CSCI5409 - MECHANISMS, CRITICAL ANALYSIS AND RESPONSE

Group Name – Apes Together Strong

Instructor – Dr. Lu Yang

Dhrumil Amish Shah (B00857606)
Jaspreet Kaur Gill (B00879409)
Nachiket Niranjanbhai Panchal (B00882397)
Samiksha Narendra Salgaonkar (B00865423)

1. For each category (Compute, Network, Storage, Security, Serverless Computing), list any mechanisms offered by your cloud provider(s) that your project will use.

Answer: **TravelMemories** will use the list of services provided below:

A. Compute: For the computing tasks of our application and deployment, we will use services such as **Amazon Elastic Compute Cloud (EC2) and Amazon Elastic Container Service (ECS).**
B. Network: We will use **Amazon API Gateway** for establishing the Network component of our application.
C. Storage: TravelMemories will use **Amazon DynamoDB** and **Amazon Simple Storage Services (S3)** for storing and managing data of our application.
D. Security: Security of the application will be managed by **Amazon Cognito.**
E. Serverless Computing: Some other services used are **Amazon Lambda** and **Google Firebase.**

2. For each mechanism, describe why your group chose this technology/service:
   a. What task or role is the service providing for your software architecture?

Answer: The software architecture of the **TravelMemories** application will be formulated of the following cloud services with Amazon and Google being the cloud service providers for each of the services. The tasks each of these cloud services will perform are listed below:

A. Compute:
   a. **Amazon Elastic Compute Cloud (EC2):** Amazon's EC2 offers a platform to the developers which helps them run virtual machines. Each of these virtual machine instances will host one or more containers encapsulating user applications. We will deploy the front-end and back-end of our application separately in two different containers[5].
   b. **Amazon Elastic Container Service (ECS):** Amazon's ECS is a container management service offered by Amazon to run user applications. Amazon ECS provides flexibility in running, stopping and managing containers on clusters. The frontend and backend of the application will run in these containers[6].

B. Storage:
   a. **Amazon DynamoDB:** Amazon's DynamoDB is a NoSQL database service that stores data in a key-value pair. DynamoDB will act as a primary database storage mechanism for the application. **TravelMemories** will use this service to store the data for the application. As a part of initial design, we have decided on the following tables/collections (as called in DynamoDB)[8]:
      a. Users: The collection containing the user details of the applications will contain attributes such as userID, name, other personal details such as contact, address, age, email-address, encrypted password. The key attribute in this collection will be userID.
      b. Posts: The collection containing posts made by the users will contain attributes such as postID, userID, post-image-url, post-location and many more. The attribute userID can be stored for every post to

identify which user has made the post in the application. The key attribute in this collection will be postID.

b. **Amazon Simple Storage Services (S3):** Amazon S3 is an object storage service provided by Amazon accessed over a web interface. This service is most commonly used for storing files of any format in entities, also called as buckets. **TravelMemories** will use the Amazon S3 service for storing image files (images that will be posted by the users of the application as a part of a post) in S3 buckets. The Amazon S3 service will be invoked when the user clicks on the upload or add a post option from the frontend of the application. These images will be stored in S3 buckets and their URL will be sent across to be further stored in Amazon DynamoDB database along with the rest of the post data[7].

C. Network:
   a. **Amazon API Gateway:** Amazon's service to create, publish, maintain, monitor and secure APIs. This service is used to establish a real-time communication channel between various components of the application. This will help us establish process workflow and data flow in our application.

D. Security:
   a. **Amazon Cognito:** Amazon's Cognito helps users to build secure applications. The service basically assists in implementing the Sign-up and Sign-In features for any application. We will use the Cognito service to register our users and log them in post registration[9].

E. Serverless Computing:
   a. **Amazon Lambda:** Amazon Lambda is a platform that is event-driven. **TravelMemories** will use Amazon Lambda to trigger function when an image is uploaded in Amazon S3 bucket[11].
   b. **Google Firebase:** Google Firebase's notification allows to send notifications to the application users. **TravelMemories** will use this notification service when a user uploads an image. A trigger using a function made of Amazon's Lambda service is executed everytime a user uploads an image file in Amazon S3 bucket[10].

b. What were the alternatives?

