1. Features Stories Tasks

Soulstone Feature, Story & Task Catalog

Status: Final Version: v1.0 Last Updated: 2023-10-14

Overview

Launch Soulstone’s trusted commerce platform for ethically sourced crystals.

Enable wellness-conscious consumers in India to discover, learn about, and purchase ethically sourced crystals through a transparent, high-performing web and mobile experience reinforced by education, subscriptions, loyalty, and community partnerships.

Success Criteria

- MVP live within 3 months with web + mobile, 100+ SKUs, payments, and Learn Hub v1.
- Year■1 traction: ≥ 25,000 paying users; repeat purchase rate ≥ 30%.
- Conversion and checkout: sitewide conversion ≥ 3.5%; checkout success ≥ 65%; payment failures ≤ 2%.
- Performance and reliability: P75 page load < 2.0s (web); crash■free sessions > 99.5% (app).
- Unit economics: CAC ≤ ■500; LTV:CAC ≥ 8:1; AOV ≈ ■1,800.
- Quality and experience: returns ≤ 4% and defects < 2%; CSAT ≥ 90 with first■response time ≤ 2 minutes; partner■attributed revenue ≥ 15% and partner NPS ≥ 60 by Month 12.

Table of Contents

1. [Feature] - Engineering Foundations & Dev Environment

Description: Establish a monorepo, coding standards, and a reproducible local development environment to enable fast, consistent engineering workflows.

Acceptance Criteria:

- Monorepo structure created with apps/packages/infra as specified and workspace scripts working.
- Shared TypeScript config with strict mode and path aliases adopted by all apps.
- ESLint/Prettier configured with pre-commit hooks; commitlint and PR template in place.
- Docker Compose services (Postgres, Redis, Mailhog, etc.) start via one command; healthchecks green.
- Onboarding docs and Makefile/Taskfile allow new devs to bootstrap in < 15 minutes.

1.1. [User Story] - Monorepo and Tooling Setup

As a developer, I want a well-structured monorepo with shared tooling so that I can build, test, and collaborate consistently across apps and packages.

Acceptance Criteria:

- Workspace scripts run lint, type-check, test, and build across all packages from the root.
- Shared TypeScript, ESLint, and Prettier configs are enforced via pre-commit hooks.
- New developers can bootstrap the environment in under 15 minutes using documented scripts.

1.1.1. [Task] - Initialize monorepo (pnpm or yarn workspaces)

Description:

- Create structure: apps/api, apps/web, packages/ui, packages/config, packages/tsconfig, packages/eslint-config, infra/terraform, ops
- Configure workspace root package.json with scripts (build, dev, test, lint, format)
- Add .editorconfig, .nvmrc, .gitattributes, .gitignore

1.1.1.1. [Subtask] - Create structure: apps/api, apps/web, packages/ui, packages/config, packages/tsconfig, packages/eslint-config, infra/terraform, ops

Description:

- Apps/api
- Apps/web
- Packages/ui

1.1.1.2. [Subtask] - Configure workspace root package.json with scripts (build, dev, test, lint, format)

Description:

- Configure workspace root package.json with scripts (build
- Dev
- Test

1.1.1.3. [Subtask] - Add .editorconfig, .nvmrc, .gitattributes, .gitignore

Description:

- Add .editorconfig
- .nvmrc
- .gitattributes

1.1.2. [Task] - Configure TypeScript baselines

Description:

- Define shared tsconfig in packages/tsconfig and extend in apps
- Enable strict mode, path aliases, incremental builds

1.1.2.1. [Subtask] - Define shared tsconfig in packages/tsconfig and extend in apps

Description:

- Define shared tsconfig in packages/tsconfig
- Extend in apps

1.1.2.2. [Subtask] - Enable strict mode, path aliases, incremental builds

Description:

- Enable strict mode
- Path aliases
- Incremental builds

1.1.3. [Task] - Establish linting and formatting standards

Description:

- ESLint with TypeScript rules; Prettier with import/order
- Add lint-staged + Husky pre-commit hooks

1.1.3.1. [Subtask] - ESLint with TypeScript rules; Prettier with import/order

Description:

- ESLint with TypeScript rules
- Prettier with import/order

1.1.3.2. [Subtask] - Add lint-staged + Husky pre-commit hooks

Description:

- Add lint-staged + Husky pre-commit hooks

1.1.4. [Task] - Define commit and branch conventions

Description:

- Configure commitlint (Conventional Commits)
- Define branch protection and PR template

1.1.4.1. [Subtask] - Configure commitlint (Conventional Commits)

Description:

- Configure commitlint (Conventional Commits)

1.1.4.2. [Subtask] - Define branch protection and PR template

Description:

- Define branch protection
- PR template

1.2. [User Story] - Local Dev Environment

As a developer, I want a one-command local environment so that I can run all dependent services reliably and start coding quickly.

