University of Westminster

Department of Computer Science

**This is provisional and subject to revisions

5COSC022W	Client-Server Architectures – Coursework (2023/24)
Module leader	Hamed Hamzeh
Unit	Coursework
Weighting:	60%
Qualifying mark	30%
Description	REST API design, development and implementation.
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO1 Gain a thorough understanding of RESTful principles and their application in API design. - LO2 Acquire familiarity with the JAX-RS framework as a tool for building RESTful APIs in Java.
Handed Out:	February 2025
Due Date	28th April 2025, 13:00
Expected deliverables	**A zip file containing the developed project** **Video demonstration** **Report in a pdf format**
Method of Submission:	Electronic submission on Blackboard via a provided link close to the submission time.
Type of Feedback and Due Date:	Written feedback within 15 working days.
BCS CRITERIA MEETING IN THIS ASSIGNMENT	2.1.1 Knowledge and understanding of facts, concepts, principles & theories 2.1.2 Use of such knowledge in modelling and design 2.1.3 Problem solving strategies 2.2.1 Specify, design or construct computer-based systems 2.2.4 Deploy tools effectively 2.3.2 Development of general transferable skills 3.1.1 Deploy systems to meet business goals 4.1.1 Knowledge and understanding of scientific and engineering principles 4.1.3 Knowledge and understanding of computational modelling

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website: **http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Learning Goals:

- Define key principles of REST architecture.
- Differentiate between RESTful and non-RESTful APIs.
- Recognize the importance of resource-based interactions.
- Understand the role of JAX-RS in Java-based API development.
- Explore JAX-RS annotations for resource mapping and HTTP method handling.
- Implement basic resource classes using JAX-RS.

Introduction

This coursework is designed to provide you with practical experience in designing and implementing RESTful web services using the Java API for RESTful Web Services (JAX-RS). You will build a comprehensive "Bookstore" application API, gaining hands-on experience with core RESTful principles, JAX-RS features, and common API design patterns.

This project simulates a real-world scenario where you need to create a robust and scalable backend API to manage various entities like books, authors, customers, shopping carts, and orders. You'll learn how to structure your application using resource classes, handle different HTTP methods, process requests and responses, and implement proper error handling.

Coursework Objectives

Upon successful completion of this coursework, you will be able to:

- Understand and apply the fundamental principles of RESTful architecture.
- Design and implement RESTful APIs using the JAX-RS specification.
- Structure a JAX-RS application using resource classes and sub-resources.
- Handle various HTTP methods (GET, POST, PUT, DELETE) appropriately.
- Work with different data formats, specifically JSON, for request and response bodies.
- Implement data validation and exception handling using ExceptionMapper.
- Understand and implement basic API testing using Postman.
- Model basic e-commerce functionalities such as shopping cart and order management.

Coursework Specifications

1. Application Domain:

You will be building a RESTful API for a "Bookstore" application. This API will allow clients to interact with the following entities:

- **Books:** Products with attributes like title, author, ISBN, publication year, price, and stock quantity.
- **Authors:** Creators of books, with attributes like name and biography.
- **Customers:** Users of the bookstore, with attributes like name, email, and a simple password for this coursework.
- **Carts:** Shopping carts associated with customers, allowing them to add, remove, and update the quantity of books.
- **Orders:** Represent completed purchases, created from a customer's cart.

2. Technology Stack:

- **JAX-RS:** You **must** use the JAX-RS API for implementing your RESTful services. You can choose a specific JAX-RS implementation like Jersey or RESTEasy.
- **JSON:** You **must** use JSON as the data format for all request and response bodies.
- **Postman:** You **must** use Postman for testing and demonstrating your API.
- **In-Memory Data Storage:** You **must** use simple in-memory data structures (e.g., ArrayList, HashMap) to store your application data. **Do not use any external databases or persistence frameworks.**

3. Resource Classes and Endpoints:

Your API must include the following resource classes and endpoints:

- **BookResource (/books)**
  - POST /books
  - GET /books
  - GET /books/{id}
  - PUT /books/{id}
  - DELETE /books/{id}
- **AuthorResource (/authors)**
  - POST /authors
  - GET /authors
  - GET /authors/{id}
  - PUT /authors/{id}
  - DELETE /authors/{id}
  - GET /authors/{id}/books
- **CustomerResource (/customers)**
  - POST /customers
  - GET /customers
  - GET /customers/{id}
  - PUT /customers/{id}
  - DELETE /customers/{id}
- **CartResource (/customers/{customerId}/cart)**
  - POST /customers/{customerId}/cart/items
  - GET /customers/{customerId}/cart
  - PUT /customers/{customerId}/cart/items/{bookId}
  - DELETE /customers/{customerId}/cart/items/{bookId}
- **OrderResource (/customers/{customerId}/orders)**
  - POST /customers/{customerId}/orders

- GET /customers/{customerId}/orders
- GET /customers/{customerId}/orders/{orderId}

4. Data Models:

You need to create corresponding Java classes to represent the Book, Author, Customer entities. These classes should have appropriate attributes, data types, and constructors.

