Credit-Based Document Scanning System (CBDSS) Documentation

---

1. Project Overview

The **Credit-Based Document Scanning System (CBDSS)** is a web application that allows users to upload and compare text documents for similarity. Users are given a limited number of credits to upload and scan documents. Admins can manage user credit requests and view system analytics.

---

2. Features

User Features

1. **User Registration and Login**:
    - Users can register with a unique username and email.
    - Users can log in to access their dashboard.
2. **Document Upload and Scanning**:
    - Users can upload plain text documents.
    - Each upload deducts 1 credit from the user's account.
    - The system compares the uploaded document with existing documents and displays matches with a similarity score.
3. **Credit Management**:
    - Users start with 20 credits.
    - Users can request additional credits, which require admin approval.
4. **User Profile**:
    - Users can view their profile, including their username, email, role, and remaining credits.
5. **Logout**:
    - Users can log out, which clears their session.

Admin Features

1. **Admin Login**:
    - A single admin account is created automatically when the application starts.
    - Admins can log in to access the admin dashboard.
2. **Admin Dashboard**:
    - Admins can view all uploaded documents.
    - Admins can approve or reject user credit requests.
3. **System Analytics**:
    - Admins can view total scans, top users, and most common topics in uploaded documents.

---

3. Frontend Implementation

1. Pages

1. **Login/Signup Page**:
    - Users can register or log in.
2. **Dashboard**:
    - Users can upload documents, view their uploaded documents, and request credits.
3. **Admin Dashboard**:
    - Admins can view all uploaded documents, approve/reject credit requests, and view system analytics.
4. **About Page**:
    - Provides information about the project.
5. **Contact Page**:
    - Displays contact information.

---

2. Dynamic Navigation

- If the user is logged in:
    - Regular users see **Dashboard**.
    - Admins see **Admin Dashboard**.
- If the user is not logged in, they see **Home**.

---

4. Backend Implementation

1. Setup

1. **Initialize Project**:
    - Create a project folder:
      ```
      mkdir creditBasedDocScanner
      cd creditBasedDocScanner
      ```
    - Initialize a Node.js project:
      ```
      npm init -y
      ```
2. **Install Dependencies**:
    - Install required packages:
      ```
      npm install express bcrypt sqlite3 express-session uuid dotenv fast-levenshtein
      ```
3. **Folder Structure**:
   ```
   Copy
   creditBasedDocScanner/
   ├── backend/
   │   ├── database/
   │   │   └── database.js
   │   ├── uploads/ (for storing uploaded documents)
   │   └── server.js
   ├── frontend/
   │   ├── css/
   │   │   ├── styles.css
   ```

```
│   │   ├── loginSignup.css
│   │   ├── dashboard.css
│   │   ├── admin-dashboard.css
│   │   ├── about.css
│   │   └── contact.css
│   ├── images/
│   │   └── logo.jpg
│   ├── about.html
│   ├── contact.html
│   ├── dashboard.html
│   ├── admin-dashboard.html
│   └── loginSignup.html
├── package.json
├── .env
└── README.md
```

4. **Environment Variables**:
    - Create a .env file:

```
SESSION_SECRET=supersecretkey
ADMIN_PASSWORD=admin123
```

---

2. Backend API Endpoints

1. **User Registration**:
    - **Endpoint**: POST /auth/register
    - **Body**:

```
{
  "username": "user1",
  "email": "user1@example.com",
  "password": "password123"
}
```

2. **User Login**:
    - **Endpoint**: POST /auth/login
    - **Body**:

```
{
  "email": "user1@example.com",
  "password": "password123"
}
```

3. **User Logout**:
    - **Endpoint**: POST /auth/logout
4. **User Profile**:
    - **Endpoint**: GET /user/profile
5. **Document Upload**:
    - **Endpoint**: POST /scan
    - **Body**:

```
{
  "text": "This is a sample document text.",
  "fileName": "sample.txt"
}
```

6. **Get Matching Documents**:
    - **Endpoint**: POST /scan
7. **Credit Request**:
    - **Endpoint**: POST /credits/request
    - **Body**:

```
{
  "amount": 20
}
```

8. **Admin Analytics**:
    - **Endpoint**: GET /admin/analytics
9. **Admin Dashboard**:
    - **Endpoint**: GET /admin/dashboard

---

3. Database Schema

1. **Users Table**:
```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL,
    role TEXT DEFAULT 'user',
    credits INTEGER DEFAULT 20
);
```
2. **Documents Table**:
```
CREATE TABLE documents (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    userId INTEGER NOT NULL,
    filePath TEXT NOT NULL,
    fileName TEXT NOT NULL,
    uploadedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (userId) REFERENCES users(id)
);
```
3. **Credit Requests Table**:
```
CREATE TABLE credit_requests (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    userId INTEGER NOT NULL,
    amount INTEGER NOT NULL,
    status TEXT DEFAULT 'pending',
    requestedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (userId) REFERENCES users(id)
);
```

---

5. How to Run the Project

1. Backend

1. **Install Dependencies**:
```
npm install
```
2. **Start the Server**:
```
nodemon backend/server.js
```
3. **Test APIs**:
    - Use **Postman** to test the backend APIs.

2. Frontend

1. **Open the Frontend**:
    - Open the frontend/ folder in a browser or use a local server (e.g., live-server).

2. **Test the Application**:
    - Register, log in, upload documents, and test other features.

---

6. Deployment

1. Backend (Using Render)

1. **Push your backend code to GitHub**.
2. **Deploy to Render**:
    - Create a new web service on Render and connect your GitHub repository.
    - Add environment variables (`SESSION_SECRET`, `ADMIN_PASSWORD`).
    - Deploy the backend.

2. Frontend (Using Netlify)

1. **Push your frontend code to GitHub**.
2. **Deploy to Netlify**:
    - Create a new site on Netlify and connect your GitHub repository.
    - Deploy the frontend.

---

7. Future Enhancements

1. **File Type Support**:
    - Allow users to upload PDFs and Word documents.
2. **Email Notifications**:
    - Notify users when their credit requests are approved or rejected.
3. **Advanced Analytics**:
    - Add more detailed analytics for admins.
4. **User Roles**:
    - Add more roles (e.g., moderator) with specific permissions.
5. **Mobile App**:
    - Develop a mobile app for easier access.
