Php and Databases
Yan Cesare – IDM 5.1
Technical Document

Setting up a local virtual server

To build and test my web application, I used XAMPP, which allowed me to create a local virtual server on my computer.

I opened the XAMPP control panel and started both Apache and MySQL. Once these were running, my computer could act like a live server.

I placed my project folder inside the htdocs directory that comes with XAMPP. This is important because Apache only serves files from this folder. After doing this, I could access my project through the browser using http://localhost/studentbase/.

This allowed me to fully test my system locally before it would ever be hosted online.

Database setup

The database for my project was created using phpMyAdmin. I created a database called Student Base and then designed the tables based on an ERD that I had planned earlier.

I created tables such as:

- users
- roles
- courses
- units
- classes
- assignments
- submissions
- grades
- notifications

Each table has a primary key (for example user_id, course_id, etc.), and these were set to **auto-increment** so that each record is unique.

I also used foreign keys to link tables together. Such as:

- users table is linked to roles
- users can then be linked to a course and a class
- assignments table is linked to units
- submissions is then linked to assignments and students

These relationships help keep the data consistent and prevent invalid data from being entered into the database. I also added some initial data, such as user roles and sample courses, so the system could be tested properly.

Techniques used to build a dynamic web application

The website is dynamic, meaning the content changes depending on the user and the data stored in the database. I used PHP to generate pages dynamically instead of writing HTML. For example, user lists, courses, classes, and assignments are all loaded from the database and displayed automatically.

To avoid repeating the same code, I used includes for common parts of the website such as the header, footer, navigation bar, and sidebar. This made the site easier to manage and more consistent. I also implemented authentication and role-based access control. When a user logs in, their details are stored in a session. Depending on their role (admin, lecturer, or student), they are redirected to the correct dashboard and prevented from accessing pages they shouldn't see.

Bootstrap was used for the interface, which helped with layout, responsiveness, and components like modals, alerts, off-canvas panels, and tables. This made the website look more like a real system.

In some areas, I used JavaScript with the Fetch API to load data dynamically without reloading the page, such as filtering dropdowns or loading profile information in side panels.

Techniques used to manipulate database data through the application

The system allows users to interact with the database through the website using forms and buttons. This includes creating, viewing, editing, and deleting data.
Admins can create and manage users, courses, units, classes, and timetables. Lecturers can create assignments and grade submissions. Students can submit assignments and upload files.

All database operations are handled using PHP and MySQLi. For inserts, updates, and deletes, I used prepared statements.

Form data is sent using the POST method, validated in PHP, and then processed before being saved to the database. In some cases, I added extra checks, such as requiring students to have a course and class assigned. Some features required multiple database actions at once. For example, when creating a unit, it is inserted into the units table and then linked to courses and lecturers using intermediatory tables.

I also implemented file uploads for assignment submissions, where uploaded files are stored on the server and their paths are saved in the database.
Finally, I created a notifications system where events like assignment creation or grading can be stored in a notifications table and later shown to users in the interface.

Test Cases (following IPO Chart Process)
Notes (so it sounds like you)

- I started testing **as I built each page/feature** (Task 2 stage) instead of leaving it for the end.
- Whenever something failed, I fixed it, then **re-tested the same case** until it passed.
- I tested using:
    - Chrome browser
    - XAMPP (Apache + MySQL)
    - phpMyAdmin for verifying DB inserts/updates
    - Different user roles: Admin, Lecturer, Student

Account Management — Test Cases

A1 — Create user (Admin creates user)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Name, Email, Password, Role	Validate fields, hash password, check duplicates, insert into DB	User account created	User created + appears in Users table	Pass
Same email again	Check duplicates	Error shown: duplicate email	Error shown + user not created	Pass
Student role but no Course/Class selected	Validate student requires course+class	Warning shown, stop submit	Form submitted anyway (no warning)	Fail
Student role but no Course/Class selected	Add validation (JS + PHP safety)	Warning shown, stop submit	Warning shown, not submitted	Pass

A2 — Update user (Edit user modal)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Update name/email	Update user record	Details updated	Updated correctly in DB + UI	Pass
Change student course → load correct class list	Fetch classes by course_id	Correct class dropdown loads	Dropdown empty	Fail
Change student course	Fix fetch endpoint/IDs	Dropdown loads class list	Class list loads correctly	Pass

A3 — Deactivate user (soft delete)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Delete user	Mark deleted=1 (not remove row)	User disappears from Users page	User removed from list	Pass
Dashboard counts after delete	Count only deleted=0	Card values exclude deleted	Counts still included deleted users	Fail
Same action after fix	Add WHERE deleted=0 in counting functions	Counts exclude deleted	Numbers correct	Pass

