FACULTY OF SCIENCE

ACADEMY FOR COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	**IT08X31 - SERVICES COMPUTING**
CAMPUS	**APK**
FINAL SUMMATIVE ASSESSMENT (EXAM)	**JUNE 2021**

MEMO

DATE: xx JUNE 2021 **SESSION 08:30 – 11:30**

ASSESSOR(S) **PROF M COETZEE**

EXTERNAL MODERATOR **PROF HS VENTER (UP)**

DURATION: 3 HOURS **MARKS: 75**

2.5 hours to write, 30 min to upload/download

NUMBER OF PAGES: 4 PAGES

INSTRUCTIONS:

- Answer all the questions.
- You may use Microsoft Word (or equivalent) and Microsoft Visio (or equivalent) to answer your questions.
- Type your name at the top of the document. Clearly number the questions, leaving spaces between questions. Include the diagrams directly in the document at the correct place.

- Once you are finished writing, please save the document as a PDF and upload a single document. Create a document using your student number and surname as the file name.
- You only need Internet access to download the exam and then to upload your final submission. You do not need to be connected for the duration of the test.
- Should load-shedding be implemented during the day, we will communicate with you.
- While it is advised that you type the test, you may also write the test by hand, on paper. If you do this, please number the physical pages before scanning/taking the photos. After that, you must use CamScanner or take CLEAR photographs and upload them.
- During the exam, if you have any queries, you can contact me via our Discord server channel or email me at marijkec@uj.ac.za

---

QUESTION 1

HATEOAS promotes loose coupling.

Do you agree or disagree with this statement? Motivate your answer. **[8]**

Students to discuss both terms

Hypermedia as the Engine of Application State (**HATEOAS**) is a constraint of the REST application architecture that distinguishes it from other network application architectures. With **HATEOAS**, a client interacts with a network application whose application servers provide information dynamically through hypermedia.

Loose coupling is an approach to interconnecting the components in a system or network so that those components, also called elements, depend on each other to the least extent practicable. **Coupling** refers to the degree of direct knowledge that one element has of another.

Yes – as links are presented on the fly they are not static.
Application has one entry point

QUESTION 2

a) Define when an HTTP command is considered to be *safe*. (1)
doesn't alter the state of the server

b) Define when an HTTP command is considered to be *indempotent*. (1)
an identical request can be made once or several times in a row with the same effect while leaving the server

c) You are given the following set of endpoints. **For each endpoint**, you need to:

i. Comment on the manner in which it is designed.
Propose a better design where required. (2)

ii. Comment on how idempotent and safe the use of the HTTP command is. (2)

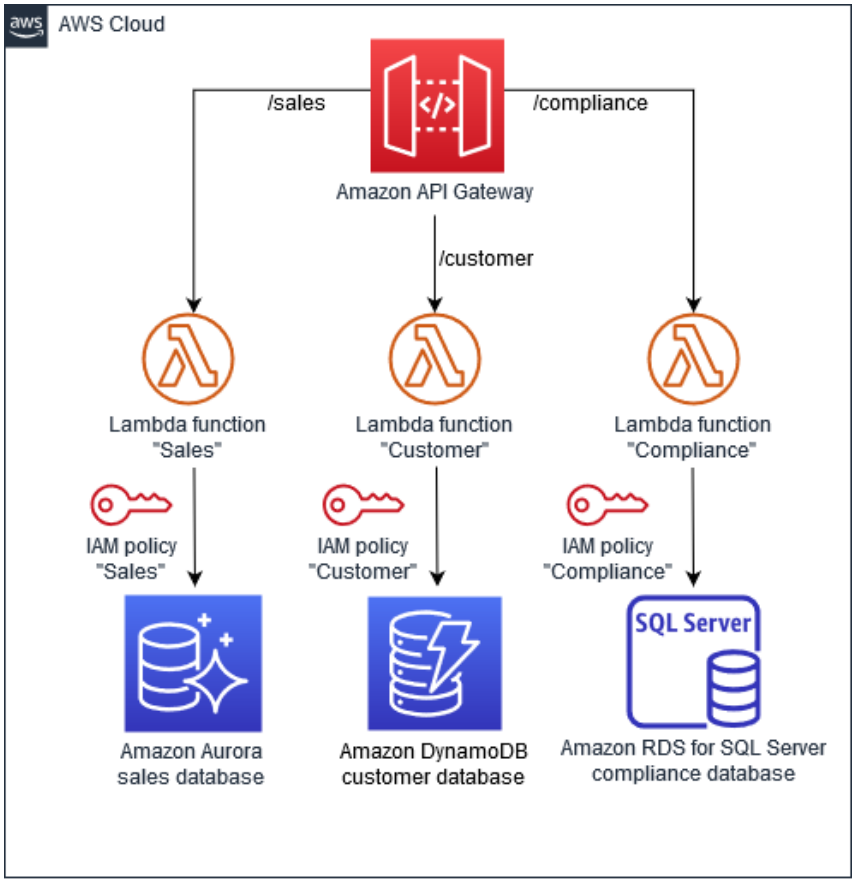
Answer your question by writing down the endpoint, and then answering i and ii for each endpoint. The answer for each endpoint therefore counts 4 marks.

