Mechanisms Critical Analysis and Response

Group Member:

Archer Zhou B00806294
Junqiao Qu B00817232
Kessel Zhang B00809478

Compute:

AWS Elastic Beanstalk:

1. We choose Elastic Beanstalk for the cloud platform to run our service application. Our running application will respond to the user's request and complete some basic functions, like login, order system. It would be the main computing platform for this project. A plan is that we can develop a flask application. Then package it in a Docker image. We could deploy the container on Elastic Beanstalk.
2. Google App Engine, Azure Web Apps
3. Elastic Beanstalk has an ease of building user interfaces. The deployment configuration is easy to work with. It only takes a couple of clicks to switch instances or change servers. And it has a native support for Docker. And compared to alternatives, the Elastic Beanstalk has a better cost efficiency.

Network:

AWS API Gateway:

1. We chose AWS API Gateway for the network part of our project because it would help us manage the api's more efficiently to reduce the potential pitfalls when different services are requested.
2. We got alternatives such as Amazon EventBridge to integrate our application. The difference between these two is API Gateway is a service that helps you to create RESTful API and EventBridge is more like a event bus, but since we are doing a student project, there won’t be too many services being included, and API Gateway is good enough for managing APIs in our project.
3. We don’t choose AWS VPC because it is too complicated for our project, we don't want to create a routing table and make things complex. One big advantage of VPC is it can create a private network which can either be used as an extension of your website but also protect your website from external requests. But it doesn't fit our project well so we won't use that service.
4. Amazon CloudFont is also mentioned in the group project file, but it is a web service that speeds up distribution of your static and dynamic web content. We are building a

simple website and there are not too many large resources so we just try to make things in an easier way and don't use that service.

Storage:

AWS S3:

1. We can use s3 buckets for hosting a static website. In many cases, we do not require a dynamic website where data changes very frequently. We can leverage the AWS S3 for hosting a static website using AWS serverless architecture. A plan is that we create a static website using HTML,CSS. Then upload static website files to an AWS S3 bucket. Then we could configure static website hosting in the AWS S3 bucket. At the current stage, a link of S3 is sufficient to access our website. In the future, we could generate custom domain names to access the website on S3.
2. Alternatives: Google Cloud Storage, Azure Blob Storage
3. S3 has a fantastic developer API, it is a reliable and stable file storage service. AWS offers an easy deployment process. It is more popular than alternatives, so when there are unexpected problems, we can easily find supporting materials other than official documents.

AWS DynamoDB:

1. DynamoDB is a well-known database in AWS, and it has been widely used for many business websites. In this project, we will store our product information in this database.
2. Alternatives:DocumentDB, RDS, Aurora…
3. There are many choices for the database, they actually do not have much difference compared to each other in our project. We choose dynamoDB for two reasons. Firstly, a nosql database is giving better performance, and it is more flexible. And some of our group members have experience on this service. So using DynamoDB again is efficient for our project.

Security:

AWS Secrets Manager:

1. A basic method we are using for the Login function before is simply comparing the user’s input with the account data in a dataset. That is convenient but the problem is the lack of security. Therefore, in this project, we plan to use AWS Secrets Manager to protect our login function. Now Users and applications retrieve secrets with a call to Secrets Manager APIs, so that we do not need to hardcode sensitive information in plain text.

2. HashiCorp Vault, Google Cloud Identity & Access Management (IAM)

3. However, there have been many functions that could work together with AWS Secrets Manager. For instance, AWS Key Management Service could be used with AWS Key Management Service (KMS) to manage and protect encryption keys. We choose AWS Secrets Manager because it's good cost efficiency. Its developer API is easy to use and has more supporting documents. Another important reason is AWS Secrets Manager is cheaper than its alternatives.

Serverless Computing:

AWS Lambda:

1. With AWS Lambda, we can run code without on-premises or managed servers. With Lambda, we can run code for virtually any type of application or backend service, and it's completely unmanaged. Just upload our code and Lambda will take care of everything needed to run and scale your highly available code. We can set our code to trigger automatically from other AWS services, or call it directly from any web or mobile app. We could use it as a computing resource for small services in addition to AWS Elastic Beanstalk.
2. Firebase Cloud Functions, Step Functions.
3. The reason why we are choosing Lambda among other alternatives such as Step Functions is mainly just because there are more examples and tutorials online. Since all the members in our group don't have experience with cloud computing, more resources online should help us a lot, and that’s the main reason we choose Lambda for serverless computing.

AWS SNS:

1. On our website, we want users to get notification from us. For example, we want them to know the update information about their order. Here we need multiple ways to contact users. AWS SNS provides us these functions so that we can reach users easily.
2. We are considering two alternatives for this function: AWS SES and AWS SQS
3. We finally decided to use AWS SNS because: AWS SES is only available for the email notification. However, we wish to contact users in multiple ways, not only emails. AWS SQS is a distributed queuing system, receivers will not see messages until they pull them from SQS. Which means that users may not immediately see our messages, that is not what we want.
4. This service is free on amazon for 1 millions requests. That is totally enough for us.

Price and Cost:

Services	Usage	Price	Following Price
AWS S3	5GB 20000 GET request 2000 PUT request	free for twelve month	0.023USD/GB
AWS SNS(email)	1000	free	2USD/100000
AWS Lambda	1million request 400,000 GB-seconds of compute time	free	Price varies due to computational power
AWS Secrets Manager	N/A	30 day free trial	0.4USD/month
DynamoDB	25 GB of Storage 25 provisioned Write Capacity Units 25 provisioned Read Capacity Units 200M requests per month	Free	Free
API Gateway	1M REST API calls 1M HTTP API calls 750000 Connection Minutes	Free	3.5USD/333 Million REST API calls 1USD/300 Million HTTP API calls

Conclusion:

We most likely don't need to spend extra money for other services, the free tier and free trial should be enough for us in this group project.