AWS Powered Online Bookstore

TEAM : SERVERLESS SEEKERS

1) **Nitya Reddy Yerram**
2) **Dhruv Khut**
3) **Pratik Kamanahalli Mallikarjuna**
4) **Rahul Dhingra**

Version 2.0
Date : May 11th, 2025

Revision History

Date	Version	Description	Author
2025-05-09	1.0	Initial project report draft created, covering scope and objectives	Nitya Reddy Yerram
2025-05-10	1.1	Added architecture diagram, AWS services flow, and infrastructure overview	Dhruv Khut
2025-05-11	1.2	Updated team contribution table; refined technical implementation details	Pratik K Mallikarjuna
2025-05-12	2.0	Finalized project report; completed proofread and formatting; prepared for submission	Rahul Dhingra

Table of Contents

1. Introduction

1.1 Purpose of this document

The purpose of this document is to provide a comprehensive and structured project plan for the development of the **AWS Powered Online Bookstore**. This project plan serves as both a technical blueprint and a management guideline for implementing a scalable, secure, and highly available serverless e-commerce web application using Amazon Web Services (AWS).

This document outlines the application's goals, target users, technical architecture, AWS services employed, integration strategies, and the key milestones throughout the development lifecycle.

In addition, this project plan provides detailed insights into the roles and responsibilities of each team member, timelines for deliverables, potential risks and mitigation strategies, testing and deployment plans, and future scalability considerations.

By documenting the project's objectives and methodology, this report ensures that all team members, supervisors, and stakeholders share a unified understanding of the system's vision, technical implementation, and expected outcomes. It also provides a foundation for future maintenance, upgrades, or scaling efforts after the initial deployment.

Furthermore, this document is intended to align technical execution with project objectives by mapping AWS service usage to business needs. It enables project scalability by providing a reusable and extensible framework that other teams or future versions of the project can build upon. It also reinforces adherence to best practices in cloud architecture, agile collaboration, and DevOps lifecycle management, ultimately contributing to a production-ready, enterprise-grade solution.

1.2 Intended Audience

This document is intended for a broad range of audiences who are directly or indirectly involved with the development, evaluation, or utilization of the **AWS Powered Online Bookstore** project.

The primary intended readers include:

- **Project Supervisor and Faculty Instructor**: Academic mentors who are overseeing the progress of the project, evaluating its technical soundness, adherence to requirements, and overall implementation quality.
- **Team Members and Project Leader**: The developers, designers, and architects collaborating on this project who will refer to this document for understanding the project scope, task allocation, timelines, technical decisions, and deployment guidelines.
- **External Reviewers and Industry Evaluators** (if applicable): Reviewers from the industry or external bodies who might assess the project for technical relevance, innovative use of cloud technologies, or its practical value as a demonstration of AWS serverless solutions.

This document not only provides technical specifications but also serves as a **communication bridge** among all stakeholders, ensuring that expectations are clearly defined, technical dependencies are understood, and milestones are transparent

1.3 Scope

The scope of the **AWS Powered Online Bookstore** project is to design, develop, deploy, and document a **fully serverless online bookstore platform** hosted entirely on AWS cloud infrastructure.

The application allows users to browse book collections, search for titles, add items to a shopping cart, complete secure checkouts, and view personalized recommendations—all within a web-based interface.

To achieve this, the platform leverages multiple AWS services to ensure scalability, security, high availability, and cost-effectiveness:

- **Frontend**: Hosted on **Amazon S3** as a static website, distributed globally using **Amazon CloudFront** for optimized content delivery and low latency access.
- **Backend**: Built using **AWS Lambda** functions exposed via **Amazon API Gateway**, enabling a fully serverless and event-driven architecture that scales automatically without provisioning servers.
- **Database**: Utilizes **Amazon DynamoDB** as the primary NoSQL database to store product catalogs, user carts, orders, and user profiles. Integration with **Amazon Neptune** supports graph-based relationships such as book recommendations.

