Digital Egypt Pioneers Initiative

React Front-End Development

ACE – Access to Careers & Education

ACE (Access to Careers & Education) is the **world's first platform** of its kind that seamlessly integrates education and career development into one unified digital system. The name **ACE** introduces an **innovative** and **pioneering** concept that **appears for the first time in this field**.
Our vision is to redefine the connection between learning and employment, transforming them into a single, continuous experience where university students are closer to their future careers, companies are closer to the qualified talents they seek, and instructors are closer to their accomplishments.

Edvance

Supervised by:

Eng. Basma Abdel Halim

Team members:

Eng. Abdalla Sobhy

Eng. Salsabeel Shehata

Eng. Malak Magdy

Eng. Mohammed Mahmoud

Eng. Ziad Mahmoud

Digital Egypt Pioneers Initiative

Integrated Education & Career Platform

Acknowledgement

We extend our heartfelt appreciation to Eng. Basma Abdel Halim for her invaluable guidance and unwavering support throughout our journey in the Digital Egypt Pioneers Initiative. Her mentorship has been instrumental in shaping our experience, providing us with the knowledge, insights, and encouragement needed to excel.

Her technical expertise, constructive feedback, and commitment to fostering innovation have played a crucial role in the development and success of this initiative. She has consistently provided thoughtful advice and practical solutions, helping us overcome challenges and refine our approach.

We are truly grateful for the opportunities to learn, grow, and collaborate under her insightful leadership. Her dedication and passion have left a lasting impact on our journey, and we sincerely appreciate her efforts in guiding and inspiring us every step of the way.

Abstract

ACE (Access to Careers & Education) is a groundbreaking platform designed to revolutionize the connection between education and employment. It uniquely integrates a Learning Management System (LMS) with a job platform, creating a unified digital ecosystem that empowers learners, facilitates career development, and enables companies to discover pre-qualified talent efficiently. The platform addresses the disconnect between academic learning and real-world job opportunities by providing personalized career paths, fostering collaboration between educational institutions and companies, and offering an interactive learning environment powered by AI and analytics. ACE aims to serve university students, general learners, teachers, parents, and companies, streamlining processes from course enrollment and skill development to job application and recruitment, thereby establishing a new global model for educational and employment efficiency.

Table of Contents

7. Final Presentation & Reports

Project Planning & Management

1.1. Project Proposal

Overview:

ACE (Access to Careers & Education) is the first platform of its kind in the world that combines a Learning Management System (LMS) with Job Platforms into one smart and efficient system. It serves as the missing bridge between "education" and "employment", transforming the academic journey into a complete professional experience.

Objectives:

Integrate education and the labor market into one connected experience.

Personalize the career path for each user based on their skills and educational level.

Enable universities and companies to collaborate in practically training students.

Provide an interactive learning environment built on AI and analytics.

Establish a new global model that unites education and employment in a single digital ecosystem.

Scope:

The system will cater to various user types, providing distinct interfaces and functionalities:

University Student Portal: Access courses, build professional profiles, apply for jobs/internships.

Company Portal: Post job opportunities, search for talent, manage recruitment pipelines.

Learner Portal (K-12): Access courses, track progress, earn rewards.

Teacher Dashboard: Create and manage courses, track student performance, monitor revenue.

Parent Dashboard: Monitor child's academic and career progress.

1.2. Project Plan

Timeline:

Phase 1: Analysis & Planning (Week 1): Market research and user needs analysis.

Phase 2: Core Development (Weeks 1–6): Building the backend system and main interfaces.

Phase 3: System Integration (Weeks 6–8): Integrating the LMS with the job system.

Phase 4: Pilot Launch (Weeks 9–12): Testing in universities.

Phase 5: Public Launch & Expansion (Weeks 13–20): Local then regional launch.

Deliverables:

A fully functional ACE platform integrating LMS and job board features.

Personalized dashboards for all user types.

Intelligent recommendation engines for jobs and courses.

Tools for collaboration between educational institutions and companies.

Detailed documentation and presentation covering technical and user aspects.

1.3. Task Assignment & Roles

Responsibilities for team members are defined across development, design, and management roles, ensuring efficient project execution. Specific assignments leverage individual expertise in frontend, backend, UI/UX, and project leadership.

Note: Detailed task assignments are managed internally and not detailed here.

1.4. Risk Assessment & Mitigation Plan

Potential risks include integration challenges between LMS and job modules, user adoption rates, data security, and scalability. Mitigation strategies involve thorough testing, phased rollouts, robust security protocols, and a flexible, scalable architecture.

1.5. KPIs (Key Performance Indicators)

User Engagement Rate (%): Percentage of active users across all roles. Target: High engagement across student, teacher, and company segments.

Job/Internship Placement Rate (%): Percentage of university students securing positions through the platform. Target: Significant year-on-year increase.

Course Completion Rate (%): Percentage of enrolled students successfully completing courses. Target: Above industry average.

System Uptime (%): Ensuring platform availability. Target: 99.9% uptime.

User Satisfaction Score (1-5): Based on feedback and reviews. Target: Consistently high satisfaction.

Successful Integrations: Smooth operation of integrated services (e.g., payment gateways, verification tools). Target: Zero critical failures.

Literature Review

2.1. Feedback & Evaluation

User feedback from pilot programs and early adopters is crucial for iterative improvement. This includes surveys, direct interviews, and in-app feedback mechanisms.

2.2. Suggested Improvements

Based on initial insights, potential improvements include enhancing AI-driven recommendation accuracy, expanding course offerings, fostering a stronger community interaction, and refining the administrative tools for institutions.

2.3. Final Grading Criteria

The project's success will be evaluated based on its ability to meet strategic objectives, user adoption, successful integration of core features, and overall impact on bridging the education-employment gap.

Requirements Gathering

3.1 Stakeholder Analysis:

University Students: Need a platform to build professional profiles, gain practical skills, and find job/internship opportunities.

Companies/Employers: Seek qualified young talent and efficient recruitment tools.

Learners (K-12): Require engaging educational content and early skill development.

Teachers/Trainers: Need tools to create, manage, and monetize courses and track student progress.

Parents: Want to monitor their children's academic and career development.

Admins: Oversee system operations, manage content, and analyze platform data.

3.2 User Stories & Use Cases:

As a university student, I want to find relevant internships based on my skills and major so I can gain practical experience.

As a company, I want to post job openings and filter candidates by specific skills to find the best fit.

As a teacher, I want to create and publish online courses, track student enrollment, and manage payments.

As a parent, I want to view my child's course progress and achievements to stay informed.

3.3 Functional Requirements:

User Authentication & Authorization (role-based).

Profile Management (Student, Company, Teacher).

Course Creation, Management, and Enrollment (LMS features).

Job Posting, Application, and Tracking System.

Recommendation Engines (Job & Course).

Payment Gateway Integration (for courses).

Notification System.

Reporting & Analytics Dashboard.

3.4. Non-functional Requirements:

Performance: Fast load times and responsive UI across devices.

Security: Robust data protection, secure authentication (JWT, OTP, AI Verification), and privacy compliance.

Scalability: Ability to handle a growing user base and increasing data volume.

Usability: Intuitive and user-friendly interface, with support for multiple themes and languages.

Reliability: High system uptime and data integrity.

