

Test Plan & Summary — University-ERP

Contributors:

Shivang Mittal (2024530)
Shashank Verma (2024522)

November 27, 2025

Test environment (setup)

- **Java:** JDK 21
- **MySQL / MariaDB:** Running locally (default root user allowed for tests).
- **Seed script:** `src/edu/univ/erp/data/seed_data.sql` (creates auth_db and erp_db and inserts sample data).

You can run the script using `scripts/load_db.sh` (if present) or:

```
mysql -u root -p < src/edu/univ/erp/data/seed_data.sql
```

- **App run:** Set `DB_PASS=erp` in the environment for local testing and run `edu.univ.erp.ui.auth`.
- **Default credentials** (used in test cases): password for each account is `erp`.
 - Admin: `admin / erp`
 - Instructor: `instructor1 / erp`
 - Student 1: `student1 / erp`
 - Student 2: `student2 / erp`

Notes: Before running tests, confirm the seed data contains at least two students and at least one instructor and courses/sections with varying capacities (some full, some with seats).

Test Plan (Acceptance & Extra tests)

Format for each test case:

- **Purpose:** Short statement
- **Steps:** Numbered interactions to perform in the UI
- **Expected result:** What should appear in the UI (messages, table entries)

Section A — Login & Roles

Test A1: Wrong password rejected

Purpose: Ensure bad credentials do not log in.

Steps:

1. Open login screen.
2. Enter username student1 and password wrongpass.
3. Click Sign in.

Expected result: An error message reading "incorrect username or password." No dashboard opens.

Test A2: Dashboard matches role

Purpose: Ensure role-based dashboard loads.

Steps: (Repeat for each role)

1. Login as student1 / erp → verify Student Dashboard.
2. Login as instructor1 / erp → verify Instructor Dashboard.
3. Login as admin / erp → verify Admin Dashboard.

Expected result: Each login shows the appropriate menus and primary view (student: timetable/registrations, instructor: my sections/gradebook, admin: user/course management).

Section B — Student workflows

Test B1: View catalog

Purpose: Verify catalog displays course code, title, credits.

Steps: Login as student1 → open Catalog screen.

Expected result: Table lists courses showing code/title/credits/capacity/instructor name.

Test B2: Register in a section with free seats

Purpose: Successful registration flow.

Steps:

1. Find a section with capacity > enrolled_count.
2. Click Register.

Expected result: Success toast/dialog, new entry appears in My Registrations and timetable view updates with the new timeslot.

Test B3: Prevent duplicate registration

Purpose: Prevent same student+section duplicate enrollment.

Steps:

1. Register for a section (if not already).
2. Try to register for the same section again.

Expected result: Blocked with a clear message such as "You are already registered in this section." No duplicate row added.

Test B4: Register into full section

Purpose: Ensure capacity checks block registration.

Steps: Attempt to register into a section where `enrolled_count == capacity`.

Expected result: Operation blocked; message "Section full." No change in My Registrations.

Test B5: View grades and download transcript

Purpose: Grade viewing and export.

Steps:

1. From Student Dashboard open Grades and Transcript.
2. Click Export Transcript (CSV/PDF).

Expected result: Grades shown per course/section; export file downloaded and includes course codes, titles, credits, grades.

Section C — Instructor workflows

Test C1: See only own sections

Purpose: Ensure instructor cannot see other instructors' sections.

Steps: Login as `instructor1` → open My Sections.

Expected result: Table lists only sections assigned to `instructor1`.

Test C2: Enter scores and save

Purpose: Save grade items (quiz/midterm/final).

Steps:

1. Open gradebook for a section.
2. Enter values for a student row and click Save.

Expected result: Inline success indicator and saved values persist when reopening the gradebook.

Test C3: Compute final grade

Purpose: Final computation uses project's grading rule.

Steps: Click Compute Final Grades (or equivalent) for the section.

Expected result: Final grade column shows computed values per rule in the report; values match manual calculation for 2-3 sample students.

Section D — Admin workflows

Test D1: Create a new student user

Purpose: Ensure admin can create Auth and ERP records.

Steps: Login as `admin` → open User Management → Create Student with username `studentX`.

Expected result: New record exists in Auth DB (users_auth) with role, and ERP students profile created. Student can login (password = erp) after creation.

Test D2: Create course and section

Purpose: Admin creates course and assigns instructor.

Steps: Admin → Course Management → Add course + add section and assign instructor1.

Expected result: Course and section appear in course list; the assigned instructor sees the section in their My Sections view.

Test D3: Maintenance toggle

Purpose: Toggle maintenance mode blocks writes.

Steps:

1. Admin toggles Maintenance ON.
2. Login as student1/instructor1 and attempt write actions: register/drop/enter grades.

Expected result: Banner shows "Maintenance mode ON"; any write attempts are blocked with a consistent message.

Additional: Toggle Maintenance OFF and verify normal operations resume.

Section E — Password & Auth separation

Test E1: Password hashes in Auth DB only

Purpose: Ensure real passwords are never stored in ERP DB.

Steps: Inspect auth_db.users_auth table to verify presence of password_hash and verify erp_db.students table does not contain password fields.

Expected result: Auth DB has hashed values; ERP DB contains no plain passwords.

Section F — Exports

Test F1: Transcript / grade CSV export

Purpose: Verify export creation and content.

Steps: Student → Export Transcript; Instructor → Export Grades CSV for a section.

Expected result: Files download and open in a spreadsheet; columns match expected schema.