Acceptance Criteria:

- Docker Compose brings up Postgres, Redis, Mailhog, and utilities with healthy checks.

- Environment variables are managed via .env files and validated at startup.
- Bootstrap scripts handle install, migrations, seeds, and starting dev servers.

1.2.1. [Task] - Provision Docker Compose services

Description:

- postgres (15), redis, mailhog, localstack (optional)
- Seed volumes and healthchecks; named networks

1.2.1.1. [Subtask] - postgres (15), redis, mailhog, localstack (optional)

Description:

- Postgres (15)
- Redis
- Mailhog

1.2.1.2. [Subtask] - Seed volumes and healthchecks; named networks

Description:

- Seed volumes and healthchecks
- Named networks

1.2.2. [Task] - Implement environment management

Description:

- .env.example with required vars; dotenv-safe loading
- Split envs: .env.dev, .env.test, .env.local

1.2.2.1. [Subtask] - .env.example with required vars; dotenv-safe loading

Description:

- .env.example with required vars
- Dotenv-safe loading

1.2.2.2. [Subtask] - Split envs: .env.dev, .env.test, .env.local

Description:

- .env.dev
- .env.test
- .env.local

1.2.3. [Task] - Create developer bootstrap scripts

Description:

- Makefile/Taskfile for install, db:up, db:migrate, db:seed, dev
- Onboarding doc covering prerequisites and commands

1.2.3.1. [Subtask] - Makefile/Taskfile for install, db:up, db:migrate, db:seed, dev

Description:

- Up
- Db:migrate
- Db:seed

1.2.3.2. [Subtask] - Onboarding doc covering prerequisites and commands

Description:

- Onboarding doc covering prerequisites

- Commands

2. [Feature] - Infrastructure as Code (AWS)

Description: Provision secure, scalable AWS foundations using Terraform for networking, compute, data, secrets, and observability.

Acceptance Criteria:

- VPC with public/private subnets, NAT, least-privilege security groups, and SSM access provisioned.
- Route53 + ACM for domains/certs; CloudFront + WAF fronting the web workload.
- ECR repos and ECS Fargate services for API and web with passing health checks.
- RDS Postgres Multi-AZ with encryption, backups, parameter groups; Redis cluster provisioned.
- Secrets structure in Secrets Manager/SSM; CloudWatch log retention and alarms wired to Slack/webhook.

2.1. [User Story] - Core Networking & Security

As a platform engineer, I want secure networking foundations so that services are isolated, reachable, and compliant.

Acceptance Criteria:

- VPC with public/private subnets and NAT is provisioned via Terraform.
- Security groups follow least privilege; access via SSM Session Manager.
- DNS/CDN (Route53, ACM, CloudFront, WAF) configured for public workloads.

2.1.1. [Task] - Provision VPC with public/private subnets and NAT

Description:

- Terraform module for VPC, subnet, route tables
- Security groups least privilege; SSM Session Manager access

2.1.1.1. [Subtask] - Terraform module for VPC, subnet, route tables

Description:

- Terraform module for VPC
- Subnet
- Route tables

2.1.1.2. [Subtask] - Security groups least privilege; SSM Session Manager access

Description:

- Security groups least privilege
- SSM Session Manager access

2.1.2. [Task] - Configure DNS and CDN

Description:

- Route 53 hosted zones; ACM certs
- CloudFront for web with S3 origin and WAF

2.1.2.1. [Subtask] - Route 53 hosted zones; ACM certs

Description:

- Route 53 hosted zones
- ACM certs

2.1.2.2. [Subtask] - CloudFront for web with S3 origin and WAF

Description:

- CloudFront for web with S3 origin
- WAF

2.2. [User Story] - Compute & Containers

As a platform engineer, I want containerized compute with health-checked deployments so that API and web run reliably and scale predictably.

Acceptance Criteria:

- ECR repos created with immutability and scanning; lifecycle policies applied.
- ECS Fargate services for API/web deploy with passing health/readiness checks.
- SSR vs SSG deployment path is chosen and documented.

