Cloud Computing Architecture

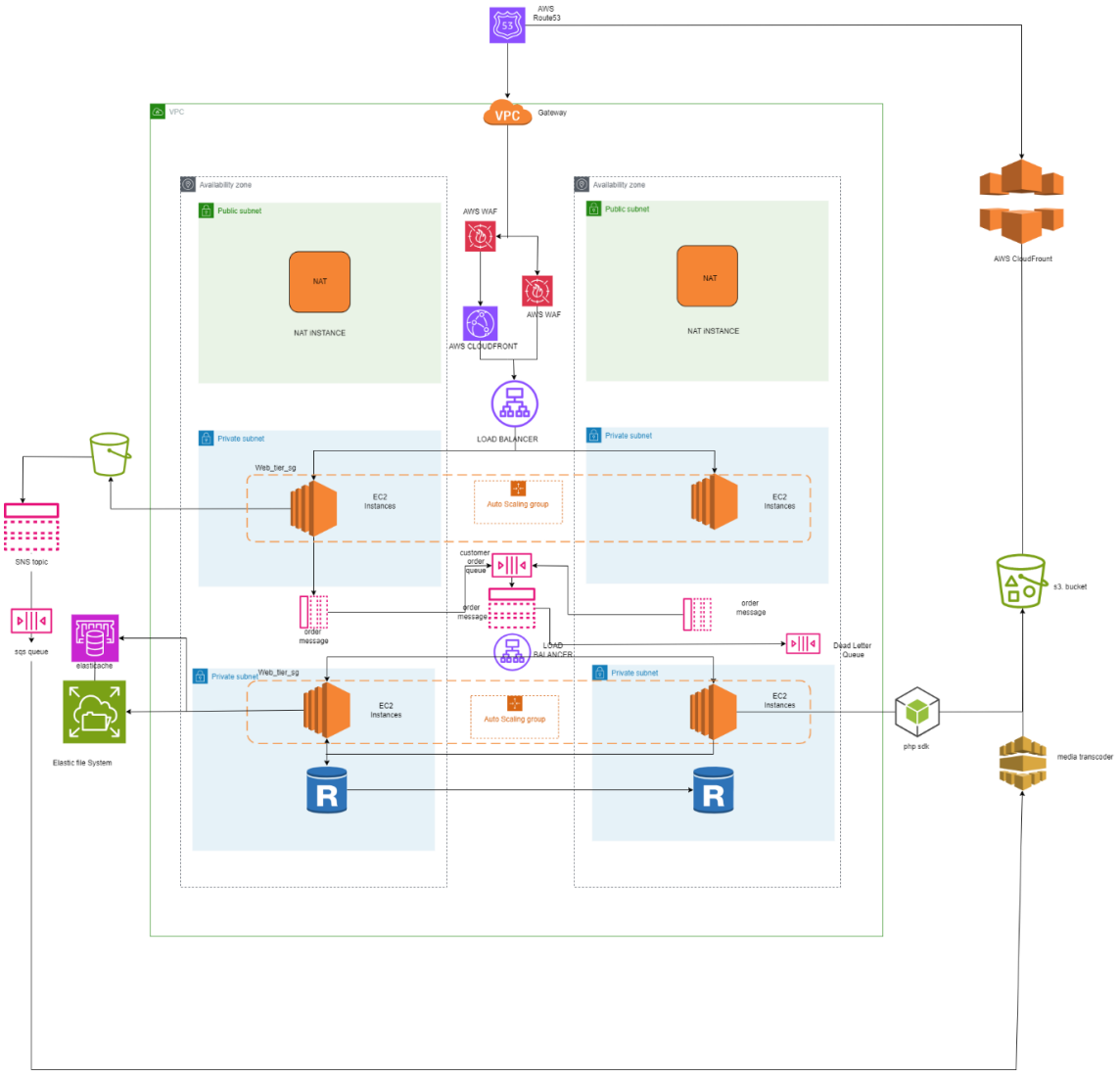
Assignment 3

Serverless/Event-driven Architectural Design Report

I. Introduction

This project focuses on enhancing our system to manage the increasing user demand of our photo album website, improve global access speeds, and streamline our costs. By leveraging advanced cloud solutions, especially those offered by Amazon Web Services (AWS), we aim to provide an optimized and scalable platform for our application. From revaluating our database choices to refining how we handle media, this endeavour aims for a comprehensive upgrade suited for our future growth.

