BCS2213 OBJECT-ORIENTED PROGRAMMING
Group Assignment (*SEPT 2025*)

Course : Degree
Commence Date : Week 11 – 8/12/2025 (Monday)
Deadline Date : Week 14 – 29/12/2025 (Friday before 6pm)
: **Project Demo on Thursday**

Unit Controller / Examiner : Vinod Sarna
Contact Number : 03-88005028
E-mail : vinod@unimy.edu.my

Assignment (110 marks) 30%

STUDENT NAME: ……..……..
STUDENT ID: ……..……..…….
PROGRAMME/COURSE: ……..
DATE: …….
EMAIL: ……..……

OBJECTIVES

- To test students' competencies and capabilities in writing a Java program.
- To assess students' skills in creating, editing, compiling and running a Java program.

SPECIFIC INFORMATION

- This group assignment contributes **30%** to the coursework.
- This project is group work and has a maximum of **TWO (2)** students.

SUBMISSION INFORMATION

- The report must be printed out using Arial font style with size of 12pt and 1.15 line spacing.
- While codes can be in `Courier New` font style.
- You must submit the report & zipped files of your project in MS Teams.

Assignment Brief: Student Enrollment Management System (SEMS)

1. Project Overview

A comprehensive **Student Enrollment Management System** to be developed by Java programming degree students. This full-stack web application will manage student records, course enrollment, and academic information using Java EE technologies. The project **must** be completed within **4 weeks.**

2. Technical Specifications

Mandatory Technologies:

- **IDE:** NetBeans (latest version)
- **Backend:** Java EE (JPA, Servlets, EJB optional)
- **Frontend:** JSP with JSTL, HTML5, CSS3, JavaScript
- **Database:** JDBC with SQL (MySQL recommended)
- **Application Server:** GlassFish Server
- **Build Tool:** Maven (integrated with NetBeans)

3. Core Requirements

A. Database Schema (SQL)

- **Students Table:** student_id, first_name, last_name, email, dob, enrollment_date
- **Courses Table:** course_id, course_code, course_name, credits, department
- **Enrollments Table:** enrollment_id, student_id, course_id, enrollment_date, grade
- **Users Table:** user_id, username, password_hash, role (admin/student)

B. Backend Functionality (Java)

1. **Data Access Layer:**
    - JDBC connection management with connection pooling
    - CRUD operations for all entities

      C - Create
      R - Read
      U - Update
      D - Delete

    - Prepared statements to prevent SQL injection
2. **Business Logic Layer:**
    - Student registration and validation
    - Course enrollment logic (prerequisite checks, capacity)
    - Grade management
    - User authentication and authorization

3. **Service Layer:**
    - RESTful web services or servlets for AJAX operations
    - Session management
    - Input validation and error handling

C. Frontend Interface (JSP/HTML/JS)

1. **Authentication Pages:**
    - Login/Logout functionality
    - Session timeout handling
2. **Student Portal:**
    - Dashboard with enrolled courses
    - Course registration interface
    - Grade view
    - Profile management
3. **Admin Portal:**
    - Student management (add/edit/delete)
    - Course management
    - Enrollment oversight
    - Report generation

4. Project Timeline (4 Weeks)

Week 1: Foundation & Database

- **Days 1-2:** Project setup in NetBeans, GlassFish configuration
- **Days 3-4:** Database design and implementation (SQL scripts)
- **Days 5-7:** JDBC layer, connection pooling, basic DAO classes

Week 2: Backend Development

- **Days 8-10:** Business logic implementation, servlet development
- **Days 11-12:** User authentication, session management
- **Days 13-14:** Service layer completion, exception handling

Week 3: Frontend & Integration

- **Days 15-16:** JSP pages development with Bootstrap/CSS
- **Days 17-18:** AJAX integration, form validation
- **Days 19-20:** Testing, debugging, security implementation
- **Day 21:** Final deployment, documentation

Week 4: Report & Demonstration.

5. Deliverables

A. Source Code:

- Complete NetBeans project folder
- SQL database creation scripts
- Configuration files (web.xml, context.xml, persistence.xml if using JPA)

B. Documentation (REPORT):

Your report **MUST** contain:

- Cover page, table of contents, page numbers, header and footer.
- A brief explanation (*on methods, functions, classes and techniques used*) of each program codes.
- Source code with the **line numbers** and **Javadoc comments**.
- The outputs. (*Screenshots*)
- References *(complete URL links, books, etc.)*

Your report **MUST** include:

- System design document
- Database schema diagram
- User manual
- Installation guide for GlassFish deployment

C. Deployment:

- WAR file ready for GlassFish deployment
- Database backup/restore script

6. Technical Requirements Details

Database Connectivity:

java

```
// Example structure expected
public class DatabaseConnection {
    private static DataSource dataSource;
    // Implement connection pooling
}
```

Security Requirements:

- Password hashing (SHA-256 or BCrypt)
- SQL injection prevention
- XSS protection
- Session fixation protection
- Role-based access control

Code Quality:

- Proper Java naming conventions
- Comprehensive commenting
- Exception handling throughout
- Logging implementation
- Unit tests for critical methods

7. Assessment Criteria

- **Functionality (40%):** All features working correctly
- **Code Quality (25%):** Clean, well-structured, documented code
- **Database Design (15%):** Normalized, efficient SQL
- **UI/UX (10%):** Intuitive, responsive interface
- **Security (10%):** Proper authentication and data protection

