I have applied my course understanding to improve the application and wrote a report, based on the report that was borrowed form **2022 CSCI-5409 Cloud Computing Course**, the application was developed by Shathish Annamalai (B00886546), Gandhi Rajan Mahendran (B00892974) Khusboo Patel (B00878765)[29]. This is the Previous Architecture diagram *Figure 1*.

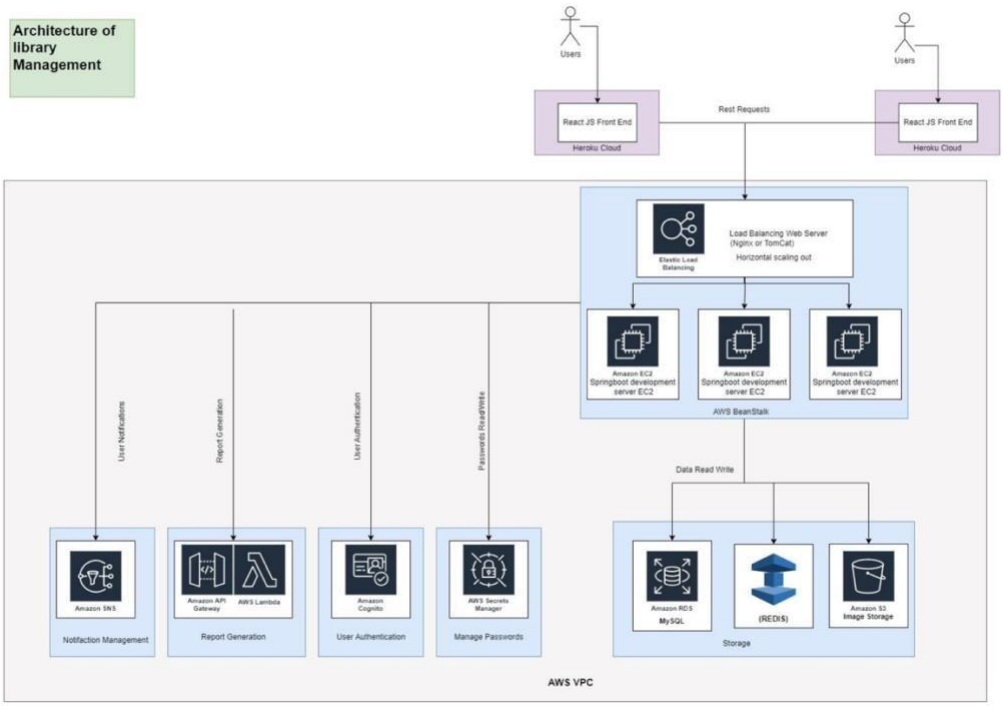

Figure 1: Previous Architecture

List of Services Previously used:

SERVICES	TYPE
1. Cognito	Identity Management
2. Secret Manager	SECURITY
3. S3	STORAGE
4. SNS	NOTIFICATION
5. Elastic cache REDIS	DATABASE
6. RDS MySQL	DATABASE
7. VPC	NETWORKING
8. API Gateway	NETWORKING
9. Load Balancer	COMPUTE
10. Lambda function	COMPUTE
11. EC2	COMPUTE

12. HEROKU	COMPUTE

Proposed Architecture and Services:

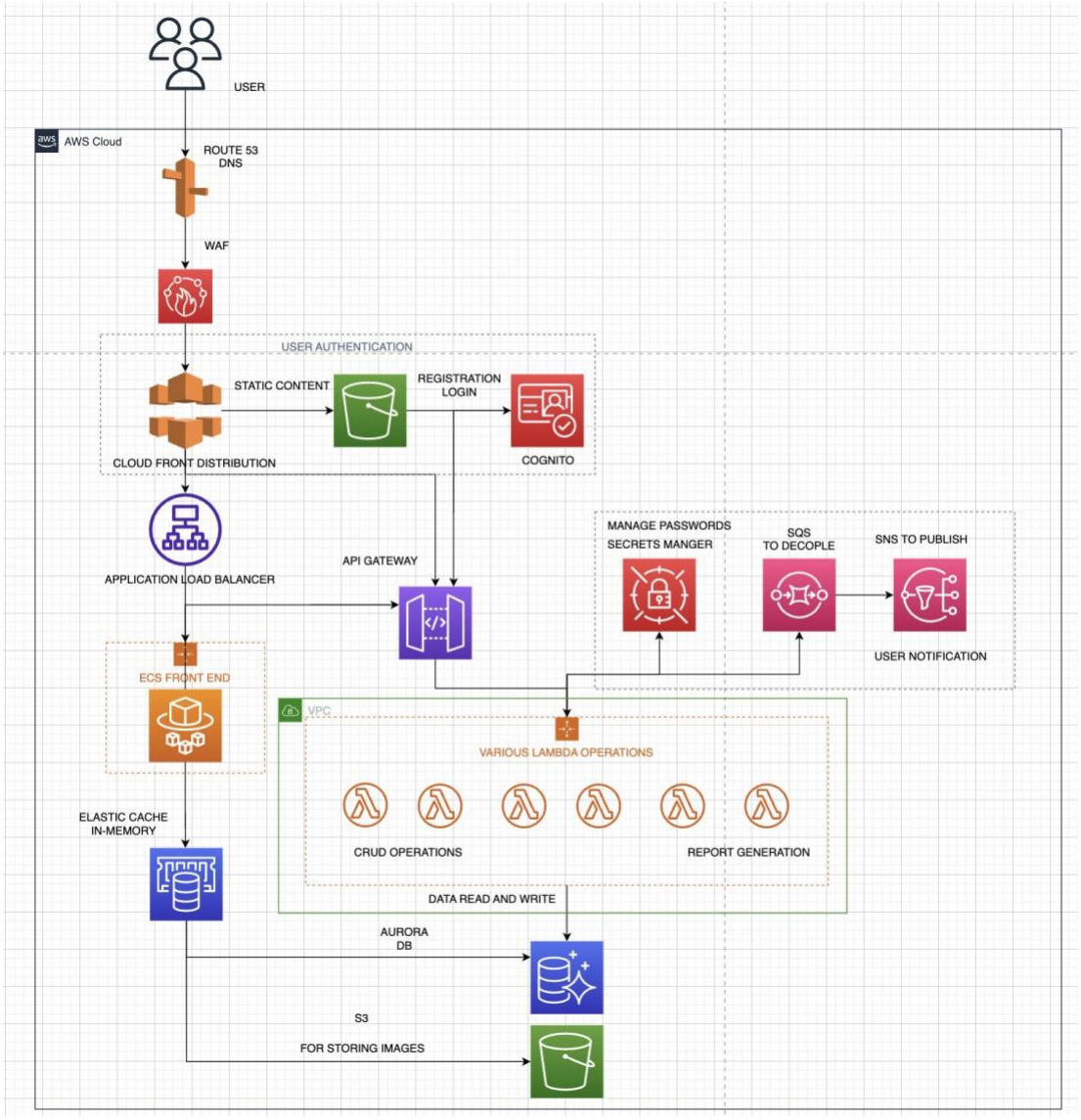

Figure 2: Proposed Architecture

SERVICES	TYPE
1. Cognito	Identity Management
2. Secret Manager	SECURITY
3. S3	STORAGE
4. SNS	NOTIFICATION
5. Elastic cache REDIS	DATABASE

6. AURORA MySQL	DATABASE
7. VPC	NETWORKING
8. API Gateway	NETWORKING
9. Load Balancer	COMPUTE
10. Lambda function	COMPUTE
11. FARGATE	COMPUTE

SECTION 1 – STORAGE:

1. What cloud services did you use?

For Storage I primarily used **S3 bucket**, also I used **Cognito and Secret Manager** though they are **not storage services**, they store some part of my application data, that is the reason I am mentioning under storage service.