2.2.1. [Task] - Create ECR repositories and lifecycle policies

Description:

- Private ECR with immutability and tag scanning

2.2.1.1. [Subtask] - Private ECR with immutability and tag scanning

Description:

- Private ECR with immutability
- Tag scanning

2.2.2. [Task] - Provision ECS Fargate services

Description:

- API service behind ALB; health/readiness checks

- Web SSR service (if Next SSR) or static S3 hosting

2.2.2.1. [Subtask] - API service behind ALB; health/readiness checks

Description:

- API service behind ALB
- Health/readiness checks

2.2.2.2. [Subtask] - Web SSR service (if Next SSR) or static S3 hosting

Description:

- Web SSR service (if Next SSR) or static S3 hosting

2.3. [User Story] - Data Stores

As a platform engineer, I want managed data stores configured securely so that application data is durable, performant, and recoverable.

Acceptance Criteria:

- RDS Postgres Multi-AZ with encryption, backups, and tuned parameters is provisioned.
- Redis cluster for sessions/cache is available with appropriate limits and alerts.
- Monitoring and alerts exist for connections, storage, replication, and failures.

2.3.1. [Task] - Provision RDS PostgreSQL (Multi-AZ) with security

Description:

- Parameter groups, backups, encryption at rest
- Read replica plan; connection limits and alerts

2.3.1.1. [Subtask] - Parameter groups, backups, encryption at rest

Description:

- Parameter groups
- Backups
- Encryption at rest

2.3.1.2. [Subtask] - Read replica plan; connection limits and alerts

Description:

- Read replica plan
- Connection limits and alerts

2.3.2. [Task] - Provision ElastiCache Redis for sessions/cache

Description:

- Provision ElastiCache Redis for sessions/cache

2.4. [User Story] - Secrets & Keys

As a security engineer, I want centralized secret management so that credentials are rotated, scoped by environment, and never hard-coded.

Acceptance Criteria:

- Secrets are organized by environment in Secrets Manager/SSM Parameter Store.
- Rotation policies and IAM boundaries are applied; access is logged and auditable.
- Applications read secrets at runtime without storing plaintext in code or images.

2.4.1. [Task] - Define Secrets Manager/SSM Parameter Store structure

Description:

- Namespace by env (dev/stage/prod)
- Rotation policy and IAM access boundaries

2.4.1.1. [Subtask] - Namespace by env (dev/stage/prod)

Description:

- Namespace by env (dev/stage/prod)

2.4.1.2. [Subtask] - Rotation policy and IAM access boundaries

Description:

- Rotation policy
- IAM access boundaries

2.5. [User Story] - Observability Base

As an SRE, I want baseline logs, metrics, and alerts so that issues are detectable and actionable from day one.

Acceptance Criteria:

- CloudWatch log groups have retention set; error/latency metric filters exist.
- Key alarms route to Slack/webhook with actionable thresholds and runbooks.
- Health and metrics endpoints are instrumented and visible in dashboards.

2.5.1. [Task] - Create CloudWatch log groups and metrics

Description:

- Log retention policies; metric filters for errors/latency

2.5.1.1. [Subtask] - Log retention policies; metric filters for errors/latency

Description:

- Log retention policies
- Metric filters for errors/latency

2.5.2. [Task] - Configure alarms and notifications

Description:

- SNS + Slack/webhook integration for alerts

2.5.2.1. [Subtask] - SNS + Slack/webhook integration for alerts

Description:

- SNS + Slack/webhook integration for alerts

3. [Feature] - CI/CD Pipelines

Description: Automate build, test, security scanning, artifact creation, and deployments with preview environments.

Acceptance Criteria:

- GitHub Actions run lint, type-check, unit tests, and build with caching on every PR.
- Dependency, image, and secret scans gate merges; SBOMs produced for images.
- Docker images are non-root, multi-stage, with healthchecks and graceful shutdown.
- API deploys via blue/green with auto-rollback; web deploys with invalidations; DB migrations gated.
- Per-PR preview environments spin up and tear down automatically.

3.1. [User Story] - Continuous Integration

As a developer, I want consistent CI checks so that code quality and builds are validated on every change.

Acceptance Criteria:

- Lint, type-check, unit tests, and build jobs run in GitHub Actions with caching.
- Security scans (dependencies, images, secrets) gate merges.
- PR status reflects all checks; failures block merges until fixed.

