CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Backend Development
Student Management System

LECTURER: HOK TIN
TEAM 06: LIM PANHA, LENG SOKPUNLEU, KHORN VANNDA

Table of Contents

01 **Introduction**

02 **Frontend**

03 **Backend**

04 **Database**

05 **Conclusion & Future Work**

06 **Demo**

FinalProject (Backend Course)

Responsibility Roles

Team 06	Responsible For
• Khorn Vannda	Frontend, 3 roles (admin, teacher, student)
• Leng Sokpunleu	Controllers, Routes, Authentication (Log in)
• Lim Panha	Database, Middleware, Models

1. Introduction

- A **fullstack** web-based application to manage core academic operations in a school or university.
- Solves the common pain points of manual data handling in academic settings.
- Supports **three roles:** Admin, Teacher, Student — each with specific tasks and permissions.
- **Technologies used:** React (**Vite**) + Node.js (**Express**) + Supabase (**MySQL**) as the database with Sequelize ORM.

2. Frontend (React + Vite + Tailwind)

Folder: src/

Role-based Structure:

- components/Admin/: Manage users, view dashboards
- components/Teacher/: Manage classes, view students
- components/Student/: View subjects, profiles, grades

Key Features:

- AuthContext.tsx: Stores logged-in user and role
- ProtectedRoute.tsx: Restricts unauthorized access by role
- Tailwind.css: Ensures clean, responsive design

3. Backend (Node.js + Express + JWT Auth)

Folder: server/

Key Parts:

- Controllers: Handle business logic (e.g., studentController.js)
- Routes: Role-based (admin, student, teacher)
- Middleware: JWT token validation (middleware/auth.js)
- Models: Sequelize models representing DB schema (e.g., User.js, ExamResult.js)

4. Database (MySQL via Supabase)

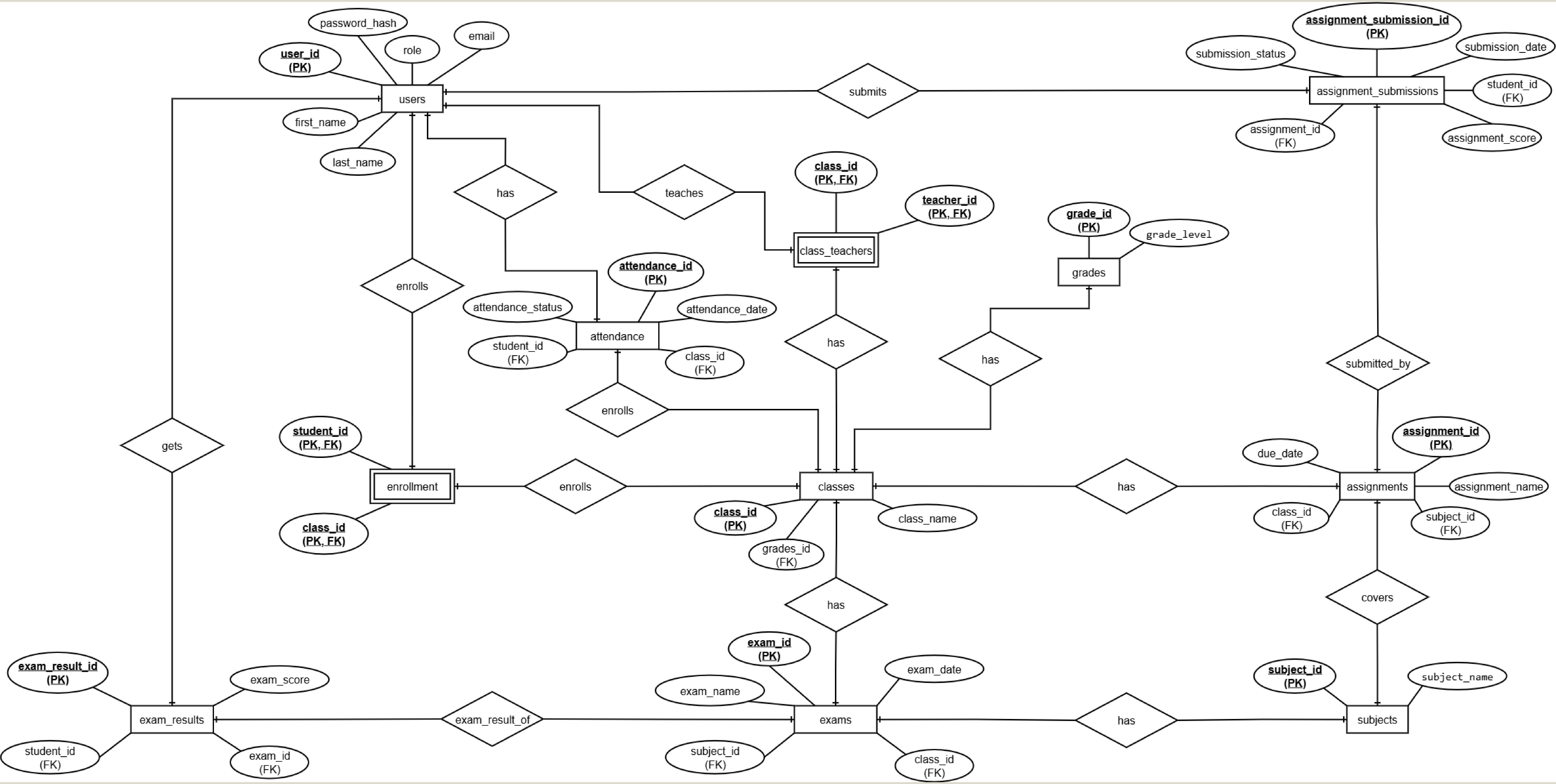
Folder: supabase/

Key Relations:

- 11 tables with 12 relationships

Main Tables:

- Users (role, email, password)
- Subjects, Classes, Grades, Attendance
- Assignments, Assignment_Submissions, Exams, Exam_Results
- Composite Key = (Enrollment, Class_teachers)


password_hash
assignment_submission_id (PK)
role
email
user_id (PK)
submission_status
submission_date
users
submits
assignment_submissions
student_id (FK)
first_name
assignment_id (FK)
assignment_score
last_name
class_id (PK, FK)
has
teaches
teacher_id (PK, FK)
grade_id (PK)
grade_level
class_teachers
grades
attendance_id (PK)
enrolls
attendance_status
attendance_date
attendance
has
has
student_id (FK)
class_id (FK)
submitted_by
enrolls
gets
student_id (PK, FK)
due_date
assignment_id (PK)
enrollment
enrolls
classes
has
assignments
assignment_name
class_id (PK, FK)
class_id (PK)
class_name
class_id (FK)
subject_id (FK)
grades_id (FK)
has
covers
exam_id (PK)
exam_result_id (PK)
exam_score
exam_date
subject_id (PK)
subject_name
exam_name
exam_results
exam_result_of
exams
has
subjects
student_id (FK)
exam_id (FK)
subject_id (FK)
class_id (FK)

5. Conclusion & Future Work

Conclusion:

- This project helped us build a fully functional and secure fullstack system.
- Taught us how to manage logic between frontend, backend, and relational DB.
- Each role’s needs are clearly addressed with role-based UI and APIs.

Future Work:

- Add real-time notification system for assignments
- Add chat for student–teacher interaction
- Add charts/analytics for grades and attendance
- Add file uploads for assignment submissions

DEMO

Thank you for your kind listening!