

SOFTWARE TESTING & VALIDATION COURSE

Final Project Assignment

Project Title: Exam Security System (Identity Verification + Seating Plan + Violation Logging)

1. Purpose of the Final

This final project evaluates students' ability to build a working software system and demonstrate testing & validation practices.

Unlike the midterm (analysis-focused), the final emphasizes end-to-end implementation, integration of a simple computer-vision / ML component, and producing test cases and unit tests for critical logic.

2. Project Scenario

You will develop a web-based system named **Exam Security System**.

The system supports exam-day identity verification, seating compliance, and incident/violation recording.

Core Actors: Student, Proctor (Invigilator), Exam Coordinator (Admin)

Core Goals:

- Verify that a student entering the exam room matches the registered student.
- Check that the student sits in the correct seat according to the seating plan.
- Record violations and generate reports.

3. ML / Computer Vision Component (Required, Simple)

A simple image-recognition component must be integrated. Allowed implementations (choose one):

- Face verification (known vs unknown / match vs no match) using a library-based approach.
- ID photo vs live photo comparison (basic similarity decision via embeddings).

Notes:

- This is not an AI competition.
- Training a deep model from scratch is not required.
- Grading focuses on integration, validation, and workflow correctness.

4. Working Format

- Groups of 3–4 students.
- All outputs must be submitted through a Bitbucket repository.

5. Required Deliverables (Summary)

- A) Requirements & Rules
- B) System Diagrams (Use Case, ERD, Sequence) + Bonus Activity (optional)
- C) JIRA Planning (structure decided by the team)
- D) Implementation (features listed below)
- E) Database (schema + dummy data)
- F) Testing & Validation (test cases document + unit tests)

5D. Minimum Required Implementation Features

- Authentication & role-based access (Proctor/Admin)
- Exam creation (exam, room, date/time)
- Student roster import or entry (basic)
- Seating plan creation (rows/columns or seat codes)
- Check-in workflow: capture/upload photo, ML verification decision, seat compliance check, record result and timestamp
- Violation logging (reason, notes, evidence image optional)
- Basic reporting (check-ins, mismatches, violations)

5F. Testing & Validation Requirements

Test Cases Document: Functional + Negative + Edge cases (e.g., missing image, duplicate check-in, multiple faces).

Recommended format: Test ID, Precondition, Steps, Input, Expected Result.

Unit Tests (mandatory): At minimum test seating compliance logic, validation rules (required fields, role permissions, duplicate check-in), and the ML integration wrapper/service logic using mock inputs.

Note: You are not expected to unit-test the ML model itself; you must unit-test the wrapper/service your system calls.

6. Submission Repository Structure

analysis/ (requirements + business rules)

diagrams/ (usecase, erd, sequence, activity optional)

jira/ (sprint board screenshot)

database/ (schema + dummy data)

src/ (application code)

tests/ (unit tests)

test-docs/ (test cases document)

README.md (run + tests + demo scenario)

7. Evaluation (Base Score: 100 Points)

Requirements & Rules (15), Diagrams (15), JIRA & Planning (15), Implementation (25), Database (10), Testing & Validation (20) = 100.

Bonus: Activity Diagram +10 (not included in base 100).

8. Academic Integrity & Use of ChatGPT

Students MAY use ChatGPT to support understanding concepts. Copy-paste answers are strictly forbidden.

Directly copying Epic/Story/Task names from AI tools is NOT allowed. All AI-assisted content must be reviewed and personalized.

The instructor may evaluate originality and coherence.

Instructor Evaluation Checklist (Final Project)

Area	Check (✓/X)	Notes
Requirements		Functional + non-functional; business rules & validations documented
Diagrams		Use Case, ERD, Sequence present and consistent with requirements
JIRA		Coherent backlog; tasks assigned; realistic sprint plan; evidence screenshot
Implementation - Auth/Roles		Role-based access works; permissions enforced
Implementation - Exam/Room/Seating		Exam entities created; seating plan represented correctly
Implementation - Check-in Flow		Photo capture/upload; ML decision; seat compliance check; saved with photo
Implementation - Violations		Violations logged with reason/notes (evidence optional)
Implementation - Reporting		Basic reports for check-ins, mismatches, violations
Database		Schema + dummy data; constraints support business rules
Testing - Test Cases Doc		Functional, negative, edge cases; expected results are clear
Testing - Unit Tests		Critical logic tested; ML wrapper/service tested via mocks
Academic Integrity		No copy-paste; content is coherent; team can explain choices
Bonus		Activity diagram provided (optional)