browsing**:
   * Explore featured and most downloaded books.
   * Search and filter resources by title, author, year, and category (IEEE-style filtering).
3. **Resource Details**:
   * Displays full metadata of a resource.
   * Includes a share button and a download button.
4. **Resource Access**:
   * Non-premium resources can be downloaded freely.
   * Premium resources require an active subscription.
   * Authentication flow:
     + If logged in, download starts immediately.
     + If not logged in, redirect to login/signup before proceeding.
5. **User Authentication**:
   * Allows account creation with email, username, and password.
   * Students log in using credentials provided by their university.

**University Portal**

1. **University Admin Authentication**:
   * Universities log in using admin credentials created by the main admin.
   * University admins can change their password.
2. **Student Management**:
   * Add/remove students.
   * Generate unique credentials for students.
3. **Subscription Management**:
   * Manage subscription status and view usage statistics.

**Admin Panel**

1. **Resource Management**:
   * CRUD operations on resources.
   * Categorization of resources.
2. **User Management**:
   * Manage individual and university user accounts.
3. **Subscription Management**:
   * Manage both university and individual subscriptions.
4. **Analytics**:
   * Track downloads, user activity, and subscription usage.

**Tech Stack**

**Frontend**

* **Next.js** – Public website.
* **React Admin Panel** – Admin and university portal.

**Backend**

* **Node.js** – Server logic and API development.
* **Express.js** – RESTful API framework.

**Database**

* **MySQL** – Stores user data, subscriptions, and resources.

**File Storage**

* **Amazon S3** / **DigitalOcean Spaces** – Secure storage for PDFs, DOCX, etc.

**Hosting**

* **DigitalOcean Droplets** – Hosts the Node.js server and MySQL database.
* **Vercel** – Hosts the Next.js frontend (optional).

**Authentication**

* **JWT (JSON Web Tokens)** – Secure authentication.
* **Bcrypt** – Password hashing.

**Database Design**

**Tables**

1. **Users**
   * user\_id, username, email, password\_hash, created\_at
2. **Universities**
   * university\_id, name, email, created\_at
3. **UniversityAdmins**
   * admin\_id, university\_id, username, password\_hash, created\_at
4. **UniversityStudents**
   * student\_id, university\_id, username, password\_hash, email, created\_at
5. **Resources**
   * resource\_id, title, author, description, file\_url, is\_premium, created\_at
6. **Subscriptions**
   * subscription\_id, name, price, duration\_days, max\_users
7. **UserSubscriptions**
   * user\_sub\_id, user\_id, university\_id, subscription\_id, start\_date, end\_date
8. **Categories**
   * category\_id, name
9. **ResourceCategories**
   * resource\_cat\_id, resource\_id, category\_id
10. **Favorites**

* favorite\_id, user\_id, resource\_id, created\_at

1. **Downloads**

* download\_id, user\_id, resource\_id, downloaded\_at

**File Storage**

**Cloud Storage**

* Use **Amazon S3** or **DigitalOcean Spaces** to store resources.
* Store only **file URLs** in the database.
* Use **signed URLs** for secure access to premium resources.

**Workflow**

1. User uploads a PDF.
2. Backend uploads the file to S3 and saves the URL in the database.
3. Users access the file via the stored URL.

**Authentication Flow**

**User Authentication**

1. **Students**:
   * Log in using university-provided credentials.
   * Select university from a dropdown list.
2. **Individual Users**:
   * Sign up with email and password.
3. **Universities**:
   * Log in via university admin credentials.

**Secure Access**

* **JWT authentication** for all requests.
* **Bcrypt password hashing**.
* **Session persistence** for returning users.

**Subscription Management**

**University Subscriptions**

1. Universities purchase subscriptions.
2. Admins manage student access.
3. Students access premium resources under their university plan.

**Individual Subscriptions**

1. Users purchase a subscription.
2. Subscription status determines access to premium resources.

**Implementation Steps**

1. **Set Up Hosting**:
   * Create a DigitalOcean Droplet.
   * Set up cloud storage for file hosting.
2. **Develop Backend**:
   * Build RESTful APIs with Node.js and Express.js.
   * Implement authentication using JWT and Bcrypt.
   * Integrate cloud storage for file uploads.
3. **Develop Frontend**:
   * Build public website with Next.js.
   * Develop admin panel with React.
4. **Set Up Database**:
   * Create MySQL schema and tables.
   * Populate with initial data (categories, subscriptions, etc.).
5. **Testing & Deployment**:
   * Thoroughly test system functionality.
   * Deploy frontend to Vercel, backend to DigitalOcean.

**Security Best Practices**

1. **Encryption**:
   * Use HTTPS for secure data transmission.
   * Encrypt sensitive database fields.
2. **Access Control**:
   * Restrict access to admin and university portals.
   * Use **signed URLs** for premium resources.
3. **Backups**:
   * Implement automatic database and file backups.

**Conclusion**

The Somali Digital Library is a **scalable, secure, and user-friendly** platform for accessing educational resources. This documentation provides a structured guide for developing and maintaining the system.

Let me know if you need further refinements! 🚀