DALHOUSIE
UNIVERSITY

FACULTY OF COMPUTER SCIENCE

TERM ASSIGNMENT

CSCI 5409: ADVANCED TOPICS IN CLOUD COMPUTING

by

Jay Jagdishbhai Patel

B00981520

Submitted to

Prof. Lu Yang

Department of Computer Science

Dalhousie University.

Date: 7th August 2024

Table of Contents

Introduction

Project Overview:

MedEasy is a cloud-based platform that has been made available on the cloud to simplify the medicine ordering process through the automation of the workflow that starts from prescription upload to the final order fulfillment. The main purpose of MedEasy is to provide a seamless and efficient solution for users to order medicines online by simply uploading their prescriptions. Relevantly, the system uses highly specialized AWS services to ensure that it extracts the medicine's details accurately, do real-time inventory checks, and automate order processing. This allows individuals to have an easy and smooth user experience.

Purpose and Functionality:

MedEasy is a platform that allows users to upload pictures of their prescriptions, which are then automatically transcribed into a list of medications that need to be ordered. The platform then accesses the inventory of different sellers and completes the order if the medications are in stock. As soon as users' orders are confirmed, they receive an email with order details that include any unavailable goods and a link to the prescription used for the order.

Target Users:

- **Individual Consumers:** Seeking a convenient way to order medications without manually ordering medicine from website or visiting physical pharmacies.
- **Pharmacists and Sellers:** Wanting to keep track of inventory easily and to attract more clients.
- **Healthcare Providers:** Needing a fast and secure way to process prescriptions and make drug orders for patients.

Performance Targets

- **Scalability:** Through the use of the Lambda service, it is possible to have an architecture that is serverless and can easily be expanded to serve 100,000 users at the same time [1].
- **Response Time:** Employing AWS API Gateway, the system achieves API response times of less than 300 milliseconds guaranteeing quick user interactions [2].

- **Reliability:** Aiming for 99.9% uptime with AWS infrastructure, results in very little downtime and disruptions, thus reliability is a key objective.
- **Order Accuracy:** AWS Textract is used for accurate text extraction from prescriptions, achieving an order accuracy rate of 99% [3].
- **Notification Speed:** Real-time messaging through AWS SNS and EventBridge is used for sending delivery confirmations and updates within seconds [4].
- **Event Handling:** Event-driven workflows are managed by AWS EventBridge which also ensures automatic scaling for efficient event processing [4].

Meeting Menu Item Requirements

In developing MedEasy, a thorough choice was made to define which suitable AWS services would satisfy the requirements of the project in terms of both functionality and performance. The chosen services were evaluated against alternative AWS options to ensure optimal scalability, reliability, and efficiency.

1. Compute and Processing

- **Selected Service: AWS Lambda**
    - **Justification:** AWS Lambda offers a serverless computing environment that dynamically adjusts its capacity based on demand, which provides cost-effectiveness and eliminates the need for manual resource management. It can deal with the asynchronous workflows needed in MedEasy very well [1].

- **Alternative: Amazon EC2**
    - **Comparison:** EC2 gives you more server environment control but it also involves the maintenance and manual scaling which can result in the higher costs and complexity. AWS Lambda's serverless model is more suitable for MedEasy's needs of scalability and cost-effectiveness, especially when processing sporadic and event-driven workloads [5].

2. Workflow Orchestration

- **Selected Service: AWS Step Functions**
    - **Justification:** AWS Step Functions grant the possibility of orchestrating complicated workflows by coordinating several AWS services, for example, utilizing functions in serverless environments. It provides an easy way to manage the state, handle errors, and retries as well, thus

making the execution of multi-step processes like those in MedEasy easier [6].

- **Alternative: AWS Simple Workflow Service (SWF)**
  - **Comparison:** While SWF provides similar orchestration capabilities, Step Functions are easier to integrate with Lambda and some other AWS services, and provide a more user-friendly experience of visual workflow design. This makes the Step Functions options more suitable for MedEasy's purposes, where speedy development and flexibility are a must [7].