5. Exception Handling:

You must implement custom exception classes and use ExceptionMapper to handle the following scenarios gracefully:

- BookNotFoundException
- AuthorNotFoundException
- CustomerNotFoundException
- InvalidInputException
- OutOfStockException
- CartNotFoundException

Map these exceptions to appropriate HTTP status codes (e.g., 404 Not Found, 400 Bad Request) and provide informative error messages in JSON format.

6. Code Structure:

- Organize your code into the specified resource classes.
- Follow best practices for JAX-RS development.
- Use meaningful names for variables, methods, and classes.

7. Report Structure:

This report requires you to document the test cases you have designed and executed for your RESTful Bookstore API. The report will consist primarily of tables that clearly outline each endpoint, its expected behavior, and the results of your tests.

Your report should have the following structure:

1. Introduction (Brief - Approximately 1/4 page)

- Briefly introduce the purpose of the report, which is to document the test cases for your Bookstore API.
- Mention the technologies used to build the API (JAX-RS, JSON) and the tool used for testing (Postman).
- State that the report focuses on presenting test cases in a tabular format.

2. API Endpoint Test Case Tables (Main Content of the Report)

- For **each endpoint** in your API, create a separate table to document its test cases.
- Organize the tables by resource class (e.g., a section for BookResource test cases, a section for AuthorResource test cases, etc.).
- Use the following table structure for each endpoint:

Sample Test Case Table: POST /books Endpoint:

Endpoint	Description	HTTP Method	Request Body (JSON)	Expected HTTP Status Code	Expected Response Body (JSON)	Actual Result (Pass/Fail)
POST /books	Create a book with valid data.	POST	{ "title": "The Lord of the Rings", "authorId": 1, "isbn": "978-0-618-05326-7", "publicationYear": 1954, "price": 20.99, "stock": 100 }	201 Created	{ "id": <generated_id>, "title": "The Lord of the Rings", "authorId": 1, "isbn": "978-0-618-05326-7", "publicationYear": 1954, "price": 20.99, "stock": 100 }	
POST /books	Create a book with invalid author ID (not found).	POST	{ "title": "The Lord of the Rings", "authorId": 999, "isbn": "978-0-618-05326-7", "publicationYear": 1954, "price": 20.99, "stock": 100 }	404 Not Found	{ "error": "Author Not Found", "message": "Author with ID 999 does not exist." }	
POST /books	Create a book with invalid publication year (future).	POST	{ "title": "The Lord of the Rings", "authorId": 1, "isbn": "978-0-618-05326-7", "publicationYear": 2025, "price": 20.99, "stock": 100 }	400 Bad Request	{ "error": "Invalid Input", "message": "Publication year cannot be in the future." }	
GET /books	Get all books.	GET		200 OK	[{ "id": 1, "title": "Book 1", ... }, { "id": 2, "title": "Book 2", ... }, ...]	
GET /books/{id}	Get book by valid ID.	GET		200 OK	{ "id": 1, "title": "Book 1", ... }	
GET /books/{id}	Get book by invalid ID.	GET		404 Not Found	{ "error": "Book Not Found", "message": "Book with ID 999 does not exist." }	

Explanation of Columns:

- **Endpoint: The specific API endpoint being tested (e.g., POST /books, GET /books/{id}).**
- **Description:** A brief description of what the test case is doing.
- **HTTP Method:** The HTTP method used (e.g., POST, GET, PUT, DELETE).
- **Request Body (JSON):** The JSON payload sent in the request body (if applicable).
- **Expected HTTP Status Code:** The HTTP status code you expect the API to return.
- **Expected Response Body (JSON):** The expected JSON payload in the response body.
- **Actual Result (Pass/Fail):** Indicate whether the test case passed or failed during testing.

Important Reminders:

- **Comprehensive Test Cases:** Ensure that your test cases cover a wide range of scenarios for each endpoint, including valid inputs, invalid inputs, edge cases, and error conditions.
- **Postman for Execution:** You must use Postman to execute your test cases and record the results in the tables.
- **Focus on Documentation:** The primary goal of this report is to clearly document your test cases and their results.