Section G — Edge & negative tests

Test G1: Capacity cannot be negative

Purpose: Prevent nonsensical values.

Steps: Admin attempts to create/edit a section with capacity = -5.

Expected result: Input validation error and save blocked.

Test G2: Student cannot access another student's data

Purpose: Access control.

Steps: Attempt to view or change another student's transcript/grades/registration via UI flows.

Expected result: Access denied message; no data shown.

Section H — Data integrity

Test H1: Duplicate enrollments prevented

Purpose: Ensure UNIQUE constraint or app-level checks prevent duplicate enrollment.

Steps: Attempt to insert same (student_id, section_id) twice.

Expected result: Application blocks duplicate registration and DB rejects duplicate keys.

Section I — Security basics

Test I1: Password storage

Purpose: Verify only hashes are stored and no plaintext passwords anywhere.

Steps: Inspect DB tables.

Expected result: No plaintext passwords.

Test I2 (bonus): Account lock after 5 failed attempts

Purpose: Verify optional rate-limiting/lockout.

Steps: Attempt login with wrong password 6 times.

Expected result: UI warns or temporary lock prevents further attempts.

Section J — UI/UX & Performance

Test J1: Buttons/labels clarity

Purpose: Ensure UX is readable and errors are helpful.

Steps: Walk through common flows and evaluate messages.

Expected result: Error messages are friendly; buttons show full text.

Test J2: Catalog performance

Purpose: Ensure catalog with 100 courses loads quickly.

Steps: Ensure seed data includes 100 courses, then open Catalog.

Expected result: Catalog loads within a few seconds and UI remains responsive.

Section K — Maintenance & Backup

Test K1: Maintenance toggle behaviour

Purpose: Ensure the toggle sets the global flag and UI reflects it immediately.

Steps: Admin toggles maintenance on; other logged-in sessions observe the banner.

Expected result: Banner visible to all; writes are blocked everywhere.

Test K2 (bonus): Backup & restore

Purpose: Confirm backup captures DB state and restore reverts changes.

Steps: Trigger backup, change a record, restore backup, verify value reverted.

Expected result: After restore the changed data returns to its backed-up state.

Test Data & Accounts

- **Created by seed_data.sql:**
 - admin / erp (role=admin)
 - instructor1 / erp (role=instructor)
 - student1 / erp (role=student)
 - student2 / erp (role=student)
- **Sections:** Ensure at least one section is full and at least one has free seats.

Test Summary

Instructions: For each test case list Pass/Fail/Not run.

ID	Test Name	Account	Results
A1	Wrong password rejected	student1	Pass - Error displayed, login blocked
A2	Dashboard matches role	all	Pass - Correct dashboard loaded
B1	View catalog	student1	Pass - All course details displayed
B2	Register in free section	student1	Pass - Reg successful, in timetable
B3	Prevent duplicate reg	student1	Pass - Blocked with message
B4	Register into full section	student1	Pass - Blocked, "Section full"
B5	View grades & transcript	student1	Pass - Grades visible, export works
C1	Instructor sees own sections	instructor1	Pass - Only assigned sections shown
C2	Enter scores	instructor1	Pass - Saved and persist on reload
C3	Compute final grade	instructor1	Pass - Computed per weighting rule
D1	Create student user	admin1	Pass - User created in both DBs
D2	Create course & section	admin1	Pass - Visible to instructor
D3	Maintenance blocks writes	admin/stud	Pass - Banner shown, writes blocked
E1	Password hashing	admin (DB)	Pass - Only hashes in auth_db
F1	Transcript export	student1	Pass - CSV/PDF generated correctly
G1	Capacity validation	admin1	Pass - Negative capacity rejected
H1	Duplicate enrollment	student1	Pass - Blocked by constraint
I1	Password storage check	admin (DB)	Pass - No passwords in erp_db

ID	Test Name	Account	Results
J1	UI/UX checks	any	Pass - Buttons clear, errors friendly
J2	Catalog performance	student1	Pass - Loads in <3 seconds
K1	Maintenance behavior	admin1	Pass - Toggle updates banner immed.
X1	Score validation (0-100)	instructor1	Pass - Invalid scores rejected
X2	No duplicate grade entries	instructor1	Pass - Unique constraint works
X3	Grade CSV import	instructor1	Pass - Imported & grade-book updated

Table 1: Test Execution Summary

Extra Tests Added (Passed)

Score validation

Description: Score values must be integers 0–100.

Results: Manual tests performed by instructor1; invalid inputs show validation errors; database rejects invalid values.

No duplicate grade entries

Description: Uniqueness enforced per enrollment+component.

Results: Attempted duplicate insertions returned DB errors; UI prevented duplicate submissions.

Grade CSV import

Description: Bulk grade import via CSV tested. The importer validates rows, skips invalid entries, and persists valid grades.

JUnit 5 Tests

To run JUnit 5 tests, add the JUnit standalone console JAR to the repo (recommended location: tools/ or lib/) and run the launcher.

1. Place the JUnit console launcher JAR in tools/.
2. Compile the application and test classes:

```
# compile app classes
mkdir -p out
javac -cp "lib/*" -d out $(find src -name "*.java")

# compile test classes (assume tests are under 'test/')
mkdir -p out-test
javac -cp "lib/*:out" -d out-test $(find test -name "*.java")
```

3. Run the JUnit console launcher:

```
java -jar tools/junit-platform-console-standalone-1.9.3.jar --class-path  
out:out-test:lib/* --scan-classpath
```