3. Optical Character Recognition (OCR)

- **Selected Service: AWS Textract**
  - **Justification:** AWS Textract is one of the most powerful OCR services which specializes in the extraction of text and data from the documents. The main reason for choosing this service is that it can analyze prescription images and extract structured data with such high accuracy that it is the best for MedEasy [8].

- **Alternative: Amazon Rekognition**
  - **Comparison:** While Amazon Rekognition provides image and video analysis, Amazon Rekognition offers video and image analysis services, but it mainly focuses on object detection, scene detection, and face detection. The Textract service of AWS is a more specialized solution that is aligned with the specific needs of MedEasy, which is extracting the precise details from the prescriptions [8].

4. API Management

- **Selected Service: AWS API Gateway**
  - **Justification:** API Gateway offers a highly scalable and secure platform for managing RESTful APIs. It provides features like traffic management, authorization, and monitoring, making it the standard choice for API management within AWS [2].

- **Alternative: No direct AWS alternative**
  - **Explanation:** API Gateway is the primary service in AWS for handling API requests, and its tight integration with other AWS services like Lambda makes it an unmatched choice for MedEasy's API management needs.

5. Database and Storage

- **Selected Service: Amazon DynamoDB**
  - **Justification:** Amazon DynamoDB provides a fully managed NoSQL database solution with fast and predictable performance. It is particularly well-suited for managing dynamic inventory and order data with varying access patterns [9].

- **Alternative: Amazon RDS (Relational Database Service)**
  - **Comparison:** RDS offers structured SQL database management but requires more complex schema definitions and maintenance. DynamoDB's flexibility and scalability make it a better fit for MedEasy's unstructured and scalable data needs [9].

- **Selected Service: Amazon S3**
  - **Justification:** Amazon S3 is the go-to service for scalable object storage in AWS, offering high durability and availability. It is ideal for storing prescription images and order data as JSON documents [10].

- **Alternative: No direct AWS alternative**
  - **Explanation:** S3 is the primary storage service in AWS for object storage, providing unmatched durability, scalability, and integration capabilities with other AWS services.

6. Notification and Event Management

- **Selected Services: AWS Simple Notification Service (SNS) & AWS EventBridge**
    - **Justification:** SNS and EventBridge are designed for real-time event-driven architectures. SNS handles messaging and notifications, while EventBridge facilitates seamless event orchestration across AWS services, providing a highly scalable and reliable solution for MedEasy's notification and event handling needs [4] [11].

- **Alternative: No direct AWS alternatives for combined functionality**
    - **Explanation:** The combination of SNS and EventBridge is unmatched within AWS for handling complex event-driven workflows, offering deep integration with other AWS services and robust scalability.

Cloud Deployment Model for MedEasy

MedEasy is deployed using a **Public Cloud** model on AWS, utilizing the suite of managed services provided by Amazon Web Services. This deployment model is particularly suitable for MedEasy due to several key reasons:

Key Reasons for Choosing Public Cloud:

- **Cost Efficiency:**
    - **No Upfront Costs:** MedEasy benefits from a pay-as-you-go pricing model, which eliminates the need for large capital investments in hardware and infrastructure [12].
    - **Operational Costs:** The public cloud offers a reduction in operational costs by leveraging AWS's economy of scale, making it more affordable for startups and growing businesses [12].

- **Scalability and Elasticity:**
    - **Automatic Scaling:** AWS provides automatic scaling capabilities, allowing MedEasy to handle varying loads seamlessly. This is crucial for handling peak times when user demand may surge, such as during health crises or promotional events [12].
    - **Elastic Resources:** The platform can easily scale resources up or down as needed, providing flexibility to adapt to changing workloads without manual intervention [12].

- **Global Reach:**
  - **Geographical Distribution:** AWS’s global infrastructure allows MedEasy to deploy its application across multiple regions, ensuring low latency and high availability for users worldwide [12].
  - **Accessibility:** Users can access MedEasy's services from anywhere, providing a consistent experience regardless of location [12].

