Project title: Accreditation and Data Management Assistant for Institutions

Name: VIDYA SINHA 21, DHRUVI PATEL 29

1. Project Summary

The Accreditation and Data Management Assistant is an automated, centralized platform designed to overcome the inefficiencies and errors associated with manual data handling during institutional accreditation processes such as NBA and NAAC. The system's primary objective is to automate self-assessment score calculations and provide standardized dashboards that serve different stakeholders, including HODs, Deans, Administrators, and Accreditation Officers. By automating repetitive tasks and integrating data-driven reporting, the project ensures transparency, reduces human error, and significantly saves time in evaluation processes.

Core Objectives: Automate self-assessment score calculations and provide standardized dashboards and reporting tools for stakeholders (HODs, Deans, Administrators, Accreditation Officers).

Key Outcomes

1. Efficiency & Automation- Eliminates repetitive manual work through rule-based algorithms (Python), reducing errors and evaluation time.

2. Centralized Data Management- Uses PostgreSQL (Supabase) to securely store and retrieve large volumes of institutional data (students, faculty, research, achievements).

3. Security & Compliance- Implements encryption, authentication, and Role-Based Access Control and encrypted storage of documents to ensure data privacy and restricted access.

4. Reporting- Supports multiple accreditation bodies with live verification dashboards for instant score validation, gap analysis, and evidence linking.

2. System Design and Architecture

The system follows a three-tier, modular design to ensure scalability, maintainability, and compliance with ICT engineering principles.

Architecture Overview

Layer	Technology Stack	Functionality
Presentation (Frontend)	React with TypeScript, Tailwind CSS	User interface, dashboards (scores, compliance), data entry & file uploads. Enforces type safety.
Application (Backend)	Flask (Python), SQLAlchemy ORM, REST APIs	Orchestrates business logic, manages API requests, and executes scoring algorithms. Uses Blueprint for modular routing.
Data/Infrastructure	Supabase (PostgreSQL & Storage)	Relational data management, secure file storage (with separate buckets for evidence), authentication, and backups.

Communication: RESTful APIs over HTTPS for secure and standardized client–server interaction.

Implementation Highlights:

Modular Code Organization- Flask Blueprint pattern separates controllers (students, faculty, enrollment, etc.) for maintainability.

Secure File Upload System-Handles large volumes (~8 GB/year for 1000 students).
Validates file type, generates unique filenames, and securely stores in Supabase with controlled access URLs.

Database Management- SQLAlchemy ORM abstracts SQL queries, ensures maintainable and readable code, and manages complex queries across entities (Users, Institutions, Students, Faculty, Publications).
Quality Assurance- Code quality ensured via consistent conventions, separation of concerns, structured error handling, and unit testing with pytest

3. Scalability Strategies

The system is built to support 500+ students per year with the following strategies:

To ensure scalability, several strategies were embedded into the system design. Since the RESTful APIs are stateless, the backend can be horizontally scaled by deploying multiple instances behind a load balancer. Database performance is enhanced through indexing of frequently queried fields and the use of read replicas to handle high-volume dashboard operations. Additionally, caching mechanisms such as Redis are applied for frequently accessed

data, and rate limiting is enforced on resource-intensive endpoints such as file uploads to prevent overload. These strategies ensure that the system can effectively handle growth beyond 1000 students per year without compromising performance.

User Manual

Primary Use Case: Uploading Student Admission Documents (Admin/Staff Role)

steps:

1. Access the System
   Navigate to: http://localhost:5173

2. Login
   Enter username & password. Authentication uses session-based RBAC.

3. Navigate to Enrollment
   Go to *Student Data Management* → *Admissions/Enrollment* → *Upload Documents*.
   Corresponds to API route: criteria4/student/admissions/upload-docs.

4. Enter Student Identifier
   Input the student's Enrollment Number (e.g., 12345).

5. Upload Files
   Select
   Registration Form (reg.pdf)
   10th Marksheet (10th.pdf)
   12th Marksheet (12th.pdf)
   Entrance Exam Marksheet (gujcet.pdf)

6. Submit: Click *Upload/Submit*. System validates file type and securely stores in Supabase Storage.

7. Confirmation: Success message: *"Documents uploaded successfully"* (HTTP 200).

Code Documentation

The codebase is located in the **MADMS/ directory** and adheres to a **three-tier design** for clarity, extensibility, and debugging.

Backend (Flask) Organization

controllers/ → Route handlers & business logic (e.g., enrollment_controller.py, faculty_controller.py).
models/ → Database schemas via SQLAlchemy ORM.
database/ → Database initialization & connections.
services/ → Reusable utilities (logging, external APIs, data processing).

Frontend (React/TypeScript) Organization

components/ → Reusable UI (forms, dashboards, buttons).
pages/ → Main application views (Login, Enrollment, Dashboard).
services/ → Handles async REST API communication.

. Key Modules and Dependencies:

Component	Dependency/Tool	Role
Backend	Python	Primary programming language
Backend	Flask	Primary web framework
Backend	SQLAlchemy ORM	Database abstraction layer
Testing	`pytest`	Test runner and framework
Frontend	Node.js	Runtime environment
Frontend	React, TypeScript	Framework and language
Infrastructure	Supabase	Cloud Database/Storage/Auth