ShopSmartlytoday – Production-Ready Architecture

This document translates the provided storefront HTML/CSS into a scalable, secure, and observable e-commerce platform. It is opinionated for AWS (fully portable to Azure/GCP).

---

1) High-Level Overview

Goal: Serve a global, performant e-commerce experience with reliable search, secure checkout, and operational excellence.

Architecture style: Modular microservices with an API Gateway + event-driven backbone.

Core tenets: - **Performance first:** CDN, edge caching, image optimization, aggressive read caching. - **Security by default:** Zero-trust networking, managed auth, WAF, secrets vault, least privilege. - **Observability:** End-to-end tracing, structured logs, SLOs, synthetic tests. - **Automation:** IaC, CI/CD, canary/blue-green, automated rollbacks.

---

2) Reference Diagram (Mermaid)

```
flowchart LR
  subgraph Edge
    DNS[Route53 DNS]
    CDN[CloudFront CDN]
    WAF[AWS WAF]
  end

  subgraph Web
    FE[Next.js/React SPA+SSR]
    ASSET[S3 Static Assets]
  end

  subgraph Platform
    APIGW[API Gateway]
    AUTH[Cognito / Keycloak]
    BFF[Node.js BFF]
  end

  subgraph Services
    CAT[Catalog Service]
    SRCH[Search Service (OpenSearch)]
```

```
  CART[Cart Service]
  ORD[Order Service]
  PAY[Payments Adapter]
  INV[Inventory Service]
  REC[Recommendations]
  UMS[User/Accounts]
  CMS[CMS (admin)]
end

subgraph Data
  PG[(PostgreSQL/Aurora)]
  REDIS[(ElastiCache/Redis)]
  OS[(OpenSearch)]
  OBJ[(S3 Product Media)]
  BUS[(Kafka/MSK)]
  WH[(Redshift/Snowflake)]
end

subgraph Observability
  LOGS[(CloudWatch/ELK)]
  MET[Prometheus/Grafana]
  TRACE[OpenTelemetry->Jaeger/X-Ray]
end

DNS-->CDN-->WAF-->FE
FE-->ASSET
FE-->APIGW
APIGW-->AUTH
APIGW-->BFF

BFF-->CAT
BFF-->SRCH
BFF-->CART
BFF-->ORD
BFF-->PAY
BFF-->INV
BFF-->UMS
BFF-->REC
BFF-->CMS

CAT-->PG
CART-->REDIS
ORD-->PG
INV-->PG
SRCH<-->OS
PAY-->|Stripe/Adyen|EXT[Payment PSP]
CAT-->OBJ
```

```
ORD--events-->BUS
INV--events-->BUS
PAY--events-->BUS
BUS-->WH

Services-->LOGS
Services-->MET
Services-->TRACE
```

---

3) Mapping UI → Services

- **Header/Search bar** → `Search Service` (autocomplete, typeahead, filters), `Catalog` for categories.
- **Category tiles** → `Catalog Service` (taxonomy), `CMS` for promotional copy/images.
- **Featured Deals/Recommended** → `CMS` + `Recommendations` (rules + ML). Edge cached.
- **Cart/Orders** → `Cart Service` (ephemeral Redis) + `Order Service` (Aurora, transactional).
- **Auth/Account** → `UMS` + `Cognito/Keycloak` (OIDC, social login).
- **Images** → Stored in `S3`, delivered with `CloudFront` and responsive image variants.

---

4) Services & Responsibilities

4.1 Backend-for-Frontend (BFF)

- Shapes data for the storefront, aggregates calls, enforces caching, rate limits, and A/B flags.
- **Tech:** Node.js (TypeScript), Fastify/Express + OpenAPI.

4.2 Catalog Service

- Product, category, attributes, pricing, promotions, inventory summary view.
- **DB:** Aurora Postgres. Write-through to OpenSearch for text/synonym search.

4.3 Search Service

- OpenSearch indexes: products, categories, suggestions.
- Features: synonym sets, analyzers, typo tolerance, popularity boosts, personalization hooks.

4.4 Cart Service

- Stateless API with carts in Redis (TTL 30 days). Supports anonymous carts (cookie) and merge on login.

4.5 Order Service

- Reserves inventory, commits payment intents, creates order records, issues events (`OrderCreated/Cancelled/Refunded`).
- **DB:** Aurora (ACID), outbox pattern → Kafka for reliability.

4.6 Payments Adapter

- Integrations: Stripe/Adyen/Braintree via tokenization. PCI SAQ-A maintained (no card data on servers).