Table 1: Table displays the list of selected services along with their alternatives

Services Selected	Services Alternatives Available
AWS S3	Google Cloud Storage, Azure Cloud Storage
AWS DynamoDB	Google Cloud Firestore, MongoDB Atlas, Azure Cosmos DB
AWS Cognito	Google Identity Platform
AWS EC2 & ECS	Google Compute Engine, Google Cloud Run
AWS Lambda	Google Cloud Functions

c. Why did your group pick this one over the alternatives? Make sure your decision-making process is clearly described along with the tradeoffs you considered.

Answer:

a. **AWS S3**

S3(Simple Storage Service) provided by AWS, Google Cloud Storage, and Azure Cloud Storage are categorized as "Cloud Storage" services. Because of following reasons we chose S3:

1. Excellent developer API support that includes AWS command line, GUI and library functions.
2. Best in terms of Reliability and Scalability with wider developer community support.
3. Solution offered by AWS S3 for static website hosting is chepeast, highly stable, and best in performance.
4. Cost in terms of efforts is significantly reduced since the team is already familiar with AWS S3 [1].

b. **AWS DynamoDB**

DynamoDB database service by AWS, Google Cloud Firestore, MongoDB Atlas, and Azure Cosmos DB are categorized as "NoSQL Cloud Database" services. Because of following reasons we chose DynamoDB:

1. For large scale applications, the performance of database is not affected. (i.e., Read/Write operations are not affected)
2. Provides built-in security, backup and restores as well as supports wide array of platforms.
3. In-memory caching mechanism reduces the the response time from millisecond to microsecond [2].

c. **AWS Cognito**

Google Firebase is a single cloud service with multiple features like authentication, database, messaging, etcetera. In addition, it is mainly focused on mobile applications. On the other hand, AWS congnito comes with two integrations which are AWS UI and Self hosted UI. The Cognito UI integration might be easy interms of implementaion because Firebase Authentication does not provide any UI. Even more, Aws Cognito has larger support than Firebase Authentication in-terms of web applications. On these basis, we have decided to go with AWS Cognito as our application's authentication service [3].

d. **AWS EC2 & ECS**

1. Amazon EC2 and ECS are services offered by Amazon for computing tasks. We will use the virtual-machine based and container-based services to deploy our applications.
2. Amazon offers flexibility over choosing the best choice across numerous options available and hence it is a better suitable cloud service provider for compute as compared to the other service providers such as Google.

e. **AWS Lambda**

1. AWS Lambda and Google Functions both supports custom runtime environments.

2. If we try to compare both the services for performance, AWS Lambda allows to run one thousand lambda functions concurrently and for Google Functions there is no official limit specified.
3. Now, let's compare both the services for a cold start. AWS Lambda takes average time of less than 1 second and Google Functions take approximately from a split-second to two seconds. So, AWS Lambda is winner for cold start comparision.
4. AWS Lambda has maximum timeout of 15 minutes where Google Cloud Functions is slightly behind and provides maximum 9 minutes timeout.
5. Finally the memory comparision, AWS Lambda and Google Cloud Functions both provides dymanic memory allocation according to computation requirements. However, Google Functions provides maximum of 4GB memory allocation where AWS offers maximum 10GB memory allocation for compuatation and we can see a huge difference here.
Considering all the above points we have chosen AWS lambda for better performance and scalability [4].

f. **AWS API Gateway**
As we are going to use majority of AWS services the AWS API Gateway is the best and only option to isolate our AWS services.

3. How much does it cost to use each of the mechanisms in your project? Did price affect your choices?

Answer:

A. Compute:

a) **Amazon Elastic Compute Cloud (EC2):**
On-Demand rates for EC2 instances start at $0.013 per hour ($9.50 per month) and come in three sizes (micro, small, and medium) [5].

b) **Amazon Elastic Container Service (ECS):**
Our team is using the ECS service to run the application hosted on EC2 instances and there are no additional charges for ECS service when it is utilized with EC2 instances [6].

We have planned to host our application on EC2 and run it using ECS so we have used these two services from the computing category and have not considered the pricing model.

B. Storage:

a) **Amazon Simple Storage Services (S3):**
Storage cost: charged per GB / month. ~ $0.03 / GB / month, charged hourly
API cost for operation of files: ~$0.005 / 10000 read requests write requests are 10 times more expensive
Data transfer outside of AWS region: ~$0.02 / GB to different AWS region, ~$0.06 / GB to the internet [7].

b) **Amazon DynamoDB:**

AWS DynamoDB service is charged based on the read request units. It costs $0.25 per million read request units [8].

