CSMIT FRONTEND

Comprehensive System Analysis & Requirement Documentation

Tech Stack: Angular 16+ **Environment:** Development/Staging **Status:** Active Analysis

1. Project Infrastructure

CSMIT is a modular frontend built using Angular 16. It is designed to act as a centralized management system for educational institutes, streamlining workflows for students, trainers, and administrative staff.

Development Tools

Node.js, npm, Angular CLI, RxJS

Core Scripts

ng serve, ng build, ng test

External Libs

SweetAlert2, html2pdf.js, FontAwesome

2. Functional Module Breakdown

A. Administrative ERP Complex

The core control hub for institute operations. Handles high-level data management.

- ✓ **User Management:** Multi-role (Admin/Trainer/Student) CRUD operations.
- ✓ **Academic Engine:** Creation and scheduling of Courses and Batches.
- ✓ **Content Management:** CMS for Success Stories, Blogs, and Career postings.
- ✓ **Analytics:** Reporting interface for institutional oversight.

B. Student Learning Portal Highly Interactive

Designed for student engagement and professional development.

- ✓ **Exam System:** Specialized timed assessment UI with focus-guard logic.
- ✓ **Career Tools:** Dynamic ATS Resume Generator using client-side PDF rendering.
- ✓ **Micro-Learning:** "Tech Shorts" vertical video feed for quick skill updates.

✓ **Course Management:** Enrolled batch tracking and performance analytics.

C. Trainer Intelligence Suite Operational

Equips trainers with tools to manage classroom efficiency.

✓ **Dashboard:** Real-time batch strength and timing countdowns.
✓ **Student Ops:** Attendance tracking and assignment management.
✓ **HR Workflow:** Self-service leave application and schedule calendar.

3. Technical Risk Scan (Critical Feedback)

Note for Developers: The following architectural issues were identified during analysis and must be addressed immediately.

[HIGH] **AppModule Misconfiguration:** Components are incorrectly placed in `imports` instead of `declarations`. This will cause compilation failure.

[HIGH] **Hard-coded API URLs:** Localhost endpoints (e.g., port 8000) found in components. Must move to `environments.ts`.

[MED] **Type Strictness:** Excessive use of `any` in file handlers and API responses weakens TypeScript's benefits.

[MED] **Debug Logs:** console.log instances left in production-bound files (e.g., profile-setting).

4. Future Roadmap & Improvements

✓ **Centralized State:** Implementation of NgRx or Akita as the application complexity grows.
✓ **Automated CI/CD:** Integration of GitHub Actions for automated building, linting, and testing.
✓ **API Standardization:** Moving all HTTP calls to a unified Interceptor-based system.
✓ **Enhanced E2E:** Adding Cypress/Playwright tests for critical paths (Exam flow, Resume generation).

5. Design & UX Standards

The platform follows a **Modern Professional** aesthetic. Key requirements for UI/UX designers include:

- **Responsiveness:** All dashboards must adapt seamlessly from 4K desktops to mobile devices.
- **Debouncing:** Implement 300ms delays on all search inputs to save server resources.

- **Accessibility:** High contrast ratios and aria-labels for the timed exam portal.

MADE BY CSMIT TEAM