NOTE: Though Cognito and Secret Manager are not storage services, they store some part of my application data, so I mentioned them under storage section.

a. Why did you use those cloud services?

I used Cognito which is a **user management service**, I used it in my application for **Sign-Up and Login authentications**. As it stores user data in **user pools**, which are the directories for authentication. So these user pool stores my user's *credential like username, password, address, phone number* and so on [1].

I have also used **Secret manager**, which is a **security service**, I used to *store my API, DB credentials and secret passwords*, it is the right way to do it instead of **hardcoding the values**.

Also I have used **S3 bucket**, where I will store my **book images/covers**, each book has cover picture, instead of storing it in Database, I stored it in S3 bucket, and will pass the URL to the database.

b. After reviewing your project, are you going to use the same services? Why?

Yes, **I will be using the same services**, the previously used services are the exact services to manage the storage, as for S3, images are typically larger in size, each image can vary in size as well, so storing in s3 seems more efficient than database, also considering the performance, retrieving image directly can impact performance when there is a large volume of data. S3 is more **scalable and quicker for image retrieval**.

As for Secret manager, the passwords and credentials need to be always protected, secret manager sole purpose is to secure the sensitive information. With the **fine-grained access** to manage and built-in functionality of rotating secrets, gives me secure storage with less management from my side.

Cognito offers built in storage for authentication, this saves me a lot of time because it automatically stores the user data and verification is also performed. Also with additional features like MFA, Integration with third party services it is very suitable for the application. It is fully managed, which makes my work easy.

c. If you didn't use any services in this category, why? If you think know you will use some services in this category, what are they and why?

I am not using any new services as the existing services are all the capable of performing the exact need for the application. All are being serverless it well fits in my architecture. **As I am building a serverless architecture**, these services go hand in hand with my application.

2. Did you apply any security measures to the services?

So apart from Cognito and Secret manager which are the mainly Security and User Authentication services, I didn't apply any security for S3 buckets previously.

a. If yes, do you think your security measures were enough? Why? If your answer is no, what new security measures are you going to apply?

No, I haven't applied any security services previously. However, I will apply security in S3 and Cognito to make the data more secure.

b. If no, why? What will you do to make your services secure?

I will make my S3 bucket more secure by implementing new security measures like first I start with least privilege principle, with help of IAM I will create user polices to **grant minimum permissions required** to access s3 bucket, this will help reduce the help of *accidental data exposure[*2]. I will also make use of bucket polices to restrict access to s3 bucket, will only allow respective admin or developer to access[2]. Most Importantly I will enable **server-side encryption** for the S3 bucket to encrypt all objects stored in the bucket using *KMS to manager keys*[2]. For additional security, *enable encryption at rest and in transit using TLS*[2].

As for the Secrets Manager, by **default all the credentials are encrypted** with a unique key, however for additional safety I will follow same **least privilege principle by IAM policies** to control access to the secrets stored in AWS Secrets Manager [3]. Also will enable secret rotation as they will help me reduce the risk of compromising [3].

For Cognito I will start by implementing strong password policy, along with account verification with email, also make use of MFA, which is the better way to secure the users login. More importantly I will use TLS encryption to protect data during transit[1].

3. Was your application monolithic?

The Storage services are independent in nature, they are not monolithic, the storage services perform single operations as they are not dependent. However, we can further decouple them to achieve more maintainability.

a. If yes, how will you decouple the components in your solution/architecture?

No, the components are already decoupled, and they have their own set of individual operations to execute when required.

b. If no, can you further decouple the components in your solution/architecture? Why and how?

We can further decouple the components to enhance the reusability, and it will be easy for us to maintain the services as they are not dependent. The purposed architecture is to make them **serverless so it will be event driven**, every storage resources can be independent, and will be used based on the events, we can make use **S3 Event Notification to trigger** when there is an event, as for Cognito we can use user pool triggers to execute lambda codes for user authentication. By using **asynchronous communication** system like SNS and SQS we can decouple the events between services, like example instead of *lambda fetching data from secrets manger, it can use SQS to retrieve the message from secrets manager.*

This services and plans can make the architecture loosely coupled, and promote reusability, so for further updates it would be easy.

4. Did you have a disaster plan?

a. If yes, can you further improve your disaster plan? How?

No, for my previous application I didn't have any disaster recovery plan, as it was not part of the requirement.

b. If no, how will you make your disaster plan?

For S3, my strategy would be first to ensure the **versioning** is turned on, so that it will have the previous version of my data. Also will apply **cross region replication** on the s3 bucket, so in case of disaster, the data will be still present in another region. More than that, I will implement AWS **Backup** to back up my s3 bucket, which indeed will create a s3 bucket with encryption. Along with this I will create a documentation of step-by-step procedures for disaster recovery[4].

Disaster recovery for user registration using Cognito, we can make use of Cognito User Profiles Export, where it *periodically exports user profiles, groups, and group memberships from an Amazon Cognito user pool in a primary AWS Region to an Amazon DynamoDB global table in the same Region* [4]. This way we can get the backup of data in the database.

For credentials in secrets manager we can plan on replicating them to another region with the option of *Replicate secret to other region* [6].

Note: The replicated secret will be promoted to standalone secret [6].

5. Were the services highly available and reliable?

Yes, the services like S3, Cognito, secrets manager are highly available by default.

a. If yes, can you further improve the availability and reliability? How?

S3 is designed for high availability, it has *99.99% availability*[7], by default it is resilient and resistant to disruption. To improve the high availability I can make use of cross region replication or same region replication, versioning, and store the data in multiple AZ's. Even further I don't think this is necessary for the application, but we can *use Multi-Region Access Points to get the benefit of Global Accelerator*, where with one endpoint we can access multiple buckets from different AZ. This provides more availability and low latency.[7]

For secrets manager we can implement replication of data to other region, and frequent rotation of the key will provide us more availability. With minimum set of access to IAM we can ensure the key is highly available.

From Cognito user pool we can copy the data and store it in S3 for high availability, as S3 is highly available than any other aws services, it would be safe.

b. If no, how will you make the services highly available and reliable?

My services are highly available, also I have mentioned few additional approaches which will make my services even highly available.

6. Did you consider the cost of the services?

Yes, I did consider cost for my services and had a rough estimate for initial and monthly expenses. **However the costs where not specific to Storage service**.

a. If yes, can you further improve the cost effectiveness? How?

As I mentioned, the cost for S3, Cognito and Secrets manager were not explicitly covered in our previous application.

b. If no, how will you make your services more cost effective?

