High-Level Design Document

Web Application Project on AWS

Version: 1.0

Date:

Author:

# Document Versioning

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 |  |  | Initial Draft |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Document Versioning 2](#_Toc197076627)

[1. Introduction 4](#_Toc197076628)

[2. System Overview 5](#_Toc197076629)

[3. Architecture Design 6](#_Toc197076630)

[4. Component Design 7](#_Toc197076631)

[5. Data Flow and Interactions 8](#_Toc197076632)

[6. Security Design 9](#_Toc197076633)

[7. Scalability and Availability 10](#_Toc197076634)

[8. Monitoring and Logging 11](#_Toc197076635)

[9. Cost Overview and Optimization Strategy 12](#_Toc197076636)

[10. Appendix 13](#_Toc197076637)

# 1. Introduction

This document provides a high-level overview of the architecture and design approach for a cloud-based web application deployed on AWS. It outlines system components, architectural choices, and AWS services leveraged to deliver a scalable and secure solution.

# 2. System Overview

The application is a web-based platform that enables users to register, log in, and access personalized content. It is designed as a cloud-native solution for global reach and high availability, utilizing managed AWS services to reduce operational overhead.

# 3. Architecture Design

The architecture includes a multi-tier design: a presentation layer, an application logic layer, and a data storage layer. Key AWS services include Amazon S3 for static web hosting, AWS Lambda or EC2 for compute, RDS or DynamoDB for storage, and CloudFront for content delivery. The solution is deployed across multiple Availability Zones for resilience.

# 4. Component Design

- Frontend: Static content served from Amazon S3 with CloudFront CDN for low latency.

- Backend: RESTful services hosted using AWS Lambda or EC2 instances behind an Application Load Balancer.

- Database: Data is stored in Amazon RDS (PostgreSQL) or Amazon DynamoDB depending on use case.

- Third-Party Integrations: Integration with external APIs for authentication or data enrichment.

# 5. Data Flow and Interactions

End-users interact via a browser or mobile app. Requests are routed through CloudFront to the frontend, which interacts with backend services over secured endpoints. These backend services retrieve and store data in the database tier. Key interactions are governed by IAM roles and security groups.

# 6. Security Design

- Authentication and authorization handled using Amazon Cognito or IAM roles.

- All communication is encrypted using TLS.

- Data at rest is encrypted using AWS KMS.

- A Web Application Firewall (AWS WAF) is configured on the Application Load Balancer.

# 7. Scalability and Availability

- AWS Auto Scaling and Load Balancers ensure horizontal scalability.

- Resources are deployed in multiple Availability Zones to tolerate failures.

- AWS Global Accelerator or Route 53 may be used to optimize global traffic distribution.

# 8. Monitoring and Logging

- Amazon CloudWatch is used to monitor application health and resource utilization.

- AWS CloudTrail records API activity.

- Logs are stored in CloudWatch Logs for visibility and troubleshooting.

# 9. Cost Overview and Optimization Strategy

- The estimated monthly cost of the application is derived using the AWS Pricing Calculator, based on expected usage and service tiers.

- Major cost contributors include:

• Amazon EC2 or AWS Lambda for compute

• Amazon RDS or DynamoDB for data storage

• Amazon S3 and CloudFront for static hosting and CDN

- Reserved Instances or Savings Plans are considered for predictable workloads.

- Budgets and CloudWatch alarms are configured to monitor and alert on unexpected spend.

- Opportunities for cost optimization include:

• Right-sizing compute instances

• Using S3 lifecycle rules to archive or delete infrequently accessed data

• Leveraging serverless services where applicable to reduce idle costs

# 10. Appendix

Contains architecture diagrams and glossary of AWS services.