**University of Westminster**
Department of Computer Science

| 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):<br>- LO1 Gain a thorough understanding of RESTful principles and their application in API design.<br>- LO2 Acquire familiarity with the JAX-RS framework as a tool for building RESTful APIs in Java. |
| Handed Out: | May 31 2024 |
| Due Date | 1st July 2024, 13:00 |
| Expected deliverables | **A zip file containing the developed project**<br>**Video demonstration**<br>**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<br>2.1.2 Use of such knowledge in modelling and design<br>2.1.3 Problem solving strategies<br>2.2.1 Specify, design or construct computer-based systems<br>2.2.4 Deploy tools effectively<br>2.3.2 Development of general transferable skills<br>3.1.1 Deploy systems to meet business goals<br>4.1.1 Knowledge and understanding of scientific and engineering principles<br>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.

# Overview of the project Scenario:

This coursework involves designing and implementing a RESTful API for a holiday booking system using JAX-RS in Java. The system will allow customers to book hotel rooms online by providing dates and destinations. This coursework is designed to align with specific student learning goals, focusing on REST API design and implementation using JAX-RS. The objectives aim to equip students with practical skills and knowledge that directly contribute to their understanding of modern software development practices.

## System Architecture

The system architecture will follow a client-server model with a RESTful API serving as the interface between the client and the server. The architecture will be divided into the following components:

- **RESTful API:** The core of the system, implemented using JAX-RS, which handles HTTP requests and responses.
- **Data Access Layer (DAO):** Manages interactions with the data structure.
- **Data structure:** Stores data related to customers, bookings, hotels, rooms, and payments.

## Entities and Relationships

The primary entities in the system are Customer, Hotel, Room, Booking, and Payment. The relationships between these entities are as follows:

- **Customer:** Represents a user of the system who can make bookings.
- **Hotel**: Represents a hotel with multiple rooms.
- **Room**: Represents a room in a hotel that can be booked.
- **Booking**: Represents a reservation made by a customer for a room in a hotel.
- **Payment**: Represents a payment made by a customer for a booking.

## Entity Relationships

- A Customer can have multiple Bookings (One-to-Many).
- A Hotel can have multiple Rooms (One-to-Many).
- A Room can have multiple Bookings (One-to-Many).

- A Booking can have one Payment (One-to-One).
- A Customer can have multiple Payments (One-to-Many).

## 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 **July 8, 2024, 13:00**. Late submissions will not be accepted.
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.

## Video:

You need to record a video demonstrating all endpoints through Postman. Please remember to initialize data structures in each DAO class for testing. The video length can be 10 minutes. Please do not explain codes, just focus on demonstrating the responses from endpoints.

## Report:

The report contains two parts:

- The first part of the report, you need to create a UML class diagram, showing all entities with attributes, methods and all possible relationships between classes/entities.
- The second part includes a table to cover test cases including endpoints in each resource class.

## Marking Criteria:

- **Model Classes (%5):** All model classes are well-implemented with correct attributes, getters, setters, and constructors. Proper use of inheritance with clear hierarchies and relationships among classes.

- **DAO Classes (%30):** Well-implemented DAO classes with all CRUD methods for corresponding entities. Proper error handling, and initializing the data structures.

- **Resource Classes (%40):** All resource classes follow RESTful principles correctly. Resource methods for retrieving details by ID, searching, scheduling, managing related entities, and handling operations are accurately implemented. Exceptions are handled considering the ExceptionMapper classes. Proper logging is used. Appropriate HTTP methods (GET, POST, PUT, DELETE) are used for each operation. When application runs, necessary information based on the defined endpoints are displayed correctly.

- **ExceptionMapper and Logging (%5):** Comprehensive exception handling and logging are consistently implemented throughout the codebase, providing detailed information for debugging and auditing. Proper implementation of different ExceptionMapper classes are provided.

- **Code Formatting (%5):** Code is well-organized, modular, and follows best practices for separation of concerns. Code is extensively commented, providing clear explanations for complex logic or algorithms.

- **Video Demonstration (%5):** A comprehensive video demonstration of the API is provided, showcasing its functionality, proper usage, and interaction with Postman tests. The video is clear, well-structured, and covers all essential aspects of the API.

- **Report (%10):** A comprehensive UML class diagram has been included, demonstrating all possible entities including their attributes, methods, and all possible relationships between classes. Thorough testing is conducted with Postman, covering various endpoints and scenarios.

## Notes:

When implementing the classes, you must adhere to object-oriented principles like inheritance, encapsulation, etc. For example, when you invoke an endpoint for the Payment, the expected JSON output should be something as follows:


```
{
  "id": 401,
  "amount": 1500.00,
  "paymentDate": "2024-05-25",
  "paymentMethod": "Credit Card",
  "customer": {
    "id": 1,
    "name": "John Doe",
    "email": "john.doe@example.com"
  },
  "booking": {
    "id": 101,
    "startDate": "2024-06-01",
    "endDate": "2024-06-10",
    "room": {
      "id": 201,
      "roomNumber": "101",
      "type": "Deluxe",
      "hotel": {
        "id": 301,
        "name": "Grand Hotel",
        "address": "123 Main St, Anytown"
      }
    }
  }
}
```


The expected JSON response for the customer would be:


```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "bookings": [
    {
      "id": 101,
      "startDate": "2024-06-01",
      "endDate": "2024-06-10",
      "room": {
        "id": 201,
        "roomNumber": "101",
        "type": "Deluxe",
        "hotel": {
          "id": 301,
          "name": "Grand Hotel",
          "address": "123 Main St, Anytown"
        }
      },
      "payment": {
        "id": 401,
        "amount": 1500.00,
        "paymentDate": "2024-05-25",
        "paymentMethod": "Credit Card"
      }
    }
```

```
  ]
}
```

The expected JSON response for the hotel would be:

```
{
  "id": 301,
  "name": "Grand Hotel",
  "address": "123 Main St, Anytown",
  "rooms": [
    {
      "id": 201,
      "roomNumber": "101",
      "type": "Deluxe"
    },
    {
      "id": 202,
      "roomNumber": "102",
      "type": "Standard"
    }
  ]
}
```

The expected JSON response for the room would be:

```
{
  "id": 201,
  "roomNumber": "101",
  "type": "Deluxe",
  "hotel": {
    "id": 301,
    "name": "Grand Hotel",
    "address": "123 Main St, Anytown"
  },
  "bookings": [
    {
      "id": 101,
      "startDate": "2024-06-01",
      "endDate": "2024-06-10",
      "customer": {
        "id": 1,
        "name": "John Doe",
        "email": "john.doe@example.com"
      },
      "payment": {
        "id": 401,
        "amount": 1500.00,
        "paymentDate": "2024-05-25",
        "paymentMethod": "Credit Card"
      }
    }
  ]
}
```

The expected JSON response for the booking would be:


```
{
  "id": 101,
  "startDate": "2024-06-01",
  "endDate": "2024-06-10",
  "customer": {
    "id": 1,
    "name": "John Doe",
    "email": "john.doe@example.com"
  },
  "room": {
    "id": 201,
    "roomNumber": "101",
    "type": "Deluxe",
    "hotel": {
      "id": 301,
      "name": "Grand Hotel",
      "address": "123 Main St, Anytown"
    }
  },
  "payment": {
    "id": 401,
    "amount": 1500.00,
    "paymentDate": "2024-05-25",
    "paymentMethod": "Credit Card"
  }
}
```