React Frontend (Vercel)

Host: https://prodemyx.vercel.app
Consumes REST APIs from Node.js backend.

MySQL Database

Schema is here:
Tables:

- users
- categories
- courses
- course_schedules
- purchases
- user_course_access

Complete Workflow of the System

User Flow: Authentication & Access

Frontend uses login screens like:

Login Process

1. User enters email/password.
2. Frontend calls Node backend: /auth/login.
3. Backend verifies against users table.
4. Backend returns auth token + role (admin, instructor, student).
5. Frontend routes user to appropriate dashboard.

Roles matter heavily:

- Students → Course catalog, enrollment
- Instructors → Course schedules, teaching tools
- Admin → Full control panel (courses, users, analytics)

Admin Workflow (Your uploaded UI is 90% Admin)

Your admin interfaces include:

- Dashboard:

- Create Course:
- Enrollments:
- Category Creation:
- Course Management:
- User Creation:
- User Management:
- Reports:

Your Admin Portal workflows look like this:

A. Categories → Courses → Instructor Assignment → Scheduling

1. Create Category

Using interface:
Admin creates a category → stored in categories table.

2. Create Course

From:
Fields map to DB table courses:

UI Field	DB Field
Title	title
Description	description
Category	category_id
Duration/Price	duration, price
Photo	photo

3. Assign Instructor

From same page: "Assign Instructor" dropdown
Maps to instructor_id in course_schedules table.

4. Scheduling

Your DB has a course_schedules table:

Fields:

- meeting_title
- meeting_link
- meeting_date
- meeting_time
- instructor_id
- course_id

This is for online class sessions.

Your admin UI supports creating schedules via the “Scheduling & Logistics” section.

B. User Management

Using:

- Create User:
- Manage Users:

Users map directly to users table.

Roles determine permissions:

- Admin: everything
- Instructor: only their courses
- Student: only enrolled courses

C. Enrolments

Using:

This ties into:

1. purchases table

Tracks payments:

- user_id
- course_id
- payment_id
- status

2. user_course_access table

This grants actual permissions:

- user_id
- course_id

Admin can:

- Enroll students manually
- Edit enrollment status
- Remove/drop students

These operations map directly to inserts/updates on user_course_access and purchases.

D. Reports

Using:

Reports require backend aggregation:

- Completion rate
- Course performance
- Active users
- Instructor performance
- System usage

Student Workflow

Students never use these admin HTML files — they use your React frontend at Vercel.

Their flow:

1. Browse Courses

Frontend fetches /courses → shows cards.

2. Purchase Course

Triggers insert into purchases.

3. Access Content

Backend checks user_course_access.

4. Join Live Sessions

Schedules pulled from course_schedules.

5. Track Progress

Instructor Workflow

Instructors should:

- View courses assigned to them
- Manage schedules
- Upload content
- Track student progress
- Host sessions (Zoom/Meet links)

Database Relationships

Users ↔ Purchases ↔ Courses

Purchases show payment history.

Users ↔ user_course_access ↔ Courses

Access table shows what the student can view.

Courses ↔ Categories

Organized by category.

Courses ↔ course_schedules ↔ Users (instructors)

Schedules for classes.

Final Summary

Your project's workflow currently operates like this:

1. Admin sets up categories.

2. Admin creates courses.
3. Admin assigns instructors.
4. Admin configures schedules.
5. Students browse/purchase courses on React frontend.
6. Purchases trigger access grants.
7. Students attend sessions.
8. Admin uses reports to track activity.