1. `GET /v1/all-offices` should be offices only
2. `DELETE /v1/remove-offices/{office_id}` should be offices only
3. `POST /v1/offices/delete/{office_id}` wrong command

[16]

QUESTION 3

In the following illustration, different AWS databases are used by the “Sales,” “Customer,” and “Compliance” microservices. These microservices are deployed as AWS Lambda functions and accessed through an Amazon API Gateway API. Each microservice uses a database type that meets its individual requirements; for example, "Sales" uses Amazon Aurora, "Customer" uses Amazon DynamoDB, and "Compliance" uses Amazon Relational Database Service (Amazon RDS) for SQL Server.



a) Name and describe the pattern used by this application to manage data. (5)

Database per service

b) What is the main reason why organisation would implement this pattern? Motivate your answer well. (4)

Ensure true loose coupling

c) Describe two solution that can be implemented when queries are run that need to integrate data from the Sales and Customer databases. (6)

API composition and CQRS patterns

API Composer (API Composition pattern)

- The application performs the join rather than the database.
- For example, a service (or the API gateway) could retrieve a customer and their orders by first retrieving the customer from the customer service and then querying the order service to return the customer's most recent orders.

Command Query Responsibility Segregation (CQRS)

- Maintain one or more materialized views that contain data from multiple services. The views are kept by services that subscribe to events that each services publishes when it updates its data.
- The event sync in white takes the written model data and sends it asynchronously to another module in order to save the same data to the materialized view

[15]

QUESTION 4

Describe the role of the API Gateway in securing microservices transactions. **[8]**

- An API Gateway is the single point of entry for any microservice call.
- It can work as a proxy service to route a request to the concerned microservice.
- It can also offload the authentication/authorization responsibility of the microservice.
- The API gateway is the single entry point for client requests. It authenticates requests, and forwards them to other services, which might in turn invoke other services.
- The API Gateway authenticates the request and passes an access token (e.g. JSON Web Token) that securely identifies the requestor in each request to the services. A service can include the access token in requests it makes to other services
- Can integrate with an Identity Provider component - takes care of all the security-related tasks like
    - User management
    - Key and token management (OAuth2 and OIDC)
    - Entitlement management (XACML)
    - Authentication

- Authorization

QUESTION 5

A Delivery microservice is in charge of delivering customer orders. The Delivery microservice calls the Order microservice to update order data, but a delay is experienced, causing a backlog. It is difficult to understand what is causing the delay. There is speculation that it may be the Delivery microservice that is overloaded and runs slowly. The Delivery microservice may have a bug, or the network has firewalls that are purposely slowing traffic. There could be many more reasons for the delay to occur. It was clear that the system was not capable of running in partially failed conditions.

a) Give three patterns that could be used to attempt to solve this problem. Motivate why they could be helpful. (3)

Client-side load balancing: Give the client the list of possible endpoints and let it decide which to call.

Service discovery: A mechanism for finding the periodically updated list of healthy endpoints for a particular logical service.

Circuit breaking: Shedding load for a period of time to a service that appears to be misbehaving.

Bulk heading: Limiting client resource usage with explicit thresholds (connections, threads, sessions, etc.) when making calls to a service.

