



UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)

Academic Year 2026 – 3rd Year Examination – Semester 6

IT6505(R) – Middleware Architecture (Repeat)
Structured Question Paper

(TWO HOURS)

To be completed by the candidate

BIT Examination Index No:

Important Instructions:

- The duration of the paper is **2 (Two) hours**.
- The medium of instruction and questions is English.
- This paper has **4 questions and 11 pages**
- **Answer all questions.** All questions carry **equal** marks.
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
If a page is not printed, please inform the supervisor immediately.
- All kinds of electronic devices including calculators are **not** allowed.
- *All Rights Reserved.*

Questions Answered

Indicate by a cross (x), (e.g. ☐) the numbers of the questions answered.

To be completed by the candidate by marking a cross (x).	Question numbers			
	1	2	3	4
To be completed by the examiners:				

1)

(a)

Differentiate between **syntactic transparency** and **semantic transparency**.

(6 marks)

ANSWER IN THIS BOX

- **Syntactic transparency:** Remote and local procedure calls have the same syntax.
- **Semantic transparency:** Remote and local calls behave identically in execution.

(b)

List and briefly explain the following *four (04)* **non-functional requirements** of distributed systems.

(8 marks)

ANSWER IN THIS BOX

1. **Scalability** – Ability to handle growth by adding resources
2. **Openness** – Easy extension via well-defined interfaces
3. **Heterogeneity** – Supports multiple platforms and technologies
4. **Fault tolerance** – Continues functioning despite failures

- (c) List and explain *three (03)* types of **failure models**?

(6 marks)

ANSWER IN THIS BOX

1. **Halting failure** – Process stops completely
2. **Send-omission failure** – Message is not sent
3. **Byzantine failure** – Arbitrary or malicious incorrect behavior

- (d) Why are **distributed systems** considered less reliable than **centralized systems**? Discuss at least four (04) points.

(5 marks)

ANSWER IN THIS BOX

- Multiple components can fail independently
- Network introduces unreliability
- Communication failures are possible
- Difficult fault detection and recovery

- 2)
(a) Explain the concept of **integration infrastructure**.

(5 marks)

ANSWER IN THIS BOX

Integration infrastructure refers to the hardware and software required to connect system components. It supports communication without assuming specific deployment.

- (b) Explain the use of **protocols** in middleware.

(5 marks)

ANSWER IN THIS BOX

Protocols define rules for communication between systems. Network protocols transmit data, while middleware protocols handle dialogue between applications and ensure reliability and performance.

- (c) Explain the meaning of marshallling. Describe the importance of this operation.

(6 marks)

ANSWER IN THIS BOX

Marshalling is the process of converting parameters into a network-compatible format for transmission. It ensures communication between systems with different data representations.

- (d) List *four (04)* actions performed by a **message broker**.

(4 marks)

ANSWER IN THIS BOX

- Routing messages to destinations
- Transforming message formats
- Aggregating and decomposing messages
- Invoking services or responding to events

- (e) Explain the **publish - subscribe** model.

(5 marks)

ANSWER IN THIS BOX

Publish – subscribe is a messaging pattern where publishers send messages without knowing receivers, and subscribers receive relevant messages based on their interests without knowing publishers.

3.
(a) List the steps of invoking remote operations on object references used in OOM?

(6 marks)

ANSWER IN THIS BOX

A client first obtains a reference to a remote object.
The reference acts like a pointer to invoke methods remotely.
Calls appear similar to local object invocation.
Naming services are often used to locate object references.

- (b) Discuss the role of **IDL (Interface Definition Language)** in OOM?

(6 marks)

ANSWER IN THIS BOX

- Defines object interfaces in a platform-independent manner
- Generates client stubs and server skeletons
- Supports object types, inheritance, and exceptions
- Enables interoperability between different systems

- (c) Discuss *two (02)* reasons why **CORBA** did not gain widespread adoption?

(4 marks)

ANSWER IN THIS BOX

Serialization is the process of converting an object into a message for storage or transmission. It is a broader concept that includes marshalling, especially in object-oriented systems.

- (d) What is meant by a **container** in transactional middleware.

(4 marks)

ANSWER IN THIS BOX

A container is an environment that manages services such as transactions, resource pooling, and security for components.