To be cost-effective I will set our Storage class to the **Intelligent tier** for the first few months, this provides *free monitoring from AWS*, and the data will be moved around different storage classes automatically based on the access. Once we know how frequently the data is accessed, we can create **lifecycle rules** to move data. We can start with the **S3 standard** as it has more availability with high throughput and low latency it will be useful in content distribution. Going forward we can move to S3 **standard-Infrequent access** as it will be our backup option in case of disaster and files less accessed will be moved here. Further, we can move our old files to **S3 Glacier and deep archive.**

The best practice to be cost-effective is to follow the lifecycle rule, remove unused data, generate S3 inventory, compress huge files, eliminate incomplete multi-part uploads, and delete non-current versions.

As for Secrets manager it charges for number of secrets, requests and amount of data stored, I can leverage the free tier for the limited set of traffic, as the application grows, I can change to standard plan It ranges from $0.40 per secret per month to $2.00 per secret per month[8]. This would be sufficient to handle the traffic and it is cost effective.

Cognito charges for Identity management and data synchronization, so to be cost effective, first I can make use of free tier which gives me 50,000 monthly active users. This will be more than enough for my application. In case the user limit exceeds only pay $0.0055 per monthly active users[1].

NOTE: We should make use of the free tier services to our advantage, so that we can be more cost effective. Also we should make use of aws cost allocation report, set a budget, cost explorer to our advantage, these services are useful to figure out the cost and monitor them.

7. Did you use monitoring and/or logging tools to give you graphs, messages, metrics, or alerts of the services?

No, I didn't use any monitoring tool specifically for storage components, to check or log the storage service.

a. If yes, can you further improve the monitoring and logging in your solutions/architectures?

No for S3, Cognito and Secrets manager I didn't use any monitoring, as they were not essential in the application. However, it is always good to monitor the services we use, so I will apply some mechanism for monitoring.

b. If no, what monitoring and logging tools will you apply? Why?

For monitoring and maintaining the features first, I will use **Tags** to distinguish the buckets, so it is easy to track. We can use *cloud watch to monitor the logs and health of S3*. Moreover, we can utilize *the Cloud Trail to track our activities*. S3 Event Notification comes in handy when an event is triggered, as I am planning on implementing serverless architecture this will come in handy.

For Secrets manager I can leverage the cloud trail to access the logs, this will give me the API activity, also any CRUD operations performed on the secrets will be recorded, I can set up a cloud watch rule to trigger when there is any event happening.

Similarly we can use Cloud watch to check the user pool for Cognito, also schedule a trigger for events that can impact the application. Also leverage Cloud trail to capture API calls and even catch IP address in the request[9].

Note: Cloud watch and Cloud Trail these both services can be more helpful to discover the errors and logs of our application, moreover they give us metrics, alerts, graphs all in understandable format.

SECTION 2 – Compute:

1. What cloud services did you use?

I have previously used **Elastic Beanstalk for hosting my backend and Heroku to host my front end**. In elastic beanstalk we have used EC2 instances with horizontal scaling, as for front end we have implemented NGINX webserver. And added Load balancer to handle the incoming traffic.

a. Why did you use those cloud services?

I have used Beanstalk because it is easy to manage, *developer friendly*, and makes it easy to deploy an application without much work. As we can configure the deployment in UI with very few steps, also with the java platform versions, it made my work easier. Moreover, with the benefits of automatically handling the capacity provisioning, load balancing, scaling, and application health monitoring[10].

To be specific I have used EC2 T2 micro instance in beanstalk, as it comes under free tier, also our application was not used in real time it made sense to stick with free tier.

The previous team suggested using Heroku as part of *multi cloud environment* would increase the *high availability* also at the same time it is simple and easy-to-use interface for deploying your front-end application[11]. It is very similar to beanstalk, as in Heroku also we must upload our code, it will automatically handle the deployment, scaling, and monitoring of our application

b. After reviewing your project, are you going to use the same services? Why?

No, after thinking about the deployment approach, I am planning to **completely move to serverless architecture**. As the previous deployment models is very developer friendly, but we need to maintain sthe underlying infrastructure like EC2, load balancers and *there are security patches, updates, and backup, this will take significant amount of time.*

As for front end deployment, Haruko is very user friendly, and it provides multi cloud environment. However with multi cloud there is **security concerns and integration problems**. It also provides *limited customization and limited scalability*. Considering all these things in mind, also with today's tech moving **towards serverless architecture**, it would be wise to make use of serverless services.

c. If you didn't use any services in this category, why? If you think know you will use some services in this category, what are they and why?

So as I mentioned on using Serverless architecture, I will use **AWS Lambda to perform my backend works**, it will be an **event driven architecture**, with help of *API Gateway* I can configure my lambda with my front end. So basic workflow will be frontend to API gateway and to lambda. So lambda is single unit of code which follows *single responsibility principle*, I can have many lambda functions to perform different operations. This way I can focus on writing code and stop worrying about infrastructure, it **scales and handles traffic large amount of traffic automatically also with cost effective of pay for the compute time option**. In addition, with the flexibility of using different programming language and integration with other aws resources.

Front end will be deployed using **ECS Fargate**, it is a *serverless compute engine for containers* where we can containers without having to manage servers. This means I don't have to manage infrastructure. Also The reason to opt for *containerization resource is because of the portability and security it offers to the front end*. In future I can use *Kubernetes to orchestrate the containers* if necessary. ECS Fargate gives me **fine grained control over scaling and resource allocation** to the containers[12]. Moreover, the cost effectiveness and integration with other services provide are quite useful.

2. Did you apply any security measures to the services?

Yes, we did apply some level of security measure to our compute services, however, we could apply more security to ensure the data security.

a. If yes, do you think your security measures were enough? Why? If your answer is no, what new security measures are you going to apply?

No, I think the security measures weren't enough, we have previously used VPC to restrict the services to public and private subnets based on the requirement, like web server in public subnet, database, cache in private subnet. Also security groups are used to restrict the traffic into the backend server, by giving access to certain ports following least privilege principle. I will add

some additional security measure to make the infrastructure more secure. **The services I am planning on using are NACL, VPC Flow logs, IAM and WAF.**

b. If no, why? What will you do to make your services secure?

To make it more secure we can implement **NACL at subnet level** so that we can control traffic between subnets and *restrict access based on IP addresses and ports*. This provides additional layer of security for out VPC. It is stateless we must explicitly set both inbound and outbound rules. With **VPC flow logs we can capture the IP traffic passing through our VPC**, this is very valuable for monitoring and analysing. This can additionally help us with compliance requirement as well. Also we can use *separate route tables for every subnet*, this way it would be easier to manage the traffics.

We should use **HTTPS by default** to encrypt data in transit between the user and the front-end application. We can use **Web Application Firewall** (WAF) this can protect our application from malicious attack or any SQL injunctions. These security measure will make our compute services more secure.

3. Was your application monolithic?

I would not say it is completely monolithic, it is a combination of both monolithic and micro services, most of the **services are distributed and independently deployable**. However, we can make the service completely micro service and loosely coupled.

a. If yes, how will you decouple the components in your solution/architecture?