System Analysis & Design

4.1. Problem Statement & Objectives:

Problem Statement:

The significant disconnect between traditional education systems and the demands of the job market leads to graduates lacking essential practical skills and companies struggling to find qualified talent. This results in a gap between academic learning and professional readiness, hindering career progression and economic growth.

Objectives:

ACE aims to bridge this gap by providing an integrated platform that:

Connects learning pathways directly to career opportunities.

Empowers students with relevant skills and market insights.

Enables efficient talent discovery for employers.

Facilitates collaboration between educational bodies and industries.

4.2. Database Design & Data Modeling:

The database is designed using a normalized approach (3NF) with MySQL as the RDBMS, optimized for scalability and performance. Key entities include Users (with distinct types: Company, University Student, Teacher, Parent), Job Postings, Job Applications, Courses, Course Lessons, Course Enrollments, and Payments. JSON columns are used for flexible attributes like skills and experience. Referential integrity is maintained via foreign keys, and strategic indexing supports efficient querying for job searches, application management, and course discovery. Soft deletes are implemented for data recovery and audit trails.

And here is the Database Design in detail, and every single attribute in the database.

```
# ACE Platform - Database Design Document

## Table of Contents
1. [Executive Summary](#executive-summary)
2. [Database Overview](#database-overview)
3. [Entity Relationship Diagram](#entity-relationship-diagram)
4. [Detailed Schema Design](#detailed-schema-design)
5. [Relationships & Cardinalities](#relationships--cardinalities)
6. [Indexes & Performance Optimization](#indexes--performance-optimization)
7. [Data Integrity & Constraints](#data-integrity--constraints)
8. [Security Considerations](#security-considerations)
9. [Scalability & Future Enhancements](#scalability--future-enhancements)

---

## Executive Summary

The ACE (Adaptive Career Education) platform is a comprehensive system that
bridges education and career development. The database design supports two
primary domains:

1. **Educational Management System (LMS)** - Course creation, enrollment, and
learning management
2. **Job Recruitment Platform** - Connecting university students with career
opportunities

This document presents a normalized, scalable database design optimized for both
transactional operations and analytical queries.

---

## Database Overview

### Technology Stack
- **RDBMS**: MySQL 8.0+ / PostgreSQL 13+
- **ORM**: Laravel Eloquent
- **Key Features**:
  - JSON column support for complex data structures
  - Full-text search capabilities
  - Transaction support for data integrity
  - Soft deletes for audit trails
```

~~~
### Design Principles
1. **Normalization**: 3rd Normal Form (3NF) with selective denormalization for
performance
2. **Flexibility**: JSON columns for evolving data structures (skills,
experience, etc.)
3. **Integrity**: Foreign key constraints with appropriate cascade rules
4. **Performance**: Strategic indexing based on query patterns
5. **Audit Trail**: Timestamps and soft deletes on all major tables

---

## Entity Relationship Diagram

```
┌─────────────────┐
│     USERS       │
│─────────────────│
│ PK id           │
│    first_name   │
│    last_name    │
│    email (UQ)   │
│    password     │
│    user_type    │◄───────────────────────────┐
│    is_approved  │                            │
│    created_at   │                            │
│    updated_at   │                            │
│    deleted_at   │                            │
└─────────────────┘                            │
         │                                     │
         │                                     │
         │ 1:1                                 │
         ├──────────────────┬──────────────────┼──────────────────┐
         │                  │                  │                  │
         ▼                  ▼                  ▼                  ▼
┌────────────────┐  ┌──────────────────┐   ┌─────────────┐  ┌──────────┐
│   COMPANIES    │  │ UNIV_STUDENT_    │   │  TEACHERS   │  │ PARENTS  │
│────────────────│  │    PROFILES      │   │─────────────│  │──────────│
│ PK id          │  │──────────────────│   │ PK id       │  │ PK id    │
│ FK user_id (UQ)│  │ PK id            │   │ FK user_id  │  │ FK user_id│
│    company_name│  │ FK user_id (UQ)  │   │    bio      │  │    phone │
│    industry    │  │    faculty       │   │    expertise│  │ created_ │
│    company_size│  │    university    │   │ created_at  │  │ updated_ │
│    location    │  │    year_of_study │   │ updated_at  │  │ deleted_ │
│    website     │  │    gpa           │   │ deleted_at  │  └──────────┘
│    logo_path   │  │    bio           │   └─────────────┘
│    is_verified │  │    cv_path       │
~~~

```
│    linkedin_url│  │    cv_filename    │
│    benefits[]  │  │    skills[] (JSON)│
│    registration│  │    languages[]    │
│    created_at  │  │    experience[]   │
│    updated_at  │  │    projects[]     │
│    deleted_at  │  │    certifications│
└────────┬───────┘  │    linkedin_url   │
         │          │    github_url     │
         │ 1:N      │    portfolio_url  │
         │          │    is_public      │
         ▼          │    looking_for_   │
┌────────────────┐  │       opportunities│
│ JOB_POSTINGS   │  │    preferred_job_│
│────────────────│  │       types[]     │
│ PK id          │  │    profile_views  │
│ FK company_id  │  │    cv_downloads   │
│    title       │  │    created_at     │
│    description │  │    updated_at     │
│    requirements│  │    deleted_at     │
│    responsibil │  └────────┬──────────┘
│    skills_req[]│           │
│    skills_pref │           │ 1:N
│    job_type    │           │
│    work_location│          │
│    salary_min  │           │
│    salary_max  │◄──────────┼─────────┐
│    experience_ │           │         │
│       level    │           │         │
│    positions_  │           │         │
│       available│           │         │
│    application_│           │         │
│       deadline │           │         │
│    faculties_  │           │         │
│       preferred│           │         │
│    education_  │           │         │
│       requirement│         │         │
│    is_active   │           │         │
│    views_count │           │         │
│    applications│           │         │
│       _count   │           │         │
│    created_at  │           │         │
│    updated_at  │           │         │
│    deleted_at  │           │         │
└────────┬───────┘           │         │
         │                   │         │
         │ 1:N                │         │
         │                    │         │
         ▼                    ▼         │
┌────────────────────────────────────┐  │
│      JOB_APPLICATIONS              │  │
├────────────────────────────────────┤  │
│ PK id                              │  │
│ FK job_posting_id                  │  │
│ FK student_id (→ univ_student_     │  │
│                   profiles)        │  │
│    cover_letter                    │  │
│    status (ENUM)                   │  │
│    status_history[] (JSON)         │  │
│    interview_date                  │  │
│    interview_location              │  │
│    interview_notes                 │  │
│    company_notes                   │  │
│    viewed_at                       │  │
│    is_favorite                     │  │
│    created_at                      │  │
│    updated_at                      │  │
│    deleted_at                      │  │
│ UQ (job_posting_id, student_id)    │  │
└────────────────────────────────────┘  │
                                        │
         ┌──────────────────────────────┘
         │
         │ 1:N
         ▼
┌─────────────────┐
│    COURSES      │
├─────────────────┤
│ PK id           │
│ FK teacher_id   │
│    title        │
│    description  │
│    category     │
│    level        │
│    price        │
│    duration     │
│    thumbnail    │
│    is_published │
│    max_students │
│    created_at   │
│    updated_at   │
│    deleted_at   │
└────────┬────────┘
         │
         │ 1:N
         ├────────────────┬─────────────────┐
         │                │                 │
         ▼                ▼                 ▼
┌────────────────┐  ┌───────────────┐  ┌──────────────┐
│ COURSE_LESSONS │  │COURSE_        │  │LIVE_SESSIONS │
├────────────────┤  │ ENROLLMENTS   │  ├──────────────┤
│ PK id          │  ├───────────────┤  │ PK id        │
│ FK course_id   │  │ PK id         │  │ FK course_id │
│    title       │  │ FK course_id  │  │    title     │
│    content     │  │ FK student_id │  │    scheduled_│
│    order       │  │    status     │  │       at     │
│    duration    │  │    progress   │  │    duration  │
│    video_url   │  │    enrolled_at│  │    meeting_id│
│    is_free     │  │    completed_ │  │    status    │
│    created_at  │  │       at      │  │    created_at│
│    updated_at  │  │ created_at    │  │    updated_at│
│    deleted_at  │  │ updated_at    │  │    deleted_at│
└────────────────┘  │ deleted_at    │  └──────────────┘
                    └───────┬───────┘
                            │
                            │ 1:N
                            ▼
                    ┌───────────────┐
                    │   PAYMENTS    │
                    ├───────────────┤
                    │ PK id         │
                    │ FK enrollment_│
                    │       id      │
                    │    amount     │
                    │    currency   │
                    │    payment_   │
                    │       method  │
                    │    transaction│
                    │       _id     │
                    │    status     │
                    │    paid_at    │
                    │ created_at    │
                    │ updated_at    │
                    └───────────────┘

