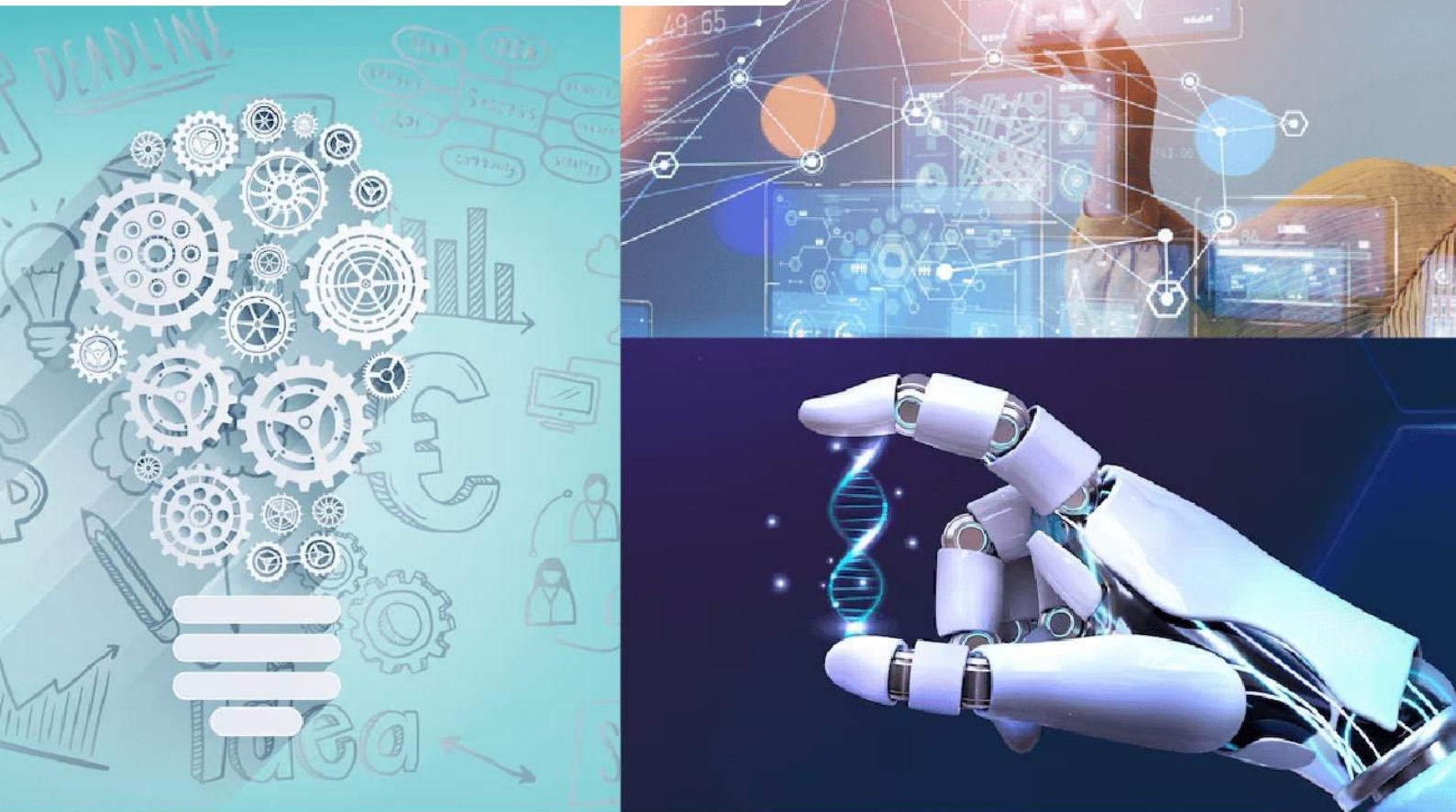




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 2, March - April 2026

Impact Factor: 8.152



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



Building and Securing Serverless Applications Using AWS Lambda and API Gateway

P. Chandra Sekhar¹, S. Navya², P. Amulya Reddy³, Pranavi Sree.CH⁴

Assistant Professor, Department of Computer Science & Engineering, Guru Nanak Institute Of Technology, Ibrahimpatnam, RR District, Telangana, India¹.

Student, Department of Computer Science & Engineering, Guru Nanak Institute Of Technology, Ibrahimpatnam, RR District, Telangana, India^{2,3,4}.