- **Innovation and Agility:**
  - **Access to Advanced Services:** The public cloud provides access to cutting-edge technologies such as AWS Lambda, Textract, and AI services, enabling MedEasy to innovate quickly [12].
  - **Rapid Deployment:** New features and updates can be rolled out rapidly, keeping MedEasy at the forefront of technology advancements [12].

- **Focus on Core Business:**
  - **Managed Services:** By leveraging AWS's managed services, MedEasy can focus on its core business processes and user experience rather than managing infrastructure and IT operations [12].

Cloud Delivery Model for MedEasy

MedEasy uses the SaaS delivery model to provide a seamless and efficient medicine ordering experience for its users. The decision to adopt this model is based on several key advantages that align with MedEasy's goals and requirements:

Key Advantages of SaaS:

- **Accessibility and Convenience:**
  - **Anywhere, Anytime Access:** Users can access MedEasy from any device with an internet connection, allowing them to order medicines conveniently without being tied to a specific location or hardware [13].
  - **No Installation Required:** SaaS eliminates the need for users to download or install software, making it easy to use and reducing potential technical barriers [13].

- **Scalability and Flexibility:**
    - **Elastic Resources:** MedEasy can scale its services up or down based on demand, ensuring optimal performance and availability during peak usage times [13].
    - **Rapid Deployment:** New features and updates can be rolled out quickly without requiring manual intervention from users, keeping the platform up-to-date with the latest enhancements [13].

- **Cost-Effectiveness:**
    - **Reduced IT Costs:** SaaS minimizes the need for extensive IT infrastructure and maintenance, lowering operational costs for both MedEasy and its users [13].
    - **Subscription-Based Pricing:** MedEasy can offer flexible pricing models, such as pay-as-you-go or subscription plans, making it more accessible to a wider range of users [13].
- **Security and Compliance:**
    - **Centralized Security Management:** Security measures are managed centrally by AWS, ensuring consistent protection of sensitive data across all users [13].
    - **Compliance with Standards:** SaaS providers like AWS adhere to industry standards and regulations, helping MedEasy meet compliance requirements for handling healthcare data [13].

- **Focus on Core Business:**
    - **Managed Infrastructure:** By relying on AWS to manage the infrastructure, MedEasy can focus on improving its core business processes and user experience rather than dealing with technical complexities [13].

Final Architecture

Cloud Mechanisms and Service Integration

- **AWS API Gateway:** API Gateway acts as the entry point for HTTP requests and triggers AWS Step Functions workflows. It securely manages and routes user requests, enabling seamless communication between the frontend and backend services. API Gateway ensures reliable handling of incoming requests, managing authentication, authorization, and throttling as needed [2].

- **AWS Step Functions:** AWS Step Functions orchestrates a workflow consisting of five Lambda functions to complete the process, among them, storing prescriptions, extracting medicine details, medicine matching with available inventory, processing orders, and sending order confirmation events. This service provides error handling, retries, and state management, ensuring a smooth execution of each step in the process [6].

- **AWS Lambda:** Lambda functions are in charge of the individual tasks in the Step Functions workflow. They run the code to do particular things: saving prescription images in S3, using AWS Textract to extract medicine details from the prescription image, medicine matching with the inventory in DynamoDB, processing the orders, and sending the order confirmation event to the EventBridge [3].

- **Amazon S3:** Amazon S3 is utilized for keeping the images of users' prescriptions uploaded by them and order data as JSON documents. It has highly durable and scalable object storage that makes sure the data is stored safely and is easily retrievable and processable when required [10].

- **Amazon DynamoDB:** DynamoDB is the main database used for the storage of the medicine inventory and order details. It comes with fast, consistent performance that has automatic scaling, thus, MedEasy can easily handle large volumes of data with minimal latency [9].

- **AWS Textract:** AWS Textract analyzes prescription images to extract structured text data, such as medicine names, power of medicine (mg), type (tube, tablets, capsule, syrup), and quantities. This service automates data extraction, reducing manual input errors and speeding up the ordering process [8].

- **AWS EventBridge:** EventBridge manages event-driven communication, triggering notifications and workflow processes based on specific events, such as order confirmations, status updates, publish data to SNS topics, and periodic low inventory alerts to sellers. It enables efficient, real-time event handling across AWS services [4].