Timeouts: Enforcing time limitations on requests, sockets, liveness, etc., when making calls to a service.

Retries: Retrying a failed request

b) Describe a solution that could assist in supporting service to service communication to solve these problems. (8)

Describe Service Mesh

- A ***service mesh*** is a distributed application infrastructure that is responsible for handling network traffic on behalf of the application in a transparent, out-of-process manner.
- A given microservice does not directly communicate with the other microservices.
- An abstracted layer which sits on top of the microservices and handles service-to-service communication
- The service proxies form the “data plane” through which all traffic is handled and observed.
- The **data plane** is responsible for establishing, securing, and controlling the traffic through the mesh.

- The management components that instruct the data plane how to behave is known as the "control plane".
- The **control plane** is the brains of the mesh and exposes an API for operators to manipulate the network behaviours.
- The interconnected set of proxies in a service mesh that control the **inter-services communication** represents its **data plane.**
- The data plane is the data path and provides the ability to forward requests from the applications.
- A data plane may also provide more sophisticated features like health checking, load balancing, circuit breaking, timeouts, retries, authentication, and authorization.
- The specialized proxy that is created for each service instance (i.e., sidecar proxy) performs the runtime operations needed for enforcing security (e.g., access control, communication-related), which are enabled by injecting policies (e.g., access control policies) into the proxy from the control plane.
- This also provides the flexibility to dynamically change policies without modifying the microservice's code.
- A **control plane** is a set of APIs and tools used to **control and configure data plane** (proxy) behavior across the mesh.
- The control plane is where users specify authentication policies and naming information, gather metrics (in general telemetry collection), and configure the data plane as a whole.
- The intelligence, data, and other artifacts required for implementing all e.g. security functions lie in the control plane
-

c) Why would an API Gateway not be the ideal solution to this problem? (4)

A centralized system through which traffic travels and can become a source of bottlenecks

All internal traffic between services can be traversing the API gateway requiring two hops between services.

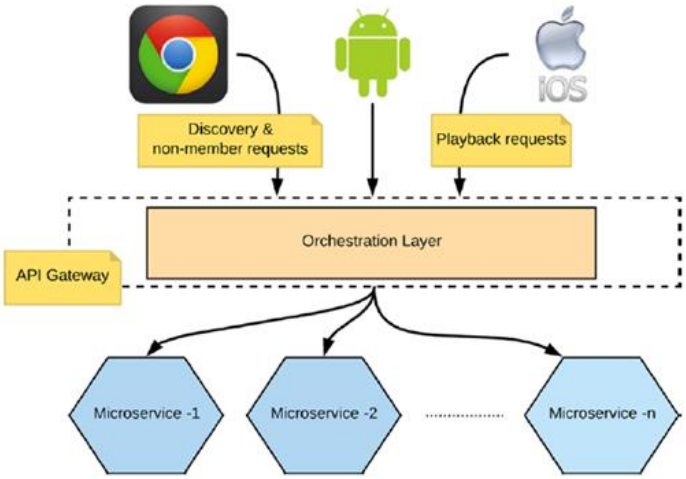
Impacts network overhead and latency and security

API gateway does not always implement resilience capabilities such as circuit breakers or bulkheading.

[15]

QUESTION 6

Given the following older Netflix integration approach:



a) Explain how Netflix implemented service intergration and what their problems were with this approach. (4)

the orchestration layer, which is a monolithic component, contains a significant portion of the business logic in this scenario

b) Explain what active and reactive service compositions are. (6)

Active Composition or Orchestration

microservice actively calls several other services (can be core or composite service). The business logic and the network communication are built as part of the integration service. The integration microservice should formulate business functionality out of the composition that it does.

Reactive Composition or Choreography

With the reactive communication style, we do not have a service that synchronously calls other services. Instead, all the interactions between services are implemented using the asynchronous event-driven communication style.

c) To solve their integration problems, Netflix implemented a new soloution. Explain what the solution was. Classify the solution very clearly according to one of the two patterns listed in question (b). (5)

shouldn't be using an API gateway as a monolithic runtime to put the business logic.

The service integration or composition logic must be part of another microservice (either at the API gateway layer or at the services layer).

[15]