EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

Practice Assignment – Quarter 4, 2024-2025
Course Name: Backend Development
Course Code: CSW 306
Student's Full Name:
Student ID:

Practice Assignment 3

To help you build a project using ASP.NET Core API and develop all the features needed for managing a library management system.

LIBRARY MANAGEMENT SYSTEM
**Create a Web ASP.NET Core API application project to work through exercise 1 to 7.*
Learning Objectives:

- Understand the structure of Controllers in ASP.NET Core.
- Use basic routing and HTTP verbs (GET, POST, etc.).
- Return JSON-formatted responses.
- Practice designing RESTful APIs without a database.

***Exercise* 1: Create the Book model**

Requirements:

- Create a class Book inside the Models/ folder.
- Include the following properties:
  - int Id
  - string Title
  - string Author
  - int Year
  - string Genre

Note: This model will be used to create a hardcoded list of books.

***Exercise* 2: Create the BooksController**

Requirements:

- Create a controller named BooksController inside the Controllers/ folder.
- Add the attributes [ApiController] and [Route("api/[controller]")].
- Inside the controller, initialize a hardcoded List<Book> with at least 3 sample books.

Example book:

new Book { Id = 1, Title = "Clean Code", Author = "Robert C. Martin", Year = 2008, Genre = "Programming" }

***Exercise* 3: Implement GET /api/books**

Requirements:

- Create a method GetAllBooks() with the [HttpGet] attribute.
- Return the full list of books in JSON format.

Expected: Accessing GET /api/books should return the sample book list.

***Exercise* 4: Implement GET /api/books/{id}**

Requirements:

- Create a method GetBookById(int id) with [HttpGet("{id}").
- Search for the book by id.
  - If found → return it.
  - If not → return 404 NotFound.

Example: GET /api/books/2 should return the book with Id = 2.

***Exercise* 5: Implement POST /api/books**

Requirements:

- Create a method AddBook(Book newBook) with [HttpPost].
- Use [FromBody] to accept book data in JSON format from the request body.
- Add the new book to the list.
- Return the newly added book in the response.

Note: No database is used. The list only exists during runtime.

Submission Requirements:

- A working ASP.NET Core Web API project.
- All endpoints must be testable using Swagger or Postman:
  - GET /api/books
  - GET /api/books/{id}
  - POST /api/books
- At least 3 hardcoded books + ability to add new books via API.
- Clean code, correct use of routes and HTTP verbs, include comments if needed.

Exercise 6: Implement PUT /api/books/{id} (Update **Book)**

Requirements:

- Create a method UpdateBook(int id, Book updatedBook) with [HttpPut("{id}")].
- Search for the book by id.
    - If found → update its properties with values from updatedBook.
    - If not → return 404 NotFound.
- Return the updated book in the response.

Example: PUT /api/books/1 should update the book with Id = 1.

Exercise 7: Implement DELETE /api/books/{id} (Delete Book)

Requirements:

- Create a method DeleteBook(int id) with [HttpDelete("{id}")].
- Search for the book by id.
    - If found → remove it from the list.
    - If not → return 404 NotFound.
- Return a confirmation message (or 204 No Content).

Example: DELETE /api/books/3 should remove the book with Id = 3.