# LABORATORY EXERCISE 2

# DATABASE DESIGN AND MIGRATION

**Learning Objectives**

By the end of this laboratory exercise, students should be able to:

* Design a database schema for a Learning Management System (LMS).
* Identify and define core database tables and relationships.
* Implement database migrations using CodeIgniter’s migration feature.
* Seed the database with sample data for testing purposes.
* Use GitHub for version control and tracking database changes.

**Prerequisite student experiences and knowledge**

Before starting this exercise, students should have:

* Basic knowledge of relational database concepts.
* Familiarity with MySQL database creation and management.
* Understanding of CodeIgniter framework basics.
* Experience using Git and GitHub for project version control.
* Ability to run and configure a local web server (XAMPP/WAMP/LAMP).

**Background**

This laboratory focuses on creating the database backbone for the LMS project. The process includes identifying core tables, defining their structure, setting up migrations for automated schema creation, and seeding data for testing. Using migrations ensures version-controlled database changes that can be easily shared among team members through GitHub.

**Materials/Resources**

* **Personal Computer with Internet Access**
* **XAMPP/WAMP/LAMP server installed**
* **CodeIgniter Framework (latest version)**
* **Bootstrap (via CDN or downloaded files)**
* **Visual Studio Code or any code editor**
* **Git and GitHub Account**
* **Web Browser (Chrome, Firefox, etc.)**

**Laboratory Activity**

**Step 1: Identify Core Tables**

List the primary tables for the LMS:

1. users – stores student, instructor, and admin information.
2. courses – contains course details.
3. enrollments – manages student enrollment in courses.
4. lessons – stores lesson content linked to courses.
5. quizzes – contains quiz questions linked to lessons.
6. submissions – stores quiz submissions and results.

**Step 2: Create Database**

1. Open phpMyAdmin.
2. Create a new database named: lms\_lastname
3. Do not create tables manually — migrations will handle this.

**Step 3: Enable CodeIgniter Migrations**

1. In `application/config/migration.php`, set:

$config['migration\_enabled'] = TRUE;

$config['migration\_type'] = 'sequential';

1. In `application/config/config.php`, set your database connection details.

**Step 4: Create Migration Files**

1. In the terminal, navigate to your project folder.
2. Create migration files for each table:

php spark make:migration CreateUsersTable

php spark make:migration CreateCoursesTable

php spark make:migration CreateEnrollmentsTable

php spark make:migration CreateLessonsTable

php spark make:migration CreateQuizzesTable

php spark make:migration CreateSubmissionsTable

1. Define the schema in each migration file.

**Step 5: Run Migrations**

Run the following command:

* php spark migrate

**Step 6: Seed Sample Data**

1. Create a seeder: **php spark make:seeder UserSeeder**
2. Add sample users (students, instructors, admin).
3. Run the seeder: **php spark db:seed UserSeeder**