Some parts of the services are dependent coupled in code wise, we can decouple and distribute them using AWS Lambda and API gateway, we can write **separate lambda functions for all operations**, this way we can follow the design principles (SRP) and with *API Gateway we can create micro service architecture that can be deployed and scaled independently*. All the operations will have their own lambda which will be **triggered based on events**. This way it will be cost effective, as only necessary code will be executed and the rest will be idle. Moreover we can use **CICD to automate the deployment pipeline** so it will be easier. We can make use *SQS queuing service to build loosely couple architecture.*

Also in the previous architecture, I am using load balancer for web traffic, we can also **use elastic load balancer to between services, as it will make our system decoupled.**

b. If no, can you further decouple the components in your solution/architecture? Why and how?

The above-mentioned method will help further decouple the components in my architecture, this will make loosely coupled and more distributive architecture.

4. Did you have a disaster plan?

a. If yes, can you further improve your disaster plan? How?

No, for my previous application I didn't have any disaster recovery plan, as it was not part of the requirement.

b. If no, how will you make your disaster plan?

I can schedule a **regular backup**, so that application data can be backed up parodically. We can use environment's configuration offered by Beanstalk that can define the environment platform[13]. We can additionally take *snapshots of the used resources*. Also deploying our beanstalk in multiple AZ will be a great benefit for disaster recovery, this will ensure even one *AZ fails another is there to back it up*. This will beneficial. Also scaling and load balancing can improve the availability.

In case of serverless architecture, our **lambda functions can different versions and alias**, it will be useful for testing. Moreover *lambda is span across all AZ in the region*, so it will make disaster recovery easy and cost effective.

5. Were the services highly available and reliable?

By *default Elastic beanstalk with EC2 instances* are designed to be *highly available and reliable*, as it is running our application in multiple availability zone to ensure high availability. However, we can still add few configurations to ensure more availability and reliability.

a. If yes, can you further improve the availability and reliability? How?

So when we create beanstalk, it *automatically provisions resources like EC2, ELB, ASG in multiple AZ*, this way it will distribute the workload and make it as a backup in case of disaster[14]. Elastic Beanstalk also **monitors the health of our application**, and it automatically replaces the unhealthy instances to maintain the availability of your application. This automatic monitoring and replacement of instances help us to ensure that our application remains highly available and reliable[14].
To improve high availability we can deploy in **multi-region deployment and use cross-zone load balancing**. By use of **auto scaling polices** we can set the desired, max and min instances to scale up and down the instances based on traffic.

However, if we use the serverless architecture, lambda is by **default highly available and reliable**, it is designed that way. Lambda automatically *replicates our code across multiple availability zones within a region to ensure high availability and fault tolerance*. This way requests will be automatically routed to the healthy instance in the region[15].

b. If no, how will you make the services highly available and reliable?

From the above configuration we can make the service more available and reliable. This will help us in case of high traffic, disaster, up time.

6. Did you consider the cost of the services?

Yes, I did consider cost for my services and had a rough estimate for initial and monthly expenses. However the costs were just approximate estimate based on the server racks online, we had initial of 2000$ approximate for web servers for the application hosting and Load balancing server (Nginix or Tomcat) were 2000$ approximate. It is not accurate; it was based on previous reequipments.

a. If yes, can you further improve the cost effectiveness? How?

At first, we can set up the **cost explorer** to gain insights on the spendings of the resources, from that we can make use of Trusted advisor which will help us optimize by choosing right-sizing EC2 instances, optimizing load balancer configurations, or **leveraging Reserved Instances**. Also with compute optimizer we can analyze the utilization of our EC2 instances and receive recommendations on instance type and size optimizations.

So it is better to use **spot instances when we have non-critical workload**, however using reserved instances can be helpful, also it provides significant discount. We can also schedule scaling so that it can help optimize costs by ensuring that you have the right capacity at the right time.

In serverless architecture, **we follow pay-per-user pricing model**, this is like we get charged when our code runs. As it is *event driven this will reduce lot of cost, as it is not running all the time.*

b. If no, how will you make your services more cost effective?

Above mentioned cost optimization can be beneficial, and it will reduce the significant amount of cost, by choosing right instances, balancers and Auto scaling.

7. Did you use monitoring and/or logging tools to give you graphs, messages, metrics, or alerts of the services?

Yes, we did use **CloudWatch to monitor the elastic beanstalk**. However, we can still make use of some other services, to enhance the monitoring for the serverless architecture.

a. If yes, can you further improve the monitoring and logging in your solutions/architectures?

In the previous architecture we have used cloud watch to monitor the logs and set metrics for scaling, in addition to that we can make use of elastic search it provides advance logging capabilities, with this we can analyze and visualize logs.
For the proposed serverless architecture, we can use cloud watch to analyze the lambda responses, we can **create custom dashboard, alarm, and visualize metrics**, in the monitor tab we can see recent, highest invocation of the lambda and view the metrics in detailed level like error count, throttles. Additionally we can make use of *X-Ray and ServiceLens* to get end to end view of our function. These AWS lambda Insights helps us analyse and monitor the application code in granular level.

For alerts we can even configure SNS and send various type of message to alert, this is very useful, we can add it as endpoint in our lambda.

b. If no, what monitoring and logging tools will you apply? Why?

I believe cloud watch can play an essential role in monitoring and logging the necessary information and we can analyze them in granular level to get insights, so above-mentioned services are enough to improve the cloud monitoring.

SECTION 3 – DATABASE:

1. What cloud services did you use?

For database layer, I have **previously used Amazon RDS MySQL and Elastic cache**, as these services helped us to enhance performance and scalability.

a. Why did you use those cloud services?

The main purpose of **using Elastic cache is used to improve the performance and reduce the load on backend for frequent request**. As Elastic cache is fully managed, in memory caching service, it gave us less read and write time, *which indeed improved scalability*. Initially we setup and RDS instances (MySQL) and created Elastic cache with same DB configuration, so whenever the application calls, *it will first check in memory store, if there is nothing, it will go to RDS database.* This will indeed reduce number of queries made to the database.

We used REDIS as our Elastic cache because this helped us providing faster response time and performance efficiency. As it as highly available in-memory cache which decrease data access latency, increase throughput, so it took significant load of the database.

We used RDS MySQL because, we required a relational database which required scalability and security. The main advantage of using RDS is it goes hand in hand with REDIS Elastic cache service also, we can it is easy to setup and cost effective. As we were using Spring Boot application, we just had to give the credentials in the code.

b. After reviewing your project, are you going to use the same services? Why?

No. For database layer I won't use the same services, As I am planning on building a serverless architecture, I would change the traditional RDS database to **MYSQL AURORA Database**. However, I will still use the **REDIS Elastic cache** as it does perfect job managing the in-memory layer. *The reason for changing the database is, it is serverless as it is fully managed database service, and it is compatible with MYSQL*.

c. If you didn't use any services in this category, why? If you think know you will use some services in this category, what are they and why?

As I previously mentioned I am going to use **Aurora MYSQL instead of RDS** and will keep the REDIS Elastic cache as it is. So I not going to change the cache mechanism as it is required and makes the application to work well with handling large requests. This is *increasing our read performance*, also works well with serverless architecture, so changing it doesn't make sense.

