Dear Manthan Sonawane ,

Overview:

You will build a self-contained document scanning and matching system with a built-in credit system. Each user has a daily limit of 20 free scans, and additional scans require requesting more credits.

The project must be completed within 10 days.

Using the internet for research is allowed, but copying code from online sources or peers will result in disqualification.

Bonus: Implement AI-powered document matching using OpenAI, Gemini, or DeepSeek for better results.

Assignment Timeline (10 Days):

DayTask Planner

1. Day 1-2 Set up the project (Backend, Frontend, Database)
2. Day 3-4 Implement User Authentication & Role Management
3. Day 5-6 Develop the Credit System (Daily Free Credits & Admin Approval)
4. Day 7-8 Implement Document Upload & Text Matching (Basic Version)
5. Day 9 Build Smart Analytics Dashboard & Optimize Features
6. Day 10 Testing, Documentation, and Final Submission

Assignment Requirements:

1. User Management & Authentication

User Registration & Login (Basic username/password authentication).

User Roles: Regular Users & Admins.

Profile Section to display user credits, past scans, and requests.

2. Credit System:

Each user gets 20 free scans per day (auto-reset at midnight).

Users must request additional credits if they exceed their limit.

Admins can approve or deny credit requests.

Each document scan deducts 1 credit from the user’s balance.

3. Document Scanning & Matching:

Users upload a plain text file for scanning.

System scans and compares it against existing stored documents.

Returns similar documents based on basic text similarity algorithms.

Bonus: Implement AI-based document matching using frameworks like OpenAI, Gemini, or DeepSeek to improve accuracy.

4. Smart Analytics Dashboard:

Track the number of scans per user per day.

Identify most common scanned document topics.

View top users by scans and credit usage.

Generate credit usage statistics for admins

5. API Endpoints:

Method | Endpoint | Description

a. POST. /auth/register | User registration

b. POST/auth/login| User login (Session-based)

c. GET/user/profile| Get user profile & credits

d. POST/scanUpload document for scanning (uses 1 credit)

e. GET/matches/:docIdGet matching documents

f. POST/credits/requestRequest admin to add credits

g. GET/admin/analyticsGet analytics for admins

h. Credit System Implementation

Daily Free Credits:

Every user starts with 20 free credits at midnight (local server reset).

If credits are exhausted, the user must wait till the next day or request admin approval for more credits.

Admin Credit Management:

Users can submit a request for additional credits.

Admin can approve or deny requests.

Admin can manually adjust user credit balances.

Handling Credit Deductions:

Each document scan deducts 1 credit.

If a user has 0 credits, they cannot scan until:

The next daily reset, OR

Admin manually adds more credits.

Tech Stack (Self-Contained, No Third-Party Services):

a. Frontend: HTML, CSS, JavaScript (No frameworks)

b. Backend: Python (Flask/Django) or Node.js (Express) without external libraries

c. Database: SQLite (or JSON files for small-scale storage)

d. File Storage: Store documents locally

e. Authentication: Basic username-password login (hashed passwords)

f. Text Matching Logic: Custom algorithm using Levenshtein distance, word frequency, etc.

Bonus (AI-Powered Matching):

If you want to enhance document matching, you can integrate AI-based similarity analysis:

a. OpenAI API (GPT models)

b. Google Gemini

c. DeepSeek AI

d. Self-hosted NLP models (e.g., spaCy, BERT, Llama2)

AI models can improve semantic similarity detection beyond basic word matching.

Bonus Features (Optional):

1. Automated Credit Reset: Auto-reset free credits at midnight (local time).
2. User Activity Logs: Track when users scan or request credits.
3. Admin Dashboard: Simple UI for approving credits & viewing analytics.
4. Export Reports: Allow users to download their scan history as a text file.
5. Multi-user Support: Handle multiple users on the same local server.

Rules for Submission & Disqualification:

1. Using the internet for research is allowed.

2 Copying code from online sources or peers will result in disqualification.

3. Your implementation must be original, structured, and easy to understand.
4. Plagiarism detection tools will be used to verify submissions.
5. Ensure the code is fully functional, well-documented, and includes setup instructions.

Evaluation Criteria:

1. Functionality: Do the credit system and document matching features work correctly?
2. Performance: Can the system handle multiple users and large document datasets?
3. Security: Are user passwords and data stored securely?
4. Scalability: Can the system support multiple users on a local server?
5. Code Quality: Is the code clean, modular, and well-documented?
6. AI Bonus: If AI-powered document matching is implemented, how effectively does it improve accuracy?

Submission Guidelines:

Submit a GitHub repository with:

1. README.md (Setup instructions)
2. Code files (Frontend, Backend, and Database)
3. Test data (sample documents)

You will receive the submission link after 25 February, where you need to provide screenshots or a short demo video of the working application.