COMP3095 – Assignment 1 Demonstration Checklist

Team Members: [Student 1 Name, Student 2 Name, Student 3 Name]
Course: COMP3095 – Advanced Web Application Development (Java)
Assignment: Assignment 1 – Student Wellness Hub
Video Duration Limit: 5–10 minutes

1. Introduction Slide (Required at Start of Video)

- Display a slide with:
  - Full names, student IDs, course code (COMP3095), section, and group name.
  - Real photos of all team members (no avatars).
  - Title: "Assignment 1 – GBC_WellnessHub-".
  - Brief list of core features implemented.
- Ensure the slide is clear and professional, shown at the video start.

2. Environment Setup

- Show Docker Desktop running with all containers listed (ex. via docker ps).
- Run docker-compose up -d to start the environment, showing:
  - wellness-resource-service, goal-tracking-service, event-service, Redis, MongoDB, PostgreSQL containers.
- Verify services are running (e.g., curl http://localhost:8081/actuator/health for wellness-resource-service).
- Show docker-compose.yml file in the IDE or text editor.
- Demonstrate only the containerized environment, not IntelliJ (except for when demonstrating integration tests).

3. Wellness Resource Service Demonstration

- Use Postman to demonstrate:
  - GET /resources to list all mental health resources.
  - GET /resources?category=mindfulness to filter by category.
  - POST /resources to create a new resource (e.g., title, description, category, URL).
  - PUT /resources/{id} to update a resource.
  - DELETE /resources/{id} to delete a resource.
- Show Redis caching:
  - Call GET /resources twice, showing faster response on second call (use Postman timing).
  - Show @Cacheable annotation in WellnessResourceController or equivalent code.
- Show PostgreSQL data:
  - Query the resources table (e.g., via psql in the PostgreSQL container).
- Highlight caching and database integration as part of your demonstration.

4. Goal Tracking Service Demonstration

- Use Postman to demonstrate:
    - GET /goals to list all goals.
    - GET /goals?status=in-progress to filter by status.
    - POST /goals to create a new goal (e.g., title, description, targetDate, category).
    - PUT /goals/{id} to update a goal.
    - DELETE /goals/{id} to delete a goal.
    - PATCH /goals/{id}/complete to mark a goal as completed.
- Show MongoDB data:
    - Query the goals collection (e.g., via mongo in the MongoDB container).
- Show REST call to wellness-resource-service:
    - GET /resources?category={goal.category} to suggest resources for a goal's category.
- Demonstrate inter-service communication.

5. Event Service Demonstration

- Use Postman to demonstrate:
    - GET /events to list all wellness events.
    - GET /events?date=2025-10-01 to filter by date.
    - POST /events to create a new event (e.g., title, description, date, location, capacity).
    - PUT /events/{id} to update an event.
    - DELETE /events/{id} to delete an event.
    - POST /events/{id}/register to register a student.
- Show PostgreSQL data:
    - Query the events table (e.g., via psql in the PostgreSQL container).
- Show REST call to wellness-resource-service:
    - GET /resources?category={event.category} to link resources to an event.
- Highlight event registration and inter-service communication.

6. Integration Testing

- Run integration tests using TestContainers:
    - Show test execution in IntelliJ or command line (gradlew test).
    - Demonstrate at least two tests (e.g., one for wellness-resource-service, one for event-service).
    - Show TestContainers spinning up PostgreSQL/MongoDB containers (e.g., test logs or Docker Desktop).
- Briefly highlight a test case (e.g., WellnessResourceServiceTest.java).
- Tests must use TestContainers, not local databases.

7. Docker Compose Demonstration

- Show all services running in Docker Compose:
  - docker ps to list containers.
  - Access a service endpoint (e.g., curl http://localhost:8081/resources).
- Show docker-compose.yml contents, highlighting services and ports.
- Demonstrate stopping and restarting the environment:
  - docker-compose down && docker-compose up -d.
- All functionality must be shown in Docker, not IntelliJ.

8. Code Structure

- Show project structure in IDE:
  - Highlight wellness-resource-service, goal-tracking-service, event-service directories.
  - Show key files (e.g., WellnessResourceController.java, Goal.java, EventRepository.java).
- Briefly explain one REST endpoint implementation (e.g., GET /resources with caching).
- Show Redis configuration (e.g., RedisConfig.java or application.properties).
- Focus on microservices architecture and separation.

9. Deliverables Recap

- Show the 1-page status checklist report (PDF) in the video:
  - List completed requirements (e.g., CRUD, caching, inter-service communication).
  - List uncompleted requirements and reasons (e.g., time constraints).
- Confirm the private GitLab repository URL and professor access.
- Mention adherence to AI usage guidelines (if applicable).
- Ensure the report is concise and shown at the start.

10. Wrap-Up

- Team signs off with a brief reflection (e.g., challenges faced, favorite feature).
- Keep the video under 10 minutes with clear audio and resolution.