As for AURORA, it is expensive than RDS. however, instead of using DYNAMO DB which is NOSQL, AURORA as relational suits well for Library management application as the relationship *between the entities are essential*. Moreover, aurora gives us so many benefits like improving the performance and scalability, works well with distributed architecture for handling high read and write. Also the scaling of storage till 128 TiB, and the high availability it provides by replicating across multi-AZ along with automated backups. Overall I think when the application grows aurora will fit in for handling large audience.

2. Did you apply any security measures to the services?

Yes, we did have some default security measures for our Database Layer. As RDS and Elastic cache provides default security. We have restricted the traffic to database from selected ports, also we have placed our database layer in private subnet. Only traffic from EC2 instance can access the database, no internet and public access is given to DB layer.

a. If yes, do you think your security measures were enough? Why? If your answer is no, what new security measures are you going to apply?

I believe the previous security measure are good enough. However, when it comes to security, we can always add one or two additionally security measures to be secure.

By default *the RDS provides encryption of data in rest and in transit*, we will be managing the key using KMS[16]. We are also following the **least privilege principle**, so by giving limited access to IAM policies. As we placed our database in private subnet in the VPC, it gives isolation within our area. We also have security group for our instance to limit the traffic in inbound and outbound[16].

Similarly for Elastic cache by default data is protected in rest and transit, like **communication** with other aws **resources is done using TLS**. And by limiting the IAM polices we are maintaining the creation and modification of the cache resources[17].

b. If no, why? What will you do to make your services secure?

However we can further improve the security for both RDS and Elastic cache. For RDS as it is in private subnet, we can setup NACL to act as a firewall. We can use guard duty to protect our the RDS instance[16].

As we are proposing for serverless aurora database, we can perform the same things as we planned for RDS. As aurora is built in top of RDS, they share almost same security services, however for aurora we can make use of Backtrack and end to end encryption.

For elastic cache instead of using Redis AUTH command, we can use Role based access control, because this gives granular level security by creating and assigning users with specific permission, We can *add cache cluster to the virtual network, and control access to the cache cluster by using Amazon VPC security groups[17].* This is will indeed give additional level of security. We can additionally use AWS compliance resources or audit manager to manage the risk and compliance[17].

3. Was your application monolithic?

Yes, at some level the database layer connected with application layer can be considered as monolithic, because elastic beanstalk we previously used is handling all the backend and database responses, so it can be considered as monolithic.

a. If yes, how will you decouple the components in your solution/architecture?

Yes, we can decouple this architecture, as per my proposed serverless architecture framework we are using **lambda instead of beanstalk**, so that means we can breakdown the functionalities into micro services like smaller application. So lambda will be invoked by event driven approach that means with help of API gateway calls we can **direct to other services like database, asynchronously**. This will lead to loosely coupled, for asynchronous we can make use of *SQS, but this is good approach for non-critical workloads* like may notifications to the user. We can further make use of *SNS and Event bridge to prevent coupling and direct interactions between services.*

More importantly we can implement **load balancer between application and database layer**, this way the traffics will be automatically scaled across, also gives high availability. The separation

of data layer, where lambda will be responsible for specific tasks and interacts with the database and cache services through APIs or events. This way we can achieve decoupling.

b. If no, can you further decouple the components in your solution/architecture? Why and how?

Above mentioned decoupling solution can be implemented to achieve the decoupling required for the application, as these decoupling methods can be useful for further updates.

4. Did you have a disaster plan?

a. If yes, can you further improve your disaster plan? How?

No, for my previous application I didn't have any disaster recovery plan, as it was not part of the requirement.

b. If no, how will you make your disaster plan?

If we consider the serverless architecture, we use aurora database, which has several disaster recovery options, we can use **aurora automatic backups**, which will take continuous backup and store it in s3 which is highly available, also we can enable **Point in Time Recovery** (PITR), with this we can restore the database at any time. But the better option would be using Global database with this we can create cross region read replicas, this will act as primary database in regions in case actual DB fails. We can also enable Backtracking just to be on the safter side.

Similarly, the Redis Elastic cache also should have disaster recovery plan, we can use automated **backups or snapshots**, but it has a retention period. We can setup a *multi-AZ replication* this way our in-memory database will be highly available, as data is automatically replicated to standby by DB, also if primary fails, this will take over. However we can also use *cross region replications* using *standby replica* in another region for high availability. This is not required, as AZ wise disaster recovery will be more than enough, and it is cost efficient.

5. Were the services highly available and reliable?

The previous application wasn't specifically built to be highly available, however **RDS provides default availability**. But we can implement features so that the services can be highly available and reliable.

a. If yes, can you further improve the availability and reliability? How?

From the below configuration we can make the service more available and reliable. This will help us in case of high traffic, disaster, minimizing down time.

b. If no, how will you make the services highly available and reliable?

So initially we can set up our **multi-AZ deployment** with a standby database which will provide synchronous data flow, The good thing is both databases will share same **DNS endpoint** so if the primary DB fails, the traffic will be **directed to the standby DB**, this will minimize the downtime[18].

We can **also setup read replicas to direct all the reads**, so this would reduce **some downtime**, also we can leverage **RDS Proxy which can help minimize downtime** and improve database efficiency. It reduces failover time by 66% [18]. For Relatability, we can use automated backups, transaction logs and PITR.

But in case of Serverless aurora, it offers more high availability and reliability. Aurora stores data across **multiple AZs in a single AWS Region**. *Aurora synchronously replicates the data across AZs to six storage nodes*[19]. Also we can create up to 15 read replicas. Also it automatically fails over to a standby replica in case the primary DB instance becomes unavailable[19]. We can also leverage the Global database for region level high availability, as it spans across multiple regions enabling low latency global reads. The same RDS proxy can also be used here for failover time.

We can also make use of automated backups and snapshots with a retention period this can be useful.

For Redis Elastic cache to improve availability and reliability, we can setup multi-AZ deployment which will automatically **create read replicas up to 5**. We can also partition the data up to 500 shards. Sharding can be useful when we have large dataset. We should enable the cluster mode to use it. We can create a **replication group and scale the read replicas** based on our usage[20].

We can further use backup to restore the cluster, which will be saved in s3. For this we need to turn off the cluster mode.

6. Did you consider the cost of the services?

Yes, I did consider cost for my services and had a rough estimate for initial and monthly expenses. However the costs were just approximate estimate based on the server racks online, we had initial of Servers for Database nodes – (2000$) approximate, Server with high memory for Redis – (5000$) approximate.

a. If yes, can you further improve the cost effectiveness? How?

Yes, as we are using serverless aroura we can we improve the cost effectiveness of the databases, first we need to monitor the performance and cost usage of how the data is, so we use **cloud watch, Enhanced monitoring, or Performance insight** any services to understand the cost flow[21]. Aurora is *pay as you go model*, the main cost can occur from *capacity units and storage size and*

IOPS consumed (read and write), additionally the **backup and data transfer** can cause cost. In case we are using the **global database**, backtrack features we pay for it[21]. There is also auto scaling that requires cost too. So to be cost effective we can use tools like cost explorer, pricing calculator to calculate the cost beforehand. We can choose **auto-pause and resume option** this will help us reduce cost when the database is idle. We should set minimum and maximum scaling policies so that we *don't waste cost on un-utilized resources*. We should enable auto scaling in the storage, so we don't have fixed storage[21]. These things can help improving cost.

