Assignment Secure Coding

Overview

Module: ELE201 – Secure Coding

Continuous Assessment Weightage:

CA3 (**25%**) – Individual Project

CA4 (**20%**) – Documentation (**10%**) + Viva (**10%**)

Project and Documentation **Submission Deadline: November 10, 2025**

*[This Assignment contains **Eight** printed pages]*

Assignment 2025 ELE201

🎯 Objective

To design and develop a secure web application (students may choose e-commerce, educational, healthcare, or booking system themes) that adheres to secure coding principles. This project will require you to demonstrate your understanding of security vulnerabilities, input validation, output sanitization, error handling, and more, specifically tailored to the educational context.

Introduction

You are tasked with designing and developing any web application. The primary goal of this assignment is to demonstrate your understanding and application of secure coding principles to ensure the confidentiality, integrity, and availability of the users and platform resources.

Assignment 2025 ELE201

1. Assignment Requirements (Rubrics)

(80 marks = 25%)

Your assignment should include the following features, with a strong emphasis on security:

No.	Component	Description/ Security Requirements	Excellent	Fair	Low	Total Marks
1	**User Authentication & Authorization**	Implement bcrypt password hashing; secure login; role-based access control (RBAC).	**13–15** marks → Fully functional authentication and RBAC with secure password storage, least privilege applied.	**7–12** marks → Authentication works but lacks full RBAC or partial security flaws.	**0–6** marks → Weak or missing authenticatio n; plaintext passwords or no access control.	**15**
2	**Data Encryption & Protection**	Encrypt sensitive data (e.g student info, grades) at rest and in transit; apply access restrictions.	**13–15** marks → Proper encryption (HTTPS + AES/bcrypt), access control enforced, data confidentiality maintained.	**7–12** marks → Partial encryption (e.g., HTTPS only) or weak key handling.	**0–6** marks → No encryption; sensitive data visible or unprotected.	**15**
3	**Input Validation & Output Sanitization**	Validate all input (client & server); sanitize output to prevent SQLi/XSS.	**13–15** marks → Comprehensive client/server validation; effective sanitization; passes attack tests.	**7–12** marks → Some validation present but inconsistent or missing server-side enforcement.	**0–6** marks → Vulnerable to SQLi/XSS; no input checks or encoding.	**15**

4	**Session Management**	Use token-based or cookie sessions; apply session expiration and secure cookie flags.	**9–10** marks → Sessions secured with HttpOnly, Secure, SameSite; proper logout and expiry.	**5–8** marks → Sessions functional but missing flags or inconsistent timeout.	**0–4** marks → Persistent sessions; hijacking or fixation possible.	**10**
5	**Error Handling & Logging**	Implement try-catch handling; use Winston or similar for secure logging.	**9–10** marks → Logs well-structured, errors handled gracefully; no sensitive info leaked.	**5–8** marks → Logging works but lacks detail or error messages are too vague.	**0–4** marks → Crashes or exposes stack traces; no structured logging.	**10**
6	**Secure Communication**	Use HTTPS; secure API endpoints with authentication/authorization.	**9–10** marks → HTTPS enforced throughout; secure APIs with token or RBAC checks.	**5–8** marks → HTTPS used but some endpoints insecure or misconfigured.	**0–4** marks → Plain HTTP or open APIs; credentials transmitted insecurely.	**10**
7	**Security Testing & Auditing**	Conduct penetration testing, code reviews; comply with security standards (GDPR, HIPAA, FERPA).	**5** marks → Thorough testing with vulnerability report and fixes; documentation provided.	**3–4** marks → Basic tests done but incomplete documentation or limited fixes.	**0–2** marks → No testing or missing compliance evidence.	**5**

2. Document Requirements (20 marks = 10%)

Each student must include a comprehensive documentation file (PDF) explaining how security features were implemented and verified.
The documentation should contain the following sections:

A. Application Overview (5 Marks)

- Briefly describe the purpose of your web application (e.g., educational portal, e-commerce system, or student management platform).
- Specify the main features and user roles (Admin, Instructor, Student, etc.).
- Mention the technologies and frameworks used (e.g., React, Node.js, Express, Django, MySQL, MongoDB).

B. Security Implementation Details (8 marks)

For each of the major security components, explain how they were implemented in your project:

1. User Authentication and Authorization – bcrypt, session tokens, RBAC.
2. Data Encryption and Protection – encryption libraries, HTTPS setup.
3. Input Validation and Output Sanitization – regex, sanitizers.
4. Session Management – token expiration, cookie security.
5. Error Handling and Logging – Winston or equivalent.
6. Secure Communication – encrypted using HTTPS protocol to protect data confidentiality and integrity.
7. Security Testing and Auditing – tools used, vulnerabilities identified and fixed.

Include annotated screenshots for each component (e.g., code snippets, login form, HTTPS configuration, security test results).
Clearly label each screenshot with a title and brief description (e.g., Figure 2: Session Timeout Mechanism Implemented in Express.js).

C. Component Breakdown (4 marks)

- List all major components or modules added to the project (e.g., Authentication module, Encryption service, Validation middleware, API routes, Admin dashboard).
- For each component, provide:
    - A short description of its purpose.
    - The security measures implemented within it.
    - References to relevant screenshots or code sections in the document.

Example Format:

Component Name	Description	Security Feature Implemented	Screenshot Reference
Authentication Module	Handles login and password verification	bcrypt hashing, token-based login	Fig 1.
Encryption Service	Protects stored student data	AES encryption, environment variables	Fig 3.

D. Challenges and Solutions (2 marks)

- Describe any technical difficulties faced (e.g., handling token expiration, configuring HTTPS, database encryption).
- Explain how you identified, troubleshot, and resolved these issues.

E. Security Summary (1 mark)

- Provide a short section summarizing the overall security design of your project.
- Discuss how your implementation ensures:
    - Confidentiality: Data protection through encryption.
    - Integrity: Prevention of unauthorized modification.
    - Availability: Reliable access and error handling.

F. Repository or Drive Link (1 mark)

- At the end of your document, clearly include your:
    - Google Drive submission link (if uploaded as a zip file).
    - GitLab repository URL (if project is pushed in GitLab).
- Ensure that access permissions are correctly set (i.e., “Anyone with the link can view”).

G. File Naming Convention

When submitting your .zip file or documentation, use the following format:

ELE201_[YourFullName]_[StudentID].zip
ELE201_[YourFullName]_[StudentID].pdf

Example:
ELE201_JigmeDema_12200005.zip
ELE201_JigmeDema_12200005.pdf

H. Evaluation Reminder

🔒 Important Note:
Your project will be evaluated based on both functionality and the effectiveness of your security implementations.

Marks will be awarded for:

- Quality of secure coding practices demonstrated.
- Correct and thorough documentation with labeled screenshots and component breakdowns.
- Clear explanation of challenges and resolutions.

Incomplete submissions or missing documentation will result in mark deductions.

— "Code with care; deploy with confidence." —