Assignment 2: Advanced Room Booking and Event Management System

Team Size: 1-4 Team Members
Project Name: **GBC_EventBooking-**<*Group Name*>

Objective – Advanced Features and Scalability:

Building upon Assignment 1, the goal of Assignment 2 is to extend the functionality and robustness of the Room Booking and Event Management System for George Brown College. The focus will be on integrating advanced microservices features, including enhanced inter-service communication, secure API management, resilience patterns, and observability.

Instructions and Requirements:

Extend your environment with these additional tools and frameworks:

1. **Keycloak** for secure authentication management.
2. **Spring Cloud Gateway** as an API Gateway.
3. **Resilience4J** for implementing Circuit Breaker patterns.
4. **Kafka with Schema Registry** for schema management in inter-service communication.
5. **Grafana Stack** (Prometheus, Grafana, Loki) for observability, logging, and monitoring.
6. **Swagger/OpenAPI** for API documentation, aggregated through the API Gateway.

Below you will find the assignment requirements have been broken up for you to digest more easily.

Part 1: Secure API Gateway

Objective:

Secure the API communication and centralize entry points through an API Gateway.

- Setup Spring Cloud Gateway as the entry point to route requests to your microservices.
- Integrate Keycloak to manage authentication:
    - Configure the API Gateway to use Keycloak for securing all microservices.
    - Ensure that only authenticated users can access specific services (e.g., Booking and Approval services).
    - Role-based access should be enforced through Keycloak
        - Research & Implement → Only staff can access **ApprovalService**.

Part 2: Enhanced Inter-Service Communication with Kafka and Schema Registry

Objective:

Introduce Kafka for asynchronous messaging with strict schema adherence.

Implement Kafka for asynchronous event handling:

- **BookingService** should publish booking events to Kafka when a booking request is confirmed.
- **EventService** should consume these events to register events for the confirmed bookings.

Use Schema Registry to enforce message schemas:

- Register and manage schemas in the Schema Registry to validate the structure of Kafka messages.
- Ensure schema compatibility between services to prevent breaking changes.

Part 3: Circuit Breaker Pattern with Resilience4J

Objective:

Implement Circuit Breaker for resilience in service communication.

1. **Apply Circuit Breaker** to critical inter-service calls:
   - Configure Circuit Breaker in **BookingService** to handle potential failures in **RoomService** when checking room availability.
   - Configure Circuit Breaker in **ApprovalService** when fetching user roles from **UserService** to ensure fallback responses during downtimes.

2. **Fallback Mechanisms**:
   - Implement fallback strategies for each Circuit Breaker configuration to handle degraded services and prevent cascading failures.

Part 4: Observability with Grafana Stack

Objective:

Implement observability to monitor system performance, health, and logs.

1. **Prometheus and Grafana:**
   - Set up **Prometheus** to collect metrics from each microservice and the API Gateway.
   - Create a **Grafana dashboard** to visualize metrics, such as request rates, response times, and error rates.

2. **Logging with Loki:**
   - Set up **Loki** to collect and centralize logs from all microservices.
   - Integrate **Grafana Loki** to enable log analysis and correlate logs with metrics for end-to-end monitoring.

Part 5: API Documentation with Swagger

Objective:

Provide accessible documentation for all REST APIs through the API Gateway.

1. **Integrate Swagger** into each microservice to document available endpoints and methods.

2. **Aggregate Swagger Documentation**:
   - Configure the API Gateway to aggregate Swagger documentation from each microservice.
   - Ensure that the complete API documentation is accessible through a single endpoint on the Gateway.

Deliverables:

- A **private** git repository containing all your code, properly documented.
  - Please ensure to add me as a collaborator (**Sergio.Santilli@georgebrown.ca**)
- A docker compose file to launch the entire system.
  - Make sure all components of your solution are containerized in docker.
  - I will be running the docker compose file to test your system **exclusively**. I will **NOT** run your solution environment in any other way, so you must make sure your docker compose brings up working environment correctly.
  - Please note: If I cannot run your docker compose, you will lose **substantial** marks
- You **must** demonstrate in your video recording, you're running **docker-compose environment**. This is the principal environment, and only environment that should be utilized during the demonstration (**not Intellij**)
- Commit your postman collection to your code report to showcase and test your various endpoints of your application.
- You must to a thorough job of demonstrating your solution in your video.
- A brief **report** explaining the architecture, challenges faced, and lessons learned.
  - Include this as an individual attachment upload as part of your submission.
- **Please review the final page (below) for submission guidelines**

Evaluation Criteria:

- Code Quality: Clear, readable, and maintainable code.
- Functionality: All described features should be implemented and working.
- Robustness: Proper error handling and usage of the Circuit Breaker pattern.
- Documentation: Both in-code comments and the accompanying report.
- System Robustness: The system should handle errors gracefully and ensure data consistency.
- Demonstration via your video

Assignment Submission Guidelines:

1. Video Requirement
   a. Create a Short Video presentation. Your presentation should start with an introduction, where it must display a PowerPoint (or Google Presentation), that is 1 (single) slide. The slide introduces each member of your group, again, at the very start of your video.

   b. The first (and only) slide of your presentation must include current images of you and your partner(s) (no avatars allowed) that are displayed appropriately. You must also include your Full Names, Student IDs, the Course Code, Course Name, Course Section and your Assignment information.

   c. Within the recording, you or your partner(s) will take turns demonstrating your program's functionality. You must show your solution working properly. You will also construct an assignment status report, a single page checklist/report. Use the report during the video, to facilitate communication confirming where requirements where successfully implemented and/or where requirements failed to be implemented and why.

   d. You and your partner(s) share the responsibility to demonstrate your functioning solution. Please note, the entire objective of the video is to demonstrate the functionality of your solution, this even more so then explaining your code. You will likely need to use Postman to demonstrate your services. When demonstrating your solution, I want to only see your docker compose environment running, that is, the environment that is demonstrated and tested, is running via your completely containerized environment.

   e. You and your partner(s) will each share the responsibility in describing the code in your solution that drives the functionality of your program – you will want to do this part well and be very clear. Be intelligent/selective on what code segments you describe; I do not need to know how every line of code works so keep this somewhat minimal.

   f. Sound for your video must at an appropriate level so that your voices may be clearly heard, and your screen resolution should be set so that your program's code and console details are clearly visible. In short, QA your videos. If your video is poor, assignment failures can/will be assigned.

   g. Your video should run no more than **~5-10** minutes. If you exceed this time, I simply will **not** be able to watch them... resulting in a grade of **zero**.

2. The 1 team lead, must submit the following **3** components to **Brightspace** on behalf of the entire group:
   a. The 1-page project report – **mandatory**
   b. The (zipped) project source code – **mandatory**
      i. Please make sure your docker-compose file is in your project
   c. A Postman collection to showcase and test the various endpoints of your application.
   d. A brief report (<=1 page) explaining the **architecture**, **challenges faced**, and **lessons learned**.
      i. Include this in a folder, located within your project
   e. The group video file - **mandatory**
      i. You may find OBS Studio useful to create this

3. Be cautious **DO NOT** share your application with others. Complete failures will be assigned if code is shared. All assignments will be reviewed and analyzed strictly within these regards.

4. Late assignments are assigned a penalty as described below,
   a. -20% per day (Max of 5 Days)

Exception to this is **end-of-year final assignments. Late submissions for final assignments are **NOT** accepted (**assigned a mark of zero**).

5. Groups that exceed the maximum number of group members, will receive a penalty (-25%) for each added extra person. -10% deduction for those who do not fulfil the minimum group size.

Good luck, and remember to enjoy the process!