Note: Aurora is expensive compared to traditional RDS, but it is more powerful and highly scalable.

As for elastic cache we need to determine the *optimal size of the cache nodes*, which means we need to analyze the application workload first, then we can implement right type of instance, as for our application we can choose general purpose instance, as it is cost effective and handles the workload well[22]. M instance family is suitable and provides low-cost high throughput. We can also improve cost by choosing data tiering which **reduces almost 60% of cost saving**[22]. It uses SSD so we can have least used data in SSD this way we can be cost effective[22].

b. If no, how will you make your services more cost effective?

Above mentioned cost optimization can be beneficial, and it will reduce the significant amount of cost, by choosing right instances, backups and Auto scaling.

7. Did you use monitoring and/or logging tools to give you graphs, messages, metrics, or alerts of the services?

Yes, **we did use CloudWatch to monitor the elastic cache** . However, we can still make use of some other services, to enhance the monitoring for the serverless architecture.

a. If yes, can you further improve the monitoring and logging in your solutions/architectures?

To monitor the aurora database we can use cloud watch, this is an obvious necessity we can get *CPU utilization, storage usage* and so on. Then we can make use of *Performance Insights*, specifically used for monitoring and analyzing database performance. We should also turn on the enhanced monitoring while configuring the aurora, as it gives *granular level metrics*. For monitoring logs we can make use of **RDS logs and cloud trail to know the API calls**. Also mainly to monitor the real time database activity we can make use of **Database Activity Streams.** For monitoring the Threats we can implement **amazon guard duty** with RDS protection as well. These options are convenient for us to keep track of data and monitor the resources.

NOTE: By default the performance insight is turned on. It works on DB level. We can aggregate multiple DB's together.

As for elastic cache, we can use cloud watch, with this we *set threshold on metrics and trigger notifications*, in elastic cache we must monitor the health which can be done by analyzing the CPU

metrics, memory and network. We can make use of *engine-level metrics and host-level metrics*[23]. Other than that we can monitor elastic cache events which can be configured with SNS, this way we can know the events. Also by logging API calls with cloud trail is useful to determine the recent events.

b. If no, what monitoring and logging tools will you apply? Why?

I believe cloud watch can play an essential role in monitoring and logging the necessary information and we can analyze them in granular level to get insights, so above-mentioned services are enough to improve the cloud monitoring.

SECTION 4 – NETWORKING:

1. What cloud services did you use?

So for networking layer we have mainly used **Security groups and VPC to restrict** and **isolate** the resources.

a. Why did you use those cloud services?

These *security group act as a firewall to our EC2 instance*, both inbound and outbound rules are well defined so only restricted IP's can be accessed. As for VPC is a virtual network which helps to *isolate our resources*. We used these to restrict unauthenticated access to our servers. So initially we had 2 subnets a public and private subnet, so in *public we kept our servers which needed internet access*, and our *backend server*, *database those things are kept in private subnet*. This way we are not exposing the things that are not needed to be accessed by the users. Also by segmenting the resources help us protect the data efficiently. We have attached *route tables along to direct the traffic*. For accessing the internet we have kept an internet gateway in the public subnet. This way our servers in public subnet can access the internet. The main reason for networking is where each resource can communicate with a secure line, it can be within same VPC or different VPC.

b. After reviewing your project, are you going to use the same services? Why?

Yes, I will be using the same services, but I will add some additional features to enhance the networking and security at the same time. The main reason to keep these services are they are basic block of networking. Moreover, they are very essential. I will discuss what services I will use in the next question.

c. If you didn't use any services in this category, why? If you think know you will use some services in this category, what are they and why?

As I mentioned previously, we are using serverless architecture, we will enhance the networking by **creating own VPC**, **not using default VPC**. We will create subnets public and private, for each AZ. Only the **web layer will be exposed to internet using public subnet**, rest are in private subnet. We will also create NACL for VPCs it will act as a firewall along with security groups to

protect the lambda. Along with that we can create separate route tables for each subnet, and attach NAT gateway for private subnets, so other services can access internet using NAT.

The most important networking feature for serverless architecture is **API gateway**, they act as a **front-end interface to expose the Restful APIs** and they route the traffic to lambda functions. We can also leverage the Event bridge as a networking service to trigger and communicate with other services.

2. Did you apply any security measures to the services?

Yes, we did apply security measures for the services, as I mentioned earlier, we are using subnets, security groups to limit the access, for instance we only allow public access to port 80 and 443 to serve web requests and database servers are provided incoming access on 3306 ports.

a. If yes, do you think your security measures were enough? Why? If your answer is no, what new security measures are you going to apply?

We can improve the security measures by addition some additional layer of security and creating individual subnets and route tables for each layer, this we can assure more security. As we know the *NACL is stateless*, we will create rules for both *inbound and outbound to allow certain IPs*, and via route table we will communicate efferently with other services. Most importantly we follow the **least privilege principle** and *give limited access to users* who is accessing the services. Also instead of storing the values in environment variable we will store them in secrets manager or parameter store. We can implement **WAF (web application firewall) to protect against the web application attack**. We can use AWS Network Firewall to monitor and protect your VPC by filtering inbound and outbound traffic[24]. We can size our **VPC CIDR blocks** , we much ensure they are **not overlapping.**

Most importantly we use *HTTPS*, so that *hackers can't listen to our packets*. And initialize VPC Flow Logs to monitor the IP traffic going to and from a VPC, subnet, or network interface[24]. Then we *reserve some address space for future use*, as it is better not allocating all network address.

b. If no, why? What will you do to make your services secure?

I think the above security measure will be good enough for securing network layer of the application.

3. Was your application monolithic?

Yes, the **network layer was tightly coupled, because we have used default VPC and using default route table.** This can lead to cohesion. However, the security groups are loosely coupled.

a. If yes, how will you decouple the components in your solution/architecture?

To decouple the VCP, we can create our VPC and create separate route tables for all subnets we are creating, we must create one subnet for each layer both public and private, for example if there

is 3 layers with multi-AZ *we need 6 subnets, probably 2 public acts as web servers, 4 private which acts as secured from internet*. Each subnet should have own route tables, and NAT gateways, this way they won't have to depend on the other services. Same goes for security groups, creating individual security group based on the communication. We need to **avoid circular dependency**. This way we can have more modular and manageable security infrastructure[24].

b. If no, can you further decouple the components in your solution/architecture? Why and how?

I believe the above-mentioned features are enough to decouple the resource, if required we can create a separate subnet and put all essential resources there like a dedicated subnet or VPC. This enhances security and reduces the risk of unauthorized access[24].

4. Did you have a disaster plan?

a. If yes, can you further improve your disaster plan? How?

No, for my previous application I didn't have any disaster recovery plan, as it was not part of the requirement.

b. If no, how will you make your disaster plan?

We can implement a disaster recovery plan for our network layer, we can start with by backing up the *configuration of our VPC, subnets, route tables using AWS backup*, the better option is utilizing the *cloud formation scripts as they versioning*. Also *by enabling multi-AZ deployment* for our subnets we can ensure if one zone fails still all our traffic will be forward to other zone. In case to communicate with the VPC in other region we can use **VPC peering**, this comes with data *replication and failover capabilities*. Even if a region is out, this will help as the other region will take the current place. As in our serverless application we use API gateway often so to have high availability we can *implement regional deployment with caching* enabled this will improve performance as well. Along with this we can implement cross region EventBridge rule replication this will ensure event *routing and processing continuity in case of a regional failure*. These methods can be helpful to test our availability and resiliency of our application.