Login System — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Correct email+password	Verify credentials, create session	Redirect to dashboard	Redirected correctly based on role	Pass
Invalid email format	Detect invalid format	"Invalid email format"	Browser validation blocked form	Pass
Email not registered	Check email exists	"Email not found"	"No user" error shown	Pass
Wrong password	Verify password	"Incorrect password"	Wrong password message	Pass
Not logged in → open admin URL	Require session + requireRole(1)	Redirect to login	Redirects to login	Pass
Logout	Destroy session	User fully logged out	Logout works but back button showed page	Fail
Logout + browser back	Improve auth checks	Back shows login page (blocked)	Fixed (page blocked)	Pass

Forgot Password System — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Email exists	Generate token, insert into reset table	"Reset link sent"	Token inserted in DB	Pass
Email not found	Check email exists	"Account not found"	Correct error shown	Pass
Expired token	Check expiry	"Token expired"	Token rejected	Pass
Reused token	Check used flag	"Link no longer valid"	Correct message	Pass

Course Management — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Course name/desc/code	Validate fields, insert	Course created	Created & visible in table	Pass
Edit course	Update record	Course updated	Updates correctly	Pass
Delete course (normal delete)	Delete from DB	Course removed	Foreign key prevented delete	Fail
Delete course after fix	Delete dependent rows first / restrict if in use	Delete works or warning shown	Working as expected	Pass

Unit Management — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Unit name + course	Insert unit + link course_unit	Unit created + linked	Works correctly	Pass
Assign lecturers	Insert into unit_lecturers	Lecturers assigned	Lecturers didn't insert	Fail
Assign lecturers again	Fix insert loop + field names	Lecturers insert correctly	Works & shows on UI	Pass
View units per course	Filter by course_id	Only matching units show	Correct units show	Pass

Class Register — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Add class (course/year/group)	Validate + insert	Class created	Created correctly	Pass
View students in class	Query users where class_id	Student list shows	List shows correctly	Pass
Offcanvas user profile from class list	Fetch profile fragment	Offcanvas opens & loads	Link opened new page instead	Fail
Retest after applying openUserProfile	Use same JS + offcanvas as Users page	Offcanvas works	Works correctly	Pass

Timetable System — Test Cases

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Add lesson (class/unit/lecturer/day/time)	Validate + insert timetable row	Lesson appears in grid	Appears correctly	Pass
Add lesson with end < start	Validate time logic	Error: “End must be later”	Correct error returned	Pass
Add lesson clash (same class/time overlap)	Check class clashes	Error: “Class clash”	Detected clash correctly	Pass
Add lesson lecturer double-booked	Check lecturer schedule	Error: “Lecturer clash”	Detected correctly	Pass
Edit lesson	Update row	Lesson updated	Updated correctly	Pass
Delete lesson with confirmation modal	Delete timetable row	Removed from grid	Works correctly	Pass

Assignment Management — Test Cases (Lecturer)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Create assignment (title/desc/unit/due)	Validate + insert	Assignment created	Created & visible	Pass
Upload brief	Upload file + store path	Brief visible to students	Brief visible	Pass
Edit assignment	Update record	Assignment updated	Not implemented yet	Fail
Edit after adding edit modal + update script	Update record	Updates correctly		Pass
Delete assignment	Delete record	Removed from list		Pass

Submission Management — Test Cases (Student)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Upload 1 file	Validate type/size, upload, insert submission/files	Submission saved	File not uploading	Fail
Retest after fixing form enctype + upload path	Same process	Submission saved		Pass
Upload multiple files	Save each file row linked to submission_id	All files saved	Works & all files appear	Pass

Grading Management — Test Cases (Lecturer)

Input	Process	Output (Expected)	Actual Result	Pass/Fail
Lecturer clicks View submission	Load submission modal	Submission loads	"Invalid submission"	Fail
Fix submission_id auto increment	Load again	Submission loads		Pass
Save grade	Insert/update grade record	Grade saved	Saves correctly	Pass
Student sees grade	Fetch grades for student	Grade visible	Visible correctly	Pass

Notification System — Test Cases

Trigger	Process	Output (Expected)	Actual Result	Pass/Fail
Lecturer creates assignment	Insert notifications for students in unit/course	Students receive notif	No row inserted	Fail
Fix insert logic to notify correct users	Insert notifications	Rows inserted in notifications table		Pass
Student loads navbar	Fetch unread count + list	Badge updates + dropdown list	Badge + list works	Pass
Mark notification read	Update is_read=1	Badge count drops	Works correctly	Pass