┌──────────────────┐
```

```
│  NOTIFICATIONS   │
├──────────────────┤
│ PK id            │
│    type          │
│    notifiable_   │
│       type       │
│    notifiable_id │
│    data (JSON)   │
│    read_at       │
│    created_at    │
│    updated_at    │
└──────────────────┘
```

## Detailed Schema Design

### 1. Users Table

Purpose: Central authentication and user management

Column	Type	Constraints	Description
id	BIGINT UNSIGNED	PK, AUTO_INCREMENT	Unique identifier
first_name	VARCHAR(100)	NOT NULL	User's first name
last_name	VARCHAR(100)	NOT NULL	User's last name
email	VARCHAR(255)	UNIQUE, NOT NULL	Email address (login credential)
phone	VARCHAR(20)	NULLABLE	Contact phone number
password	VARCHAR(255)	NOT NULL	Hashed password (bcrypt)
user_type	ENUM	NOT NULL	'company', 'university_student', 'teacher', 'parent'
is_approved	BOOLEAN	DEFAULT FALSE	Admin approval status
email_verified_at	TIMESTAMP	NULLABLE	Email verification timestamp
remember_token	VARCHAR(100)	NULLABLE	Session persistence token
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time
updated_at	TIMESTAMP	ON UPDATE	Last modification time
deleted_at	TIMESTAMP	NULLABLE	Soft delete timestamp

Indexes:
- PRIMARY KEY (id)
- UNIQUE INDEX (email)
- INDEX (user_type)
- INDEX (is_approved)

```
---

### 2. Companies Table

**Purpose**: Company profile information for employers

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| user_id | BIGINT UNSIGNED | FK (users.id), UNIQUE, NOT NULL | Reference to user account |
| company_name | VARCHAR(255) | NOT NULL | Official company name |
| industry | VARCHAR(100) | NULLABLE | Industry sector |
| company_size | VARCHAR(50) | NULLABLE | Employee count range |
| location | VARCHAR(255) | NULLABLE | Company headquarters/location |
| website | VARCHAR(255) | NULLABLE | Company website URL |
| logo_path | VARCHAR(255) | NULLABLE | Path to company logo |
| linkedin_url | VARCHAR(255) | NULLABLE | LinkedIn company page |
| benefits | JSON | NULLABLE | Array of employee benefits |
| registration_number | VARCHAR(100) | NULLABLE | Official registration number |
| is_verified | BOOLEAN | DEFAULT FALSE | Verification status by admin |
| about | TEXT | NULLABLE | Company description |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Indexes**:
- PRIMARY KEY (id)
- UNIQUE INDEX (user_id)
- INDEX (is_verified)
- INDEX (industry)
- FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE

---

### 3. University Student Profiles Table

**Purpose**: Detailed student profiles for job matching

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| user_id | BIGINT UNSIGNED | FK (users.id), UNIQUE, NOT NULL | Reference to user account |
| faculty | VARCHAR(255) | NULLABLE | Faculty/department name |
```

~~~
| university | VARCHAR(255) | NULLABLE | University name |
| year_of_study | INTEGER | NULLABLE | Current academic year (1-5) |
| gpa | DECIMAL(3,2) | NULLABLE | Grade point average (0.00-4.00) |
| bio | TEXT | NULLABLE | Student bio/summary |
| cv_path | VARCHAR(255) | NULLABLE | Path to CV file |
| cv_filename | VARCHAR(255) | NULLABLE | Original CV filename |
| skills | JSON | NULLABLE | Array of skills |
| languages | JSON | NULLABLE | Array of language proficiencies |
| experience | JSON | NULLABLE | Array of work experience objects |
| projects | JSON | NULLABLE | Array of project objects |
| certifications | JSON | NULLABLE | Array of certification objects |
| linkedin_url | VARCHAR(255) | NULLABLE | LinkedIn profile URL |
| github_url | VARCHAR(255) | NULLABLE | GitHub profile URL |
| portfolio_url | VARCHAR(255) | NULLABLE | Personal portfolio URL |
| is_public | BOOLEAN | DEFAULT TRUE | Profile visibility to companies |
| looking_for_opportunities | BOOLEAN | DEFAULT TRUE | Open to job opportunities |
| preferred_job_types | JSON | NULLABLE | Array of preferred job types |
| profile_views | INTEGER UNSIGNED | DEFAULT 0 | Number of profile views |
| cv_downloads | INTEGER UNSIGNED | DEFAULT 0 | Number of CV downloads |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**JSON Structure Examples**:

```json
// skills
["JavaScript", "Python", "React", "Machine Learning"]

// languages
[
  {"language": "Arabic", "proficiency": "Native"},
  {"language": "English", "proficiency": "Fluent"}
]

// experience
[
  {
    "company": "Tech Corp",
    "position": "Software Intern",
    "start_date": "2023-06",
    "end_date": "2023-09",
    "description": "Developed web applications..."
  }
~~~

~~~
]

// projects
[
  {
    "title": "E-commerce Platform",
    "description": "Built a full-stack e-commerce...",
    "technologies": ["React", "Node.js", "MongoDB"],
    "url": "https://github.com/user/project"
  }
]

// certifications
[
  {
    "name": "AWS Certified Developer",
    "issuer": "Amazon",
    "issue_date": "2024-01",
    "credential_id": "ABC123"
  }
]

// preferred_job_types
["full_time", "internship", "remote"]
```

**Indexes**:
- PRIMARY KEY (id)
- UNIQUE INDEX (user_id)
- INDEX (is_public, looking_for_opportunities)
- INDEX (university, faculty)
- FULLTEXT INDEX (skills) - For skill-based search
- FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE

---

### 4. Job Postings Table

**Purpose**: Job listings created by companies

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| company_id | BIGINT UNSIGNED | FK (companies.id), NOT NULL | Reference to
company |
~~~