5. Were the services highly available and reliable?

So by default the *VPCs and security groups are highly available and reliable by default*, we have not specifically configured anything to ensure the high availability of the network layer in the previous architecture. However we can improve that in our new serverless architecture.

a. If yes, can you further improve the availability and reliability? How?

So initially when we create an VPC, it will be **spans across multiple AZ within same region**[25]. This helps us to ensure even on AZ fails, the another will be there for backup. Similarly for security

groups they are **automatically replicated and managed across all the AZs within the region**[25]. We can use load balancers *to distribute the traffic to different AZs*, this will ensure if one AZ fails another can handle. We can use **VPC endpoints, they provide high availability to connect privately as well**. Also by adding **Route 53 before the service is the better and ideal approach**, because when it comes to high availability route 53 provides the most. We can also add health checks to it. For communication between VPCs we can make use of VPC peering as they are highly available.

For API gateway, we can use regional deployment, as it distributes APIs across multiple regions, this way we can achieve availability and reliability. Caching can help reduce the load and improve reliability. The deployment stages come in handy to managed different versions. Moreover the *SSL certificates and custom domain names enhance the security and trustworthiness of API endpoints*, adding to their reliability[25]. We can also consider using Global accelerator for multi region deployment.

Elastic IP can be used, this way we can stick with same IP. However it is not recommended to use for all scenarios. They tend to incur more cost and have limited pool of access. Not used in serverless architecture.

b. If no, how will you make the services highly available and reliable?

Above mentioned methods can be quite helpful in making the network layer highly available and reliable, using multi-AZ compared to multi-region can be quite cost effective as well.

6. Did you consider the cost of the services?

Yes, we did consider the cost of the networking layer, However the services we used in the previous application all comes under free tier. However for our serverless architecture we can improve the cost effectiveness.

a. If yes, can you further improve the cost effectiveness? How?

So first the basic cost will come from data transfer charges, this may occur when the data is going out, AWS may incur charges in the range of $0.08-$0.12 per GB[26]. Though **VPC doesn't cost us, the services used inside will cost** us as we discussed in other sections, like database, S3, lambda. Apart from this the NAT gateway can incur charges, it can be either hourly or data processing charges it will be around $0.045 per hr[26]. Also to ensure secure network we will be using firewalls that can incur charges around $0.395 for *Network Firewall Endpoint Hourly Charges*[26]. For API gateway we will be using REST APIs which will cost around $3.50, but these are after the free tier. For caching we will be using 0.5 GB size which will cost around $0.02. We can leverage Event bridge free tier option and make use of scheduling and triggering. In case of *using VPC peering within same AZ then it is free*, so for our network layer we can leverage the free tier to maximum. We may use Route 53 so we can integrate that with a monthly cost of $0.50 per hosted zone.

The good thing is API gateway and other resources are all can be utilized in free tier, efficiently like one million API calls per month for up to 12 months[26]. We can make use of cost optimizer, and cost explorer to track the cost allocation tags, always use tags to easily identity the resources.

b. If no, how will you make your services more cost effective?

Above mentioned cost optimization can be beneficial, and it will reduce the significant amount of cost, by making use of free tier and staying inside a single region.

7. Did you use monitoring and/or logging tools to give you graphs, messages, metrics, or alerts of the services?

No, **we didn't use any particular services for monitoring or logging the networking layer**. But we can implement resources to monitor the networking layer of our serverless application.

a. If yes, can you further improve the monitoring and logging in your solutions/architectures?

No, I will discuss the new resources that we will implement for the serverless application in the next question.

b. If no, what monitoring and logging tools will you apply? Why?

First, we can make use of cloud watch to collect default metrics from Amazon VPC (NAT gateways), Amazon Route 53, AWS Network Firewall. We can capture **non-real-time metadata information about the IP traffic for our VPC** [27]. We can make use of **VPC flow logs** which will be integrated with cloud watch, this will help us know about the IP traffic going to and from network interfaces[27]. With this we can monitor network traffic patterns and identify any potential issues or security concerns. One more thing is we can set up **cloud watch alarms**, when the threshold is crossed. For API gateway also cloud watch helps by monitoring the number of requests, latency, and error rates and so on, this way we can monitor the performance and try to enhance them further. Event bridge is closely integrated with cloud watch events, and we can monitor, and log events generated by **EventBridge rules**[27]. Same goes for *Route 53 where we can monitor the public DNS query log and resolver query logs*, these are quite useful. It is essential to monitor and keep tracks of network logs as they provide number of packets passed, dropped, received to the services. We should configure network firewall as they can inspect, and filter traffics in our VPC[27].

We can use cloud trail to keep of track of auditing and API activity, we can know the security group or VPC changes, with guard duty we can manage threat detection it provides better protection as they analyze data from VPC flow logs[27]. We can make use of Athena to query the logs, sort it and visualize the data in dashboards, as cloud watch provides different graphs it is easy to understand, we have option to create reusable graphs and visualize your cloud resources and applications in a unified view.

General Questions:

1. Assuming your application will be commercialized, and your users will be globally distributed, what would you do to enhance the security, quick and easy access, and performance of the application? Pick one or two cloud services to explain for security, quick and easy access, and performance, respectively.

To enhance the security of our application which will be used globally, I would choose Web **Application Firewall(WAF)** as it as cloud-based firewall which protects from web exploits, and bots. We can create rules to *filter web traffic* based on conditions and prevent unauthorized takeovers. This way the security can be enhanced. So another services I would suggest is using **AWS Shield**, as it offers *DDoS protection.*

In case of easy, quick access we use **Cloud front**, because it has ability to reach users across globe in milliseconds[28], it comes with edge compute capabilities and filed level encryptions. By using cloud front we can reduce the latency and improve the speed of our application. With help of edge locations we can deliver the contents to end users, we have ability control the application based on geolocation using **Geo-Restrictions**[28]. Apart from cloud front, we can also use **Global Accelerator** it also has health checks and automatic failover.

The recommend approach is using cloud front and a *route 53, this way we can improve performance and scalability of our application.*

To improve performance, there are several ways, we can use **cloud fronts content caching mechanism to enhance performance**, because frequently accessed data can be served quickly without needing to retrieve it from the server every time. As it acts as **CDN**, it accelerates the delivery. Which will indeed increase performance. Also we can configure our **lambda by using provisioned concurrency**, this way our lambda will be warm, means ready for execution, there won't be any cold start delays. (It's like pre-initialization).

We also use Elastic cache in our application, which is in-memory database, which is mainly used for increasing the performance.

2. When you design your cloud architectures/solutions, should you try to choose managed or unmanaged services?

Well using managed or unmanaged services depends upon the application requirements, and person who is working on the application. **I would like to work with managed services**, because I don't have to worry about managing the servers, monitoring or any hardware militances. I can concentrate on my *functionalities, ease deployment, it is very user friendly*. However there are certain things to consider before choosing *like Cost, security, performance, complexity, and team expertise.*

1. Explain the pros and cons of managed and unmanaged services. You can Google the answers, but you must summarize using your own words.