- **Authentication**: Implemented via **Amazon Cognito** to handle user registration, login, identity federation, and token-based access control.
- **Caching and Performance**: **Amazon ElastiCache** is employed to cache frequently accessed data like popular books and recommendations, reducing backend load and improving response times.
- **Search and Analytics**: **Amazon OpenSearch (formerly Elasticsearch)** is integrated to provide fast, full-text search functionality across book titles, authors, and descriptions, as well as analytics on user search patterns.
- **CI/CD Pipeline**: The infrastructure and deployment are automated through **AWS CodePipeline** and **AWS CodeBuild**, ensuring continuous integration and deployment practices are followed.
- **Infrastructure Management**: All AWS resources are provisioned and managed using **AWS CloudFormation** templates, allowing for repeatable, consistent, and version-controlled infrastructure deployments.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
AWS Powered Online Bookstore	The name of the project; a cloud-based online bookstore implemented using AWS services in a serverless architecture.
Project Supervisor	A person responsible for guiding and overseeing the project to ensure academic and technical objectives are met.
Project Leader	The team member accountable for coordinating the team, tracking progress, and acting as the primary point of contact for the supervisor.
Team Member	A contributor actively participating in the design, development, deployment, or documentation of the project.
Milestone	A significant achievement or checkpoint marking the completion of a project phase or deliverable.
Git	A version control system used to track and manage changes to the project's source code.
Scrum	An agile software development framework that promotes iterative progress, collaboration, and adaptation.

Kanban	A workflow visualization method used for managing and improving work in progress across development stages.
Scrum Sprint	A short, time-boxed iteration (typically 1–2 weeks) where a set of features or tasks are developed and delivered.
Scrum Master	A facilitator ensuring the Scrum process is followed, resolving team impediments, and maintaining team focus.
Product Owner	The individual responsible for defining product features, prioritizing the backlog, and ensuring the final product meets stakeholder needs.
CloudFormation	AWS service used to provision and manage infrastructure resources through declarative templates.
API Gateway	AWS service that provides a fully managed interface to create, publish, maintain, monitor, and secure APIs for the backend.
Lambda Function	Serverless compute service used to run code in response to events without provisioning or managing servers.
DynamoDB	AWS NoSQL database service used to store application data such as product catalogs, user profiles, and orders.
Neptune	AWS graph database service used for managing relationships like book recommendations.
ElastiCache	AWS caching service used to improve application performance by reducing latency for frequently accessed data.
Cognito	AWS identity service used for user sign-up, sign-in, and access control for the bookstore application.
CloudFront	AWS content delivery network (CDN) used to distribute the web application globally with low latency.
CodePipeline	AWS continuous integration and continuous delivery (CI/CD) service for automating code deployment workflows.

1.4.2 Acronyms and abbreviations

Acronym or Abbreviation	Definitions
AWS	Amazon Web Services
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Delivery
IAM	Identity and Access Management
CDN	Content Delivery Network
DB	Database
S3	Amazon Simple Storage Service
VPC	Virtual Private Cloud
SNS	Simple Notification Service
SES	Simple Email Service
RDS	Relational Database Service
WAF	Web Application Firewall

1.5 References

1. Amazon Web Services. (n.d.). *AWS Bookstore Demo App (Serverless)*. GitHub Repository. https://github.com/aws-samples/aws-bookstore-demo-app
2. Amazon Web Services. (n.d.). *AWS Lambda Documentation*. Retrieved from https://docs.aws.amazon.com/lambda
3. Amazon Web Services. (n.d.). *Amazon API Gateway Documentation*. Retrieved from https://docs.aws.amazon.com/apigateway
4. Amazon Web Services. (n.d.). *Amazon DynamoDB Documentation*. Retrieved from https://docs.aws.amazon.com/dynamodb
5. Scrum Alliance. (n.d.). *Scrum Guide*. Retrieved from http://www.scrum.org/
6. Kunagi. (n.d.). *Agile Project Management Tool*. Retrieved from http://kunagi.org/
7. Amazon Web Services. (n.d.). *AWS CloudFormation Documentation*. Retrieved from https://docs.aws.amazon.com/cloudformation
8. Amazon Web Services. (n.d.). *Amazon Cognito Documentation*. Retrieved from https://docs.aws.amazon.com/cognito

2. Background and Objectives

2.1 Background

The AWS-Powered Online Bookstore project was conceived as a modern, scalable, and serverless web application to address the evolving needs of online retail platforms. With the rapid growth of e-commerce and the increasing demand for reliable and cost-effective cloud-based solutions, this project aims to leverage Amazon Web Services (AWS) to build an end-to-end bookstore platform capable of handling product catalog management, shopping cart functionality, user authentication, order processing, and recommendations, all without managing traditional servers.