```
| title | VARCHAR(255) | NOT NULL | Job title |
| description | TEXT | NOT NULL | Detailed job description |
| requirements | TEXT | NULLABLE | Job requirements |
| responsibilities | TEXT | NULLABLE | Job responsibilities |
| skills_required | JSON | NULLABLE | Array of required skills |
| skills_preferred | JSON | NULLABLE | Array of preferred skills |
| job_type | ENUM | NOT NULL | 'full_time', 'part_time', 'internship',
'contract', 'remote' |
| work_location | VARCHAR(255) | NULLABLE | Job location |
| salary_min | DECIMAL(10,2) | NULLABLE | Minimum salary |
| salary_max | DECIMAL(10,2) | NULLABLE | Maximum salary |
| currency | VARCHAR(3) | DEFAULT 'USD' | Salary currency code |
| experience_level | ENUM | NULLABLE | 'entry', 'mid', 'senior', 'lead' |
| positions_available | INTEGER UNSIGNED | DEFAULT 1 | Number of openings |
| application_deadline | DATE | NULLABLE | Application deadline |
| faculties_preferred | JSON | NULLABLE | Array of preferred faculties |
| education_requirement | VARCHAR(255) | NULLABLE | Education level requirement |
| is_active | BOOLEAN | DEFAULT TRUE | Job posting status |
| views_count | INTEGER UNSIGNED | DEFAULT 0 | Number of views |
| applications_count | INTEGER UNSIGNED | DEFAULT 0 | Number of applications |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Indexes**:
- PRIMARY KEY (id)
- INDEX (company_id)
- INDEX (is_active, application_deadline)
- INDEX (job_type)
- INDEX (experience_level)
- INDEX (created_at)
- FULLTEXT INDEX (title, description)
- FOREIGN KEY (company_id) REFERENCES companies(id) ON DELETE CASCADE

---

### 5. Job Applications Table

**Purpose**: Track student applications to job postings

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| job_posting_id | BIGINT UNSIGNED | FK (job_postings.id), NOT NULL | Reference
to job posting |
```

~~~
| student_id | BIGINT UNSIGNED | FK (university_student_profiles.id), NOT NULL |
Reference to student |
| cover_letter | TEXT | NOT NULL | Application cover letter |
| status | ENUM | NOT NULL, DEFAULT 'pending' | 'pending', 'reviewing',
'shortlisted', 'interviewed', 'accepted', 'rejected', 'withdrawn' |
| status_history | JSON | NULLABLE | Array of status change events |
| interview_date | DATETIME | NULLABLE | Scheduled interview date/time |
| interview_location | VARCHAR(255) | NULLABLE | Interview location/link |
| interview_notes | TEXT | NULLABLE | Interview notes from company |
| company_notes | TEXT | NULLABLE | Internal company notes |
| viewed_at | TIMESTAMP | NULLABLE | When company first viewed |
| is_favorite | BOOLEAN | DEFAULT FALSE | Favorited by company |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Application submission
time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Status History JSON Structure**:
```json
[
  {
    "status": "pending",
    "changed_at": "2024-01-15T10:30:00Z",
    "notes": "Application submitted"
  },
  {
    "status": "reviewing",
    "changed_at": "2024-01-16T14:20:00Z",
    "notes": "Under review by hiring team"
  },
  {
    "status": "interviewed",
    "changed_at": "2024-01-20T09:00:00Z",
    "notes": "Interview completed"
  }
]
```

**Constraints**:
- UNIQUE (job_posting_id, student_id) - Prevent duplicate applications
- CHECK (LENGTH(cover_letter) >= 50) - Minimum cover letter length

**Indexes**:
- PRIMARY KEY (id)
- UNIQUE INDEX (job_posting_id, student_id)
~~~

- INDEX (student_id, status)
- INDEX (job_posting_id, status)
- INDEX (status, created_at)
- INDEX (is_favorite)
- FOREIGN KEY (job_posting_id) REFERENCES job_postings(id) ON DELETE CASCADE
- FOREIGN KEY (student_id) REFERENCES university_student_profiles(id) ON DELETE CASCADE

### 6. Courses Table

Purpose: Educational courses in the LMS

Column	Type	Constraints	Description
id	BIGINT UNSIGNED	PK, AUTO_INCREMENT	Unique identifier
teacher_id	BIGINT UNSIGNED	FK (users.id), NOT NULL	Reference to teacher
title	VARCHAR(255)	NOT NULL	Course title
description	TEXT	NULLABLE	Course description
category	VARCHAR(100)	NULLABLE	Course category
level	ENUM	NULLABLE	'beginner', 'intermediate', 'advanced'
price	DECIMAL(10,2)	DEFAULT 0.00	Course price
currency	VARCHAR(3)	DEFAULT 'USD'	Price currency
duration_hours	INTEGER	NULLABLE	Estimated duration in hours
thumbnail_path	VARCHAR(255)	NULLABLE	Course thumbnail image
is_published	BOOLEAN	DEFAULT FALSE	Publication status
max_students	INTEGER UNSIGNED	NULLABLE	Maximum enrollment limit
enrollment_count	INTEGER UNSIGNED	DEFAULT 0	Current enrollment count
rating_average	DECIMAL(3,2)	DEFAULT 0.00	Average rating (0.00-5.00)
rating_count	INTEGER UNSIGNED	DEFAULT 0	Number of ratings
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time
updated_at	TIMESTAMP	ON UPDATE	Last modification time
deleted_at	TIMESTAMP	NULLABLE	Soft delete timestamp

Indexes:
- PRIMARY KEY (id)
- INDEX (teacher_id)
- INDEX (is_published, category)
- INDEX (rating_average)
- FOREIGN KEY (teacher_id) REFERENCES users(id) ON DELETE CASCADE

### 7. Course Lessons Table

```markdown
**Purpose**: Individual lessons within courses

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| course_id | BIGINT UNSIGNED | FK (courses.id), NOT NULL | Reference to course |
| title | VARCHAR(255) | NOT NULL | Lesson title |
| content | TEXT | NULLABLE | Lesson content/description |
| order | INTEGER UNSIGNED | NOT NULL | Lesson order in course |
| duration_minutes | INTEGER UNSIGNED | NULLABLE | Lesson duration |
| video_url | VARCHAR(255) | NULLABLE | Video content URL |
| attachments | JSON | NULLABLE | Array of attachment objects |
| is_free | BOOLEAN | DEFAULT FALSE | Free preview lesson |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Indexes**:
- PRIMARY KEY (id)
- INDEX (course_id, order)
- FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE

---

### 8. Course Enrollments Table

**Purpose**: Track student course enrollments

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| course_id | BIGINT UNSIGNED | FK (courses.id), NOT NULL | Reference to course |
| student_id | BIGINT UNSIGNED | FK (users.id), NOT NULL | Reference to student |
| status | ENUM | NOT NULL, DEFAULT 'active' | 'active', 'completed', 'dropped' |
| progress_percentage | DECIMAL(5,2) | DEFAULT 0.00 | Course completion % (0.00-100.00) |
| enrolled_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Enrollment date |
| completed_at | TIMESTAMP | NULLABLE | Completion date |
| last_accessed_at | TIMESTAMP | NULLABLE | Last access timestamp |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Constraints**:
```

```
- UNIQUE (course_id, student_id) - One enrollment per student per course

**Indexes**:
- PRIMARY KEY (id)
- UNIQUE INDEX (course_id, student_id)
- INDEX (student_id, status)
- FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE
- FOREIGN KEY (student_id) REFERENCES users(id) ON DELETE CASCADE

