1. Database Setup

The student_system database was structured to represent real-world entities such as users, classes, assignments, and submissions. It contains **10 interrelated tables**, each with a clear purpose and proper foreign key relationships to maintain data integrity:

- **users** – stores login credentials and role associations.
- **roles** – defines access levels: Admin, Teacher, Student.
- **subjects** – a list of school subjects.
- **classes** – links a subject with a teacher, room, and schedule.
- **enrollments** – many-to-many mapping of students to classes.
- **assignments** – tasks assigned by teachers to classes.
- **assignment_files** – optional resources uploaded with assignments.
- **submissions** – student-uploaded files with comments, grades, and timestamps.
- **attendance** – class attendance per student.
- **grades** – stores individual student assessment results if separate from `submissions`.

Key setup decisions included:

- All relevant foreign keys were indexed and enforced (e.g., `user_id`, `class_id`, etc.).
- Tables included `created_at` timestamps for auditability.
- `ON DELETE RESTRICT` was applied where needed to prevent accidental data loss (e.g., when a class has student grades).

2. Techniques for Data Manipulation

The system uses **PHP with PDO (PHP Data Objects)** to handle all database operations securely. The approach includes:

- **Prepared Statements**: All queries use placeholders to prevent SQL injection.
- **POST & GET Handling**: Forms throughout the application submit data via POST, which is then validated and passed to the database.
- **Session Management**: Each user type (Admin, Teacher, Student) is authenticated and assigned access through session variables.
- **Data Binding**: Dropdowns, tables, and views are dynamically populated with database data (e.g., subject lists, assignment tables).
- **Validation & Error Control**:
    - Forms include required fields and fallback messages.
    - Server-side validation checks for logic like empty teacher selection or invalid dates.
    - Errors (like deletion attempts on graded classes) are caught and converted into user-friendly messages.

---

3. Virtual Server Setup (Localhost)

The application was developed and tested using a **local virtual server** created with **XAMPP** on a Windows machine. The setup process included:

- Installing XAMPP, which bundles **Apache (web server)** and **MySQL (database server)**.
- Placing project files inside the `htdocs/` directory.
- Launching the local server via the XAMPP control panel.
- Accessing the application via `http://localhost/student-system/`.

- Managing the database using **phpMyAdmin**, a GUI that allows for easy table creation, query testing, and relationship configuration.

This mimicked a live production environment, allowing for fast iterations and offline development.

---

4. Dynamic Web Application Techniques

To make the web application dynamic and role-aware, the following development techniques were applied:

- **Role-Based Rendering**: Each user sees a custom dashboard and pages (e.g., only teachers can grade, only students can submit).
- **Reusable Components**: Layouts like header, footer, and dashboard wrapper were modularized using PHP `include` statements for maintainability.
- **Data-Driven Content**: Tables and forms are generated based on live data, allowing real-time display of assignments, grades, attendance, etc.
- **Validation & Interactions**:
    - HTML5 form features like required, datetime-local, and select were combined with JavaScript prompts.
    - PHP fallback logic ensured server-side control even if client-side validation failed.
- **Responsive UI & Styling**: CSS was used to ensure layouts looked good on both large screens and smaller devices. The footer, for example, remains at the bottom using flex-based layout logic.

✅ **Test Cases Table (17 Total)**

#	Test Action	Expected Output	Actual Output	Pass/ Fail	Notes
1	Login with valid admin credentials	Redirect to admin dashboard	Directed to admin dashboard	Pass	Session starts, role loaded
2	Login with password under 4 characters	Show red validation prompt	Red prompt appears instantly	Pass	Handled by frontend form
3	Hover over buttons	Button color darkens	Buttons are darkened over hover	Pass	Shows interactivity via CSS
4	Add a new subject as admin	Subject appears in dropdown	Appears immediately	Pass	Inserted into `subjects` table
5	Create new class with valid inputs	Class appears in class list	Class saved in DB	Pass	Joins `classes`, `teachers`, `subjects`
6	Attempt to create class without assigning teacher	User-friendly error shown	Prompt: “Please assign teacher”	Pass	Validated client + server side

7	Delete a class with graded submissions	Block deletion, show warning	Message: " "student submissions exist."	Pass	FK constraints respected
8	Enroll student into class	Student name becomes checked & appears in their classes	Appears on student dashboard for student & checked from admin side	Pass	New row in `enrollments`
9	Create assignment for a class	Students see assignment	Appears in "My Assignments"	Pass	`assignments` and `classes` joined correctly
10	Student submits assignment file	File stored and listed	Downloadable in "My Submissions"	Pass	Stored in `uploads/submissions/`
11	Submit without attaching a file	Block submission, show error	Prevents submit	Pass	HTML `required` validation works
12	Teacher grades a submission	Student sees grade and feedback	Grade appears in submission row	Pass	`UPDATE` query executed
13	View submission with grade = NULL	Pending" shown instead	Renders "Pending" placeholder	Pass	Null-safe display logic used

14	Admin changes a user's role	Affected dashboard changes	Correct role interface shown	Pass	`users.role_id` updated
15	View attendance record for a class	List with ✓ / ✗ per student	Check or Cross is shown depending on status	Pass	Pulled from `attendance` table
16	Attempt to access student page with no session	Redirect to login page	Redirect successful	Pass	Auth check blocks access
17	Teacher submit feedback on graded work	Student receives grade/feedbac k & green confirmation propmt	Displays feedback text “Grade updated” in green	Pass	Teacher input saved to `submissions.feedback`
18	Teacher saves attendance for a class	Green success prompt appears with date	Message ”Attendance saved/updated successfully for today	Pass	Confirms successful `INSERT` or `UPDATE` to `attendance` table
19	Delete assignment with submissions	Block deletion with message	Message “Cannot delete this assignment”	Pass	Prevents FK violation

20	Add assignment without title or date	Block submission	Form prevents save	Pass	HTML5 required attribute + backend check