Week 2 Deep Dive: Data Modeling & Architecture

Project: Techmentee 702 - College Placement Management Portal

Phase Focus: Database Schema, Entity Relationships, and State Management

1. Core Database Schema (Entity-Relationship Design)

To support the workflows and future analytics, we need a normalized relational database (like PostgreSQL or MySQL). Here is the exact table structure, including Primary Keys (PK) and Foreign Keys (FK).

A. User Management & Authentication

Table Name	Column	Data Type	Constraint/Note
Users	User_ID	UUID	**PK**
	Email	VARCHAR	Unique
	Password_Hash	VARCHAR	Encrypted
	Role	ENUM	('Student', 'Recruiter', 'Admin')
	Is_Active	BOOLEAN	Default: False (pending verification)

B. Profile Entities

Table Name	Column	Data Type	Constraint/Note
Students	Student_ID	UUID	**PK, FK** (References Users.User_ID)
	College_ID	VARCHAR	Unique
	Department	VARCHAR	e.g., 'Computer Science'
	CGPA	DECIMAL(3,2)	e.g., 8.75
	Resume_URL	VARCHAR	Cloud Storage Link (S3/Azure)
Recruiters	Recruiter_ID	UUID	**PK, FK** (References Users.User_ID)
	Company_Name	VARCHAR	
	Industry	VARCHAR	e.g., 'Fintech', 'IT Services'
	Is_Verified	BOOLEAN	Managed by Admin

C. Transactional Entities (The Core Business Logic)

Table Name	Column	Data Type	Constraint/Note
Jobs	Job_ID	UUID	**PK**
	Recruiter_ID	UUID	**FK** (References Recruiters.Recruiter_ID)
	Role_Title	VARCHAR	
	Min_CGPA	DECIMAL(3,2)	Eligibility criteria
	Salary_LPA	DECIMAL(5,2)	Standardized to numerical (e.g., 6.50)
	Status	ENUM	('Pending_Admin', 'Published', 'Closed')
Applications	App_ID	UUID	**PK**
	Job_ID	UUID	**FK** (References Jobs.Job_ID)
	Student_ID	UUID	**FK** (References Students.Student_ID)

	Applied_At	TIMESTAMP	Auto-generated
	Status	ENUM	See State Machine below

2. The Application State Machine

To track where a student is in the hiring process (which feeds our "Pipeline Drop-off" reports in Module 8), the Applications.Status column must follow a strict, linear progression.

- **State 1: Applied** (Triggered when a student clicks apply)
- **State 2: Shortlisted** (Triggered by Recruiter filtering)
- **State 3: Interview_Scheduled** (Triggered when Recruiter sets a date)
- **State 4: Interview_Completed** (Triggered post-interview)
- **State 5: Offered OR Rejected** (Terminal states)

Data Rule: Every time this state changes, a trigger should log the App_ID, Old_Status, New_Status, and Timestamp into a separate Application_History table. This is how we calculate "Time-to-Hire" metrics.

3. Data Flow Architecture (How Systems Talk)

- **Frontend (React/Angular):** Students and Recruiters interact with the UI.
- **Backend API (Node.js/Python):** Validates the requests (e.g., *Does this student meet the Min_CGPA for this job?*).
- **Database (PostgreSQL):** Stores the relational data securely.
- **File Storage (AWS S3):** Holds the heavy assets (Company Logos, Student Resumes, Offer Letters) and returns a simple URL to store in the database.

Lead Analyst Note:

By setting up the schema this cleanly in Week 2, our Week 3 tasks (building the analytical queries for Placement %, Highest Package, etc.) become incredibly straightforward. We won't have to clean up messy data later because we are enforcing strict data types and constraints right at the point of entry.