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:	March 2024
Due Date	6th May 2024
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.

Overview of the project Scenario:

The coursework presents students with the challenge of designing and implementing a Health System API that addresses the complex requirements of modern healthcare management. The API will serve as the foundation upon which various healthcare applications and systems can be built, providing essential functionalities for patient management, appointment scheduling, medical record keeping, prescription management, and billing.

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 and the role of RESTful APIs in the context of healthcare systems.

System Entities:

1. **Person:**
    - Represents a generic individual with attributes such as name, contact information, and address.
2. **Patient:**
    - Extends the Person entity to include specific details relevant to patients, such as medical history and current health status.
3. **Doctor:**
    - Also extends the Person entity to include information about healthcare professionals, including their specialization and contact details.
4. **Appointment:**
    - Represents scheduled appointments between patients and doctors, including details like date, time, and associated participants.
5. **Medical Record:**
    - Holds comprehensive medical information about patients, covering diagnoses, treatments, and other relevant data.
6. **Prescription:**

- Records information about prescribed medications, including dosage, instructions, and duration.

7. **Billing:**

    - Manages financial transactions related to healthcare services, including invoices, payments, and outstanding balances.

Marking Guidelines:

1. **Model Classes (7 Marks):**

    - All model classes (Person, Patient, Doctor, Appointment, MedicalRecord, Prescription, Billing) are well-implemented with correct attributes, getters, setters, and constructors. Proper use of inheritance with clear hierarchies and relationships among classes. All classes have well-defined constructors with appropriate parameters and initialization of attributes.

2. **DAO Implementation (25 Marks):**

    - Create well-implemented DAO classes (PatientDAO, DoctorDAO, AppointmentDAO, MedicalRecordDAO, PrescriptionDAO, BillingDAO) with all CRUD methods for corresponding entities. Implement comprehensive exception handling and validation using HTTP response codes.

3. **Restful resource implementation (45 Marks):**

    - Resource classes for Patient, Doctor, Appointment, Medical Record, Prescription and Billing are created. Each Resource class includes all HTTP requests and methods. Appropriate endpoints to implement different operations such as retrieving patient/doctor/billing/prescription/appointment details by ID, creating doctors/patients/billing/prescription, searching for available appointments, scheduling appointments, accessing medical records, handling billing follow RESTful principles and are accurately implemented. Resources are clearly identified and named according to RESTful conventions. URIs are intuitive, hierarchical, and self-explanatory. Appropriate HTTP methods (GET, POST, PUT, DELETE, etc.) are used for each operation on resources. Methods align with CRUD operations. Possible exceptions are caught along with logging. Logs provide detailed information suitable for debugging.

4. **Code Organization (3 Marks):**

    - Organize code following best practices for separation of concerns, modularity, and maintainability. Providing clear explanations for codes.

5. **Documentation (10 Marks):**

    - A comprehensive class diagram is included, demonstrating clear and accurate relationships between all classes, along with attributes and methods. A full list of URIs is included, providing a clear overview of all endpoints. A comprehensive test plan is provided, covering all endpoints and ensuring thorough testing of the entire system.

6. **Video Demonstration (5 Marks):**

    - Provide a comprehensive video demonstration showcasing the API's functionality, proper usage, and interaction with Postman tests. The video should be clear and understandable.

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 appointments, the expected JSON output should be something as follows:

```
{
  "id": 123,
  "date": "2024-03-12",
  "time": "10:00",
  "patient": {
    "id": 456,
    "name": "John Doe"
    // ...other simplified patient info
  },
  "doctor": {
      "id": 789,
      "name": "Dr. Jane Smith"
      // ...other simplified doctor info
  }
}
```

As you can see, when the endpoint is invoked, the information for the doctor and patient is also presented along with **id**, **date** and **time** for appointment. Please also note that you may use any data structure like List and Map to store data for each entity.

Possible relationships between classes:

Here's an explanation of the information that should be shown in the response for each of the relationships:

Patient-Doctor Relationship:
When retrieving information about a patient or a doctor, the response should include details about their associated doctors or patients, respectively. For example, when fetching patient details, the response may include a list of doctors the patient is currently consulting, including their names, specialties, and contact information. Similarly, when fetching doctor details, the response may include a list of patients the doctor is currently treating, along with relevant patient information such as names and contact details.

Appointment-Patient Relationship:
When retrieving appointment details, the response should include information about the patient associated with the appointment. This includes the patient's name, contact information, and any other relevant details. Additionally, the response should include details about the appointment itself, such as the date, time, reason for the appointment, and the doctor involved.

Appointment-Doctor Relationship:
Similarly, when retrieving appointment details, the response should include information about the doctor associated with the appointment. This includes the doctor's name, specialty, contact information, and any other relevant details. Additionally, the response should include details about the appointment, such as the date, time, reason for the appointment, and the patient involved.

Medical Record-Patient Relationship:
When retrieving a patient's medical record, the response should include comprehensive medical information about the patient, covering diagnoses, treatments, medications, and other relevant data. This includes details such as past medical history, current conditions, allergies, surgeries, prescribed medications, and treatment plans.

Prescription-Patient Relationship:
When retrieving prescription details, the response should include information about the patient associated with the prescription. This includes the patient's name, contact

information, and any other relevant details. Additionally, the response should include details about the prescription itself, such as the prescribed medication, dosage, instructions, duration, and the doctor who issued the prescription.

Prescription-Doctor Relationship:
Similarly, when retrieving prescription details, the response should include information about the doctor associated with the prescription. This includes the doctor's name, specialty, contact information, and any other relevant details. Additionally, the response should include details about the prescription itself, such as the prescribed medication, dosage, instructions, duration, and the patient for whom the prescription was issued.

Billing-Patient Relationship:
When retrieving billing information, the response should include details about the patient associated with the billing transaction. This includes the patient's name, contact information, and any other relevant details. Additionally, the response should include details about the billing transaction itself, such as the invoice number, transaction date, billed amount, payment status, and any outstanding balances.

Billing-Appointment Relationship: If billing transactions are associated with appointments, the response should include details about the appointments that resulted in billed services. This includes information such as the appointment date, time, reason, involved doctor, and any relevant patient details. Additionally, the response should include details about the billing transaction itself, such as the invoice number, transaction date, billed amount, payment status, and any outstanding balances.

Billing-Doctor Relationship:
If billing transactions are associated with doctors, the response should include details about the doctor responsible for the billed services. This includes the doctor's name, specialty, contact information, and any other relevant details. Additionally, the response should include details about the billing transaction itself, such as the invoice number, transaction date, billed amount, payment status, and any outstanding balances.

Billing-Invoice Relationship:
When retrieving billing transactions, the response should include details about the invoice associated with each transaction. This includes information such as the invoice number, issue date, due date, billed amount, payment status, and any outstanding balances.

Additionally, the response should include details about the individual billing transactions included in the invoice, such as transaction dates, billed amounts, and payment status.

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 **May 6, 2024**. 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.