**Step 7: Push to GitHub**

1. Stage and commit your migration and seeder files:

**git add .**

**git commit -m "Added database migrations and seeders"**

**git push**

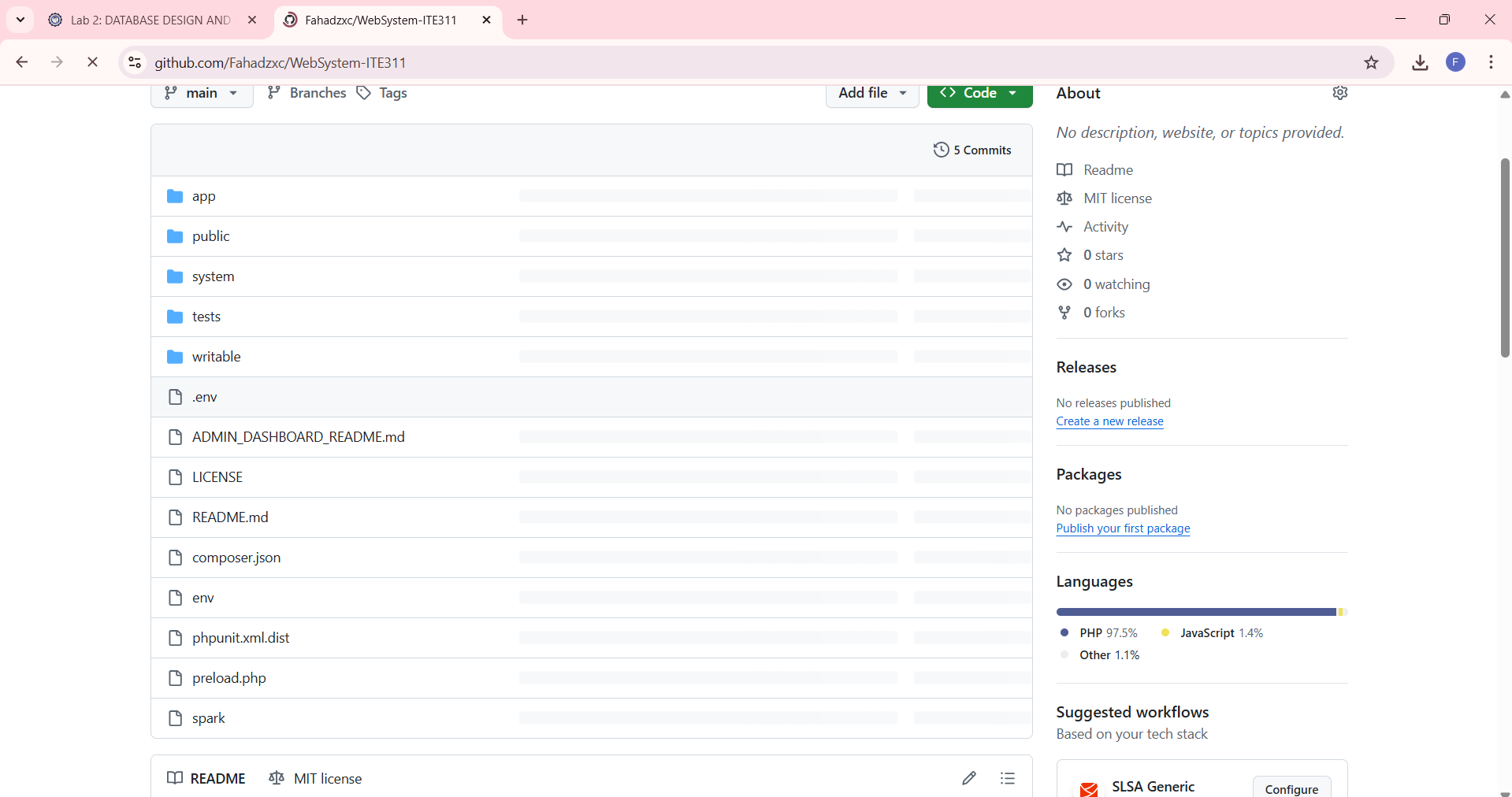
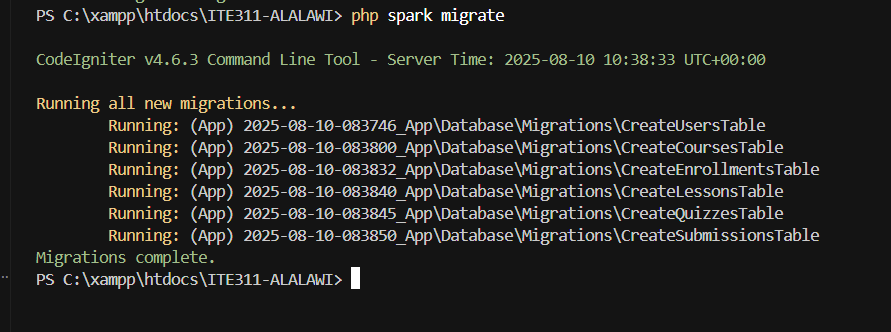
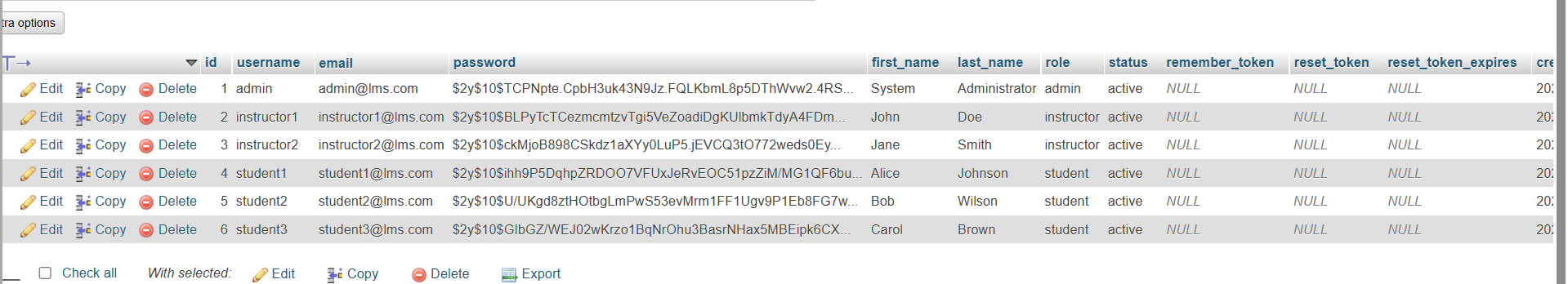
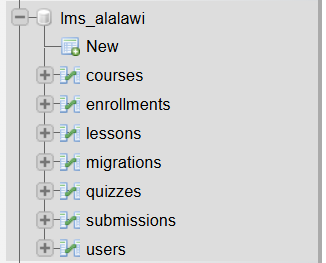
Output / Results

* Screenshot of successful migration in phpMyAdmin.
* Screenshot of seeded sample data in phpMyAdmin.
* Screenshot of GitHub repository with migration files.

**QUESTIONS:**

1. Why are migrations important in database development?Migrations make it easy to update and manage your database structure without manually editing tables. They keep changes organized, trackable, and consistent.
2. How does using GitHub with migrations help in team projects?Using GitHub with migrations lets the whole team share and apply the same database changes. This avoids confusion and keeps everyone’s database in sync.
3. What is the advantage of seeding sample data during development?  
   Seeding adds ready made sample dataso developers can test features without manually entering information each time. It speeds up testing and makes development smoother.

**Output / Results**



**Conclusion**  
Migrations keep databases consistent, GitHub makes teamwork easier, and seeding provides quick testing data. Together, they make database development faster, more organized, and more reliable.