II. Architecture Diagram



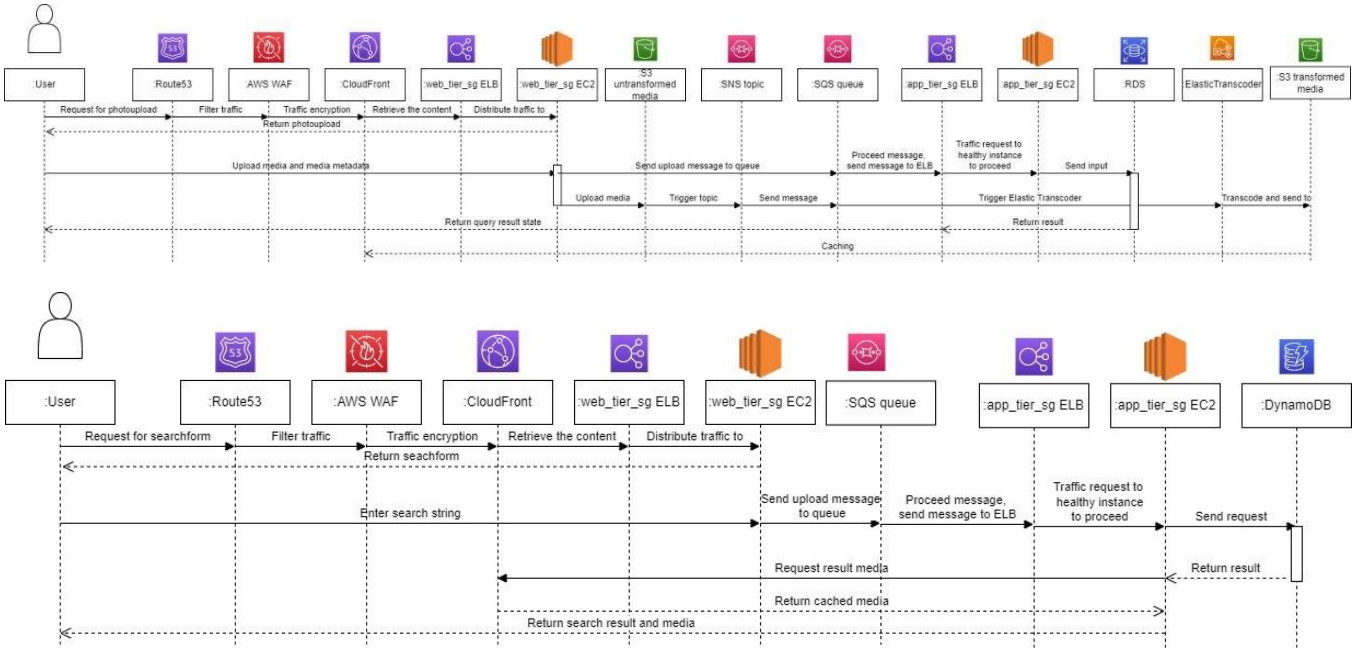
a) *Design Justification:*

Fulfillment of Business Scenario using Proposed Services

- ***AWS Route53****: Directs user requests to the appropriate endpoint (like CloudFront or EC2).*
- ***AWS CloudFront****: Efficiently serves user content, caches frequently accessed data, and provides a secure connection.*
- ***AWS WAF****: Web application firewall that helps protect web apps from common web threats.*
- ***VPC with Private and Public Subnets****: Ensures a secure and isolated network environment.*
- ***Load Balancers (ELB)****: Distributes incoming traffic to maintain application performance.*
- ***EC2 Instances with Auto Scaling****: Virtual servers to run applications, which can be automatically scaled based on traffic.*
- ***S3 Buckets****: Stores static assets and backups.*
- ***SQS (Simple Queue Service)****: Decouples and scales microservices, distributed systems, and serverless applications.*
- ***RDS****: Managed relational database service.*
- ***ElastiCache****: Caches frequently queried data for faster access.*
- ***Elastic File System (EFS)****: Scalable file storage for EC2 instances.*