The project utilizes a wide range of AWS services including **Lambda, API Gateway, DynamoDB, Cognito, S3, CloudFront, ElastiCache, Neptune, and CodePipeline**, demonstrating how serverless technologies and managed services can simplify application development and maintenance. By adopting infrastructure as code (IaC) with CloudFormation templates, the bookstore ensures reproducible deployments, scalability, and high availability across AWS regions.

This application also serves as a hands-on learning opportunity for students to apply their knowledge of cloud architecture, DevOps pipelines, CI/CD automation, and agile project management. By integrating multiple AWS services into a cohesive architecture, the team gained practical experience in cloud-native development practices while solving real-world challenges of building a feature-rich online store.

2.2 Objectives

The main objectives of the AWS Online Bookstore project are:

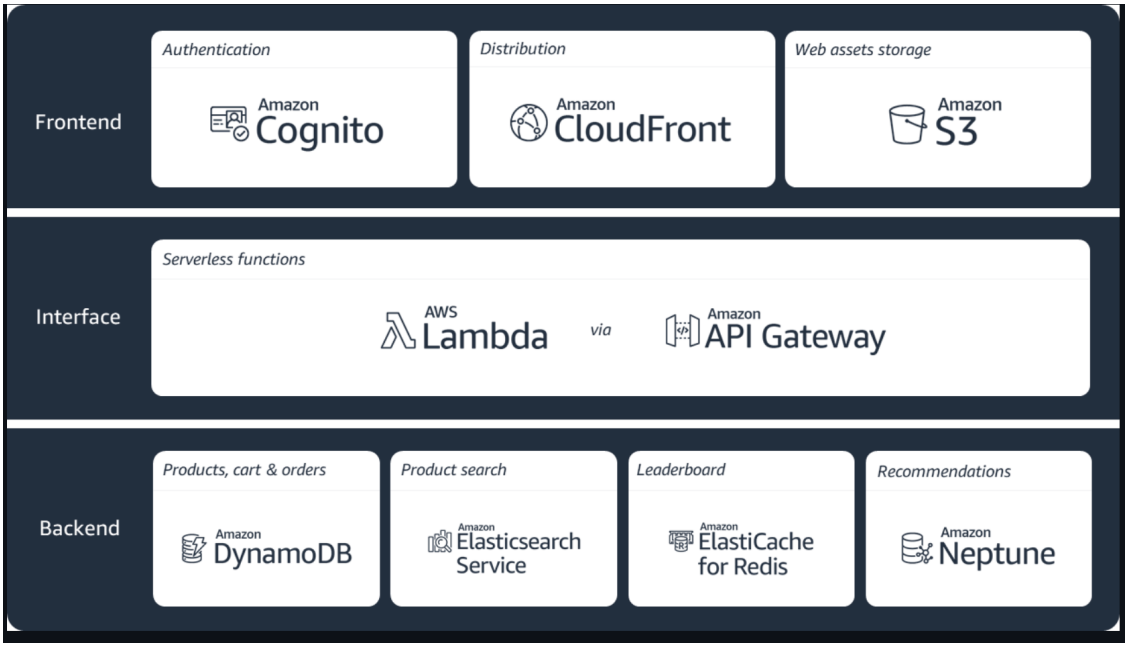
1. **To develop a fully serverless, scalable, and reliable online bookstore application using AWS services**
    - Eliminate the need for managing physical or virtual servers by relying on managed services.
    - Use **Lambda functions** for backend logic to automatically scale based on request volume.
    - Store product, order, and user data in **DynamoDB** for low-latency access.
2. **To implement modern authentication and user management using Amazon Cognito**

    - Enable secure user sign-up, sign-in, and access control.
    - Simplify identity management by integrating Cognito with API Gateway authorizers.
3. **To optimize content delivery and web performance through S3 and CloudFront**
    - Host the static frontend website on **Amazon S3**.
    - Use **CloudFront** to distribute content globally with reduced latency and caching.
4. **To enable real-time recommendations and caching using Neptune and ElastiCache**
    - Use **Amazon Neptune** to store and query book relationship graphs for recommendations.
    - Implement **ElastiCache (Redis)** to improve response times for frequently accessed queries.
5. **To establish a robust CI/CD pipeline for continuous integration and delivery using CodePipeline and CodeBuild**
    - Automate infrastructure deployment and updates with **CloudFormation templates**.
    - Enable automatic deployment of new backend Lambda functions and frontend assets.
6. **To provide a comprehensive demonstration of AWS serverless architecture for educational purposes**
    - Serve as a reference implementation of a cloud-native e-commerce platform.
    - Help future students and developers understand real-world use cases of AWS services integration.
7. **To apply agile project management and collaboration practices (Scrum methodology)**
    - Organize the project work into sprints.
    - Conduct sprint planning, reviews, and retrospectives using **Kunagi**.