8. Getting Started Instructions

1. **Environment Setup:**
    - NetBeans
    - GlassFish Server 5.x or 6.x
    - MySQL Server
    - JDBC connection pool in GlassFish
2. **Project Structure:**

```
SEMSProject/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── dao/
│   │   │   ├── model/
│   │   │   ├── service/
│   │   │   └── util/
│   │   └── webapp/
│   │       ├── WEB-INF/
│   │       ├── css/
│   │       ├── js/
│   │       └── views/
└── sql/
    └── database_creation.sql
```

9. Grading

- Project 100 marks
- Report 10 marks

Project Grading Rubric: Student Enrollment Management System (SEMS)

Total Points: 100

A. FUNCTIONALITY (40 points)

1. Core Features (25 points)

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (1-2)	Score
User Authentication	Login/logout works perfectly. Sessions managed securely. Password hashing implemented.	Minor issues with session management or password security.	Basic login works but security features missing.	Authentication not working or completely insecure.	_____/5
CRUD Operations	All CRUD operations work flawlessly for students, courses, enrollments.	Most operations work with minor bugs.	Basic operations work but some features missing.	CRUD operations incomplete or buggy.	_____/5
Enrollment System	Full enrollment workflow with validation, capacity checks, conflict detection.	Enrollment works but missing some validation.	Basic enrollment without validation.	Enrollment system not functional.	_____/5
Data Display	All data displays correctly with proper formatting, sorting, filtering.	Data displays with minor formatting issues.	Basic display without sorting/filtering.	Data display incomplete or broken.	_____/5
Navigation & Flow	Intuitive navigation between all pages. Clear user flow.	Minor navigation issues.	Navigation works but not intuitive.	Navigation confusing or broken.	_____/5

2. Database Operations (15 points)

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (1-2)	Score
JDBC Implementation	Efficient connection pooling, prepared statements, proper resource management.	Connection pooling implemented with minor issues.	Basic JDBC without connection pooling.	JDBC implementation flawed or incomplete.	_____/5
SQL Design	Well-normalized schema, appropriate indexes, foreign key constraints.	Good schema design with minor normalization issues.	Basic schema with some design flaws.	Poorly designed schema or missing relationships.	_____/5
Data Integrity	All constraints enforced, transactions where needed, referential integrity.	Most constraints enforced.	Basic constraints only.	No data integrity measures.	_____/5

B. CODE QUALITY & ARCHITECTURE (25 points)

1. Java Code Structure (15 points)

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (1-2)	Score
Layer Separation	Clear separation of DAO, Service, Controller layers. MVC pattern followed.	Good separation with minor mixing of concerns.	Some separation but layers not clearly defined.	No architectural separation (spaghetti code).	_____/5
Code Organization	Clean package structure, logical class organization, proper naming conventions.	Well-organized with minor issues.	Basic organization but inconsistent.	Poor organization, confusing structure.	_____/5
Error Handling	Comprehensive exception handling, user-friendly error messages, logging.	Good error handling with minor gaps.	Basic exception handling only.	Minimal or no error handling.	_____/5

2. Maintainability & Standards (10 points)

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)	Score
Java Conventions	Perfect adherence to Java naming, commenting, and style conventions.	Good adherence with minor deviations.	Basic conventions followed.	Poor or inconsistent conventions.	_____/4
Code Comments	Comprehensive Javadoc for all classes/methods. Inline comments for complex logic.	Good documentation with minor gaps.	Basic comments only.	Minimal or no comments.	_____/3
Reusability	Code designed for reusability, avoid duplication, utility classes where appropriate.	Good reusability with some duplication.	Basic reusability.	Significant code duplication.	_____/3

C. DATABASE IMPLEMENTATION (15 points)

Criteria	Excellent (5)	Good (4)	Satisfactory (3)	Needs Improvement (1-2)	Score
SQL Scripts	Complete, well-documented SQL scripts. Includes both schema and sample data.	Good scripts with minor documentation issues.	Basic scripts only.	Incomplete or missing scripts.	_____/5
Query Efficiency	Optimized queries, appropriate joins, efficient data retrieval.	Efficient queries with minor optimization issues.	Basic queries that work.	Inefficient or poorly written queries.	_____/5
Data Validation	Database-level constraints, triggers if needed, proper data types.	Good validation at database level.	Basic data types and constraints.	Minimal database validation.	_____/5

D. USER INTERFACE & USER EXPERIENCE (10 points)

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)	Score
Design & Layout	Professional, consistent, responsive design. Good use of CSS/Bootstrap.	Good design with minor inconsistencies.	Functional but basic design.	Poor or confusing layout.	_____/4
Usability	Intuitive interface, clear labels, helpful messages, easy navigation.	User-friendly with minor issues.	Basic usability.	Difficult to use or confusing.	_____/3
Form Validation	Comprehensive client-side and server-side validation with helpful feedback.	Good validation with minor gaps.	Basic validation only.	Minimal or no validation.	_____/3

E. SECURITY & DEPLOYMENT (10 points)

Criteria	Excellent (4)	Good (3)	Satisfactory (2)	Needs Improvement (1)	Score
Security Measures	Password hashing, SQL injection prevention, XSS protection, session security.	Good security with minor vulnerabilities.	Basic security measures only.	Major security flaws or none.	_____/4
GlassFish Deployment	Successfully deployed on GlassFish, proper configuration, working WAR file.	Deployed with minor configuration issues.	Basic deployment but issues.	Not deployable or major issues.	_____/3
Error Pages & Handling	Custom error pages, secure error messages, proper HTTP status codes.	Good error handling.	Basic error pages.	Default error messages or broken.	_____/3