b) *Architecture Service Breakdown:*

- ***Route53:*** *AWS Route53 is a scalable domain name system (DNS) web service. It's designed to route end-users to internet applications by translating human-readable names, like www.example.com, into the numeric IP addresses needed for data transfer.*
- ***CloudFront:*** *AWS CloudFront is a content delivery network (CDN) service. It distributes content globally with low latency, providing a seamless experience for users by caching content at edge locations.*
- ***Web Application Firewall (WAF):*** *AWS WAF is a security service that monitors web requests, protecting your web applications from common web exploits. By defining customizable security rules, it aids in blocking malicious traffic.*
- ***Elastic Load Balancing (ELB):*** *ELB automatically distributes incoming web traffic across multiple Amazon EC2 instances. It ensures that the application remains available even if one or more instances fail.*
- ***Security Groups****: These act as a virtual firewall for Amazon EC2 instances, controlling inbound and outbound traffic. They define which traffic is allowed or denied to the instances.*
- ***Simple Storage Service (S3)****: Amazon S3 provides scalable object storage. It's designed to store and retrieve any amount of data from anywhere, making it a foundational piece for backup, analytics, and more.*
- ***Simple Queue Service (SQS):*** *SQS is a fully managed message queuing service. It decouples the components of a cloud application, ensuring data is consistently transmitted between components without loss.*
- ***Simple Notification Service (SNS)****: AWS SNS is a fully managed pub/sub messaging service. It facilitates the sending of messages, notifications, or alerts to a distributed set of recipients.*
- ***Elastic File System (EFS):*** *EFS provides a scalable and elastic NFS file system. It can be mounted on several EC2 instances, allowing for data sharing and long-term data persistence.*
- ***ElastiCache:*** *A web service that aids in deploying, managing, and scaling an in-memory cache in the cloud. This helps in enhancing the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches.*
- ***Elastic Transcoder:*** *This service provides media transcoding in the cloud. It's designed to convert media files from their source format into versions that'll playback on devices like smartphones, tablets, and PCs.*



III. ALTERNATIVE SOLUTIONS

a) Virtual Machine vs. Container vs. Serverless Computing

- ***Virtual Machine (EC2)**:*
  - *Performance: Allows granular control over installations and configurations. Needs consideration for memory, CPU, and tenancy configurations.*
  - *Reliability: Paired with Elastic Load Balancer to prevent single points of failure. Continues running until deliberately halted.*
  - *Security: Ensures network isolation via VPC service and offers isolation at the physical host level.*
- ***Container (ECS)**:*
  - *Performance: Offers choice for server management and supports varied backend applications without admin interference.*
  - *Reliability: Can be set up across multiple Availability Zones for enhanced availability. Application always ready but runs when triggered.*
  - *Security: Controls network traffic through security groups and private subnets.*
- ***Serverless Computing (Lambda)**:*
  - *Performance: It's event-driven with certain limits on runtime and volume. Doesn't require backend administration.*
  - *Reliability: Manages availability using VPC and Availability Zones. Offers advanced features like concurrency control and network access.*
  - *Security: Uses IAM for resource access management. It's safeguarded by AWS's extensive network security.*

***Decision**: Due to cost efficiency, Virtual Machines (EC2) were chosen over the other options.*

b) SQL vs. NoSQL

- ***SQL (RDS)**:*
  - *Performance: Fits a wide range of database tasks.*
  - *Reliability: Features automated backups and database snapshots.*
  - *Security: Uses AWS KMS and AWS CloudHSM for encryption. Offers network isolation through Amazon VPC.*
- ***NoSQL (DynamoDB)**:*
  - *Performance: Supports key-value and document databases.*
  - *Reliability: Uses CloudWatch and Enhanced Monitoring for instance tracking. Provides point-in-time recovery and on-demand backup.*
  - *Security: Data is encrypted by default using AWS KMS.*

***Decision:** Due to superior security and cost-effectiveness, Amazon RDS was selected.*

c) Caching Options

- ***ElastiCache**:*
  - *Performance: Speeds up applications by caching data, reducing latency.*
  - *Reliability: Boosts application response times and provides a buffer against database downtimes.*
  - *Security: AWS is partially responsible for data security, but users must also secure their credentials.*
- ***Container (ECS)**:*
  - *Performance: Acts as a CDN, minimizing latency for content delivery, including streaming.*
  - *Reliability: Augments application availability and diminishes latency by caching dynamic content at edge locations.*
- ***Route53**:*
  - *Performance: Directs users to optimal servers, ensuring minimal latency.*
  - *Reliability: Offers consistent routing to applications, enhancing user experience.*