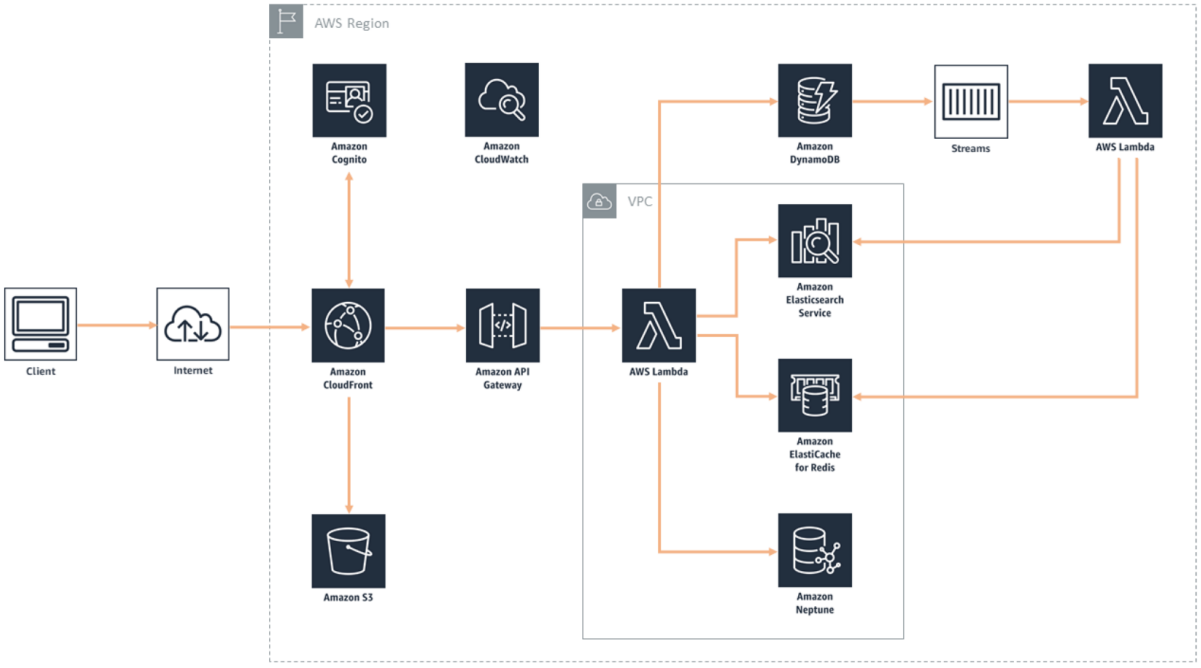
By achieving these objectives, the project aims to deliver a cloud-based solution that is **cost-efficient, highly available, performant, and maintainable**, while aligning with modern software engineering practices.

3. Architecture & High Level Design

3.1 Summary Architecture Diagram



3.2 High-level end to end diagram



4. Organization

The organization of the AWS Powered Online Bookstore project is structured around a collaborative team model, where each member took ownership of a specific domain of the application stack. The team functioned under an agile development approach with clearly assigned responsibilities, enabling parallel development and seamless integration.

4.1 Project group

Name	Initials	Responsibility (roles)
Nitya Reddy Yerram	NRY	Backend Developer: Implemented AWS Lambda functions, API Gateway integration, and DynamoDB operations for product & orders
Dhruv Khut	DK	Frontend Developer: Developed S3-hosted website UI, configured CloudFront distribution, integrated frontend with API Gateway
Pratik Kamanahalli Mallikarjuna	PKM	DevOps Engineer: Created CloudFormation templates, set up CI/CD pipelines using CodePipeline & CodeBuild, configured ElastiCache & Neptune
Rahul Dhingra	RD	Authentication & Data Integration Engineer: Set up Amazon Cognito for user authentication, integrated Elasticsearch for search, linked DynamoDB streams to Elasticsearch

4.2 Customer

The primary customers for the AWS Powered Online Bookstore include:

- **General Consumers**: Individuals looking to browse, search, and purchase books online through a reliable, scalable web platform with personalized recommendations and seamless checkout experiences.
- **Bookstore Administrators**: Internal users who manage product listings, inventory, order statuses, and customer data through backend services integrated with DynamoDB, Elasticsearch, and Neptune databases.
- **Third-party Sellers/Publishers**: Vendors or publishers who wish to list and manage their books on the platform, leveraging APIs to update content, prices, and availability.
- **Developers and DevOps Teams**: Technical users utilizing the project as a reference architecture or demo for learning AWS serverless technologies, CI/CD pipelines, and infrastructure-as-code deployment practices.
- **Academic/Research Users**: University students or researchers exploring serverless architectures, distributed systems, and cloud-native application design for educational purposes.

5. Development process

The development process for the **AWS Powered Online Bookstore** followed an **agile, iterative approach** to ensure continuous integration, flexibility, and incremental improvements. The team worked collaboratively in **scrum-based sprints**, with each sprint focusing on delivering functional modules while incorporating feedback from peer reviews and testing.

Key characteristics of the development process:

1. **Agile Methodology with Scrum Framework**

   The project was managed using **Scrum**, breaking down the work into 2-week sprints. Each sprint included sprint planning, daily stand-ups, sprint reviews, and retrospectives to track progress and adapt priorities. This allowed the team to quickly respond to changes in requirements, technical challenges, or feedback.

2. **Continuous Integration and Continuous Deployment (CI/CD)**

A robust CI/CD pipeline was implemented using **AWS CodePipeline** and **AWS CodeBuild** to automate the build, test, and deployment process. Every commit to the GitHub repository triggered a pipeline that validated infrastructure changes (CloudFormation templates), deployed Lambda functions, and updated the frontend hosted on **S3** and served via **CloudFront**.

3. **Serverless-First Architecture**

   The project followed a **serverless-first design philosophy**, leveraging **AWS Lambda**, **API Gateway**, **DynamoDB**, **Elasticsearch Service**, **Neptune**, and other managed AWS services to reduce operational overhead, increase scalability, and lower costs. No EC2 or traditional server management was required.

4. **Infrastructure as Code (IaC)**

   All infrastructure was provisioned using **AWS CloudFormation templates**, enabling repeatable, version-controlled deployments across environments (dev, test, prod). Changes to infrastructure were peer-reviewed before being merged and deployed via the CI/CD pipeline.

5. **Security by Design**

   The development process incorporated security best practices from the start. This included **IAM least-privilege roles**, secure API Gateway configurations, use of **Cognito** for authentication and authorization, and enabling **WAF** rules for protection against common web exploits.

6. **Automated Testing and Monitoring**

   Testing was embedded into the CI/CD pipeline, including:

   - Unit testing of Lambda functions
   - API testing using **Postman** or **AWS API Gateway test tools**
   - Integration testing across components

   Logging and monitoring were implemented using **Amazon CloudWatch** to capture application logs, metrics, and alerts.

7. **Collaborative Tools and Version Control**

The team collaborated via GitHub for version control, **Kunagi** for agile project management, and **Slack/Teams** for real-time communication. Code reviews were mandatory for every pull request to maintain quality and consistency.

8. **Incremental Delivery and Deployment**

   Every sprint aimed to deliver a **deployable increment** of the product. Early sprints focused on core backend APIs and data models, while later sprints added frontend integration, search features (Elasticsearch), and personalization (Neptune recommendations).

6. Project risks

Possibility	Risk	Preventive Action
High	AWS service limits exceeded (e.g., Lambda concurrency, DynamoDB throughput)	Monitor AWS quotas, request quota increases in advance, optimize function usage
Medium	Deployment failure due to CloudFormation template errors	Validate templates with cfn-lint, test in sandbox environment before production
Medium	Misconfigured IAM roles leading to access issues	Apply least-privilege principle, review policies regularly
Low	Data inconsistency across databases (DynamoDB, Elasticsearch, Neptune)	Implement automated data validation scripts, schedule periodic consistency checks
Medium	Team unfamiliarity with AWS CI/CD tools causing delays	Allocate training time, use AWS samples and documentation, pair programming
Medium	Cost overruns from unused or un-deleted resources	Enable billing alerts, automate cleanup scripts, monitor AWS Cost Explorer
Low	Security vulnerabilities in API Gateway or Cognito	Apply WAF rules, enable API throttling, use Cognito password policies

The table above outlines the key risks identified during the development of the AWS Powered Online Bookstore project. Each risk was assessed based on its likelihood and potential impact on project success. Appropriate preventive actions were planned to mitigate these risks proactively. Ongoing risk monitoring and mitigation strategies were incorporated into the development workflow to ensure resilience, maintain system reliability, and avoid unnecessary delays or cost overruns. Any new risks identified during implementation will be documented and reviewed by the team for timely action.

