Boda Norbert (bnim2219), Software engineering

Objective Description

The goal is to design and implement a small system using Event-Driven Architecture (EDA) principles. The implementation must include two scenarios: a Fan-Out scenario and a Choreography scenario. The system will feature separate services running in Docker containers that communicate through events using an asynchronous messaging system like RabbitMQ, Kafka, or a similar tool.

Technology Stack:

- **Messaging System**: RabbitMQ for event brokering.
- **Languages**: Java for one service and Python for the other services

Implementation Details

A. Fan-Out Scenario

Description: One producer emits an event consumed by 2 consumers.

Flow:

- Service A publishes a “New Landmark Added” event to the broker.

- Service B and Service C subscribe to the event and process it independently.

Docker Services:

- Service A (producer-java): Java application publishing the event.

- Service B (consumer_producer_python) and C (consumer-python1): Python services consuming the event.

B. Choreography Scenario

Description: Involves at least three services with one event triggering a cascade of dependent events.

Flow:

- Service A publishes a “New Landmark Added” event.

- Service B subscribes, processes it, and emits “Landmark Details Updated” event

- Service C subscribes to the second event and processes it

Docker Services:

- Service A (producer-java): Java application that produces the “New Landmark Added” event.

- Service B (consumer-producer-python): Python service that consumes the first event and produces the second.

- Service C (consumer-python2): Python service that consumes the second event and processes it