***Decision**: All listed caching options were incorporated to optimize the architecture.*

d) Push vs. Pull Message Handling Options for Decoupling

- ***Push Message Handling (SNS)**:*

        - *Performance: Allows topics and policies setup, enabling time-sensitive messages to decoupled subscribers.*
        - *Reliability: Offers rapid response during high request influx.*
        - *Security: Direct database communication from clients is prohibited.*
    - ***Pull Message Handling (SQS)**:*
        - *Performance: Decouples application elements, permitting asynchronous processing for speedier outcomes.*
        - *Reliability: Prevents bottlenecks and single point failures. Messages stored in queues enable easy recovery post failures.*
        - *Security: SQS messages are encrypted with SSE and decrypted only when reaching an authorized recipient.*

e) 3-tier vs. 2-tier Architecture

- ***3-tier**:*
    - *Performance: Processes application logic in a dedicated layer, optimizing resource utilization.*
    - *Reliability: Provides swift responses during traffic spikes.*
    - *Security: Prevents clients from communicating directly with the database.*
- ***2-tier**:*
    - *Performance: Handles application logic in either the web or database layer, potentially causing performance bottlenecks during high traffic.*
    - *Reliability: May exhibit slower responses during increased user activity. System vulnerabilities arise if a component fails.*
    - *Security: Lesser security as clients can interface with the database via the web layer.*

Decision: A 3-tier architecture was chosen for its superior performance, resilience against systemic failures, and heightened security.

IV. Design Criteria:

a) ***Performance, scalability (extensibility, decoupling, etc.)**: Auto Scaling, Load Balancers, and SQS ensure performance and scalability. Decoupling is achieved with SQS.*

b) ***Reliability**: Multi-AZ deployments in RDS and EC2, along with the distributed nature of S3, ensure high availability.*

c) ***Security**: VPC provides network isolation. WAF protects against web threats, while CloudFront offers SSL/TLS encryption. IAM roles and security groups restrict unauthorized access.*

d) ***Cost**: The architecture I'm considering involves various AWS services, each with its distinct cost structure. It's essential to note that predicting the exact cost of AWS services is intricate due to several considerations. The region of service usage, the specific sizes and types of instances chosen, the volume of data transfer, the choice between reserved and on-demand pricing, among other factors, can significantly influence the total cost.*

For those looking for an accurate and tailored estimate, the AWS Pricing Calculator is an invaluable tool. AWS also offers consultation to provide detailed cost breakdowns.

However, for a general overview, the table provided offers a ballpark estimate of the expected costs. It's predicated on commonly utilized instance types and services. While it provides a tangible sense of potential costs, remember that this is a high-level approximation. Real costs can differ significantly, depending on the actual usage and configurations.

The table lists components such as AWS Route53, Internet Gateway, AWS CloudFront, and several others. Each service has an associated estimated monthly and annual cost. For instance, EC2 Instances, a core component of many AWS architectures, can vary greatly in price depending on the chosen instance type and size. Some components, like VPC, come with minimal costs, being mostly free unless specific configurations or additional services are required. On the other end of the spectrum, services like RDS, which offers relational database capabilities, can be more costly depending on the database type and size chosen.

In sum, while the total estimated annual cost provided is approximately $4926, this is a rough gauge. For precise financial planning, it's recommended to use the official tools provided by AWS and, if possible, consult directly with AWS or an AWS-certified partner.

Component	**Estimated Monthly Cost**	**Estimated Annual Cost**	**Cost Efficiency Remarks**
AWS Route53	$1	$12	Centralized domain management reduces complexity and redundant queries.
Internet Gateway	$30	$360	Efficient traffic routing minimizes data transfer costs.
AWS CloudFront	$50(depends on traffic)	$600	Optimized content delivery, reducing origin fetches & latency.
AWS WAF	$5	$60	Custom security rules prevent overhead and enhance protection.
VPC	Minimal (mostly free)	Minimal	Proper network design reduces overlapping resources and costs.
EC2 Instance	$100 (varies greatly)	$1200	Use of appropriate instance types optimizes cost vs performance.
Auto Scaling	Depends on instance	Depends on instance	Dynamic scaling reduces over-provisioning and saves costs
S3 Bucket	$25 (varies by storage)	$300	Lifecycle policies and storage classes optimize storage costs.
SNS topic	$1	$12	Efficient notification system reduces overhead in message delivery.
SQS	$10	$120	Managed queuing service reduces infrastructure and maintenance costs
Load Balancer	$20	$240	Distributes incoming traffic, optimizing server use & reducing downtime.
NAT Instance	$40	$480	Efficient path for private resources to access the internet.
Elastic File System	$30 (depends on size)	$360	Scalable file storage reduces provisioning overheads
ElastiCache	$60	$720	Managed in-memory cache improves app speed, reducing server costs.
RDS	$120 (depends on type)	$1440	Managed DB service reduces admin overhead & optimizes performance.
PHP SDK	N/A (software kit)	N/A	Development kit saves on integration costs.
Media transcode	$25 (depends on usage)	$300	Pay-per-use model eliminates wasted resources and scales with demand.

