BS-Software Engineering

Software Development & Construction

Report

Instructor: Dr Zunnurain Hussain **Teaching Assistant: Umair Makhdoom**

Project Title: Cloud Based E – Commerce Platform

Group Members:

- Muhammad Numan Tahir – BSSE23069
- Muhammad Zubair Akhter – BSSE23079

Contents

Project Report: ShopSwift E-Commerce Platform

1. Executive Summary

ShopSwift is a high-performance, modern e-commerce application designed for scalability and seamless user experiences. It serves two primary user archetypes: Customers, who browse and purchase products, and Administrators, who manage inventory and oversee order fulfillment. The system is built with a decoupled architecture, integrating with the Commercetools commerce engine and providing a blueprint for AWS Cloud deployment.

2. Technology Stack

- Frontend Framework: React 19 (ESM-based)
- Styling: Tailwind CSS (Utility-first CSS)
- Language: TypeScript (Type-safe development)
- Icons: Lucide React
- Commerce Engine: Commercetools (Headless Commerce API)
- Cloud Blueprint: AWS (ECS, Kinesis, S3, CloudFront)
- Build Tool: Vite / esm.sh

3. Project Architecture

The application follows a Headless Architecture pattern:

1. Frontend (Presentation Layer): A responsive Single Page Application (SPA) that manages state, user interactions, and client-side routing between views (Shop, Cart, Admin).
2. API Service Layer: A centralized communication hub (services/api.ts) that handles OAuth2 authentication and manages requests to Commercetools.

3. Backend Proxy (Blueprint): A Node.js ECS-ready service designed to secure sensitive API credentials and handle server-side logic.

4. Feature Set

Customer Experience

- **Dynamic Storefront:** Real-time product discovery with high-quality imagery and detailed descriptions.
- **Reactive Shopping Cart:** Persistent cart management with automatic subtotal, tax, and total calculations.
- **Streamlined Checkout:** Secure shipping information capture and order submission.
- **Live Notifications:** Toast-based feedback system for all major interactions.

5. Administrative Management

- **Inventory Dashboard:** Real-time view of product projections (staged and published).
- **Price Control:** Dynamic price updating with immediate synchronization to the commerce backend.
- **Order Fulfillment Center:** A sophisticated interface to track order lifecycles (Pending → Paid → Shipped → Delivered).
- **Optimistic UI:** Immediate interface updates for admin actions to ensure a lag-free management experience.

6. Technical Implementation Details

Authentication & Authorization

The system implements a robust role-based access control (RBAC) model.

- Admins gain access to sensitive inventory and order management endpoints.

- Customers utilize me endpoints for personal profile and order history management.
- OAuth2 Flow: Implements Client Credentials flow with token caching to minimize latency and API overhead.

7. Data Management

- Commercetools Integration: Utilizes Product Projections for high-speed frontend retrieval and the Products API for administrative accuracy.
- Event-Driven Analytics: Captures user journeys (VIEW_HOME, ADD_TO_CART, PURCHASE) to provide data for recommendation engines and sales reporting.