ABSTRACT: Serverless architectures are transforming how modern applications scale and integrate across cloud environments. This project focuses on designing and implementing a secure and scalable serverless API using AWS services such as AWS Lambda and Amazon API Gateway. It emphasizes efficient identity management through IAM roles, ensuring secure access control. The system enforces input validation and rate limiting to protect against misuse and enhance reliability. Integration with Amazon S3 and Amazon DynamoDB is implemented to support scalable storage and data persistence. Advanced deployment strategies such as blue-green deployment are used to ensure seamless updates with minimal downtime. Monitoring and logging are handled using Amazon CloudWatch Logs for real-time insights into system performance. The project also analyzes cold start performance to optimize response times in serverless environments. Furthermore, it highlights best practices for API security and robust error handling. Overall, the study demonstrates a high-availability, efficient, and secure serverless architecture on AWS.

KEYWORDS: Serverless Architectures, Scalable Applications, Secure API, AWS Lambda, Amazon API Gateway, Identity and Access Management (IAM), IAM Roles, Input Validation, Rate Limiting, Amazon S3, Amazon DynamoDB, Data Persistence, Blue-Green Deployment, Lambda Deployment Strategies, Amazon CloudWatch Logs, Monitoring, Cold Start Performance, API Security.

I. INTRODUCTION

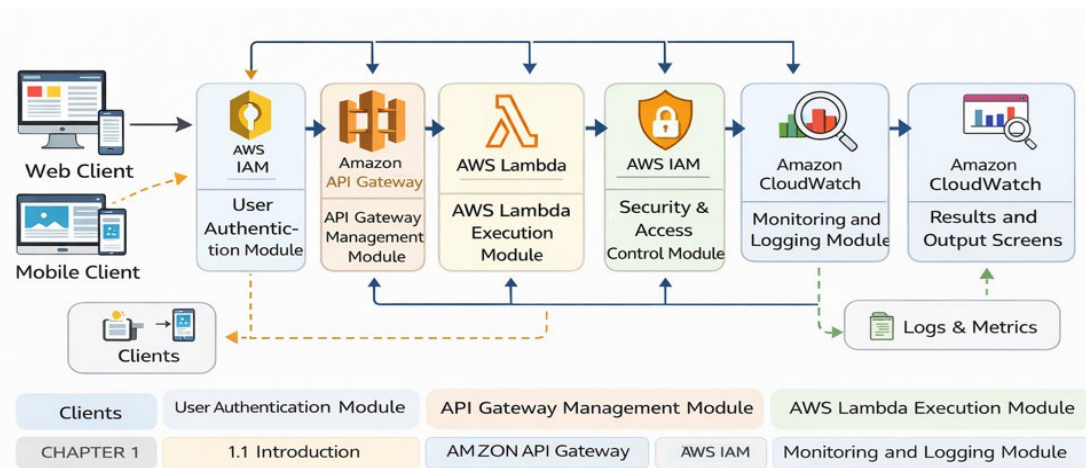
In recent years, the rapid evolution of cloud computing has transformed the way modern applications are developed, deployed, and maintained. Traditional server-based architectures require continuous infrastructure management, including server provisioning, capacity planning, maintenance, and scaling. These responsibilities increase operational complexity and cost, making such systems inefficient for applications with variable or unpredictable workloads. To address these challenges, serverless computing has emerged as a revolutionary cloud computing model. In a serverless architecture, developers focus only on writing application logic, while the cloud provider automatically manages the underlying infrastructure. This model enables applications to scale automatically, remain highly available, and operate on a pay-per-use pricing model. AWS Lambda and Amazon API Gateway are core services that enable the development of serverless applications on the Amazon Web Services (AWS) platform. AWS Lambda allows execution of backend code in response to events without provisioning or managing servers, while API Gateway provides a secure and scalable interface for handling client requests. Together, they form an event-driven architecture capable of handling millions of requests with minimal latency. Security is a major concern in serverless environments due to public exposure of APIs and distributed execution of functions. AWS provides robust security and API Gateway, demonstrating how cloud-native architectures can deliver scalable, cost-effective mechanisms such as Identity and Access Management (IAM), encryption, authentication, authorization, and monitoring through Amazon CloudWatch. By integrating these services, serverless applications can be built with strong security and compliance standards. This project focuses on designing, implementing, and securing serverless applications using AWS Lambda, and secures solutions for modern applications.

