Microservice Development Practical Test (Candidate Instructions with Marks)

Time Allowed: 45 minutes

Role: Java Spring Boot Developer – Practical Round

Objective: Build two Spring Boot microservices, make an inter-service call, and implement one logic-based API. Marks for each task are shown so you can prioritize. Total = 100 marks.

Services to Create

Service Name	Port	Purpose
Course-Service	8081	Provides course details
Student-Service	8080	Manages students & communicates with Course-Service

Part A — Course-Service (20 marks)

• Use an in-memory list/map to store courses (no database).

• Preload 4–5 course records (e.g., ids 101–105).

Endpoints:

- GET /courses/{id} — Get course by ID

- GET /courses?ids=101,102 — Get multiple courses by IDs

• Return 404 if course not found.

Marks Breakdown:

Item	Marks
Seed 4–5 courses in memory	3
GET /courses/{id} returns course	7
GET /courses?ids=... returns matching list	5
404 with clear message when not found	5

Part B — Student-Service CRUD (25 marks)

• Use H2 database.

• Student entity: id (auto), name (required), email (valid).

• CRUD Endpoints:

- POST /students

- GET /students/{id}

- PUT /students/{id}

- DELETE /students/{id}

Item	Marks
Entity + H2 configured	5
POST /students creates record, returns 201	6
GET /students/{id} returns record or 404	6
PUT /students/{id} updates existing	4
DELETE /students/{id} removes record	4

Part C — Enrollment + Inter-Service Call (30 marks)

• Maintain enrollment in Student-Service memory: Map<studentId, Set<courseId>>

• Endpoints:

- POST /enrollments/{studentId}/courses/{courseId}

- GET /enrollments/{studentId} → returns student + course details fetched from Course-Service

• If a course id is missing upstream, skip and include "missingCourseIds":[...].

Item	Marks
Enroll endpoint stores mapping	8
WebClient/RestTemplate used to call Course-Service	10
Enrollment response contains student + courses array	6
missingCourseIds included when applicable	6

Part D — Logic Task (10 marks)

Endpoint in Student-Service:

POST /utils/first-unique-char

Body: {"text":"swiss"} → Response: {"firstUnique":"w"}

Item	Marks
Correct O(n) algorithm using frequency map	7
Handles edge cases (empty, no unique → null)	3

Part E — Quality, Validation & Errors (15 marks)

- Use @Valid for Student (name not blank, email format).
- Centralize errors via @ControllerAdvice returning JSON.
- Use appropriate HTTP status codes and ResponseEntity where needed.

Item	Marks
Bean validation on Student (name/email)	5
@ControllerAdvice returns structured errors	5
Consistent HTTP status codes & clean DTOs	5

TOTAL: 100 marks

Submission

- Provide curl/Postman samples or a short README with run steps.
- Ensure Course-Service on 8081, Student-Service on 8080.
- No authentication/security required.