|  |  |
| --- | --- |
| **Name:** | **Lim Chee Seng** |
| **Email:** | **lim\_chee\_seng@mpa.gov.sg** |

**Question 1**

1. Name a (or more) cryptographic algorithm you would use to perform the following
   1. encryption

Triple DES (Data Encryption Standard)

Advanced Encryption Standard (AES)

RSA

Blowfish

Twofish

* 1. non repudiation

Hashing + sign with private key - MD5, SHA1, SHA256

Asymmetric Key Encryption

* 1. no tempering

Hashing + sign with private key - MD5, SHA1, SHA256

1. Name the following JWT registered claim names (see <https://tools.ietf.org/html/rfc7519#page-9>)
   1. unique JWT identifier

4.1.7. "jti" (JWT ID) Claim

* 1. cannot be used before a certain date

4.1.5. "nbf" (Not Before) Claim

* 1. issue date

4.1.6. "iat" (Issued At) Claim

* 1. token recipient

4.1.3. "aud" (Audience) Claim

* 1. mobile number

this is a claim (goes into the payload), proprietary field

**Question 2**

You are developing a hotel reservation application. After your user have successfully booked a hotel, the application can (opt in) update a user’s Google calendar with the stay’s detail and alerts. The application needs to create, update and delete calendar entries.

What are the required steps to allows the reservation system to update the a user’s calendar?

1. Create project in GCP
2. Enable API Services for Google Calendar in GCP
3. Copy the created API key

**Question 3**

You and a few friends have co-founded a hot social media startup. Like any good social media startups, you will need a new feed. A news feed is a list of post that is constantly updated with stories, activities, polls, etc from your friends. A post content includes the poster, text, images, videos, simple questionnaires, links, locations, etc.

The post will also include likes, the number of people reacted to it.

Two REST endpoints have been designated for users to publish and retrieve their feeds.

Publish a post

POST /api/v1/feed/me

Retrieving a feed

GET /api/v1/feed/me

The endpoints are secured with JWT.

Each user of your social site can have up to a total of 1000 friends/followers. You anticipate 5 million daily active users with about 70% of them posting at least 1 post.

Design a system that will support your REST endpoints along with the given requirements. Be as details a possible with your design.

1. System architecture to support this API

Request

Post / feed

Ingress

LB

Post

Post

Post

Post

Media

Text

Post / feed

Media goes to the S3 Object Storage

Document

{poster:”vid”

Date: “”

Tedxt

Attributes {happy…}

Media

User

User

User

User

Following

follow

Graph DB e.g. neo4j

(user {name:”fred”}) – [Following {circle=family}]->(user{name:”barney”})

match(u:user{name:”fred})-[FollowingP+{circle:”family”}]->(u:user)

Return V

Redis

Post\_id

Like:happy:….

1. Request comes in
2. Ingress with check for POST feed and send to load balancer. Then send to POST. Post has information of person, contect (media & text) and properties.

Media Id: Media goes to S3 bucket object store. Need to know the Id to communicate with e CDN.

Post Id: Text stored in Jason format, poster as user id, date, text and other attributes

{

poster: “vid”

date: “---” text

attributes: {happy,geography}

media:[------]

}

Redis

post\_id

like:

happy

1. Drop the combination of poster and posterId in the queue.
2. Group of users can access from the queue. Users have the feedlist and put to feed.

After getting the post, find the followers and post to them. Another method is when the user access the account, then look for user whom he follow and pull the post.

Alternate Answer:-

1) Have 1 system to handle all POST request and another to handle all GET request. Each system have a LB to distribute the load.

2) Store the medias in a S3 store that is behind a CDN

3) Store the document in a separate server which has a media ID reference the media in the S3 store

4) Store the “likes” and comments in redis

**Submission**

Copy this Word document to your repository and commit it.

git add .

git commit -m ‘worksheet03’

git push origin master