3.1.1. [Task] - Implement GitHub Actions workflows

Description:

- Jobs: lint, type-check, unit tests, build
- Cache node modules and build artifacts

3.1.1.1. [Subtask] - Jobs: lint, type-check, unit tests, build

Description:

- Lint
- Type-check
- Unit tests

3.1.1.2. [Subtask] - Cache node modules and build artifacts

Description:

- Cache node modules
- Build artifacts

3.1.2. [Task] - Integrate security scanning in CI

Description:

- eslint-plugin-security, npm audit, trivy image scan
- Secret scanning (gitleaks)

3.1.2.1. [Subtask] - eslint-plugin-security, npm audit, trivy image scan

Description:

- Eslint-plugin-security
- Npm audit
- Trivy image scan

3.1.2.2. [Subtask] - Secret scanning (gitleaks)

Description:

- Secret scanning (gitleaks)

3.2. [User Story] - Build & Release Artifacts

As a release engineer, I want reproducible images and artifacts so that deployments are consistent and traceable.

Acceptance Criteria:

- Multi-stage Dockerfiles produce minimal non-root images with healthcheck and SIGTERM handling.
- Images are tagged with SHA/semver; SBOMs are generated and stored.
- Versioning and changelogs follow semantic versioning.

3.2.1. [Task] - Create multi-stage Dockerfiles for API and Web

Description:

- Non-root user, minimal base (alpine or distroless)
- Healthcheck and SIGTERM handling

3.2.1.1. [Subtask] - Non-root user, minimal base (alpine or distroless)

Description:

- Non-root user
- Minimal base (alpine or distroless)

3.2.1.2. [Subtask] - Healthcheck and SIGTERM handling

Description:

- Healthcheck
- SIGTERM handling

3.2.2. [Task] - Implement image versioning and SBOM generation

Description:

- Tagging with sha/semver; syft/grype SBOM

3.2.2.1. [Subtask] - Tagging with sha/semver; syft/grype SBOM

Description:

- Tagging with sha/semver
- Syft/grype SBOM

3.3. [User Story] - Continuous Delivery

As a release manager, I want automated, safe deployments so that releases roll out quickly with rollback safety.

Acceptance Criteria:

- API deploys via blue/green with auto-rollback on health check failure.
- Web deploys via S3+CloudFront invalidations (SSG) or ECS (SSR) as appropriate.
- Per-PR preview environments spin up and shut down automatically; DB migrations are gated.

3.3.1. [Task] - Deploy API to ECS with blue/green

Description:

- Auto rollback on health check failure

3.3.1.1. [Subtask] - Auto rollback on health check failure

Description:

- Auto rollback on health check failure

3.3.2. [Task] - Deploy Web

Description:

- S3+CloudFront invalidations (SSG) or ECS (SSR)

3.3.2.1. [Subtask] - S3+CloudFront invalidations (SSG) or ECS (SSR)

Description:

- S3+CloudFront invalidations (SSG) or ECS (SSR)

3.3.3. [Task] - Gate database migrations in deploys

Description:

- Gate deploy on successful migrate; rollback plan

3.3.3.1. [Subtask] - Gate deploy on successful migrate; rollback plan

Description:

- Gate deploy on successful migrate
- Rollback plan

3.3.4. [Task] - Provision preview environments per PR

Description:

- Ephemeral API + web URLs; tear-down job

3.3.4.1. [Subtask] - Ephemeral API + web URLs; tear-down job

Description:

- Ephemeral API + web URLs
- Tear-down job

4. [Feature] - Security & Compliance

Description: Implement application security controls, privacy rights, fraud protection, and pen-test remediation workflows.

Acceptance Criteria:

- OWASP controls enforced (validation, headers, CORS, CSRF where relevant).

- IP/user rate limits and circuit breakers protect critical routes.
- Consent management and privacy rights portal operational (export/delete ≤ 7 days SLA).
- Pen-test triage, remediation, verification, and reporting tracked to closure.
- Pre/post-payment risk scoring with actions (hold/cancel/manual review) enabled.

4.1. [User Story] - AppSec Baseline

As a security engineer, I want application security controls enforced so that common vulnerabilities are prevented and threats are mitigated early.

Acceptance Criteria:

- Input/output validation is enforced (Zod); queries are parameterized to prevent injection.
- Security headers (Helmet/CSP, HSTS) and a strict CORS allowlist are configured.
- Rate limiting and circuit breakers are applied to sensitive routes with safe errors.