4.7 Inventory Service

- Stock levels, reservations, backorders. Event-driven updates from WMS/ERP if present.

4.8 User/Account (UMS)

- Profiles, addresses, preferences, wishlists. PII encrypted at rest. RBAC for admin endpoints.

4.9 Recommendations

- Phase 1: Rule-based (best sellers, trending, similar category). Phase 2: ML with user-behavior features.

4.10 CMS (Headless)

- Manage home page banners, copy, SEO metadata, feature flags per locale.
- Options: Strapi/Sanity/Contentful.

---

5) Data Model (Sketch)

```
-- products
CREATE TABLE product (
  id UUID PRIMARY KEY,
  sku TEXT UNIQUE,
  title TEXT,
  description TEXT,
  price_cents INT,
  currency CHAR(3),
  category_id UUID,
  attributes JSONB,
  images JSONB,
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now()
);

-- orders
```

```
CREATE TABLE orders (
  id UUID PRIMARY KEY,
  user_id UUID NULL,
  status TEXT CHECK (status IN
('pending','paid','shipped','cancelled','refunded')),
  total_cents INT,
  currency CHAR(3),
  payment_intent_id TEXT,
  shipping_addr JSONB,
  items JSONB,
  created_at TIMESTAMPTZ DEFAULT now()
);
```

OpenSearch `products` index fields: `title`, `keywords`, `category`, `price`, `popularity`, `in_stock`, `attributes.*`.

---

6) API Surface (Selected)

- `GET /bff/home` → hero, categories, featured, recommended.
- `GET /search?q=&filters=&sort=&page=`
- `GET /catalog/products/{id}`
- `POST /cart/{cartId}/items` | `DELETE /cart/{cartId}/items/{itemId}`
- `POST /orders` → creates order (idempotency keys)
- `POST /payments/intents` → PSP client secret
- `GET /users/me` `PUT /users/me`

All APIs documented with **OpenAPI 3.1**, generated SDK for the frontend.

---

7) Frontend (Production)

- **Framework:** Next.js 14 (SSR/SSG + ISR) for SEO (maps perfectly to the static HTML you supplied).
- **Styling:** Tailwind + component library (Radix/HeadlessUI). Your existing CSS can be preserved in global.css.
- **State:** React Query for data caching; feature flags (Unleash/ConfigCat) for experiments.
- **i18n & locales:** next-intl; currencies with Dinero.js.
- **Images:** next/image with CloudFront loader; WebP/AVIF responsive.
- **Accessibility:** Lighthouse ≥ 95; semantic HTML, focus traps, color contrast.

---

8) Caching Strategy

- **Edge (CloudFront):**
- Cache HTML for SSG/ISR pages (TTL 60–300s) with `stale-while-revalidate`.

- Cache JSON API responses that are read-mostly: catalog/search (TTL 30–120s); bypass for personalized endpoints.
- Cache product images, CSS, JS with immutable hashes (1 year).
- **App Cache:** Redis for cart/session and computed fragments (e.g., home page modules).
- **Client:** HTTP caching + React Query staleTimes.

---

9) Security & Compliance

- **AuthN/Z:** OIDC (Cognito/Keycloak), JWT access tokens, short TTL refresh tokens, PKCE.
- **Network:** Private subnets for services; Public only for CDN, API Gateway; Service-to-service via mTLS.
- **AppSec:** WAF managed rules + custom IP reputation; rate limiting at API GW; OWASP ASVS checks in CI.
- **Data:** PII encryption (KMS), field-level encryption for addresses, secrets in AWS Secrets Manager.
- **Payments:** PCI SAQ-A (JS tokenization), no PANs processed or stored.
- **Privacy:** GDPR/CCPA controls (export/delete data APIs), consent banner, cookie partitioning.
- **Audit:** CloudTrail, immutable logs in S3 with Object Lock (WORM).

---

10) Observability & SRE

- **SLIs/SLOs:**
- Homepage p95 TTFB < 400ms
- Search API p95 < 200ms; error rate < 0.2%
- Checkout availability ≥ 99.95%
- **Tracing:** OpenTelemetry SDK in FE & BE → X-Ray/Jaeger.
- **Metrics:** RED/USE metrics per service; autoscaling alarms.
- **Logs:** Structured JSON; correlation IDs propagated.
- **Synthetic monitoring:** Canary tests (every 1–5 minutes) for search and checkout.
- **Runbooks:** For common incidents (cache stampede, PSP outage, index lag).

