Serverless Quiz Application with AWS DynamoDB and

Lambda

*A Project Based Learning Report Submitted in partial fulfilment of the requirements for the award of the degree*

*of*

**Bachelor of Technology**

**in the Department of Computer Science & Engineering**

**Cloud Based AI/ML Speciality (22SDCS07A)**

Submitted by

**2210030380: Chandra Chinmayee**

Under the guidance of

**Ms. P. Sree Lakshmi**



Department of Computer Science and Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075

March - 2025.

**1. Introduction**

* **Overview of the Project**

In the era of digital transformation and big data, serverless computing has emerged as a transformative technology for building scalable and cost-efficient applications. A serverless architecture enables developers to build and deploy applications without managing underlying infrastructure, leading to reduced operational overhead and improved agility. Unlike traditional server-based applications, which require provisioning, maintenance, and scaling of servers, serverless computing abstracts these complexities, allowing applications to scale automatically based on demand.

The **Serverless Quiz Application** is designed to provide an interactive, scalable, and cost-efficient quiz system using **AWS Lambda** and **Amazon DynamoDB**. This cloud-based solution eliminates infrastructure management, allowing seamless auto-scaling based on demand.

Users can take quizzes, receive scores instantly, and track progress in real-time. The backend logic, including quiz processing and scoring, is handled by **AWS Lambda**, while **DynamoDB** stores questions, answers, and user scores.

This quiz application is an excellent use case for serverless computing due to its event-driven nature. Questions, answers, and user progress can be stored in DynamoDB, while AWS Lambda functions handle user interactions, scoring, and analytics. By integrating AWS API Gateway, users can interact with the application using a web or mobile frontend. By eliminating infrastructure management complexities, businesses and educators can deploy interactive and cost-effective quiz applications that scale effortlessly, ensuring a smooth user experience.  
By leveraging **AWS API Gateway**, the quiz application enables web and mobile access, ensuring an engaging experience. The serverless approach significantly reduces costs, improves system performance, and enhances security compared to traditional hosted quiz platforms.

* **AWS Services Being Used and Justification**

1. **AWS Lambda** - Executes quiz logic, handles user input, and processes scores in a serverless environment.
2. **Amazon DynamoDB** - Stores quiz questions, answers, and user progress efficiently in a NoSQL format.
3. **Amazon API Gateway** - Facilitates communication between frontend applications and backend Lambda functions.
4. **AWS Cognito** - Provides authentication and user management, ensuring secure access.

These services ensure a highly scalable, secure, and cost-effective quiz application.

* **Project Purpose and Expected Outcome**

The purpose of this project is to develop an interactive quiz platform that eliminates server management complexities while offering real-time scoring, analytics, and user tracking. The expected outcome is a fully functional, scalable, and secure quiz system that can be integrated into educational platforms, corporate training programs, and certification assessments.

**2. Methodology**

* **Architecture and Workflow**

The Serverless Quiz Application follows a three-tier architecture

1. **Frontend Layer**

* Users interact with the quiz UI, submitting answers through a responsive interface.

1. **Backend Layer (AWS Lambda & API Gateway)**

* AWS Lambda executes quiz logic, calculates scores, and retrieves data from DynamoDB.
* API Gateway enables secure HTTP communication between the frontend and backend.

1. **Database Layer (Amazon DynamoDB)**

* Stores quiz questions, answers, and user progress in a structured NoSQL format.
* **Explanation of AWS Services Interaction**

**User Access:**

Users authenticate via AWS Cognito and request a quiz session through API Gateway.

**Quiz Execution:**

API Gateway routes requests to AWS Lambda, which fetches quiz questions from DynamoDB.

**User Input Processing:**

User responses are processed by Lambda, which updates scores in DynamoDB.

**Results & Analytics:**

Performance metrics are logged in Amazon CloudWatch for monitoring.

* **Justification for AWS Service Selection**

Selecting the right AWS services is crucial for building a scalable, secure, and cost-efficient **Serverless Quiz Application**. The primary objective of choosing AWS services is to ensure seamless performance, automatic scaling, and minimal operational overhead while optimizing costs.

AWS Lambda is the backbone of the serverless architecture, handling all backend logic, including user interactions, answer validation, and score calculations. Its event-driven nature ensures that functions execute only when triggered, eliminating the need for continuously running servers. Additionally, it provides automatic scaling, adjusting resources dynamically based on user load, making it a cost-effective solution. Since AWS Lambda is billed per execution time, it significantly reduces infrastructure costs compared to traditional server-based applications.

To store quiz-related data such as questions, answers, and user scores, Amazon DynamoDB is chosen for its **low-latency** and **highly scalable** NoSQL database capabilities. Unlike relational databases, DynamoDB eliminates complex joins, ensuring quick data retrieval, which is essential for real-time quiz applications. Its **auto-scaling** feature allows it to handle varying loads efficiently, while the **on-demand pricing model** ensures cost-effectiveness by charging only for read/write operations performed.

For secure and efficient communication between the frontend and backend, **Amazon API Gateway** is implemented. It acts as an intermediary that routes HTTP requests to AWS Lambda functions, enabling seamless integration. The API Gateway also enforces security through **JWT-based authentication** via AWS Cognito, preventing unauthorized access. Additionally, it supports **rate limiting** to prevent abuse and provides **CORS (Cross-Origin Resource Sharing)** configurations, allowing interaction between different domains, ensuring a smooth user experience.

Authentication and user management are critical for the quiz platform, especially for tracking user progress and providing personalized quiz experiences. **AWS Cognito** is selected for its built-in user authentication capabilities, allowing secure login and registration without custom backend authentication logic. It supports **multi-factor authentication (MFA)** for enhanced security and allows users to sign in using third-party providers such as **Google and Facebook**.

Finally, **AWS CloudWatch** is integrated to monitor and log application activity. It provides **real-time monitoring** of API requests, Lambda execution times, and database performance, helping developers identify potential bottlenecks and optimize the system. CloudWatch also sends alerts when system performance deviates from expected thresholds, ensuring quick response to failures.

By leveraging these AWS services, the Serverless Quiz Application achieves **scalability, security, and cost-efficiency** while delivering a seamless user experience. The combination of serverless compute, managed databases, and API management enables businesses and educators to deploy interactive quiz platforms without worrying about infrastructure management, making serverless computing the ideal choice for this use case.

**3. Implementation Steps**

* **AWS Infrastructure Setup**

1. **Create a DynamoDB Table**

Define attributes: question\_id, question\_text, options, correct\_answer, users.

1. **Develop Lambda Functions**

Function 1: Fetch quiz questions from DynamoDB.

Function 2: Evaluate responses and update scores.

1. **Setup API Gateway**

Configure HTTP endpoints to interact with Lambda.

Enable CORS for frontend access.

1. **User Authentication with Cognito**

Create a user pool for secure sign-in.

* **Security Policies, IAM Roles, and Access Controls**

1. **IAM Roles for Lambda**

Assign fine-grained permissions for Lambda to access DynamoDB.

1. **API Gateway Security**

Enable JWT-based authentication using Cognito.

Restrict access via API keys and rate limiting.

1. **DynamoDB Security**

Implement IAM policies to prevent unauthorized access.

* **Automation and CI/CD Pipeline**

1. **AWS SAM (Serverless Application Model)**

Automates deployment of Lambda functions and API Gateway.

1. **AWS CodePipeline**

Automates code updates with integration to GitHub or AWS CodeCommit.

1. **Monitoring & Logging**

Use AWS CloudWatch to track API performance.