4.1.1. [Task] - Enforce OWASP Top 10 controls

Description:

- Input/output validation (Zod) and parameterized queries
- Helmet/CSP, HSTS, CORS allowlist

4.1.1.1. [Subtask] - Input/output validation (Zod) and parameterized queries

Description:

- Input/output validation (Zod)
- Parameterized queries

4.1.1.2. [Subtask] - Helmet/CSP, HSTS, CORS allowlist

Description:

- Helmet/CSP
- HSTS
- CORS allowlist

4.1.2. [Task] - Implement rate limiting and abuse prevention

Description:

- Per-IP and per-token quotas; circuit breaker

4.1.2.1. [Subtask] - Per-IP and per-token quotas; circuit breaker

Description:

- Per-IP and per-token quotas
- Circuit breaker

4.2. [User Story] - Privacy & DPDP/GDPR

As a privacy officer, I want consent and data rights tooling so that we comply with DPDP/GDPR and respect user choices.

Acceptance Criteria:

- Consent is captured and enforced; SDKs and pixels are gated by preferences.
- Data rights portal supports export/delete within 7 days with audit logs.
- Preference changes propagate across channels and are honored in downstream systems.

4.2.1. [Task] - Implement consent management and preferences store

Description:

- Cookie banner and SDK gating

4.2.1.1. [Subtask] - Cookie banner and SDK gating

Description:

- Cookie banner
- SDK gating

4.2.2. [Task] - Build data rights portal

Description:

- Export/delete with audit log; SLA ≤ 7 days

4.2.2.1. [Subtask] - Export/delete with audit log; SLA ≤ 7 days

Description:

- Export/delete with audit log
- SLA ≤ 7 days

4.3. [User Story] - Penetration Testing & Remediation

As a security lead, I want a repeatable pen-test remediation workflow so that findings are triaged, fixed, and verified within SLA.

Acceptance Criteria:

- Findings are classified by severity with owners and documented SLAs.
- Remediations are verified with re-tests and CI regression coverage.
- Reports, approvals, and changes are tracked with an auditable trail.

4.3.1. [Task] - Establish triage and severity classification

Description:

- SLAs by severity; owner assignment and tracking

4.3.1.1. [Subtask] - SLAs by severity; owner assignment and tracking

Description:

- SLAs by severity
- Owner assignment and tracking

4.3.2. [Task] - Execute remediation and verification

Description:

- Re-test and regression tests wired into CI

4.3.2.1. [Subtask] - Re-test and regression tests wired into CI

Description:

- Re-test
- Regression tests wired into CI

4.3.3. [Task] - Implement reporting and change management

Description:

- Approvals, audit trail, and post-mortems

4.3.3.1. [Subtask] - Approvals, audit trail, and post-mortems

Description:

- Approvals
- Audit trail

- Post-mortems

4.4. [User Story] - Fraud & Risk Scoring Hooks

As a risk analyst, I want risk scoring hooks so that suspicious orders trigger review, holds, or cancellations before loss occurs.

Acceptance Criteria:

- Pre/post-payment risk evaluation uses device/IP/email heuristics and velocity.
- A manual review queue exists; actions include auto-hold/cancel with audit logs.
- Decisions and outcomes are recorded with reasons for reporting and tuning.

4.4.1. [Task] - Implement pre/post-payment risk evaluation

Description:

- Signals: device/IP/email heuristics, velocity, coupon abuse

4.4.1.1. [Subtask] - Signals: device/IP/email heuristics, velocity, coupon abuse

Description:

- Device/IP/email heuristics
- Velocity
- Coupon abuse

4.4.2. [Task] - Configure actions and workflows

Description:

- Manual review queue; auto-cancel/hold rules; audit logs

4.4.2.1. [Subtask] - Manual review queue; auto-cancel/hold rules; audit logs

Description:

- Manual review queue
- Auto-cancel/hold rules
- Audit logs

5. [Feature] - Database & Data Modeling (PostgreSQL + Prisma)

Description: Define robust relational models and performance patterns for core entities with repeatable migrations.

Acceptance Criteria:

- Prisma schema covers core domain models with keys, constraints, and enums.
- Migrations apply cleanly across dev/stage/prod; rollback procedure documented.
- Indexes exist for search/facets and hot queries; partial indexes where appropriate.
- Seed data and anonymized fixtures available for dev/test.
- Automated backups configured; quarterly restore drill documented.

5.1. [User Story] - Schema Foundation

As a backend engineer, I want robust relational models so that data integrity and performance are ensured across core entities.

Acceptance Criteria:

- Prisma schema defines core models with enums, keys, and constraints.
- Migrations apply cleanly across environments; FKs, cascades, and unique constraints enforced.
- Indexes cover search/facets and hot queries to meet performance targets.

