Distributed Systems Project

Final Deadline 30.4 (2359)- Deliverable 1 due (25.4), Deliverable 2 due (30.4)

Total Points: 30

Description

Choose a system of your choice (except for e-commerce applications) and implement the solution from inception to closure using microservices architecture with the following main tasks:

Task 1: Design and Architecture (15 points)

- Describe the functional requirements of the system, and out of those requirements, identify the key services as stand-alone autonomous business units often called microservices.
- Clearly define the scope of each microservice with interservice communication patterns.
- Demonstrate the architecture with UML diagrams of your choice and a single architecture diagram describing the whole system architecture with a strong focus on communication.
- Clearly demonstrate the communication pattern used and describe the limitations around communication for microservices.

Task 2: Implementation (10 points) + Video (5 points)

- Develop at least 3 microservices among all the microservices identified in Task 1 using a technology stack of your own choice.
- Implementation in multiple technologies and programming languages is preferred but not mandatory.
- Front-end implementation is optional for all the microservices.

Deliverables:

1. **DELIVERABLE 1:** (Deadline: 25.4)
   Design and architecture document (TASK 1) (**Deadline: 25.4**)

2. **DELIVERABLE 2:** (Deadline: 30.4)
   i. Link to the Github code (**Deadline: 30.4**)
   ii. A 10-minute video explaining the 3 microservices in a running condition (Note: Information about Deliverable 1 is not required in the video) **(Deadline: 30.4)**

The deliverables will be evaluated based on the following criteria.

- Timely task completion **(10%)**
- System design and problem-solving approach **(30%)**
- Quality of documentation (D1) and code/presentation (D2) **(50%)**
- The idea of the solution adopted (Creativity) **(10%)**

Guidelines

Design and Architecture:

1. Identify the functional requirements of the system and define them clearly. These requirements should be broken down into smaller, independent tasks that can be accomplished by individual microservices. (How to write good software requirments)
2. Identify the key services required for the system and define their scope clearly. These services should be autonomous, with well-defined APIs for communication.
3. Determine the communication patterns for inter-service communication, such as REST or message queues, and clearly demonstrate them with UML diagrams. Also, create an architecture diagram that describes the whole system architecture with a strong focus on communication patterns.
4. Describe any limitations around communication for microservices, such as network latency or data consistency.

Implementation:

1. Develop microservices as autonomous, independent of each other and as small as possible (boundary is crucial)
2. SOLID Design Principles should be followed (About SOLID: https://www.freecodecamp.org/news/solid-design-principles-in-software-development/)
3. Stick to the purpose (Avoid coupling between services)
4. The microservices developed should be autonomous, with well-defined APIs for communication.
5. By following these guidelines, you should be able to implement a system using microservices architecture of an application. Remember that the key to success with microservices architecture is to keep the services small, autonomous, and well-defined, with clear communication patterns and boundaries.