- **AWS Simple Notification Service (SNS):** SNS delivers notifications through email to buyers and sellers with notifications such as order confirmations and status updates. The service merges with EventBridge and all other AWS components so that communication is always on time [11].

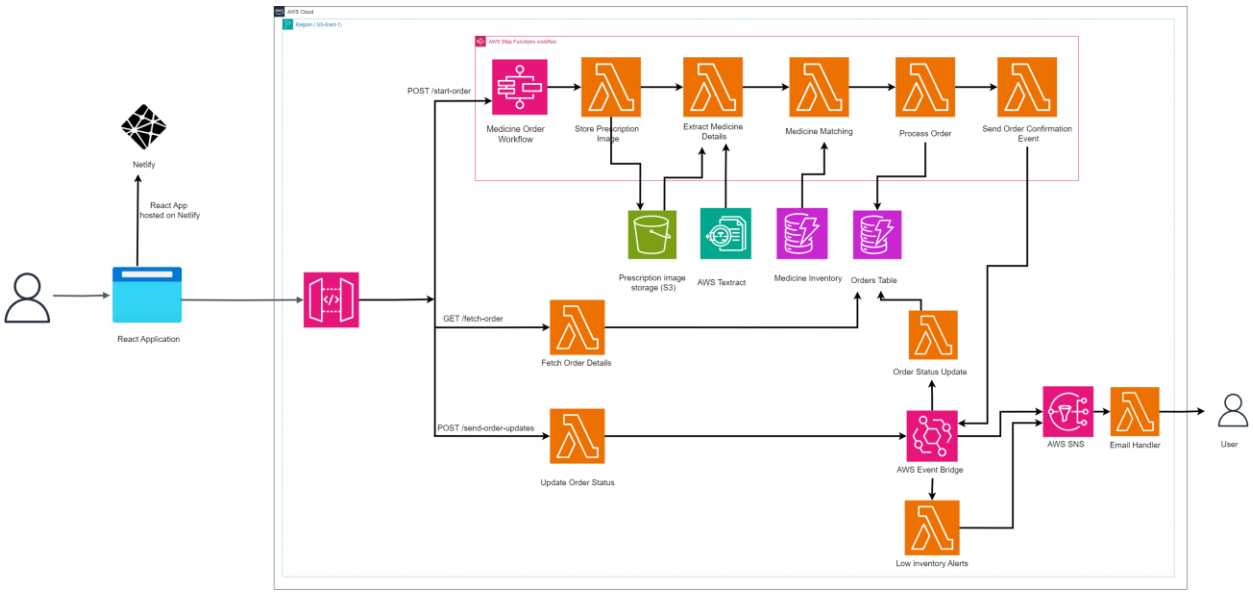

Figure 1: MedEasy Architecture Diagram

Data Storage

- **Amazon S3:** It stores customer order data documents in the format of JSON and prescription images, and it is a durable and reliable storage option that is scalable.
- **Amazon DynamoDB:** Holds medicine inventory and order details, offering fast and reliable access to dynamic data.

Programming Languages and Code Requirements

- **Node.js:**
  - **Used For:** Writing AWS Lambda functions since it is asynchronous and non-blocking. The integration of AWS services should be done seamlessly and with high performance [14].
- **JavaScript (React):**
  - **Used For:** Building the frontend application, offering a responsive and dynamic user interface that interacts with the backend services through API Gateway.

Deployment to the Cloud

Frontend Deployment:

- **Platform:** Deployed on **Netlify**, which provides a scalable and performant delivery of the React-based frontend application to users globally. Netlify handles continuous deployment and offers features like CDN integration, ensuring fast load times and high availability.

Backend Deployment:

- **Platform:** The entire serverless infrastructure is deployed on **AWS** using **AWS CloudFormation** for Infrastructure as Code. The templates of CloudFormation automate the deployment and management of all AWS services including Lambda functions, Step Function, Event Bridge, SNS, API Gateway configurations, DynamoDB tables, S3 buckets and so on. Furthermore, they make sure that the infrastructure provisioning is consistent and repeatable.