5.1.1. [Task] - Define core models (User, Address, Product, Collection, Inventory, Cart, CartItem, Order, Payment, Review, Subscription, Article, Media)

Description:

- Prisma schema and enums; migrations
- Foreign keys, cascades, unique constraints

5.1.1.1. [Subtask] - Prisma schema and enums; migrations

Description:

- Prisma schema and enums
- Migrations

5.1.1.2. [Subtask] - Foreign keys, cascades, unique constraints

Description:

- Foreign keys
- Cascades
- Unique constraints

5.1.2. [Task] - Implement indexing for performance

Description:

- B-tree/GiST indexes for search/facets; partial indexes

5.1.2.1. [Subtask] - B-tree/GiST indexes for search/facets; partial indexes

Description:

- B-tree/GiST indexes for search/facets

- Partial indexes

5.2. [User Story] - Data Lifecycle

As a data engineer, I want seed data, backups, and restore procedures so that environments are consistent and recoverable.

Acceptance Criteria:

- Dev/test fixtures and anonymized sample content exist and are easy to load.
- Automated snapshots are configured; quarterly restore drill is documented and tested.
- Backup retention meets policy with monitoring for failures.

5.2.1. [Task] - Create seed data for dev/test

Description:

- Factories and fixtures; anonymized sample content

5.2.1.1. [Subtask] - Factories and fixtures; anonymized sample content

Description:

- Factories and fixtures
- Anonymized sample content

5.2.2. [Task] - Configure backups and run restore drills

Description:

- Automated snapshots; quarterly restore test

5.2.2.1. [Subtask] - Automated snapshots; quarterly restore test

Description:

- Automated snapshots
- Quarterly restore test

6. [Feature] - API Platform (Express/Node, TypeScript)

Description: Provide a modular, observable REST API with validation, errors, security, and auth primitives.

Acceptance Criteria:

- Express app structured by routes/controllers/services/repos with request ID logging and health/metrics endpoints.
- Standardized error shape and codes; DTO validation/sanitization via Zod.
- Auth endpoints support JWT access/refresh rotation, email verification, and password reset.
- Route-level rate limits enforced via Redis token bucket.
- Metrics and structured logs available in CloudWatch/Sentry.

6.1. [User Story] - API Server Scaffolding

As a backend engineer, I want a modular API with logging and health endpoints so that services are maintainable and observable.

Acceptance Criteria:

- Express app is structured by routes/controllers/services/repos with pino logging.
- Request IDs are correlated; health, readiness, and metrics endpoints are exposed.
- Basic metrics are visible in CloudWatch/Sentry dashboards.

6.1.1. [Task] - Implement Express app with modular structure (routes/controllers/services/repos)

Description:

- Request ID correlation and structured logging (pino)
- Health, readiness, and metrics endpoints

6.1.1.1. [Subtask] - Request ID correlation and structured logging (pino)

Description:

- Request ID correlation
- Structured logging (pino)

6.1.1.2. [Subtask] - Health, readiness, and metrics endpoints

Description:

- Health
- Readiness
- Metrics endpoints

6.2. [User Story] - Error Handling & Validation

As a developer, I want standardized error handling and input validation so that failures are predictable and safe.

Acceptance Criteria:

- Global error handler returns a standardized error shape and codes.
- DTO validation and sanitization via Zod; inputs normalized and safe.
- Errors include correlation IDs; 4xx/5xx split is consistent and logged.

6.2.1. [Task] - Implement global error handler and error shape

Description:

- Map to codes: AUTH_FAILED, FORBIDDEN, NOT_FOUND, VALIDATION_ERROR, RATE_LIMITED

6.2.1.1. [Subtask] - Map to codes: AUTH_FAILED, FORBIDDEN, NOT_FOUND, VALIDATION_ERROR, RATE_LIMITED

Description:

- AUTH_FAILED
- FORBIDDEN
- NOT_FOUND

6.2.2. [Task] - Add DTO validation

Description:

- Zod schemas; sanitize and normalize inputs

6.2.2.1. [Subtask] - Zod schemas; sanitize and normalize inputs

Description:

- Zod schemas
- Sanitize and normalize inputs

6.3. [User Story] - Auth Endpoints

As a customer, I want to sign up, sign in, and manage my session so that I can securely access my account.

Acceptance Criteria:

- JWT access/refresh with rotation and a revocation store (Redis) is implemented.

- Email verification and password reset flows work end-to-end with tokens.
- Refresh and logout endpoints invalidate tokens correctly and log events.