II. METHODOLOGY

The project follows an **event-driven and modular methodology** to ensure flexibility, scalability, and security. Each user request triggers a sequence of actions that are processed independently by cloud services. The methodology consists of the following steps:

1. The client sends an HTTP request to the application endpoint.
2. Amazon API Gateway receives and validates the request.
3. Authentication and authorization checks are performed using security policies.
4. The request is forwarded to the corresponding AWS Lambda function.
5. Lambda executes the required business logic.
6. The processed response is returned to the client.
7. Logs and performance metrics are recorded using Amazon CloudWatch.

III. MODELING AND ANALYSIS



System Architecture

MODULES DESCRIPTION

1. User Authentication Module

This module is responsible for verifying user identity before allowing access to the system. It ensures that only authorized users can invoke APIs and access backend services.

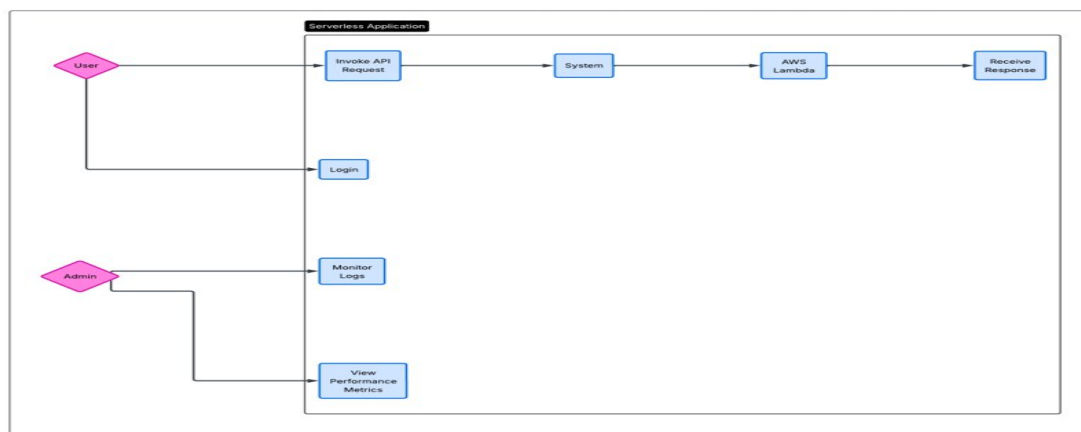
2. API Gateway Management Module

API Gateway serves as the entry point for all client requests. It manages routing, request validation, throttling, and response handling.

3. AWS Lambda Execution Module

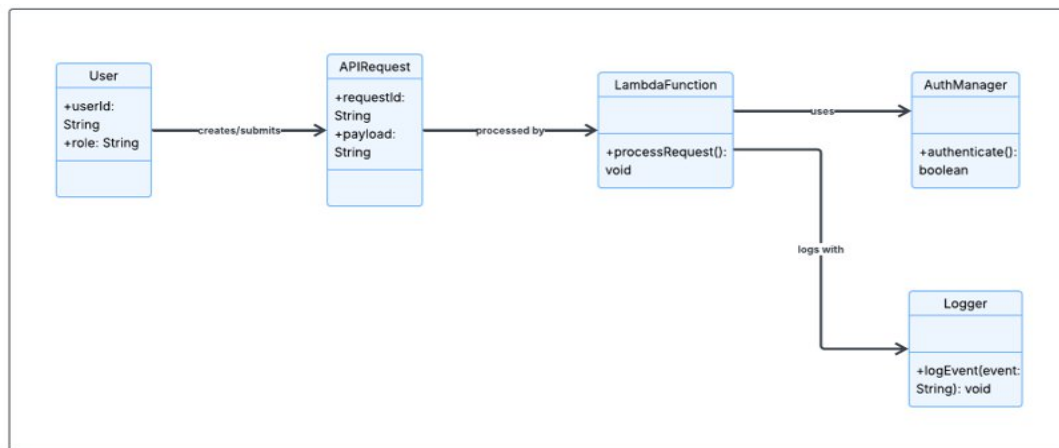
This module contains Lambda functions that process incoming requests. Each function executes independently and scales automatically based on demand.