8. Security Model

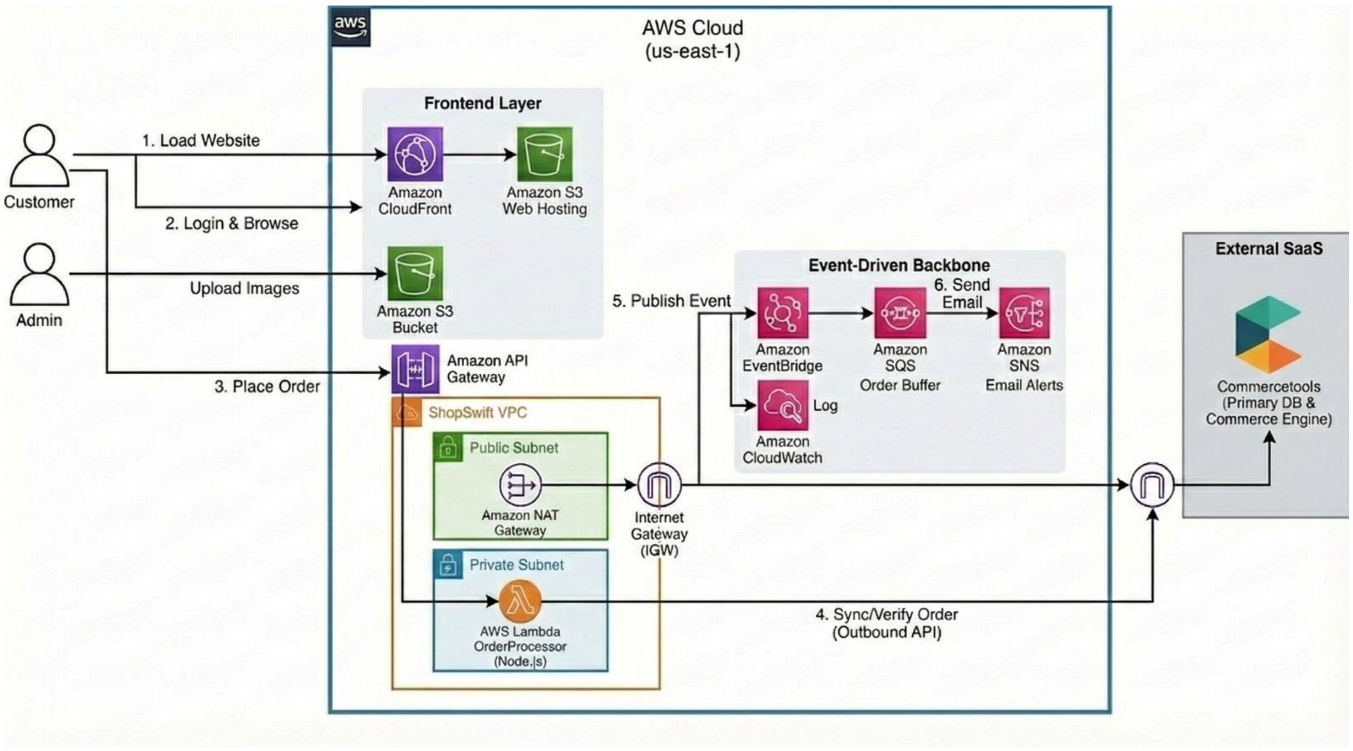
- Credential Management: The project identifies the need for a backend proxy (AWS ECS) to hide Client Secrets from the client-side bundle.
- Input Validation**:** Form data is sanitized and validated before being transmitted to the commerce engine.
- Secure Routing**:** Protected views (Admin) are shielded behind authentication checks.

9. AWS Infrastructure Blueprint

The project is designed for a highly available cloud environment:

- Frontend Hosting: AWS S3 + Amazon CloudFront for global edge delivery.
- Compute: AWS Fargate (ECS) for the Node.js API proxy.

10. Architecture Diagram:



Explanation

1. Overview

This architecture diagram represents a cloud-based e-commerce system deployed on Amazon Web Services (AWS). It illustrates how users interact with the website, how orders are processed securely, and how notifications are delivered using scalable and reliable cloud services.

2. User Interaction

Customers access the e-commerce website to browse products, log in, and place orders. Administrators manage the system by uploading product images and managing content.

3. Frontend Layer

The frontend of the application is hosted on Amazon S3, which stores static website files such as HTML, CSS, and images. Amazon CloudFront is used as a Content Delivery Network (CDN) to deliver the website efficiently and reduce latency for users across different locations. Product images uploaded by the admin are also stored in Amazon S3 buckets.

4. API Layer

Amazon API Gateway acts as the entry point for backend services. It receives requests from the frontend, such as placing an order or fetching product data, and securely forwards them to the backend for processing.

5. Backend Processing

The backend logic runs inside a secure Virtual Private Cloud (VPC).

- The public subnet contains a NAT Gateway that allows controlled internet access.
- The private subnet hosts AWS Lambda, which processes business logic such as order validation and synchronization.

This setup ensures that sensitive operations remain secure and isolated from direct internet access.

6. External Commerce System

The system integrates with Commercetools, an external Software-as-a-Service (SaaS) platform, which acts as the primary commerce engine and database. Orders are verified, stored, and managed through this external service.

7. Event-Driven Services

Once an order is placed, an event-driven workflow is triggered:

- Amazon EventBridge publishes order-related events.
- Amazon SQS temporarily stores order messages to ensure reliable processing.
- Amazon SNS sends order confirmation emails and notifications to users.

- Amazon CloudWatch logs system activities for monitoring and debugging.

8. Connectivity

An Internet Gateway (IGW) enables communication between AWS resources and external services, while the NAT Gateway allows private backend components to access the internet securely without being exposed.

Conclusion

This AWS-based architecture provides a secure, scalable, and reliable foundation for an e-commerce application. By leveraging managed cloud services and an event-driven design, the system ensures efficient order processing, high availability, and a smooth user experience.

References:

[1] Amazon Web Services, Inc., *AWS Well-Architected Framework*. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/framework/.

[2] Amazon Web Services, Inc., *Serverless Architectures on AWS*. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. Accessed: 2025.

[3] Amazon Web Services, Inc., *AWS Cloud Architecture Best Practices*. [Online]. Available: https://aws.amazon.com/architecture/. Accessed: 2025.

[4] YouTube AWS Services