Answers

Q1: If every user have 5 credits at begin (export 1 time will consume 1 credit), every 12am will gain 1 credit. Please provide your design philosophy

A1: To manage the credits for users, it would be beneficial to create a new microservice that is responsible for handling credit-related operations. This microservice would expose an API that can be used by other services to perform necessary credit-related tasks.

When a user is created, the initial credit value can be set to 5. A cron job or web framework scheduler can be scheduled to execute a SQL query at 12am every day to add one credit to each user's account.

By creating a dedicated microservice for credit management, we can ensure that credit-related operations are handled efficiently and consistently across all services. This microservice can be integrated with other services, such as the export-service, to provide seamless credit-related functionality.

Q2: Besides the design on the diagram, What kind of technique approach you can provide to achieve the same goal? What's Pros and Cons? Please provide the diagram if you prefer.

A2: I think this is very good design.

Maybe we need user authentication service and I think AWS Cognito User Pool is good option.

We can use SAM to deploy to AWS Lambda which is good for cost, server management and scalability.

But there are disadvantages such as Cold Start, Execution Time Limitation.

AWS Lambda is freezed when there are not requests for certain period so the response of first time request will be long. We need to handle these problems.

Q3: Write the sql to ...

A3:

1.

```sql
SELECT category, COUNT(DISTINCT menus.id) AS count FROM menus
JOIN muscle_menus ON menus.id = muscle_menus.menu_id
JOIN muscles ON muscle_menus.muscle_id = muscles.id
WHERE muscles.name IN ('muscle1', 'muscle2')
GROUP BY category
ORDER BY `count` DESC;
```

2.

```
SELECT category, COUNT(DISTINCT menus.id) AS COUNT FROM menus
JOIN muscle_menus ON menus.id = muscle_menus.menu_id
JOIN (SELECT id FROM muscles WHERE `name` IN ('muscle1', 'muscle2')) AS
filtered_muscles ON muscle_menus.muscle_id = filtered_muscles.id
GROUP BY category
ORDER BY `count` DESC;
```

Q4: Investigate how to implement Upload file though API

A4:

Multipart Form Data: We can use multipart form data to upload the file as part of the request body. I think this is very common way to upload file

Base64 Encoding: We can encode the file as a Base64 string and send it via API request body. But it can increase the size of the request body and may not be suitable for large files.

Streaming: We can break file down into chunks and upload each chunks as per request.

FTP/SFTP: We can use FTP/SFTP to transfer large files.

Compression: If it is possible to compress file, we can use this tech.

Q5: Investigate how to implement realtime Chat feature though API

A5: We can use WebSocket, WebRTC and XMPP communication protocols.

If Chat feature is not professional and we are using AWS and Cognito User Pool, I will implement using AWS AppSync and GraphQL to implement chat feature.

I think firebase(firestore real-time database) is also good to implement simple chat app.

Q6: Investigate how to implement APIs for mobile app? Will it different as web app?

A6: It will depends on specific requirements but I think it is good to use different api-gateway s for web and mobile APIs because mobile app has unique and different features such as network bandwidth, network connection(maybe offline support), device(camera or GPS, or push notification) and so on.

Q7: How do you design a background process run once per hour?

A7: I will do this process using cron job. If web framework supports Scheduler, I will use scheduler.

Auto Testing/Auto Deploying: We can use Jenkins or the other services to implement fully automated CI/CD pipelines but we can do this simply by using AWS CodePipeline, CodeBuild and CodeDeploy.

At the CodeBuild stage the test will be done and if passed, it will deploy or approve merge.

We need to write appspec file and buildspec file to do auto deploy/build(test).

Data Backup: We can do this simply using AWS services such as RDS Console.

Or I think we can make our own back up logic by executing script file.