Mandatory Vulnerability Categories
Each team-built or reused system must deliberately incorporate a defined set of security vulnerabilities across multiple categories. These vulnerabilities will serve as the foundation for assessment, exploitation, and subsequent patching activities during the project lifecycle.
1. Web Vulnerabilities
Teams must implement at least four (4) distinct instances from this category.
Examples include:
● Cross-Site Scripting (XSS)
● Cross-Site Request Forgery (CSRF)
● SQL Injection
● Cookie Tampering
● Broken Authentication

2. Server Security
Implement at least two (2) distinct instances from this category.
Examples include:
Only (2)
● Misconfigured HTTPS certificates
● Weak password storage or hashing
● Insecure or missing firewall rules

5. Data Exposure
Include at least one (1) instance related to improper data protection.
Examples include:
Only (1)
● Sensitive data stored or transmitted without encryption
● Misconfigured access permissions exposing confidential files

Must-have pages / UI

● Auth pages: login, logout, password reset, 2FA stub (optional).
● Student dashboard: current registrations, timetable, holds.
● Landing / Catalog: course list, search, filters
    Or
    Course detail: description, prerequisites, capacity, schedule.

Roles
● Student (register, view).
● Instructor (view rosters, grade entry stub).
● Registrar/Admin (manage courses, override).
● System/Integration (API clients).

Example API endpoints (JSON REST)

- POST /api/v1/auth/login -> {email, password} -> sets cookie sid (HttpOnly, Secure)
- POST /api/v1/auth/token -> JWT for API clients
- GET /api/v1/courses -> list / filters
- GET /api/v1/courses/{id} -> details
- POST /api/v1/enrollments -> {course_id, student_id} -> enrollment transaction
- DELETE /api/v1/enrollments/{id} -> drop
- POST /api/v1/admin/courses -> create course (admin only)
- GET /api/v1/audit?user={id} -> audit logs

Data Models:

- User: id, email, password_hash, role, student_id, last_login
- Course: id, code, title, capacity, prereq_ids[], schedule, instructor_id
- Enrollment: id, course_id, user_id, status, timestamp
- AuditRecord: id, actor_id, action, target, timestamp, details

Regression & security tests (must be provided)

- Functional tests: register/login, enroll/drop, admin create course (pytest / JUnit).
- Security tests: automated SQLi attempt, XSS reflection checks, CSRF attack simulation, cookie scope assertions, expiry checks for tokens.
- CI pipeline: run tests on branch & run “baseline” security checks.

Category	Description	Example Injection Points	Severity Guidance
Web Vulnerabilities	XSS, CSRF, SQLi, Cookie Tampering, Broken Auth	Search box reflection, comment/forum input, hidden form fields, cookie scope misconfig	Low–High
Server Security	Misconfigured HTTPS certs, weak password hashing, unsafe firewall rules, open admin ports	Self-signed certs, plaintext passwords, no rate limits	Medium–High
Data Exposure	Overly broad queries or predictable URLs	Export endpoints, file key enumeration	Medium–High