Question 1: Emad

The Selected is :
B
C
G
I
J

Question 2: Hedra
----
a-
security   use OAuth2, JWT

b-
Resilience use Hystrix

c-
Monitoring use zipkin, sleuth ,ELK

d-

Transaction use SAGA pattern, Compensating transaction, Eventual consistency

Question 3: Reda

1- Reactive System
Advantage:

- performance:no need to wait results to be available
- scaling: less thread needing

Disadvantage:

- The whole the whole calling stack need to be reactive
- Hard to debug

2- event driven architecture (messageing system)
Advantage:

- Fast
- Flexible
- Less dependencies
- Loosely coupled Producers and Consumers

- Using buffer if the service slow or down
-

Disadvatnage:

- If the consumer is temporally not available (or very slow) the message middleware has to store the messages
- Eventually the message broker will fail
- Not fault-tolerant

Question 4: Afify

A - Separation of commands and queries help us to make simpler domain models by Separating the querying from command through providing two models instead of one.

One model is built to handle and process commands
One model is built for queries needs

In the above system there is 2 valid reason why we can apply CQRS

Consistency:
Command: needs consistency
Query: eventual consistency is mostly OK

Scalability:
Command: commands don't happen very often. Scalability is often not important.
Query: queries happen very often, scalability is important

Data storage:
Command: you want a normalized schema (3rd NF)
Query: denormalized (1st NF) is good for performance (no joins)

Etnen bs ya geda3an msh el talata

B-

- Read model or DTO need not have all the fields as a command model, and a read model can have required fields by the client view which can save the capacity of the read store.

Question 5: Adam

A -

Option 1 :

Advantage :

- Full control of the synchronous flow
- Easy to monitor the process

Disadvantages:

- Coupling
- Orchestrator is single point of failure
- No parallel processing

Option 2 :

Advantage:

- Less Coupling , Easy to add/remove services without impact on other services
- Fast: parallel processing ,
- No single point of failure

Disadvantages:

- Harder to monitor the process

B-

I'd say we need to focus on business logic. Where a single point of logic does the job and the app does not need to be that scalable, while I am looking for simplicity, we do option 1, Where a problem cannot be covered by a centralized logic, or we're looking for a scalable application that will serve a huge amount of requests we are forced to go with option 2.

Question 6: Osama (DONE)

Valid reason why it is not good idea to have web client talk directly to microservices

1- Not all services support HTTP. Maybe they use only messaging, and the client does not support messaging
2- Tight coupling: Client needs to know all details of how to call a service
3- Difficult to implement crosscutting concerns: security, logging, transformation
4- Performance issue when you have chatty communication

Question 7: Elmaachi

Hystrix (circuit breaker) will use the Resilience patterns to introduce fault tolerance and latency tolerance by isolating failure and by preventing them from cascading into the other part of the system Circuit breaker will periodically check if Service A instance 1 is back online and restore it automatically once the service came back

Load balancing lets you evenly distribute network traffic to prevent failure caused by overloading a particular resource.
If one instance stopped, automatically the other instance will be used, this strategy improves the performance and availability of applications

registry/discovery :
Increase application resilience, If a service instance becomes unhealthy or unavailable, the service discovery engine will remove that instance from the list of available services.

Question 8: Hedra

A-
Microservices does not use ESB becouse ESB connect services using orchestration "central brain " but Microservices use choreography "No central brain" becouse it have small services

B-
Microservices advocate design constraints where each service is developed, deployed and scaled independently. This philosophy is only possible if you have **database** per service

Question 9: Hedra: SCI

A streaming data architecture is a framework of software components built to ingest and process large volumes of streaming data from multiple sources and this is the Unified Field is the abstract field that unites all diversity in
creation as we have the stream it self Unified all services to send and receive from it
Pure Consciousness one gets access to all intelligence of creation
As same as in stream base we have one stream and all services Producer–consumer use it