Data Security in MedEasy's Architecture

MedEasy's application architecture ensures the privacy of data being stored, transmitted, and processed through the entire architecture layer. The following is a brief explanation of how each layer is protected.

Data Security Measures

a. Data at Rest:

- **Amazon S3:**
  - **Encryption:** Data stored in Amazon S3 is encrypted using **Server-Side Encryption (SSE-S3)**, which automatically encrypts data at rest using AES-256 encryption [10].
  - **Access Control:** S3 bucket policies and IAM roles are used to restrict access to authorized users and applications only [10].

- **Amazon DynamoDB:**
  - **Encryption:** DynamoDB tables are encrypted at rest using **AWS Key Management Service (KMS)**, providing additional security for sensitive data like medicine inventory and order details [9].

b. Data in Transit:

- **TLS Encryption:**
  - **HTTPS:** All data transmitted between the client and API Gateway is secured using TLS encryption, ensuring data confidentiality and integrity during transmission [1].
  - **Internal Communication:** Data exchanged between AWS services, such as Lambda functions and DynamoDB, is also encrypted using TLS to protect against interception or tampering.

Security Mechanisms Used

a. Encryption Technologies:

- **AWS Key Management Service (KMS):**
  - **Description:** KMS is used to manage encryption keys for data at rest in DynamoDB and S3, which provides a safe solution for key management [10].
- **AES-256 Encryption:**
  - **Description:** AES-256 is used for encrypting data at rest, which provides strong encryption to protect data which is stored in S3 buckets [10].

b. Network Security:

- **Transport Layer Security (TLS):**
  - **Description:** TLS encryption is used for data in transit, which ensures security between the clients and services [1].

Reproducing MedEasy's Architecture in a Private Cloud

Hardware Infrastructure

a. Servers:

- **Purpose:** To run compute workloads similar to AWS Lambda and host databases similar to DynamoDB.
- **Requirements:**
  - **Compute Servers:** Multiple high-performance servers for running application logic, which includes CPU and memory similar to AWS Lambda's serverless capacities [15].
  - **Database Servers:** Servers for hosting NoSQL databases to replace Amazon DynamoDB [15].
- **Estimated Cost:**
  - **Compute Servers:** $30,000 - $50,000 for multiple high-performance servers.
  - **Database Servers:** $20,000 - $40,000 for database hardware.

b. Storage:

- **Purpose:** To store data similar to Amazon S3 and backups.
- **Requirements:**
  - **Network Attached Storage (NAS) or Storage Area Network (SAN):** High-capacity storage systems for storing user data, prescription images, and backup files [15].
- **Estimated Cost:**
  - **NAS/SAN Systems:** $15,000 - $30,000 for high-capacity storage solutions.

c. Networking Equipment:

- **Purpose:** To ensure safe and secure network connectivity and performance.
- **Requirements:**
  - **Routers, Switches, and Firewalls:** Enterprise-grade networking equipment for managing traffic and also ensuring security [15].

    - **Load Balancers:** Hardware load balancers to distribute traffic across servers, that ensures availability all the time.
- **Estimated Cost:**
    - **Networking Equipment:** $10,000 - $20,000.
    - **Load Balancers:** $5,000 - $10,000.

Software Infrastructure

a. Virtualization and Containerization:

- **Purpose:** To replicate AWS's serverless architecture using virtual machines or containers.
- **Requirements:**
    - **Hypervisor Software:** Such as VMware , to manage virtual machines.
    - **Container Orchestration:** Kubernetes or Docker Enterprise for containerized applications [15].
- **Estimated Cost:**
    - **Virtualization Software:** $5,000 - $15,000.
    - **Container Software:** $5,000 - $10,000.

b. Database Software:

- **Purpose:** To provide NoSQL database abilities similar to DynamoDB.
- **Requirements:**
    - **NoSQL Database Software:** Options like MongoDB for scalable database solutions [15].
- **Estimated Cost:**
    - **Database Software Licenses:** $5,000 - $10,000.

c. Storage Management:

- **Purpose:** To manage data storage, backup, and disaster recovery.
- **Requirements:**
    - **Backup Software:** For regular data backups and disaster recovery [15].
    - **Storage Management Software:** Tools for managing storage systems and ensuring data integrity [15].
- **Estimated Cost:**
    - **Backup Software:** $2,000 - $5,000.