6.3.1. [Task] - Implement auth endpoints: /auth/signup, /auth/login, /auth/refresh, /auth/logout

Description:

- JWT (RS256) access/refresh; rotation and revocation store (Redis)
- Email verification, password reset tokens

6.3.1.1. [Subtask] - JWT (RS256) access/refresh; rotation and revocation store (Redis)

Description:

- JWT (RS256) access/refresh
- Rotation and revocation store (Redis)

6.3.1.2. [Subtask] - Email verification, password reset tokens

Description:

- Email verification
- Password reset tokens

6.4. [User Story] - Rate Limiting & Security

As a security engineer, I want IP/user throttles so that abuse is contained without hurting legitimate traffic.

Acceptance Criteria:

- Redis-backed token bucket enforces per-route and per-identity limits.
- Configurable quotas return safe errors and expose metrics for tuning.

- Limit events are logged and observable to detect abuse patterns.

6.4.1. [Task] - Implement IP and user-based throttles

Description:

- Redis-backed token bucket; per-route configs

6.4.1.1. [Subtask] - Redis-backed token bucket; per-route configs

Description:

- Redis-backed token bucket
- Per-route configs

7. [Feature] - Authentication & Accounts

Description: Manage user identities, profiles, addresses, and active sessions with security controls.

Acceptance Criteria:

- Profile CRUD and Indian address validation with default address selection.
- Session/device listing and revocation available from account settings.
- Data model ensures uniqueness and referential integrity for users/addresses.
- API permissions enforced for protected profile endpoints.

7.1. [User Story] - User Profile & Address Book

As a customer, I want to manage my profile and addresses so that checkout is fast and accurate.

Acceptance Criteria:

- Profile and address CRUD works with Indian address validation and default selection.
- Default address is persisted and used at checkout; changes reflect immediately.
- Authorization is enforced; users can only view/edit their own data.

7.1.1. [Task] - Implement profile and address CRUD

Description:

- Indian address validation; default selection

7.1.1.1. [Subtask] - Indian address validation; default selection

Description:

- Indian address validation
- Default selection

7.1.2. [Task] - Implement session and device management

Description:

- List/revoke active sessions

7.1.2.1. [Subtask] - List/revoke active sessions

Description:

- List/revoke active sessions

7.2. [User Story] - Account Lifecycle (Delete/Close)

As a customer, I want to delete my account and export my data so that I control my information.

Acceptance Criteria:

- Deletion/export requests require identity verification and are queued with SLA.
- Soft-delete window and purge job with audit logging are implemented.
- Users receive status updates and confirmations for each step.

7.2.1. [Task] - Implement account deletion and data export requests

Description:

- Identity verification; queue with SLA
- Soft-delete window; purge job and audit log

7.2.1.1. [Subtask] - Identity verification; queue with SLA

Description:

- Identity verification
- Queue with SLA

7.2.1.2. [Subtask] - Soft-delete window; purge job and audit log

Description:

- Soft-delete window
- Purge job and audit log

7.3. [User Story] - Contact Changes & Reverification

As a customer, I want to change my email or phone with re-verification so that my account stays secure.

Acceptance Criteria:

- New contact is verified before switch; login identifiers migrate safely.

- Old contact is notified; a recovery lockout window is enforced.
- All attempts and outcomes are audited for security.

7.3.1. [Task] - Implement contact change with re-verification

Description:

- Verify new contact; migrate login identifiers
- Notify old contact; recovery lockout window

7.3.1.1. [Subtask] - Verify new contact; migrate login identifiers

Description:

- Verify new contact
- Migrate login identifiers

7.3.1.2. [Subtask] - Notify old contact; recovery lockout window

Description:

- Notify old contact
- Recovery lockout window

7.4. [User Story] - Anti-automation on Auth

As a security engineer, I want anti-automation on auth endpoints so that bots and attackers are throttled without harming good users.

Acceptance Criteria:

- CAPTCHA or risk checks protect signup/login/forgot; global rate limits applied.
- Device/IP heuristic scoring drives graduated challenges and throttles.

- Accessible fallbacks exist for legitimate users who fail challenges.

7.4.1. [Task] - Add CAPTCHA and risk checks to auth endpoints

Description:

- Apply to signup/login/forgot; global rate limits (see 6.4)
- Device/IP heuristic scoring and throttles

7.4.1.1. [Subtask] - Apply to signup/login/forgot; global rate limits (see 6.4)

Description:

- Apply to signup/login/forgot
- Global rate limits (see 6.4)

7.4.1.2. [Subtask] - Device/IP heuristic scoring and throttles

Description:

- Device/IP heuristic scoring
- Throttles

8. [Feature] - Catalog & Search

Description: Enable rich product discovery with facets, search, and informative PDPs.