---

11) CI/CD & Environments

- **Branching:** trunk-based + short-lived PRs.
- **Pipelines (GitHub Actions/GitLab CI):**
- Lint/Type-check/Unit tests → SAST/Dependency scan → Build → SBOM → IaC plan → Ephemeral preview → Canary → Prod.
- **Deployments:**
- **Frontend:** Static export to S3 + CloudFront invalidations; SSR via AWS Lambda@Edge or Next on ECS/Fargate.
- **Services:** Containerized (ECS/Fargate or EKS). Blue/green with automatic rollback on error budget burn.
- **Infra as Code:** Terraform + Terragrunt, drift detection.
- **Secrets:** SOPS/Chamber, rotated automatically.

12) Data & Analytics

- **Event bus:** Kafka/MSK; outbox pattern from services.
- **Warehouse:** Redshift or Snowflake; ingestion via Kafka Connect/Firehose.
- **BI:** Looker/QuickSight dashboards: funnel, AOV, LTV, search zero-results, inventory turns.
- **ML:** Feature store (SageMaker Feature Store) powering recommendations and search boosts.

---

13) Scalability & Resilience

- Multi-AZ for Aurora/OpenSearch; cross-region read replicas for DR.
- RPO $\leq$ 5 minutes (binlog/Kafka), RTO $\leq$ 60 minutes with IaC + backups.
- Rate limiting and load shedding on BFF; circuit breakers for PSP/Search.
- Bulkheads: separate autoscaling groups for checkout vs. browse.

---

14) Cost Controls

- Rightsizing + autoscaling; Graviton instances.
- Tiered storage (S3 IA/Glacier for media versions).
- OpenSearch hot-warm tiers; index lifecycle policies.
- FinOps dashboards with cost anomaly detection.

---

15) Migration Path from Current HTML/CSS

1. **Adopt Next.js** and import your HTML/CSS into components/pages; wire header/search/category links to BFF endpoints.
2. **Introduce BFF** to abstract future microservices; initially proxy to a monolith if needed.
3. **Stand up Catalog + Search** with minimal schema; ETL seed data.
4. **Add Cart + Orders + Payments**; connect Stripe test keys; enable SAQ-A.
5. **Cut over images** to S3 + CloudFront; enable next/image responsive.
6. **Enable WAF + Auth**; migrate Sign-In/Sign-Up to Cognito hosted UI.
7. **Observability & SLOs**; add synthetic tests before launch.
8. **Turn on CI/CD** with preview envs; blue/green deploys.

---

16) Non-Functional Requirements (NFRs)

- **Perf:**
- LCP < 2.5s on 75th percentile 3G/low-end mobile
- CLS < 0.1; TBT < 200ms
- **Security:** No critical/open high CVEs in runtime images; monthly pen-test cadence.
- **Availability:** 99.95% service tier; 99.9% for search.

17) Example Terraform Modules (outline)

- `modules/network` VPC, subnets, NAT, SGs
- `modules/ecs_service` service + ALB + autoscaling
- `modules/rds_aurora` with snapshots/parameter groups
- `modules/opensearch` domains/ILM
- `modules/cloudfront_s3` static hosting + OAC + WAF association
- `modules/msk` Kafka cluster + topics + SCRAM

---

18) Risk Register (initial)

Risk	Impact	Likelihood	Mitigation
OpenSearch cost growth	High	Medium	Hot-warm tiers, shard sizing, ILM
PSP outage	High	Low	Retry w/ backoff, 3DS failover, capture later
Cache stampede	Medium	Medium	Request coalescing, stale-while-revalidate
PII exposure	High	Low	Data minimization, DLP scans, field encryption

---

19) Next Steps (Execution-ready Checklist)

- [ ] Create AWS accounts (prod/stage/dev), org SCPs
- [ ] Stand up VPC + shared services (ECR, MSK, Aurora)
- [ ] Bootstrap Terraform + CI pipelines
- [ ] Scaffold Next.js app and BFF with OpenAPI
- [ ] Implement Catalog + Search MVP
- [ ] Implement Cart + Orders + Payments (Stripe)
- [ ] Configure CloudFront/S3 + image pipeline
- [ ] Enable WAF, Cognito, Secrets Manager
- [ ] Add telemetry (OTel) + dashboards + SLOs
- [ ] Run load test, pen test, and chaos drills

Appendix A: Sample OpenAPI, IAM policies, and Helm/Compose manifests can be added upon request.