- **Storage Management Software:** $3,000 - $7,000.

d. Network Security:

- **Purpose:** To ensure data security.
- **Requirements:**
  - **Firewall and Security Software:** Solutions like Cisco ASA or Palo Alto for network security [15].
  - **Encryption Software:** Tools for encrypting data at rest and in transit.
- **Estimated Cost:**
  - **Security Software:** $5,000 - $10,000.
  - **Encryption Software:** $2,000 - $4,000.

Personnel and Maintenance Costs

a. IT Staff:

- **Purpose:** To manage and maintain the private cloud infrastructure.
- **Requirements:**
  - **System Administrators:** For server and network management [15].
  - **Database Administrators:** For managing database systems and ensuring data integrity [15].
- **Estimated Cost:**
  - **Annual Salaries:** $100,000 - $200,000 for a small IT team.

b. Maintenance and Upgrades:

- **Purpose:** To keep hardware and software systems up to date and running efficiently [15].
- **Estimated Cost:**
  - **Annual Maintenance:** $20,000 - $50,000.

Total Estimated Cost:

- **Initial Setup Cost:** $120,000 - $240,000.
- **Annual Operational Cost:** $120,000 - $200,000.

Critical Cloud Mechanism for Cost Monitoring: AWS Lambda

Why AWS Lambda?

AWS Lambda is a main component of MedEasy's architecture, which is responsible for executing serverless functions that handle various tasks, such as processing prescriptions, extracting medicine details, and managing orders. While AWS Lambda offers great benefits in terms of scalability and cost-efficiency, it can also lead to unexpected charges if not properly monitored.

Potential Cost Drivers:

- **High Invocation Rates:** Lambda functions are billed based on the number of invocations and the duration of execution. If there is a spike in user activity or abnormal errors leading to retries, costs can increase very fast [16].

- **Long Execution Times:** Inefficient code or processing tasks that require long execution times can increase costs. Each millisecond of execution time adds to the overall cost, especially for high-volume tasks like text extraction or data processing [16].

- **Concurrency Limits:** If the concurrency limit is exceeded, then AWS Lambda may add additional costs for scaling beyond the pre-defined thresholds. Managing concurrency settings and optimizing function performance is essential to control costs [16].

Monitoring Strategies:

- **AWS CloudWatch:**
    - **Metrics and Alarms:** By Setting up CloudWatch metrics and alarms to keep track of the invocation counts of lambdas, execution duration, and error rates, it will allow us for real-time monitoring and alerts when thresholds are breached [16].
    - **Dashboards:** We can Use CloudWatch dashboards to visualize Lambda performance and identify patterns that may lead to cost increase.

Future Developments for MedEasy

Advanced Analytics and Insights

Feature: There is an existing feature to build data lake in Amazon S3 which can be used to implement advanced analytics, providing insights into user behavior, medicine trends, and sales patterns. This will optimize inventory management, enhance decision-making, and personalize user experiences.

- **Cloud Mechanisms:**
    - **Amazon Redshift:** Use Amazon Redshift for data warehousing to perform complex queries and gain real-time insights from the data stored in the S3 data lake [17].
    - **AWS Glue:** Employ AWS Glue for data cataloging and ETL (Extract, Transform, Load) processes, enabling seamless data movement between S3 and analytics tools [17].
    - **Amazon QuickSight:** Implement Amazon QuickSight to visualize data and create interactive dashboards, offering stakeholders actionable insights based on the analysis of the data lake [17].

Personalized Recommendations

Feature: By Utilizing the existing order data of customers from the S3 data lake to provide improved medicine recommendations, using historical order data to predict user preferences and suggest similar products.