---

### 9. Payments Table

**Purpose**: Track course payment transactions

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| enrollment_id | BIGINT UNSIGNED | FK (course_enrollments.id), NOT NULL | Reference to enrollment |
| amount | DECIMAL(10,2) | NOT NULL | Payment amount |
| currency | VARCHAR(3) | DEFAULT 'USD' | Payment currency |
| payment_method | VARCHAR(50) | NOT NULL | 'stripe', 'paypal', etc. |
| transaction_id | VARCHAR(255) | UNIQUE, NULLABLE | External transaction ID |
| status | ENUM | NOT NULL, DEFAULT 'pending' | 'pending', 'completed', 'failed', 'refunded' |
| paid_at | TIMESTAMP | NULLABLE | Payment completion time |
| metadata | JSON | NULLABLE | Additional payment metadata |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |

**Indexes**:
- PRIMARY KEY (id)
- UNIQUE INDEX (transaction_id)
- INDEX (enrollment_id)
- INDEX (status)
- FOREIGN KEY (enrollment_id) REFERENCES course_enrollments(id) ON DELETE CASCADE

---

### 10. Live Sessions Table

**Purpose**: Track live course sessions (via Agora SDK)

| Column | Type | Constraints | Description |
```

~~~markdown
|--------|------|-------------|-------------|
| id | BIGINT UNSIGNED | PK, AUTO_INCREMENT | Unique identifier |
| course_id | BIGINT UNSIGNED | FK (courses.id), NOT NULL | Reference to course |
| title | VARCHAR(255) | NOT NULL | Session title |
| scheduled_at | DATETIME | NOT NULL | Scheduled start time |
| duration_minutes | INTEGER UNSIGNED | DEFAULT 60 | Session duration |
| meeting_id | VARCHAR(255) | NULLABLE | Agora meeting/channel ID |
| recording_url | VARCHAR(255) | NULLABLE | Session recording URL |
| status | ENUM | NOT NULL, DEFAULT 'scheduled' | 'scheduled', 'live',
'completed', 'cancelled' |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |
| deleted_at | TIMESTAMP | NULLABLE | Soft delete timestamp |

**Indexes**:
- PRIMARY KEY (id)
- INDEX (course_id, scheduled_at)
- INDEX (status)
- FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE

---

### 11. Notifications Table

**Purpose**: System-wide notification management

| Column | Type | Constraints | Description |
|--------|------|-------------|-------------|
| id | CHAR(36) | PK | UUID identifier |
| type | VARCHAR(255) | NOT NULL | Notification class name |
| notifiable_type | VARCHAR(255) | NOT NULL | Polymorphic type (User model) |
| notifiable_id | BIGINT UNSIGNED | NOT NULL | Polymorphic ID (user ID) |
| data | JSON | NOT NULL | Notification payload |
| read_at | TIMESTAMP | NULLABLE | Read timestamp |
| created_at | TIMESTAMP | DEFAULT CURRENT_TIMESTAMP | Record creation time |
| updated_at | TIMESTAMP | ON UPDATE | Last modification time |

**Data JSON Structure Example**:
```json
{
  "type": "NewJobApplication",
  "title": "New Application Received",
  "message": "Ahmed Mohamed applied for Software Engineer position",
  "action_url": "/company/applications/123",
  "metadata": {
~~~

```
    "job_id": 45,
    "student_id": 78,
    "application_id": 123
  }
}
```

Indexes:
- PRIMARY KEY (id)
- INDEX (notifiable_type, notifiable_id)
- INDEX (read_at)
- INDEX (created_at)

### 12. Teachers Table

Purpose: Teacher profiles for course instructors

Column	Type	Constraints	Description
id	BIGINT UNSIGNED	PK, AUTO_INCREMENT	Unique identifier
user_id	BIGINT UNSIGNED	FK (users.id), UNIQUE, NOT NULL	Reference to user account
bio	TEXT	NULLABLE	Teacher biography
expertise	JSON	NULLABLE	Array of expertise areas
qualifications	JSON	NULLABLE	Array of qualifications
rating_average	DECIMAL(3,2)	DEFAULT 0.00	Average rating (0.00-5.00)
rating_count	INTEGER UNSIGNED	DEFAULT 0	Number of ratings
total_students	INTEGER UNSIGNED	DEFAULT 0	Total students taught
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time
updated_at	TIMESTAMP	ON UPDATE	Last modification time
deleted_at	TIMESTAMP	NULLABLE	Soft delete timestamp

Indexes:
- PRIMARY KEY (id)
- UNIQUE INDEX (user_id)
- INDEX (rating_average)
- FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE

### 13. Parents Table

Purpose: Parent accounts for monitoring student progress

Column	Type	Constraints	Description
id	BIGINT UNSIGNED	PK, AUTO_INCREMENT	Unique identifier
user_id	BIGINT UNSIGNED	FK (users.id), UNIQUE, NOT NULL	Reference to user account
phone	VARCHAR(20)	NULLABLE	Contact phone
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time
updated_at	TIMESTAMP	ON UPDATE	Last modification time
deleted_at	TIMESTAMP	NULLABLE	Soft delete timestamp

Note: Parent-student relationships may require a junction table if one parent can monitor multiple students.

Indexes:
- PRIMARY KEY (id)
- UNIQUE INDEX (user_id)
- FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE

## Relationships & Cardinalities

### Primary Relationships

Relationship	Cardinality	Description	Cascade Rules
User → Company	1:1	One user account per company	CASCADE DELETE
User → UniversityStudentProfile	1:1	One user account per student	CASCADE DELETE
User → Teacher	1:1	One user account per teacher	CASCADE DELETE
User → Parent	1:1	One user account per parent	CASCADE DELETE
Company → JobPosting	1:N	Company creates multiple jobs	CASCADE DELETE
JobPosting → JobApplication	1:N	Job receives multiple applications	CASCADE DELETE
UniversityStudentProfile → JobApplication	1:N	Student applies to multiple jobs	CASCADE DELETE
JobPosting + Student → JobApplication	M:N (via junction)	Many-to-many with attributes	UNIQUE constraint
Teacher → Course	1:N	Teacher creates multiple courses	CASCADE DELETE
Course → CourseLesson	1:N	Course contains multiple lessons	CASCADE DELETE
Course + Student → CourseEnrollment	M:N (via junction)	Many-to-many enrollment	UNIQUE constraint

```
| CourseEnrollment → Payment | 1:N | Enrollment may have multiple payments | CASCADE DELETE |
| Course → LiveSession | 1:N | Course has multiple live sessions | CASCADE DELETE |
| User → Notification | 1:N (polymorphic) | User receives multiple notifications | CASCADE DELETE |

### Business Rules

1. **Job Applications**:
   - A student can apply to a job only once (enforced by UNIQUE constraint)
   - Applications cannot be deleted, only soft-deleted or withdrawn
   - Once status is 'accepted' or 'rejected', no further changes except by admin

2. **Course Enrollments**:
   - A student can enroll in a course only once
   - Enrollment requires payment for paid courses
   - Enrollment must be active before accessing lessons

3. **User Types**:
   - Each user has exactly one role profile (company OR student OR teacher OR parent)
   - User type determines accessible features and endpoints

4. **Job Posting Lifecycle**:
   - Active jobs must have application_deadline in the future or NULL
   - Cannot delete job posting with existing applications (soft delete only)
   - Deactivating a job prevents new applications

---

## Indexes & Performance Optimization

### Index Strategy

#### 1. Primary Keys
All tables use `BIGINT UNSIGNED AUTO_INCREMENT` for primary keys to support large-scale growth.

#### 2. Foreign Keys
All foreign key columns are indexed automatically for join performance.

#### 3. Query-Specific Indexes

**Job Search & Filtering**:
```

```sql
-- Job postings
CREATE INDEX idx_active_deadline ON job_postings(is_active, application_deadline);
CREATE INDEX idx_job_type_exp ON job_postings(job_type, experience_level);
CREATE FULLTEXT INDEX idx_job_search ON job_postings(title, description);

-- Student profiles
CREATE INDEX idx_public_opportunities ON university_student_profiles(is_public, looking_for_opportunities);
CREATE FULLTEXT INDEX idx_skills_search ON university_student_profiles(skills);
```

