ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Engineering (CSE)

SEMESTER FINAL EXAMINATION **SUMMER SEMESTER, 2019-2020**

DURATION: 1 HOUR 30 MINUTES **FULL MARKS: 75**

SWE 4601: Software Design and Architecture

This is an open book exam. You are allowed to use any printed material and online resources with a computer. You are not allowed to use your keyboard or other typing methods. Mobile phone or other touch devices are not allowed. Communicating with a person online/offline is not allowed.

There are **3 (three)** questions. Answer all of them. **Please submit each answer in separate files.** Filename should be as `<your-full-id>-<question-number>.pdf`. For example **`170042003-1.pdf`. You can scan your answer script only after the exam time is over.**

In this exam, you will play the role of Chief Solution Architect of the popular e-commerce platform *Kids' Shop*. You will have to make some design decisions that may have more than one possible solution. You should choose one of them and justify your opinion (if asked).

1. Kids' shop is consistently gaining popularity and therefore facing performance issues. You have already done several horizontal scaling, however, the cost of scaling is rapidly rising. You identified that only a few parts of the application have high traffic, but you have to scale up the whole system as it is a monolith. Some R&D and a few team discussions reveal that migrating to a microservice architecture will solve this and many other problems you are currently facing. So you decide to migrate to microservice and consider the following: 15

   i. The authentication and authorization should work the same way for all the components.
   ii. You will use the technologies that suites best for a component, therefore, different components may use different technologies. Some of the technologies accept XML and the others accept JSON in the request body. However, you prefer that client-side applications always use JSON.
   iii. You want the databases to be decoupled.
   iv. You want to migrate to microservice architecture gradually, keeping the application live.
   v. The components will go through several deployments in different environments before going live. You want the configurations of the environments decoupled from the application code so that you can easily automate the deployment process.

   Link each of the points above to one or more microservice design patterns. Give a brief explanation (5 or fewer sentences each) of why you think the point links to a particular pattern. You need to consider only the following patterns:
   Strangler, API Gateway, Database per Server, CQRS, Event Sourcing, Saga, External Configuration, Circuit Breaker.

2. a) Explain "serverless" in the context of serverless architecture. 3
   b) State and explain two points why you may want to migrate to a serverless architecture. 12
   c) State and explain two points why you may not want to migrate to a serverless architecture. 12
   d) Some of your teammates are claiming that the migration to serverless would be easier if the application was currently monolith instead of microservice. Do you agree or disagree with their opinion? Explain why. 8

3. a) How can you manage the session state of the serverless application? Explain with a diagram. 12

   b) Your team is struggling to figure out how the payment processing module should be designed. So you find a possible solution and decide to present it to the team as a class diagram. 10

      i. Should you include all the connected modules?
      ii. If you include a connected module, should you include all the classes of that module?
      iii. Should you include all the classes of the payment processing module?
      iv. If you include a class, should you include all the methods and properties of that class?
      v. In general, how much detail should be included in a class diagram?

      If you answer “no” to any of the above, mention what part should be included instead. Keep in mind that someone else will write the code.

   c) How is the activity diagram different from the flow chart? 3

The class materials can be found here: https://shorturl.at/gCHSV

170042032

Ans to Q. 1

(i) this is similar to API gateway. API gateway has the same authentication structure & same endpoints for all users, but users get a different view based on the amount of privilege their role has. In this scenario, admins & normal users can use same endpoint, but admins might see different results based on privilege.

(ii) CQRS: CQRS is a responsibility segregation pattern that segregates component based on command and query. This design pattern can allow the system to use different ~~cont~~ technologies for different components but keep them ~~separat~~ separated.

(iii) Database Per Service: In this design pattern every service have their own DB. So data management can easily be decoupled. There is also scope for a shared DB if necessary.

(IV) Strangler: The strangler pattern is best suited in this scenario because this pattern keeps the legacy system alive and take requests, but then redirects them to the monolithic server. This way, there is zero downtime but we can still migrate to a monolithic system seamlessly

(V) External Configuration: This is a design pattern that decouples DB, env environment variables, data files etc from the logic code part of the application. Then there are different config files for different environments, ~~They~~ which can be installed based on need.

Ans to Q.2 (a)

Serverless Architecture refers to "Backend as a Service", where applications use cloud-based services or API to replace server application. In this architecture, developers don't write server side code. Even if they do, its usually event-triggered.