The advantage of choosing managed services is, it will provide in built automatic scaling and high availability. It will be cost efficient in long run. As it offers high availability the application can be reliable as well. They come with good and well enhanced security. They provide the technical

support like helping with cloud migration. *But remember we cannot have full control or flexibility as we can in unmanaged services*[28].

As for unmanaged services, we have complete control over our sever, we can make the technical choices which is suitable to us, we can decide on how to configure, manage, and maintain the servers. We can customize the environment based on our need we can fine tune our system to enhance performance. We can manage the OS and we can easily migrate to other platforms. Also it comes with cost effectiveness, there is less up-front cost, we can manage the resource allocation and reduce operational costs. However, the *cost can go up in a long run*[28].

2. Using your explanations in question 1), analyze if the services in your previous report were managed or unmanaged. If you think there are managed (unmanaged) services that should be replaced by unmanaged (managed) services, pick one or two to explain why. If there is none to change, pick one or two to explain why.

In the previous application, there were both managed and un-managed services, I will explain two services why they can be serverless. First is Elastic Beanstalk, instead of using that we can use **lambda**, as it is very cost effective, we can pay when we use it, lambda won't be running all day. Lambda scales automatically it can handle **1000 request concurrently**, this way the performance will be increased, we don't have to manage anything, we can focus on the implementation and functionality part, it saves time, meaning aws will take care of updates, patches, scaling.

Similarly, instead of using RDS, we can use **Aurora**, though the cost is higher, the advantages are too good, it can scale automatically based on demands and, we pay for resources we use, this way it can be cost efficient as well. No need to manage underlying infrastructure. By leveraging **Global database, we can achieve low latency and high availability**. Aurora can quickly adapt to the change in workload and scale fast. The good thing is the compatibility with RDS, *we can easily migrate from RDS to aurora.*

References:

[1] I. Amazon Web Services, "What is Amazon Cognito?," 2023. [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html. [Accessed 5 07 2023].

[2] I. Amazon Web Services, "Security best practices for Amazon S3," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html. [Accessed 05 07 2023].

[3] I. Amazon Web Services, "What is AWS Secrets Manager?," 2023. [Online]. Available: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html. [Accessed 05 07 2023].

[4] A. W. Services, "Cognito User Profiles Export Reference Architecture," 2023. [Online]. Available: https://aws.amazon.com/solutions/implementations/cognito-user-profiles-export-reference-architecture/. [Accessed 07 07 2023].

[5] AWS, "AWS SECRETS MANAGER - SECRETS REPLICATION," [Online]. Available: https://disasterrecovery.workshop.aws/en/labs/basics/secrets-manager.html. [Accessed 07 07 2023].

[6] AWS, "Amazon S3 FAQs," 2023. [Online]. Available: https://www.amazonaws.cn/en/s3/faqs/. [Accessed 07 07 2023].

[7] Myrestraining, "Uncovering the Cost of AWS Secrets Manager Pricing," 05 01 2023. [Online]. Available: https://myrestraining.com/blog/aws/uncovering-the-cost-of-aws-secrets-manager-pricing/. [Accessed 07 07 2023].

[8] A. W. Services, "Logging and monitoring in Amazon Cognito," 2023. [Online]. Available: https://docs.aws.amazon.com/cognito/latest/developerguide/monitoring.html. [Accessed 07 07 2023].

[9] AWS, "What is AWS Elastic Beanstalk?," 2023. [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html. [Accessed 11 07 2023].

[10] U. Feroze, "All You Need To Know About Heroku In 5 minutes," 1 09 2021. [Online]. Available: https://levelup.gitconnected.com/all-you-need-to-know-about-heroku-in-5-minutes-7d4ec8849114. [Accessed 11 07 2023].

[11] AWS, "What is AWS Fargate?," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/userguide/what-is-fargate.html. [Accessed 11 07 2023].

[12] AWS, "Using Elastic Beanstalk saved configurations," 2023. [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environment-configuration-savedconfig.html. [Accessed 11 07 2023].

[13] AWS, "Amazon Elastic Beanstalk Features," 2023. [Online]. Available: https://www.amazonaws.cn/en/elasticbeanstalk/features/. [Accessed 13 07 2023].

[14] AWS, "Resilience in AWS Lambda," 2023. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/security-resilience.html. [Accessed 13 07 2023].

[15] AWS, "Amazon RDS Security," 2023. [Online]. Available: https://aws.amazon.com/rds/features/security/. [Accessed 18 07 2023].

[16] AWS, "Security in Amazon ElastiCache," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/redis-security.html. [Accessed 18 07 2023].

[17] A. Deekonda, "Implementing a disaster recovery strategy with Amazon RDS," 2019. [Online]. Available: https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/. [Accessed 19 07 2023].

[18] AWS, "High availability for Amazon Aurora," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.AuroraHighAvailability.html. [Accessed 19 07 2023].

[19] AWS, "High availability using replication groups," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Replication.html. [Accessed 19 07 2023].

[20] AWS, "Amazon Aurora Pricing," 2023. [Online]. Available: https://aws.amazon.com/rds/aurora/pricing/. [Accessed 19 07 2023].

[21] R. L. R. a. S. Kulkarni, "Optimize the cost of your Amazon ElastiCache for Redis workloads," 2023. [Online]. Available: https://aws.amazon.com/blogs/database/optimize-the-cost-of-your-amazon-elasticache-for-redisworkloads/. [Accessed 19 07 2023].

[22] AWS, "Monitoring usage, events, and costs," 2023. [Online]. Available: https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/MonitoringECMetrics.html. [Accessed 19 07 2023].

[23] AWS, "Security best practices for your VPC," 2023. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html. [Accessed 25 07 2023].

[24] AWS, "High availability cluster and shared Amazon VPC A," 2023. [Online]. Available: https://docs.aws.amazon.com/sap/latest/sap-hana/sap-hana-on-aws-ha-cluster-vpc.html. [Accessed 25 07 2023].

[25] K. L. a. P. A. Thomas Smart, "AWS Data Transfer Charges for Server and Serverless Architectures," 2022. [Online]. Available: https://aws.amazon.com/blogs/apn/aws-data-transfer-charges-for-server-and-serverlessarchitectures/. [Accessed 26 07 2023].

[26] S. Tahir, "Observing and diagnosing your network with AWS," 2022. [Online]. Available:

https://d1.awsstatic.com/events/Summits/reinvent2022/NET205_Observing-and-diagnosing-your-networkwith-AWS.pdf. [Accessed 26 07 2023].

[27] AWS, "Amazon CloudFront," 2023. [Online]. Available: https://aws.amazon.com/cloudfront/. [Accessed 26 07 2023].

[28] combel, "Managed vs unmanaged cloud: what is the difference?," 2021. [Online]. Available: https://www.combell.com/en/blog/managed-vs-unmanaged-cloud-what-is-the-difference/. [Accessed 26 07 2023].

[29] G. R. M. P. Shathish Annamalai, "Group 30 (Trinity) Project Report," Halifax, 2022.