Application Management:
```sql
-- For student's application list
CREATE INDEX idx_student_status ON job_applications(student_id, status);

-- For company's application review
CREATE INDEX idx_job_status ON job_applications(job_posting_id, status);

-- For favorites
CREATE INDEX idx_favorite ON job_applications(is_favorite) WHERE is_favorite = TRUE;
```

Course Discovery:
```sql
CREATE INDEX idx_published_category ON courses(is_published, category);
CREATE INDEX idx_rating ON courses(rating_average DESC);
```

Notification Queries:
```sql
CREATE INDEX idx_notifiable ON notifications(notifiable_type, notifiable_id);
CREATE INDEX idx_unread ON notifications(read_at) WHERE read_at IS NULL;
```

#### 4. Composite Indexes

Indexes are ordered with high-cardinality columns first for optimal selectivity:

```sql
-- Job applications unique constraint doubles as index
```

```sql
CREATE UNIQUE INDEX idx_unique_application ON job_applications(job_posting_id,
student_id);

-- Course enrollments
CREATE UNIQUE INDEX idx_unique_enrollment ON course_enrollments(course_id,
student_id);
```

### Denormalization for Performance

Strategic denormalization is used for frequently accessed aggregate data:

1. **Counters**: `views_count`, `applications_count`, `enrollment_count`, `profile_views`, `cv_downloads`
   - Updated via triggers or application logic
   - Avoids expensive COUNT(*) queries

2. **Cached Aggregates**: `rating_average`, `rating_count`
   - Recalculated periodically or on updates
   - Enables fast sorting and filtering

### Query Optimization Techniques

1. **Covering Indexes**: Include frequently selected columns in indexes
2. **Partial Indexes**: Index only relevant rows (e.g., active jobs, unread notifications)
3. **JSON Indexing**: Consider generated columns for frequently queried JSON fields
4. **Pagination**: Use cursor-based pagination for large result sets

---

## Data Integrity & Constraints

### Primary Constraints

#### 1. Entity Integrity
- All tables have PRIMARY KEY constraints
- No NULL values allowed in primary keys (enforced by AUTO_INCREMENT)

#### 2. Referential Integrity
All foreign keys enforce referential integrity with appropriate cascade rules:

```sql
-- Cascade DELETE: Remove dependent records
```

~~~markdown
FOREIGN KEY (company_id) REFERENCES companies(id) ON DELETE CASCADE

-- Cascade UPDATE: Update references (default behavior)
FOREIGN KEY (user_id) REFERENCES users(id) ON UPDATE CASCADE

-- RESTRICT: Prevent deletion if references exist (for critical data)
FOREIGN KEY (teacher_id) REFERENCES users(id) ON DELETE RESTRICT
```

#### 3. Domain Integrity

**ENUM Constraints**:
```sql
user_type ENUM('company', 'university_student', 'teacher', 'parent')
job_type ENUM('full_time', 'part_time', 'internship', 'contract', 'remote')
experience_level ENUM('entry', 'mid', 'senior', 'lead')
status ENUM('pending', 'reviewing', 'shortlisted', 'interviewed', 'accepted',
'rejected', 'withdrawn')
```

**CHECK Constraints**:
```sql
-- GPA must be between 0 and 4
CONSTRAINT chk_gpa CHECK (gpa >= 0 AND gpa <= 4)

-- Salary min must be less than max
CONSTRAINT chk_salary CHECK (salary_min <= salary_max)

-- Cover letter minimum length
CONSTRAINT chk_cover_letter CHECK (LENGTH(cover_letter) >= 50)

-- Progress percentage between 0-100
CONSTRAINT chk_progress CHECK (progress_percentage >= 0 AND progress_percentage
<= 100)

-- Rating between 0-5
CONSTRAINT chk_rating CHECK (rating_average >= 0 AND rating_average <= 5)
```

**NOT NULL Constraints**:
Critical fields that must always have values:
- User credentials: email, password
- Foreign keys: user_id, company_id, course_id, etc.
- Required business data: job_title, course_title, application_status
~~~

~~~markdown
#### 4. Business Logic Constraints

**Unique Constraints**:
```sql
-- Prevent duplicate user emails
UNIQUE (email)

-- One profile per user
UNIQUE (user_id) ON companies, university_student_profiles, teachers, parents

-- One application per student per job
UNIQUE (job_posting_id, student_id) ON job_applications

-- One enrollment per student per course
UNIQUE (course_id, student_id) ON course_enrollments
```

### Data Validation

#### Application-Level Validation
1. **Email Format**: RFC 5322 compliant
2. **Phone Numbers**: International format validation
3. **URLs**: Valid HTTP/HTTPS URLs for linkedin, github, portfolio
4. **Dates**: Application deadline must be in future
5. **File Uploads**: CV file type validation (PDF, DOC, DOCX), max 5MB
6. **JSON Schema**: Validate JSON structure for skills, experience, etc.

#### Database-Level Validation
1. **Triggers**: Enforce complex business rules
2. **Stored Procedures**: Encapsulate validation logic
3. **CHECK Constraints**: Enforce data ranges

### Soft Delete Implementation

All major tables include `deleted_at TIMESTAMP NULL` for soft deletes:

**Benefits**:
- Maintains data integrity for historical records
- Enables data recovery
- Preserves foreign key relationships
- Allows audit trails

**Implementation**:
```sql
-- Soft delete query
~~~

~~~
UPDATE job_applications SET deleted_at = NOW() WHERE id = 123;

-- Exclude soft deleted records
SELECT * FROM job_applications WHERE deleted_at IS NULL;

-- Restore soft deleted record
UPDATE job_applications SET deleted_at = NULL WHERE id = 123;
```

---

## Security Considerations

### 1. Authentication & Authorization

**Password Security**:
- Passwords stored using bcrypt hashing (Laravel default)
- Minimum 8 characters required
- Password reset via email tokens with expiration

**API Authentication**:
- Laravel Sanctum for token-based authentication
- Tokens stored in `personal_access_tokens` table (Laravel default)
- Token expiration and rotation policies

**Role-Based Access Control (RBAC)**:
```sql
-- User type determines access
user_type ENUM('company', 'university_student', 'teacher', 'parent')

-- Additional permission checks
is_approved BOOLEAN -- Admin approval required
is_verified BOOLEAN -- Company verification status
```

### 2. Data Privacy

**PII (Personally Identifiable Information)**:
- Email, phone, CV files, personal profiles
- GDPR compliance considerations
- Right to be forgotten (hard delete capability)