Our team has decided to use S3 and DynamoDB for storing the application data. Post images to be stored in S3 and the other required data of the application such as user details to be stored in DynamoDB. We have not decided on these services to use in our project based on the prices but based on their usage.

C. Security:
   a) **AWS Cognito:**
   Amazon Cognito costs $0.15 for each 10,000 sync operations and $0.15 per GB of sync store per month. Straight out of the box, it is simple to operate [9].

   AWS Cognito would be used to develop the entire user management feature of the application. This service is also simple to integrate with the web-based application determining the main reason for using it but not the cost.

D. Serverless Computing:
   a) **Google Firebase:**
   Google Firebase cost on the paid tier translates to 200,000 per database, $ 5 per GB stored, and $ 1 per GB downloaded [10]**.**

   b) **AWS Lambda:**
   The price of AWS Lambda includes:
   Compute charges: $0.00001667/invocation
   Request charges: $0.2/M <requests. [11]

   We as a team decided to go with Google cloud firebase and Lambda because we are familiar with these two services and price has no role in considering these services.

E. Network:
   a) **AWS API Gateway:**
   API Gateway price is as low as $0.90 per million requests. Also, the failed requests for authorization and authentication are not charged [12].

   We have considered using API Gateway to make secure API requests without considering the pricing criteria.

References

[1] TrustRadius, "Amazon S3 vs Google Cloud Storage," TrustRadius, [Online]. Available: https://www.trustradius.com/compare-products/amazon-s3-simple-storage-service-vs-google-cloud-storage. [Accessed 23 October 2021].

[2] DataFlair, "Amazon DynamoDB Tutorial," DataFlair, [Online]. Available: https://data-flair.training/blogs/amazon-dynamodb/. [Accessed 23 October 2021].

[3] Userfront Guide, "Auth0 vs Cognito vs Okta vs Firebase vs Userfront Comparison," Userfront Guide, [Online]. Available: https://userfront.com/vs/best-auth-comparison-auth0-okta-firebase-cognito.html. [Accessed 23 October 2021].

[4] A Cloud Guru, "Serverless showdown: AWS Lambda vs Azure Functions vs Google Cloud Functions," A Cloud Guru, [Online]. Available: https://acloudguru.com/blog/engineering/serverless-showdown-aws-lambda-vs-azure-functions-vs-google-cloud-functions. [Accessed 23 October 2021].

[5] Amazon AWS, "Amazon EC2 pricing," Amazon AWS, [Online]. Available: https://aws.amazon.com/ec2/pricing/. [Accessed 23 October 2021].

[6] Spot, "AWS ECS Pricing: 3 Pricing Models and 5 Cost Saving Tips," Spot by NetApp, [Online]. Available: https://spot.io/resources/aws-pricing/aws-ecs-pricing-3-pricing-models-and-5-cost-saving-tips/. [Accessed 23 October 2021].

[7] s. logic, "Amazon S3 Simple Storage Service Cost Optimization," sumo logic, [Online]. Available: https://www.sumologic.com/insight/s3-cost-optimization/. [Accessed 23 October 2021].

[8] Amazon, "Pricing for On-Demand Capacity," Amazon, [Online]. Available: https://aws.amazon.com/dynamodb/pricing/on-demand/. [Accessed 23 October 2021].

[9] Amazon, "Amazon Cognito FAQs," Amazon, [Online]. Available: https://aws.amazon.com/cognito/faqs/. [Accessed 23 October 2021].

[10] NS804, "How Much Does Firebase Cost And Should You Use It," NS804, [Online]. Available: https://www.ns804.com/blog/how-much-does-firebase-cost-and-should-you-use-it/. [Accessed 23 October 2021].

[11] Simform, "AWS Lambda Pricing: How Much it Costs to Run a Serverless Application?," Simform, [Online]. Available: https://www.simform.com/blog/aws-lambda-pricing/. [Accessed 23 October 2021].

[12] Amazon, "API Gateway pricing," Amazon, [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-pricing.html. [Accessed 23 October 2021].

[13] TrustRadius, "Amazon S3 vs Google Cloud Storage," TrustRadius, [Online]. Available: https://www.trustradius.com/compare-products/amazon-s3-simple-storage-service-vs-google-cloud-storage. [Accessed 23 October 2021].