Fig 3.2 USE CASE DIAGRAM



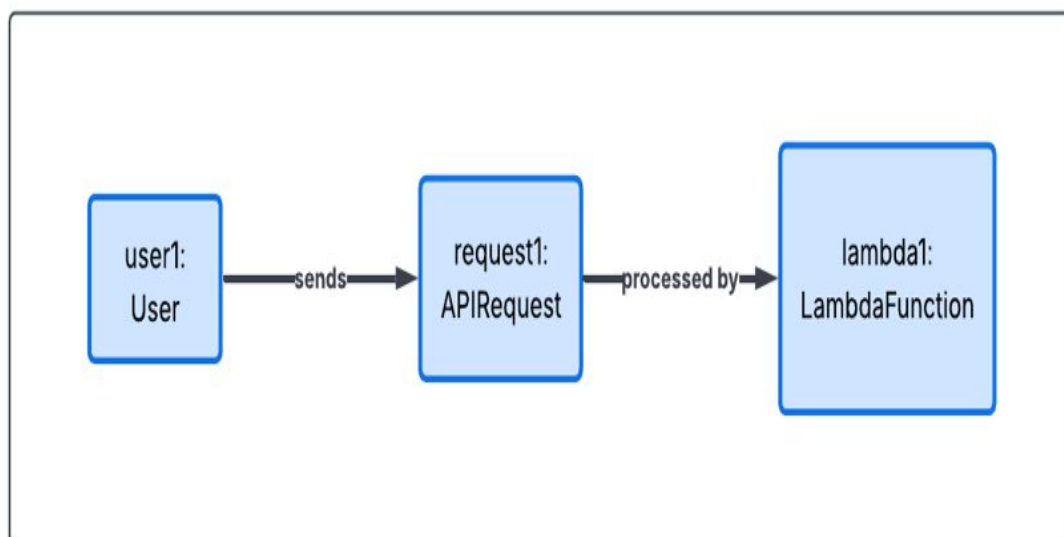
The Use Case Diagram represents the interaction between users and the serverless application. The primary actor is the user, who interacts with the system by invoking APIs and receiving responses. An administrator actor is responsible for monitoring logs and performance metrics. This diagram helps in identifying the main functionalities of the system and clearly defines the system boundary. It shows how users access the serverless application without direct interaction with backend services.

Fig 3.3 CLASS DIAGRAM



The Class Diagram illustrates the static structure of the system. It defines logical classes such as User, APIRequest, Lambda Function, AuthManager, and Logger along with their attributes and methods. Although the system is serverless, the class diagram provides a conceptual view of how data and responsibilities are organized. It helps in understanding system structure and relationships between different components.

Fig 3.4 OBJECT DIAGRAM



The Object Diagram represents a snapshot of the system at runtime. It shows instances of classes such as a specific user sending a request that is processed by a Lambda function instance. This diagram helps visualize how objects interact during a single request execution and provides clarity on dynamic behavior of the system.

Fig 3.5 ACTIVITY DIAGRAM



The Activity Diagram represents the workflow of the system. It shows activities such as receiving a request, validating the user, executing business logic, storing logs, and sending the response. Decision nodes are used to determine whether a request is authorized. This diagram highlights the control flow and decision-making process within the application.

IV. RESULTS AND DISCUSSION

The Result and Snapshots presents of the implemented serverless application to provide visual evidence of system configuration and execution. The snapshots demonstrate how different AWS services such as AWS Lambda, Amazon API Gateway, IAM, and CloudWatch are configured and used in the project. These screenshots help in understanding the practical implementation of the system and validate that the application has been successfully deployed and tested in the AWS cloud environment.