7. Communication

To further streamline coordination, the team adopted agile-based communication cycles such as sprint planning and retrospectives, which were instrumental in tracking milestones and resolving issues proactively. Task ownership was clarified during sprint standups, ensuring that each member was aligned with their responsibilities and timelines.

We also made use of **Kanban boards (e.g., Trello or GitHub Projects)** to visually manage task status—categorized as "To Do," "In Progress," and "Completed." This visual representation helped identify blockers early and provided visibility into each member's progress.

In addition, we maintained a shared **meeting notes document** where weekly action items, feedback from code reviews, and pending decisions were logged. This document served as a single source of truth for the entire team and was regularly reviewed to maintain accountability.

Our communication protocols supported both technical and non-technical coordination and created a transparent and productive environment. These practices helped the team adapt quickly to changes, maintain alignment with project goals, and deliver milestones effectively.

The communication plan included:

- Regular **weekly team meetings** (virtual/in-person) to discuss progress, blockers, and next steps
- **Daily asynchronous updates** via Slack/Teams to share individual work status and highlight urgent issues

- **Documentation repository** maintained on GitHub Wiki for shared access to architecture diagrams, deployment instructions, and technical notes
- **Email updates** for formal notifications to the project supervisor and stakeholders

Communication protocols ensured transparency, minimized misunderstandings, and facilitated quick resolution of technical and operational challenges throughout the development lifecycle.

7.1 Collaboration

The collaboration process was grounded in an agile mindset, emphasizing teamwork, accountability, and flexibility. The team employed several tools and practices to foster collaboration:

- **Version control and branching workflow** using Git and GitHub, enabling each member to work independently on features while integrating changes through pull requests and code reviews
- **Collaborative editing and documentation** on shared Google Docs and Microsoft Word for drafting reports, manuals, and diagrams
- **Scrum ceremonies** including sprint planning, sprint reviews, and retrospectives, allowing the team to iteratively reflect and adapt
- **Issue tracking and task assignments** on GitHub Issues and project boards, keeping work visible and prioritized

This collaborative approach allowed the distributed team to contribute effectively across different technical areas while maintaining alignment on shared goals.

7.2 Git

All source code, configuration files, deployment scripts, infrastructure-as-code templates, and related documentation were version-controlled using **Git** and hosted in a **GitHub repository**. The repository served as the single source of truth for the project, ensuring traceability, accountability, and easy rollback in case of issues.

Key practices in Git usage included:

- Using **feature branches** for development and merging into the main branch after peer reviews
- Leveraging **GitHub Actions** for triggering CI/CD pipelines where applicable
- Maintaining detailed **commit messages** to document changes clearly
- Using **tags and releases** to mark significant milestones (e.g., v1.0, v2.0)

Repository URL:
https://github.com/nityareedy/AWS_Powered_OnlineBookstore

8. References

1. Amazon Web Services. (2024). AWS Bookstore Demo App – GitHub Repository. Retrieved from https://github.com/aws-samples/aws-bookstore-demo-app
2. Amazon Web Services. (2024). Amazon DynamoDB Developer Guide. Retrieved from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
3. Amazon Web Services. (2024). AWS Lambda Developer Guide. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
4. Amazon Web Services. (2024). Amazon API Gateway Developer Guide. Retrieved from https://docs.aws.amazon.com/apigateway/latest/developerguide/
5. Amazon Web Services. (2024). Amazon Cognito Developer Guide. Retrieved from https://docs.aws.amazon.com/cognito/latest/developerguide/
6. Amazon Web Services. (2024). AWS CloudFormation User Guide. Retrieved from https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/
7. Amazon Web Services. (2024). Amazon Elasticsearch Service Developer Guide. Retrieved from https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/
8. Amazon Web Services. (2024). AWS CodePipeline User Guide. Retrieved from https://docs.aws.amazon.com/codepipeline/latest/userguide/