COMP3095 – ICE2 Demonstration Checklist

Project Name: microservices-parent
Module: product-service
Course: COMP3095
Lab: ICE 2 – Containerized Microservices with API Gateway & Caching
Video Duration Limit: ≤ 10 minutes

1. Introduction Slide (Start of Video)

- [ ] Names, Student IDs, Course Code, and Lab Title
- [ ] Real photos (no avatars - perferred)
- [ ] Title: 'ICE 2 – API Gateway, Caching & Containerized Microservices'
- [ ] Technologies Used: Spring Boot, Docker, Docker Compose, Redis, PostgreSQL, MongoDB, API Gateway, TestContainers, Postman

2. GitLab Project Setup

- [ ] GitLab private repository created
- [ ] Professors added with **Reporter** access
- [ ] docker-compose.yml and all services present

3. Containerized Environment (Docker Compose)

- **All containers running** in **Docker Desktop**:
  - [ ] api-gateway
  - [ ] product-service
  - [ ] order-service
  - [ ] inventory-service
  - [ ] postgres-order
  - [ ] postgres-inventory
  - [ ] redis
  - [ ] mongo
  - [ ] mongo-express
  - [ ] redis-insight
  - [ ] pgadmin
- [ ] Show docker ps and **port mappings**

4. API Testing via API Gateway Only (Postman)

No direct service calls. All traffic <u>must</u> go through api-gateway.

- [ ] **Postman Collection** loaded
- [ ] **Base URL**: http://localhost:8080 (API Gateway)
- [ ] **All requests route correctly**:

5. Interservice Communication (order → inventory)

No direct service calls. All traffic must go through api-gateway.

- [ ] Place an order via **API Gateway**

  - [] Show:
    - **order-service** calls **inventory-service** (via REST)
    - Inventory **is called**
    - Order status **succeeds or fails** based on inventory response **(true/false)**
    - [ ] Use **Logs/Postman/db** to prove communication occurred between microservices

6. Redis Caching

- [ ] Call GET /api/product **twice** via **API Gateway**
- [ ] Show:
  - First call → **hits MongoDB**
  - Second call → **hits Redis cache** **(how could you prove this?)**
- Open **Redis Insight** → show cached data

7. Integration Testing (JUnit + TestContainers)

Only IntelliJ use allowed here.

- [ ] Run integration tests in IntelliJ:
  - [ ] ProductServiceApplicationCacheTests → Redis caching verified
  - [ ] OrderServiceIntegrationTest → order-service + inventory-service
  - [ ] InventoryServiceIntegrationTest → inventory-service + postgres
- [ ] All tests PASS
- [ ] Show TestContainers spinning up real DBs

8. Submission Deliverables

- [ ] 1. Completed ICE 2 Status Checklist (this very checklist)
- [ ] 2. URL to GitLab private repo (**Reporter** access provided)
- [ ] 3. ICE 2 demonstration video (≤ 10 minutes)

9. Wrap-Up

- [ ] Student reflects on what they learned

Reminder:

- **No IntelliJ API testing. No direct service calls. Everything must go through your complete Dockized Solution + API Gateway.** If I can't run your docker-compose, and the respective containers, and/or hit all endpoints via Postman, marks will be deducted.