Ans to Q2 (b)

The following points can be a reason to migrate to a serverless architecture:

(1) Reduced Operational Cost: Since almost all the work is done by 3rd party APIs, infrastructure & labour cost cut down to nearly zero. Even database cost is competitively low because one vendor is running thousands of very similar databases. It also reduces scaling cost because I only pay for the compute that I need.

Ans. ② Reduced Development Cost: In serverless architecture, I won't have to worry about handling authentication. Which is to say, firstly I don't need to worry about which client is sending which request, secondly, ~~I don't~~ services like Auth0 can easily provide in-built authentication functionalities.

Ans to Q.2 ©

The following reasons can be applied in case of not wanting to migrate to serverless ~~and~~ architecture:

① Vendor Control: The vendor has a significant amount of control over ~~what~~ my system. ~~So this can be~~ In scenarios where there is an issue on the vendor, ~~my~~ my system may face a ~~big~~ huge downtime & I have no control over it. There also ~~are~~ are security concerns.

(11) Multilatency:- It often occurs that same vendors are serving multiple systems. Sometimes the seamless feeling being the only one served can be disrupted, if any sort of latency occurs. These problems include, being able to see other customers data, reobustness and performance slow down.

Ans to Q·2 (d)

I disagree with the statement. It's easier to implement serverless architechture in microservices than that of ~~month~~ monolith.

The serverless architechture prefers a ~~choreo~~ "choreography over orchestration" approach. Which is to say, the components are the ones concerned with listening to the events they needs. This aligns perfectly with microservices, because they have seperation of concerns. This is not only only cost-beneficial but also more flexible & amenable to change.

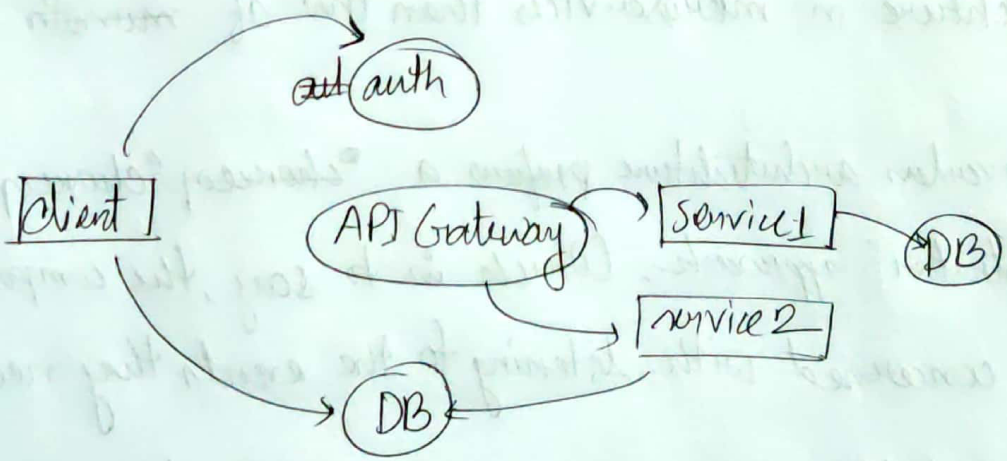
Ans to Q·3 (a)

In order to manage the session state of a serverless application. We can save session data in client side, or I can use a service that will handle authentication for me. There are many ways to do that.

(i) vendors can keep function instances alive for longer between execs.

(ii) low latency access of out-of process data.

(iii) A hybrid combination of both.



There are other approaches that can be taken from the client session state's

(1) for short session, the session data can be stored in the URL. This is not recommended for longer session.

(2) There is also a method called hidden data where we set the HTML texttype to "hidden" for session data.

(3) Rich-client apps can handle this with local storage & session storage.

Ans to Q. 3(b)

(i) no. we have to keep only the module that have an effect in the data flow of payment procesing module. For example there might be module that shows the customers current balance, this has nothing to do with payment processing, so it can be excluded.

(ii) no, we need to include the class that need changes only. For example we have a factory of payment methods and we want to alter the process for bkash only. In that case, no need to add classes for Rocket, Nagad etc.

(iii) yes

(iv) no, we will include only the methods that are necessary.

(v) we will try to keep only the necessary information in a class diagram. Here, necessary info means only the info that will help the programmer write the code properly with any confusion

170042032

8 & 9

Ans to Q. 3 (b) (c)

① ~~Active~~ Activity diagram and flow chart are almost the same. But while flow chart show the flow of ~~co~~ data in component, ~~A~~ Activity diagram explain the behaviour. For example, in the order placement module, payment & procuring are not mutually exclusive. This is something we can't show in ~~activity~~ flow chart, ~~diagram~~, hence the importance of activity diagram.