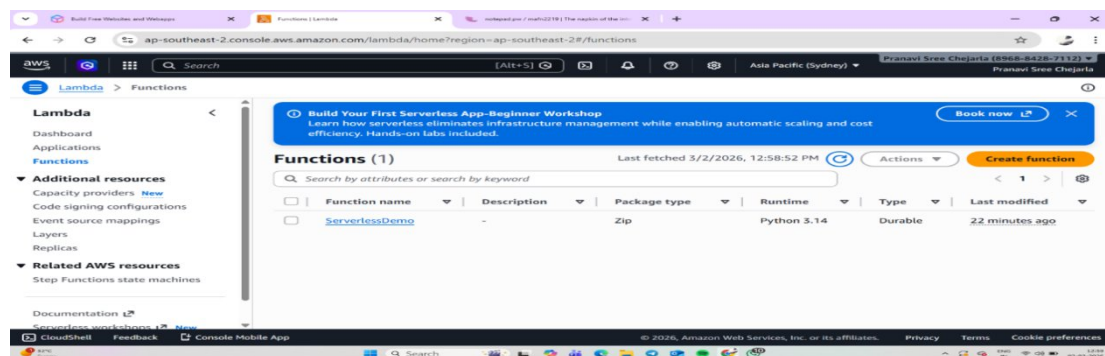


Fig 4.1: AWS Lambda Console Screenshot

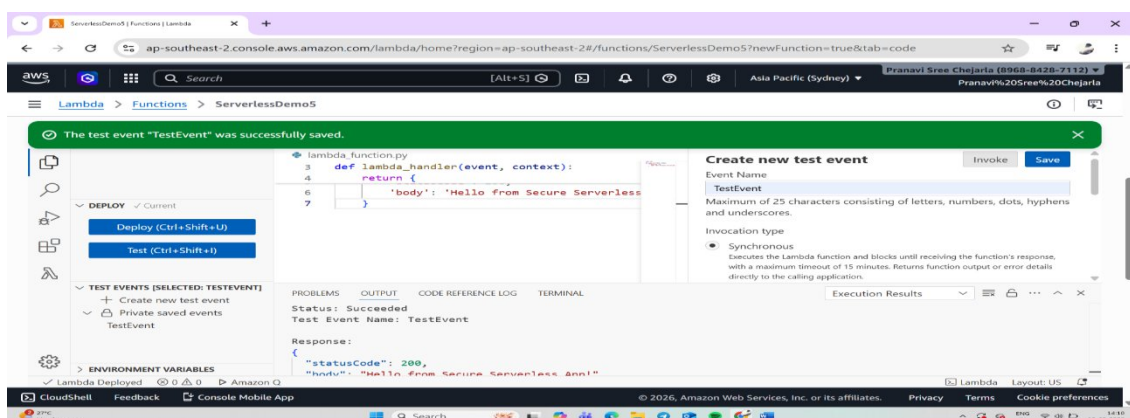


Fig 4.2: Lambda Test Output

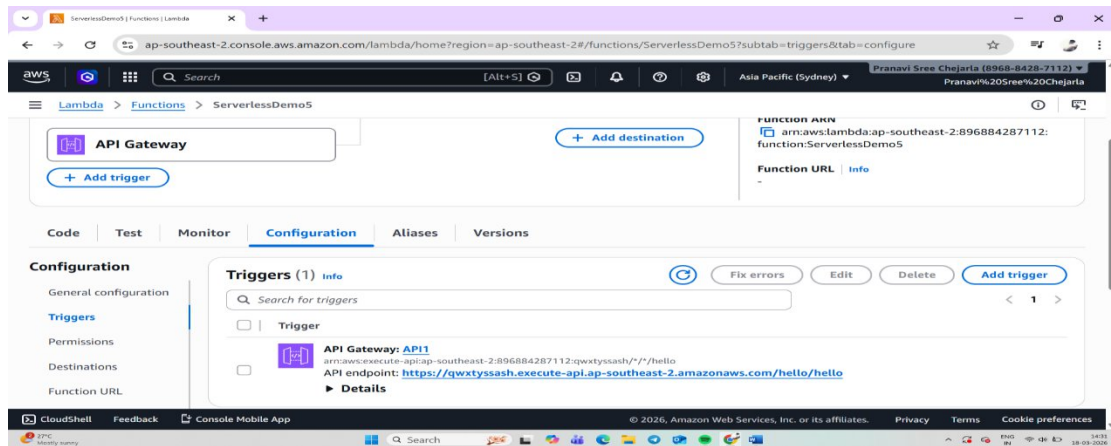


Fig 4.3: API URL



Fig 4.4: API Response Output

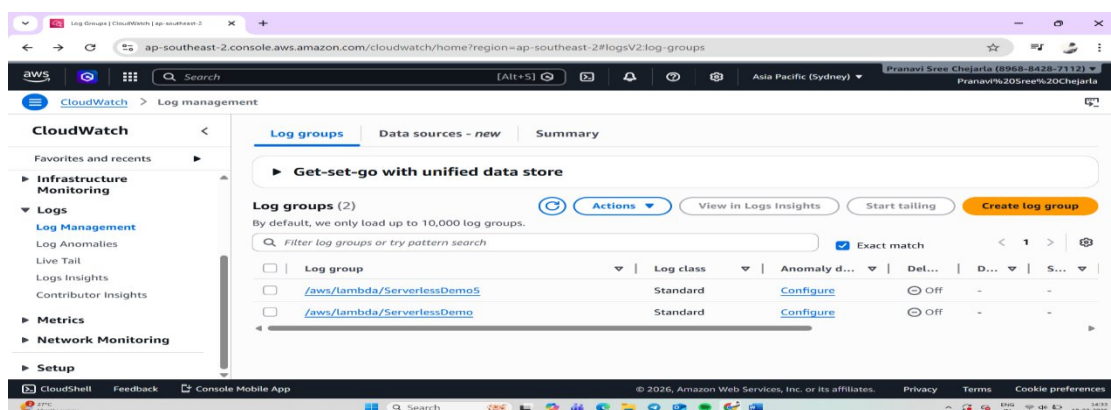


Fig 4.5: CloudWatch Logs

V. CONCLUSION

This project, Building and Securing Serverless Applications Using AWS Lambda and API Gateway, successfully demonstrated the design, implementation, and testing of a secure serverless architecture using cloud-native services. The system eliminates the need for traditional server management by leveraging AWS-managed services, thereby reducing operational overhead and infrastructure cost. The project showcased how AWS Lambda can be used to

DOI:10.15680/IJARETY.2026.1302144

execute backend logic in response to events and how Amazon API Gateway provides a scalable and secure interface for handling client requests. Security mechanisms such as AWS Identity and Access Management (IAM) and encrypted communication ensured controlled and protected access to system resources. Monitoring and logging using Amazon CloudWatch enabled real-time performance analysis and debugging. Through proper system design, modular implementation, and comprehensive testing, the proposed system achieved scalability, reliability, and security. The serverless approach proved to be efficient for applications with variable workloads and demonstrated the advantages of event-driven cloud architectures. Overall, this project highlights the effectiveness of serverless computing as a modern solution for building secure and scalable applications.

REFERENCES

- [1] J. R. Gunasekaran, A. Naseem, Dhamodhar Chatterjee and M. Patel, “Guarding Serverless Applications with SecLambda: A Security Framework for Amazon Web Services Lambda,” arXiv preprint, arXiv:2011.05322, 2023. [Online]. Available: <https://arxiv.org/abs/2011.05322> Dec. 2020
- [2] M. Yoon and H. Kim, “Performance Evaluation and Cold Start Optimization in AWS Lambda,” Access, 10.1109/ACCESS.2023.3298754 vol. 11, pp. 118945–118957, 2023. doi:
- [3] R. Patel, K. Sharma, H. Iyer and A. Nair, “Monitoring and Troubleshooting Serverless Applications Using Amazon CloudWatch,” Int. J. Cloud Comput. Appl., vol. 8, no. 2, pp. 44–53, 2023.
- [4] A. S. Gupta and L. K. Das, “High-Availability Serverless Architectures Using AWS Lambda and API Gateway,” IEEE Trans. Cloud Comput., vol. 12, no. 2, pp. 232–243, 2024. doi: 10.1109/TCC.2024.3321456
- [5] S. Rahman, “Blue/Green Deployment Strategies for Amazon Web Services Lambda,” the Medium Engineering Blog, 2024. [Online]. Available: <https://medium.com/aws-engineering/blue-green-deployment-strategies-for-aws-lambda-functions-b9e534a8e6b1>
- [6] C. A. Sill, “Serverless Architectures on Amazon Web Services: Application Programming Interface (API) Design, Security, and Scalability,” IEEE Softw., vol. 40, no. 6, pp. 72–81, Nov.–Dec. 2023. Doi : 10.1109/MS.2023.3211657
- [7] Amazon Web Services, “Tutorial: Using Amazon Web Services Lambda with Amazon API Gateway,” AWS Documentation, 2024. [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html>
- [8] Amazon Web Services, “Amazon Application programming interface Gateway Developer Guide,” Amazon Web Services Documentation, 2024. [Online]. Available: <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>
- [9] Amazon Web Services, “Best Practices for Working with Amazon Web Services Lambda Functions,” Amazon Web Services Documentation, 2024. [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html>
- [10] Amazon Web Services, “Serverless Multi-Tier Architectures with API Gateway and AWS Lambda,” https Amazon Web Services Whitepaper, 2024. [Online]. Available: <https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/web-application.html>
- [11] Amazon Web Services, “Serverless Deployment and Management using AWS SAM and AWS CDK,” https Amazon Web Services Whitepaper, 2024. [Online]. Available: <https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/serverless-deployment-and-management.html>
- [12] A. Haque, “Build and Deploy a Serverless API using AWS SAM, Lambda and API Gateway,” Dev.to, 2024. [Online]. Available: <https://dev.to/aws-builders/build-and-deploy-a-serverless-api-with-aws-sam-lambda-gateway-docker-4gb>

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152