- **Cloud Mechanisms:**
    - **Amazon Personalize:** Use Amazon Personalize to deliver tailored recommendations, enhancing user engagement and satisfaction [18].
    - **AWS Machine Learning Services:** Develop machine learning models using the data lake to analyze user behavior and predict medicine preferences or recommend health-related products [18].

References

[1] "AWS lambda," *GeeksforGeeks*, 22-Aug-2018. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-aws-lambda/. [Accessed: 06-Aug-2024].

[2] *Amazon.com*. [Online]. Available: https://aws.amazon.com/es/api-gateway/. [Accessed: 06-Aug-2024].

[3] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/textract/latest/dg/lambda.html. [Accessed: 06-Aug-2024].

[4] A. Patel, "AWS — Amazon EventBridge overview - awesome cloud - medium," *Awesome Cloud*, 07-Mar-2022. [Online]. Available: https://medium.com/awesome-cloud/aws-amazon-eventbridge-overview-what-is-aws-eventbridge-eventbus-introduction-features-use-cases-benefits-41c05f411317. [Accessed: 06-Aug-2024].

[5] *Amazon.com*. [Online]. Available: https://aws.amazon.com/ec2/. [Accessed: 06-Aug-2024].

[6] Datadog, "What is AWS Step Functions? How it Works & Use Cases," *Datadog*, 10-Jan-2022. [Online]. Available: https://www.datadoghq.com/knowledge-center/aws-step-functions/. [Accessed: 06-Aug-2024].

[7] J. Bonso, "Amazon simple workflow (SWF) vs AWS Step Functions vs Amazon SQS," *Tutorials Dojo*, 29-Apr-2019. [Online]. Available: https://tutorialsdojo.com/amazon-simple-workflow-swf-vs-aws-step-functions-vs-amazon-sqs/. [Accessed: 06-Aug-2024].

[8] "Textract vs Rekognition in detect text in picture," *Amazon Web Services, Inc*. [Online]. Available: https://repost.aws/questions/QUsCXe41EtTYq3QDaY18EnSg/textract-vs-rekognition-in-detect-text-in-picture. [Accessed: 06-Aug-2024].

[9] J. Bonso, "Amazon RDS vs DynamoDB," *Tutorials Dojo*, 14-Jan-2019. [Online]. Available: https://tutorialsdojo.com/amazon-rds-vs-dynamodb/. [Accessed: 06-Aug-2024].

[10] *Amazon.com*. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 06-Aug-2024].

[11] “AWS SNS - simple notification service,” *W3schools.com*. [Online]. Available: https://www.w3schools.com/aws/aws_cloudessentials_awssns.php. [Accessed: 06-Aug-2024].

[12] “Cloud deployment models,” *GeeksforGeeks*, 11-Jul-2021. [Online]. Available: https://www.geeksforgeeks.org/cloud-deployment-models/. [Accessed: 06-Aug-2024].

[13] *Amazon.com*. [Online]. Available: https://aws.amazon.com/types-of-cloud-computing/. [Accessed: 06-Aug-2024].

[14] “JavaScript asynchronous programming and callbacks,” *Nodejs.org*. [Online]. Available: https://nodejs.org/en/learn/asynchronous-work/javascript-asynchronous-programming-and-callbacks. [Accessed: 06-Aug-2024].

[15] A. Stanovici, “Calculating on-premise vs cloud costs,” *PMsquare*, 09-Jun-2022. [Online]. Available: https://pmsquare.com/analytics-blog/2022/6/9/calculating-on-prem-vs-cloud-costs. [Accessed: 06-Aug-2024].

[16] Moksha, “AWS lambda observability best practices,” *Cloud Native Daily*, 19-Jul-2023. [Online]. Available: https://medium.com/cloud-native-daily/aws-lambda-observability-best-practices-72cb38236086. [Accessed: 06-Aug-2024].

[17] *Amazon.com*. [Online]. Available: https://aws.amazon.com/big-data/datalakes-and-analytics/. [Accessed: 06-Aug-2024].

[18] *Amazon.com*. [Online]. Available: https://docs.aws.amazon.com/personalize/latest/dg/what-is-personalize.html. [Accessed: 06-Aug-2024].