(e) Describe how RMI makes **remote calls**.

(5 marks)

ANSWER IN THIS BOX

Uses stubs to act as proxies
 Client calls methods normally
 Network communication is hidden
 Exceptions handled transparently.

4.

(a) Explain the steps involved in creating an **RMI** application?

(5 marks)

ANSWER IN THIS BOX

1. Define interface
2. Implement server
3. Create client
4. Compile and generate stubs
5. Start registry and run programs

- (b) Identify the most suitable **HTTP verb** to be used in each of the following RESTful URIs.

(5 marks)

ANSWER IN THIS BOX

URI	HTTP Method
/teaOrder/{id}	GET
/teaOrder/add	POST
/teaOrder/delete/{id}	DELETE
/teaOrder/getAll	GET
/teaOrder/update/{id}	PUT

- (c) The following piece of code was taken from a *Controller class* in a RESTful backend application designed for a Courier Management System.

Explain The *functionality* of the code given below.

(10 marks)

```
package com.example.courierapp.controller;

import com.example.courierapp.model.Parcel;
import com.example.courierapp.service.ParcelService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/parcels")
public class ParcelController {

    @Autowired
    private ParcelService parcelService;

    @PostMapping
    public ResponseEntity<Parcel> createParcel(@RequestBody Parcel parcel) {
        Parcel createdParcel = parcelService.createParcel(parcel);
        return ResponseEntity.ok(createdParcel);
    }
}
```

```

    }

    @GetMapping
    public ResponseEntity<List<Parcel>> getAllParcels() {
        return ResponseEntity.ok(parcelService.getAllParcels());
    }

    @GetMapping("/{id}")
    public ResponseEntity<Parcel> getParcelById(@PathVariable Long id) {
        Parcel parcel = parcelService.getParcelById(id);
        return parcel != null ? ResponseEntity.ok(parcel) :
ResponseEntity.notFound().build();
    }

    @PutMapping("/{id}")
    public ResponseEntity<Parcel> updateParcel(@PathVariable Long id,
                                                @RequestBody Parcel
updatedParcel) {
        Parcel parcel = parcelService.updateParcel(id, updatedParcel);
        return parcel != null ? ResponseEntity.ok(parcel) :
ResponseEntity.notFound().build();
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteParcel(@PathVariable Long id) {
        boolean deleted = parcelService.deleteParcel(id);
        return deleted ? ResponseEntity.noContent().build()
            : ResponseEntity.notFound().build();
    }
}

```

ANSWER IN THIS BOX

The given code represents a Spring Boot REST controller for managing parcel operations in a Parcel Courier application. It is annotated with `@RestController`, which indicates that the class handles HTTP requests and returns responses in RESTful format, while `@RequestMapping("/api/parcels")` defines the base URL path for all endpoints related to parcel management. The controller uses dependency injection through the `@Autowired` annotation to access the `ParcelService`, which contains the business logic for handling parcel-related operations. The `@PostMapping` method allows clients to create a new parcel by sending parcel data in the request body, which is then passed to the service layer and returned in the response. The `@GetMapping` method without parameters retrieves a list of all parcels stored in the system, while the `@GetMapping("/{id}")` method retrieves a specific parcel using its unique identifier; if the parcel is not found, it returns a "404 Not Found" response. The `@PutMapping("/{id}")` method is used to update an existing parcel by taking both the parcel ID from the URL and updated details from the request body, returning the updated parcel if successful or a "404 Not Found" if the parcel does not exist. The `@DeleteMapping("/{id}")` method deletes a parcel using its ID and returns a "204 No Content" response if deletion is successful or a "404 Not Found" if no such parcel exists. Overall, the controller demonstrates a complete CRUD (Create, Read, Update, Delete) REST API using proper HTTP methods and `ResponseEntity` to manage responses and status codes effectively.

- (d) The following service call was found in a piece of code in a **Controller class** of a Restful backend application.

```
Order order = orderService.findOrderById(orderId);
```

Briefly explain the expected functionality of the above *service call*.

(5 marks)

ANSWER IN THIS BOX

Service classes implements the business logic (methods to create, retrieve, and manipulate data) for a particular entity. Here the route service.findOrderById method will output a order object after searching for a order using a orderID.