**Access Control**:
```sql
-- Student profile visibility
~~~

~~~
is_public BOOLEAN -- Controls company access to profile

-- Company data visibility
is_verified BOOLEAN -- Only verified companies shown to students
```

**Data Minimization**:
- Only collect necessary data
- Optional fields marked as NULLABLE
- JSON structure allows flexible data addition

### 3. SQL Injection Prevention

**Prepared Statements**:
- Laravel Eloquent ORM uses parameter binding
- Never concatenate user input into queries

**Input Validation**:
- Validate all user inputs before database operations
- Sanitize JSON inputs
- Escape special characters

### 4. File Upload Security

**CV & Logo Storage**:
```sql
cv_path VARCHAR(255) -- Stores relative path, not absolute
logo_path VARCHAR(255) -- Prevents path traversal
```

**Security Measures**:
- File type validation (whitelist)
- File size limits (5MB for CVs)
- Virus scanning before storage
- Randomized filenames to prevent guessing
- Separate storage from web root

### 5. Audit Trails

**Timestamps**:
```sql
created_at TIMESTAMP -- Record creation
updated_at TIMESTAMP -- Last modification
deleted_at TIMESTAMP -- Soft delete time
viewed_at TIMESTAMP -- First view time
~~~

~~~markdown
```

**Status History**:
```sql
status_history JSON -- Complete audit trail of status changes
```

### 6. Rate Limiting

Consider application-level rate limiting for:
- Job application submissions (prevent spam)
- CV downloads (prevent scraping)
- Profile views (prevent excessive tracking)

### 7. Data Encryption

**At Rest**:
- Encrypt sensitive columns (optional)
- Database-level encryption for CV files

**In Transit**:
- HTTPS for all API communications
- TLS for database connections

---

## Scalability & Future Enhancements

### 1. Horizontal Scaling Strategies

**Read Replicas**:
```
Primary (Master)
├── Read Replica 1 (Job Search)
├── Read Replica 2 (Application Queries)
└── Read Replica 3 (Course Queries)
```

**Sharding Opportunities**:
- Shard by user_type (separate company/student data)
- Shard by geographic region
- Shard by time (archive old applications)

### 2. Performance Optimization
~~~

~~~markdown
**Caching Strategy**:
```
Redis Cache Layers:
├── User Sessions (Sanctum tokens)
├── Job Listings (15 min TTL)
├── Course Catalog (30 min TTL)
├── Profile Data (1 hour TTL)
└── Notification Counts (real-time)
```

**Query Optimization**:
- Materialized views for complex aggregations
- Indexed views for common joins
- Partitioning large tables (job_applications by year)

### 3. Analytics & Reporting

**Data Warehouse**:
Consider ETL to separate analytics database:
- Job market trends
- Application success rates
- Course performance metrics
- User engagement analytics

**Recommended Additional Tables**:

```sql
-- Application Analytics
CREATE TABLE application_analytics (
    id BIGINT UNSIGNED PRIMARY KEY,
    date DATE NOT NULL,
    job_posting_id BIGINT UNSIGNED,
    views_count INT UNSIGNED DEFAULT 0,
    applications_count INT UNSIGNED DEFAULT 0,
    conversion_rate DECIMAL(5,2),
    INDEX (date, job_posting_id)
);

-- Student Engagement Metrics
CREATE TABLE student_engagement (
    id BIGINT UNSIGNED PRIMARY KEY,
    student_id BIGINT UNSIGNED,
    date DATE NOT NULL,
    profile_views INT UNSIGNED DEFAULT 0,
    applications_sent INT UNSIGNED DEFAULT 0,
~~~

~~~
    courses_enrolled INT UNSIGNED DEFAULT 0,
    INDEX (student_id, date)
);
```

### 4. Advanced Features

**Recommendation Engine**:
```sql
-- Job Recommendations
CREATE TABLE job_recommendations (
    id BIGINT UNSIGNED PRIMARY KEY,
    student_id BIGINT UNSIGNED NOT NULL,
    job_posting_id BIGINT UNSIGNED NOT NULL,
    score DECIMAL(5,2), -- ML-generated match score
    factors JSON, -- Why this job matches
    created_at TIMESTAMP,
    INDEX (student_id, score DESC)
);

-- Course Recommendations
CREATE TABLE course_recommendations (
    id BIGINT UNSIGNED PRIMARY KEY,
    student_id BIGINT UNSIGNED NOT NULL,
    course_id BIGINT UNSIGNED NOT NULL,
    score DECIMAL(5,2),
    created_at TIMESTAMP,
    INDEX (student_id, score DESC)
);
```

**Skills Matching**:
```sql
-- Normalized Skills Table (for better matching)
CREATE TABLE skills (
    id BIGINT UNSIGNED PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    category VARCHAR(50),
    created_at TIMESTAMP
);

CREATE TABLE student_skills (
    student_id BIGINT UNSIGNED,
    skill_id BIGINT UNSIGNED,
    proficiency_level ENUM('beginner', 'intermediate', 'advanced', 'expert'),
~~~

~~~~
    PRIMARY KEY (student_id, skill_id)
);

CREATE TABLE job_skills (
    job_posting_id BIGINT UNSIGNED,
    skill_id BIGINT UNSIGNED,
    is_required BOOLEAN DEFAULT TRUE,
    PRIMARY KEY (job_posting_id, skill_id)
);
```

**Messaging System**:
```sql
-- Direct messaging between companies and students
CREATE TABLE messages (
    id BIGINT UNSIGNED PRIMARY KEY,
    sender_id BIGINT UNSIGNED NOT NULL,
    recipient_id BIGINT UNSIGNED NOT NULL,
    subject VARCHAR(255),
    body TEXT NOT NULL,
    read_at TIMESTAMP NULL,
    created_at TIMESTAMP,
    INDEX (recipient_id, read_at),
    FOREIGN KEY (sender_id) REFERENCES users(id),
    FOREIGN KEY (recipient_id) REFERENCES users(id)
);
```

**Review System**:
```sql
-- Students review companies
CREATE TABLE company_reviews (
    id BIGINT UNSIGNED PRIMARY KEY,
    company_id BIGINT UNSIGNED NOT NULL,
    student_id BIGINT UNSIGNED NOT NULL,
    rating DECIMAL(3,2) CHECK (rating >= 0 AND rating <= 5),
    review_text TEXT,
    is_anonymous BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP,
    UNIQUE (company_id, student_id)
);

-- Students review courses
CREATE TABLE course_reviews (
    id BIGINT UNSIGNED PRIMARY KEY,
~~~~

~~~
    course_id BIGINT UNSIGNED NOT NULL,
    student_id BIGINT UNSIGNED NOT NULL,
    rating DECIMAL(3,2) CHECK (rating >= 0 AND rating <= 5),
    review_text TEXT,
    created_at TIMESTAMP,
    UNIQUE (course_id, student_id)
);
```

### 5. Archive Strategy

**Historical Data Management**:
```sql
-- Archive old applications (> 2 years)
CREATE TABLE job_applications_archive (
    -- Same structure as job_applications
    archived_at TIMESTAMP NOT NULL
) PARTITION BY RANGE (YEAR(created_at));

-- Archive completed courses
CREATE TABLE course_enrollments_archive (
    -- Same structure as course_enrollments
    archived_at TIMESTAMP NOT NULL
);
```

