Real-Time Sports Scoreboard using AWS AppSync and DynamoDB

*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

**Roll.no: 2210030478**

**NAME: B.SHAMITHA SRI**

Under the guidance of

**Ms. P. Sree Lakshmi**



Department of Computer Science & Engineering

Koneru Lakshmaiah Education Foundation, Aziz Nagar

Aziz Nagar – 500075

MARCH - 2025.

**Introduction**

1. **Overview of the Project**

The Real-Time Sports Scoreboard project aims to provide a scalable and efficient solution for tracking and updating sports scores in real time. Leveraging AWS AppSync and DynamoDB, the system ensures that users receive instant updates on game scores and statistics through a GraphQL API. The application will be designed to handle high concurrent user loads while maintaining low latency and high availability. The scoreboard will be accessible via web and mobile applications, offering a seamless experience for sports enthusiasts, broadcasters, and analysts.

1. **AWS Services Being Used and Justification**

AWS AppSync provides a managed GraphQL service that enables real-time data updates and synchronization, ensuring efficient and scalable data fetching. Amazon DynamoDB is a NoSQL database offering fast and reliable storage with automatic scaling, making it ideal for handling high-speed transactional data. AWS Lambda executes backend logic for data transformations and custom business rules. Amazon API Gateway manages API endpoints and secures access to the application. Amazon CloudFront speeds up content delivery using edge locations to ensure low-latency responses. AWS IAM (Identity and Access Management) controls permissions and security access across AWS services.

1. **Project Purpose and Expected Outcome**

The primary goal of the project is to deliver a real-time, low-latency, and highly available sports scoreboard system. The expected outcome includes a robust and scalable application capable of handling thousands of simultaneous connections while ensuring seamless and secure data interactions.

**Methodology**

1. **Architecture and Workflow**

The architecture follows a serverless design using AWS AppSync as the core API layer to interact with DynamoDB for real-time data storage. The workflow involves data ingestion through event-driven mechanisms such as webhooks, IoT devices, or manual updates. DynamoDB stores sports data with efficient indexing for quick lookups. AppSync manages real-time data fetching and synchronization for front-end applications. AWS Lambda executes business logic when complex transformations are required. API Gateway acts as an entry point for external clients, securing requests and handling authentication. CloudFront caches responses and accelerates content delivery for a smoother user experience.

1. **Explanation of AWS Services Interaction**

AppSync interacts with DynamoDB, where GraphQL queries and mutations fetch and update sports data. DynamoDB triggers Lambda functions when new scores are added, ensuring necessary backend processing. API Gateway manages requests, securing API calls and providing access control. CloudFront enhances delivery speed by caching frequently requested data to reduce latency. IAM enforces security policies by granting precise permissions to different AWS resources and services.

1. **Justification for AWS Service Selection**

The AWS services selected provide a robust, scalable, and cost-effective solution. AppSync enables real-time data synchronization without requiring additional WebSocket implementations. DynamoDB’s managed scaling and high-speed performance make it ideal for handling frequent sports data updates. The combination of Lambda, API Gateway, and CloudFront ensures a secure and optimized delivery pipeline.

**Implementation Steps**

1. **AWS Infrastructure Setup**

The first step involves creating a DynamoDB table with primary keys and global secondary indexes for fast queries. DynamoDB Streams are enabled to trigger Lambda functions for data processing. AWS AppSync is then set up by defining a GraphQL schema with queries, mutations, and subscriptions, followed by configuring resolvers to interact with DynamoDB. AWS Lambda functions are deployed to implement business logic for data validation and processing while setting up triggers from DynamoDB Streams and API Gateway. API Gateway is configured by setting up REST endpoints and integrating them with Lambda functions, along with enabling authentication mechanisms such as Cognito and IAM-based authorization. CloudFront distribution is then implemented to cache API responses and static content, optimizing for low-latency data delivery.

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

IAM roles and policies are defined to implement least-privilege access for Lambda, AppSync, and DynamoDB while restricting access using fine-grained access controls. Authentication and authorization are managed using AWS Cognito for user authentication and API Gateway security policies to protect endpoints. Data encryption and monitoring are enabled by integrating AWS CloudTrail for logging API activity and using AWS KMS for encrypting sensitive data stored in DynamoDB.

1. **Automation and CI/CD Pipeline**

Infrastructure as Code (IaC) is implemented using AWS CloudFormation or Terraform to automate deployments. The CI/CD pipeline is set up using AWS CodePipeline, CodeBuild, and CodeDeploy to automate testing and deployment processes. A blue/green deployment strategy is implemented to ensure zero-downtime updates, maintaining continuous availability and system reliability.

By leveraging AWS services effectively, this project delivers a reliable, real-time sports scoreboard with optimal performance and security.