V. This AWS Design Stands Out in Cost-Effectiveness:

Transitioning to the cloud brings forth a realm of opportunities for businesses, but it's the strategic selection and integration of services that amplify cost-effectiveness. Here's why this AWS design is a paragon of cost-efficient cloud infrastructure:

a) ***Optimized Resource Utilization:*** *By leveraging AWS services such as Auto Scaling and ElastiCache, this design ensures that resources are neither over-provisioned nor underutilized. It dynamically aligns with the actual demand, ensuring i pay only for what i use.*

b) ***Enhanced Performance with Reduced Latency:*** *With AWS CloudFront and Elastic File System, the design significantly reduces latency, offering a better user experience. This, in turn, can lead to increased user retention and engagement, offering more value for each dollar spent.*

c) ***Robust Security without the Price Tag:*** *Security is often a significant cost for businesses. However, with AWS WAF and VPC, this design offers top-notch security features without the hefty price tag, ensuring protection against threats without breaking the bank.*

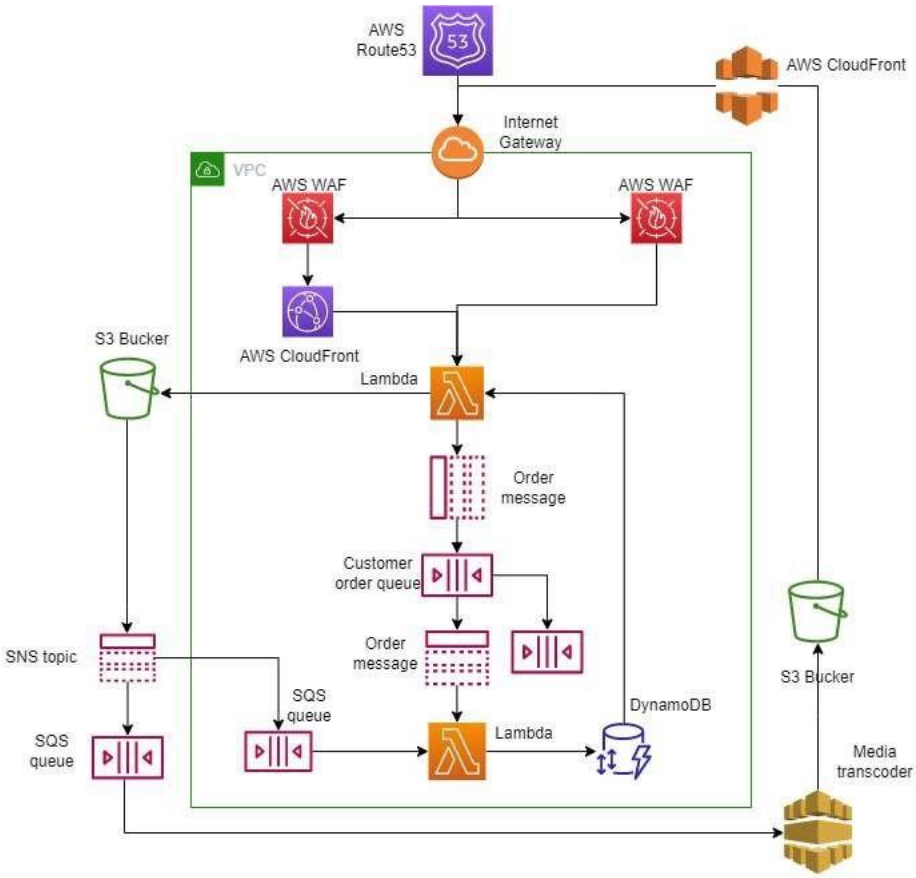
d) ***Simplified Management and Lower Overheads:*** *Centralized domain management through AWS Route53 and efficient traffic routing via Internet Gateway minimize administrative complexities. Fewer complexities often translate to fewer resources spent on management, reducing overhead costs.*