Example Test Case Tables for Other Endpoints:

You would create similar tables for other endpoints, such as:

- **GET /books**
  - Get all books (should return 200 OK and a list of books)
- **GET /books/{id}**
  - Get book by valid ID (should return 200 OK and the book)
  - Get book by invalid ID (should return 404 Not Found)
- **PUT /books/{id}**
  - Update book with valid data (should return 200 OK and the updated book)
  - Update book with invalid data (should return 400 Bad Request)
  - Update book with non-existent ID (should return 404 Not Found)
- **DELETE /books/{id}**
  - Delete a book by valid ID (should return 204 No Content).
  - Delete a book by invalid ID (should return 404 Not Found).
- **And so on for all other endpoints in AuthorResource, CustomerResource,**

CartResource, and OrderResource...

Coursework Policies and Regulations

1. **Technology Restrictions:**
    - This coursework **strictly prohibits** the use of any technologies other than **JAX-RS, JSON, and Postman** (and basic Java for data models and in-memory storage).
    - **No external databases, persistence frameworks (e.g., Hibernate), Spring, dependency injection frameworks, or any other external libraries are allowed.**
    - **Using any prohibited technology like Spring or Flask or any Database technology like SQL will result in a mark of zero (0) for the entire coursework.**
2. **Originality:**
    - All submitted code must be your own original work.
    - Plagiarism or any form of academic misconduct will not be tolerated and will result in severe penalties.
3. **Submission:**
    - You must submit a zip file containing your project code and the Postman collection.
    - Detailed submission instructions will be provided separately.
4. **Late Submissions:**
5. Late submissions will be penalized according to the standard university policy [see http://www.westminster.ac.uk/study/current-students/resources/academic-regulations].

Video Demonstration Details

A significant portion of your grade will be based on a video demonstration of your API using Postman. Here's what you need to do:

1. **Postman Collection:** Create a Postman collection that includes requests for **all** the endpoints of your API.
2. **Test Cases:**
    - Include a variety of test cases for each endpoint, covering both **positive** (successful) and **negative** (error) scenarios.

    - For example, for the POST /books endpoint, you should demonstrate creating a book with valid data, as well as attempting to create a book with missing fields, incorrect data types, or a non-existent author ID.
    - Demonstrate the scenarios where your ExceptionMapper should be triggered and verify that the correct HTTP status codes and error messages are returned.
3. **Recording:**
    - Record a video of yourself using Postman to send requests to your API and showing the responses.
    - Your video should be clear, well-paced, and easy to follow.
    - **Explain each request** as you send it, describing the purpose of the request, the expected outcome, and the actual outcome.
    - **Highlight the use of different HTTP methods** (GET, POST, PUT, DELETE) and **HTTP status codes** in your demonstration.
4. **Duration:** The video should be no longer than 15 minutes.

Grading Rubric

A detailed grading rubric has been provided separately. It outlines the specific criteria for each aspect of the coursework, including:

- Correctness of code implementation for each resource class and endpoint
- Quality and completeness of the video demonstration
- Proper exception handling
- Overall code quality

Support and Clarifications

If you have any questions or need clarification on any aspect of the coursework, please do not hesitate to ask your instructor or teaching assistant during office hours or through the designated communication channels (e.g., forum, email).

Important Reminder:

The use of any technologies beyond JAX-RS, JSON, and Postman is strictly prohibited and will result in a mark of zero for the entire coursework. This includes any external libraries, frameworks, or databases. The focus of this assessment is on your understanding of core RESTful principles and JAX-RS fundamentals.

This detailed introduction should provide a comprehensive overview of the coursework requirements, policies, and the video demonstration expectations. Good luck!

Submission deadline and guidance:

1. **Submission Format:** All coursework must be submitted as a ZIP file containing your work. The ZIP file should be named in the following format: <student-id>_<student-name>_<module>. For example, if your student ID is "12345", your name is "John Smith", and the module is "CS101", your ZIP file should be named as 12345_JohnSmith_CS101.zip.
2. **Report Submission:** Alongside the ZIP file, please submit a PDF report detailing your coursework. The report should also follow the naming convention mentioned above: <student-id>_<student-name>_<module>.pdf.
3. **Deadline:** The submission deadline is **April 28, 2025, 13:00**. Late submissions will not be accepted unless any special circumstance happens, e.g. you submit an MC form.
4. **Submission Channel:** You must submit your coursework through the Blackboard using the given link under assessments.
5. **Contact:** If you encounter any issues or have questions regarding the submission process, feel free to contact your course instructor for assistance.