### 6. Multi-Tenancy Considerations

If scaling to multiple universities/institutions:

```sql
-- Add institution/tenant support
ALTER TABLE users ADD COLUMN institution_id BIGINT UNSIGNED;
ALTER TABLE companies ADD COLUMN institution_id BIGINT UNSIGNED;
ALTER TABLE courses ADD COLUMN institution_id BIGINT UNSIGNED;

CREATE TABLE institutions (
    id BIGINT UNSIGNED PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    domain VARCHAR(255) UNIQUE,
    settings JSON,
    created_at TIMESTAMP
);
```
~~~

~~~markdown
### 7. API Rate Limiting

```sql
-- Track API usage for rate limiting
CREATE TABLE api_rate_limits (
    id BIGINT UNSIGNED PRIMARY KEY,
    user_id BIGINT UNSIGNED,
    endpoint VARCHAR(255),
    request_count INT UNSIGNED DEFAULT 0,
    window_start TIMESTAMP,
    INDEX (user_id, window_start)
);
```

### 8. Backup & Disaster Recovery

**Backup Strategy**:
- Daily full backups
- Hourly incremental backups
- Point-in-time recovery capability
- Geographic redundancy

**Critical Tables** (prioritize for recovery):
1. users
2. job_applications
3. course_enrollments
4. payments

---

## Database Initialization Scripts

### MySQL Database Creation

```sql
-- Create database with UTF-8 encoding
CREATE DATABASE ace_platform
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;

USE ace_platform;

-- Set optimal MySQL configuration
SET GLOBAL innodb_buffer_pool_size = 2147483648; -- 2GB
SET GLOBAL max_connections = 500;
~~~

~~~
SET GLOBAL innodb_file_per_table = ON;
```

### PostgreSQL Database Creation

```sql
-- Create database with UTF-8 encoding
CREATE DATABASE ace_platform
    WITH ENCODING 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TEMPLATE template0;

\c ace_platform;

-- Enable extensions
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pg_trgm"; -- For fuzzy search
```

---

## Conclusion

This database design for the ACE platform provides:

✅ **Scalability**: Supports growth from hundreds to millions of users
✅ **Performance**: Strategic indexing and denormalization for fast queries
✅ **Integrity**: Comprehensive constraints and foreign key relationships
✅ **Flexibility**: JSON columns for evolving data requirements
✅ **Security**: Multiple layers of access control and data protection
✅ **Maintainability**: Well-documented, normalized structure
✅ **Extensibility**: Clear paths for future feature additions

### Key Metrics

- **Total Tables**: 13 core tables + additional tables for features
- **Estimated Storage** (10,000 users):
  - Users & Profiles: ~50 MB
  - Job Postings & Applications: ~200 MB
  - Courses & Enrollments: ~100 MB
  - Files (CVs, videos): ~20 GB
  - **Total**: ~20.5 GB

- **Estimated Query Performance**:
~~~

```
  - Job search: < 100ms
  - Application listing: < 50ms
  - Profile load: < 20ms
  - Course enrollment: < 30ms

### Migration Path

1. **Phase 1**: Core user management and authentication
2. **Phase 2**: Job posting and application system
3. **Phase 3**: Course management and LMS features
4. **Phase 4**: Advanced features (recommendations, messaging)
5. **Phase 5**: Analytics and reporting infrastructure

---

**Document Version**: 1.0
**Last Updated**: 2024-01-15
**Database Design By**: ACE Development Team
```

4.3. Data Flow & System Behavior

Data flows through the system involving user authentication, profile creation, course interactions, job applications, and administrative management. APIs facilitate communication between the frontend and backend. User actions trigger specific backend processes, database updates, and notifications, ensuring a coherent and dynamic user experience.

Note: Specific diagrams (Data Flow, Sequence, Activity, State) are provided in the database design section source material for detailed representation there.

4.4. UI/UX Design & Prototyping:

The UI/UX design is inspired by professional development environments like GitHub (https://github.com/) and VS Code, emphasizing simplicity, clarity, and personalization. Key principles include a smooth user experience, goal focus (highlighting progress and opportunities), social interaction, and bilingual support. Visual identity uses calm colors and modern fonts. Accessibility is ensured through responsiveness and high contrast. Interactive prototypes were developed to test user flows and gather early feedback.

4.5. System Deployment & Integration:

The system is built on a modular, service-oriented architecture. The frontend uses React (Next.js) with SSR, while the backend is developed with Laravel. MySQL serves as the database. Integrations include payment gateways (Stripe, PayPal), Agora for academic email verification, and DIDIT for AI-powered identity verification, ensuring a secure and feature-rich platform.

4.6. Additional Deliverables

Key deliverables include the fully functional ACE platform, comprehensive technical documentation, user manuals, and a project presentation with a video demonstration of the system's capabilities.

Implementation (Source Code & Execution)

5.1 Source Code

https://github.com/nhahub/NHA-138

The codebase is structured with a clean folder organization (e.g., `components`, `pages`, `services`, `contexts`) and adheres to consistent naming conventions (camelCase, PascalCase). Reusable React components, utility functions, and custom hooks are employed to enhance maintainability and scalability. Code is well-commented, and linting tools like Prettier ensure quality. Robust security measures and error handling (form validation, authentication, role-based access, try-catch blocks) are implemented.

5.2 Version Control & Collaboration

The project is hosted on a private GitHub repository for source control. A clear branching strategy is followed, with features developed in separate branches. Descriptive commit messages document changes. Continuous integration is facilitated through Vercel deployment, allowing for real-time previews and feedback, though advanced CI/CD tools were not fully set up in the initial phase.

5.3 Deployment & Execution

A comprehensive README.md file guides users through installation, system requirements, configuration, and execution (local or deployed). API documentation is available in the repository. The project includes executable files or a link to the deployed version.

Testing & Quality Assurance

6.1 Test Cases & Test Plan

Test scenarios are organized into categories such as User Authentication, Admin Dashboard functionality, and Course/Job workflows. The test plan focuses on critical functionalities including user login/logout, search accuracy, application submission, course enrollment, and administrative tasks. Expected outcomes are clearly defined for each test case.

6.2 Automated Testing

While full automation was not implemented in the initial phase, the plan includes future integration of unit tests using Jest and React Testing Library to improve code stability and coverage. Manual testing of core features ensures immediate quality assurance.

6.3 Bugs

Common issues identified during testing include API response delays, UI misalignments on various screen sizes, and broken links. These are documented and prioritized for resolution.

Final Presentation & Reports

7.1 User Manual

The User Manual guides users through accessing the platform, creating accounts, navigating different sections (student profile, course catalog, job listings, teacher dashboard, parent view), enrolling in courses, applying for jobs, managing profiles, and utilizing communication features. It provides step-by-step instructions for all core functionalities.

7.2 Technical Documentation

The technical documentation details the system architecture, API endpoints, database schema, and technology stack. It includes RESTful API routes for User management, authentication, course operations, job postings, applications, and more, serving as a reference for developers and system administrators.

7.3 Presentation

https://drive.google.com/file/d/1U7QieFolTmTjrc1kD8M2k2tvdo5oSasq/view?usp=sharing

The project presentation summarizes the ACE platform's vision, mission, problem statement, solution, features, architecture, and future roadmap. It highlights the key innovations and the potential impact of the platform in bridging education and employment.

7.4 Video Demonstration

A video demonstration provides a visual walkthrough of the ACE platform, showcasing its key features and user experience across different roles.