e) ***Future-Proof Scalability:*** *One of the hidden costs businesses face is the future expansion of infrastructure. With services like RDS and S3 Bucket in the design, scalability becomes seamless, ensuring that growing demands can be met without substantial infrastructural revamps.*

f) ***Development Efficiency:*** *The inclusion of PHP SDK means that integration becomes smoother, potentially reducing development time and costs.*

In essence, this AWS design isn't just about individual service costs but the collective synergy they bring in optimizing both performance and expenditure. Companies looking for a balanced, efficient, and future-proof cloud infrastructure will find this design an attractive proposition, offering substantial returns on investment.

VI. Alternative Solution:



***Consideration of Alternative Approaches**:*

During the formulation of our architecture, we did contemplate an alternative approach that hinged on AWS Lambda and DynamoDB. This solution presented a distinct set of advantages and challenges, which have been delved into in our prior discussions. Here's an elaboration:

- ***AWS Lambda**:*
  - *Pros: One of the primary benefits of using Lambda is the liberation from server administration tasks. This serverless computing model permits us to run our code without actively managing the underlying infrastructure. This can lead to enhanced agility in deployments and potentially reduce operational overheads.*
  - *Cons: However, a major concern that arose with Lambda was the complexity it introduced to our debugging process. In a serverless environment, traditional debugging methods often don't directly apply. Moreover, there's an intrinsic need to meticulously optimize the application's size, which could stretch our development timeline.*
- ***DynamoDB**:*
  - *Pros: DynamoDB is known for its flexibility as a NoSQL database, supporting both document and key-value data structures. Its scalability and performance can be tailored for different workloads, and it's especially effective when dealing with large-scale applications that require quick data access.*
  - *Cons: From a financial perspective, integrating DynamoDB could escalate our operational costs, especially when juxtaposed with Amazon RDS. It's essential to weigh the benefits of DynamoDB's scalability and versatility against its potentially higher pricing structure.*

In summary, while both AWS Lambda and DynamoDB offer robust functionalities and cater to specific use-cases effectively, our decision was influenced by the broader context of our application's needs, the operational challenges we anticipated, and budgetary considerations.

VII. REFERENCES:

[1] "Application Architecture Checklist," Enterprise Architecture at Harvard. [Online]. Available: https://enterprisearchitecture.harvard.edu/application-architecture-checklist. [Accessed: 22-Oct-2023].

[2] "Reference Architecture Examples and Best Practices," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/architecture/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=.

[3] Amazon Web Services, Inc., "Web Application Hosting in the AWS Cloud," 2019. [Online]. Available: https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf.

[4] Amazon Web Services, Inc., "Architecting for the Cloud AWS Best Practices," 2018. [Online]. Available: https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf.

[5] "AWS Well-Architected Framework," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/framework/wellarchitected-framework.pdf.

[6] "AWS-CloudDesignPattern," Cloud Design Pattern. [Online]. Available: https://en.clouddesignpattern.org/index.php/Main_Page.html.

[7] "Architecture Icons," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/architecture/icons/.

[8] "Free Cloud Computing Services - AWS Free Tier," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/free/?trk=f181118c-0869-454a-84d2-63d0cf7146e3&sc_channel=ps&ef_id=CjwKCAjw7c2pBhAZEiwA88pOF8GkQVef3kvFoQLFYp34BJi2YMtqAmxAdpBkclhdvDC14sEimc0ZCBoC-vsQAvD_BwE:G:s&s_kwcid=AL. [Accessed: 22-Oct-2023].

[9] F. Hu et al., "A Review on Cloud Computing: Design Challenges in Architecture and Security," Journal of Computing and Information Technology, vol. 19, no. 1, pp. 25, 2011.

[10] "A Secure Cloud Computing Architecture Design," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6834978?casa_token=nk4YGfHB-toAAAAA:r98SYygvvWcM8jp3F_HllOAJz4Xzr5zxVckdIIFN-ca4VIxBMq6BghNuil9PoeSzEmMJKw7Z. [Accessed: 22-Oct-2023].

[11] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7–18, 2010.