Acceptance Criteria:

- PLP supports facets, sorting, and pagination with empty-state handling.
- PDP shows images, price, origin, authenticity, energy tags; related products present.

- Search includes typeahead, synonyms, fuzzy match, and recent searches.
- JSON-LD structured data present for PDP and breadcrumbs.

8.1. [User Story] - Catalog Browse (PLP)

As a shopper, I want to browse products with filters and sorting so that I can find relevant items quickly.

Acceptance Criteria:

- PLP supports facets, sorting, and pagination with empty-state handling.
- Performance meets budgets; analytics events are tracked for browse actions.
- Zero-result state suggests alternative intents or searches.

8.1.1. [Task] - Implement products list with facets (type, intention, chakra, price, size, origin, rating)

Description:

- Cursor/offset pagination; sort options
- Zero-result handling with suggested intents

8.1.1.1. [Subtask] - Cursor/offset pagination; sort options

Description:

- Cursor/offset pagination
- Sort options

8.1.1.2. [Subtask] - Zero-result handling with suggested intents

Description:

- Zero-result handling with suggested intents

8.2. [User Story] - Product Detail (PDP)

As a shopper, I want detailed product information so that I can buy confidently.

Acceptance Criteria:

- PDP shows images, price, origin, authenticity, and energy tags.
- Related products and structured data (JSON-LD) are present.
- Stock and variants are selectable and reflected accurately.

8.2.1. [Task] - Implement PDP data display: images, price, origin, authenticity, energy tags

Description:

- Related products by intent/type; JSON-LD

8.2.1.1. [Subtask] - Related products by intent/type; JSON-LD

Description:

- Related products by intent/type
- JSON-LD

8.3. [User Story] - Search Service

As a shopper, I want fast, relevant search so that I can find products by name, intent, or tags.

Acceptance Criteria:

- Typeahead and full-text search support synonyms and fuzzy matching.
- Recent searches are stored with a clear option for the user.
- Results paginate and meet performance targets.

8.3.1. [Task] - Implement typeahead and full-text search

Description:

- Synonyms (e.g., Amethyst <-> Jamunia); fuzzy match
- Recent searches and clear history

8.3.1.1. [Subtask] - Synonyms (e.g., Amethyst <-> Jamunia); fuzzy match

Description:

- Synonyms (e.g., Amethyst <-> Jamunia)
- Fuzzy match

8.3.1.2. [Subtask] - Recent searches and clear history

Description:

- Recent searches
- Clear history

8.4. [User Story] - Variants & Options

As a shopper, I want to select variants and options so that I purchase the exact SKU I need.

Acceptance Criteria:

- Variant modeling supports option swatches and canonical SKU selection.
- Selected options update price and availability consistently across the UI.
- Invalid combinations are disabled or clearly indicated.

8.4.1. [Task] - Implement variant modeling and selection UI

Description:

- Option swatches; canonical SKU selection

8.4.1.1. [Subtask] - Option swatches; canonical SKU selection

Description:

- Option swatches
- Canonical SKU selection

8.5. [User Story] - Indexing Pipeline

As a shopper, I want new and updated products to appear in search quickly so that catalog changes reflect promptly.

Acceptance Criteria:

- Incremental indexing runs on publish; background reindex available.
- Partial updates are supported; failures are retried with logging.
- Index freshness is monitored with alerts.

8.5.1. [Task] - Implement incremental indexing and reindex jobs

Description:

- Publish hooks; background reindex; partial updates

8.5.1.1. [Subtask] - Publish hooks; background reindex; partial updates

Description:

- Publish hooks
- Background reindex
- Partial updates

8.6. [User Story] - Search Analytics & Curation

As a merchandiser, I want zero-result analytics and synonym curation so that search outcomes improve over time.

Acceptance Criteria:

- Zero-result queries are tracked; synonyms/redirects can be curated.
- Promote/demote rules are configurable and auditable.
- Impact of curation is measured with analytics.

8.6.1. [Task] - Implement zero-result analytics and synonyms curation

Description:

- Curate synonyms/redirects; promote/demote rules

8.6.1.1. [Subtask] - Curate synonyms/redirects; promote/demote rules

Description:

- Curate synonyms/redirects
- Promote/demote rules

9. [Feature] - Cart & Pricing

Description: Provide accurate carts with discounts, tax, shipping, and guest/server persistence.

Acceptance Criteria:

- Add/update/remove items recalculates subtotal, tax, shipping, and discounts.

- Server cart persists for auth users; guest cart persists locally.
- Merge-on-login resolves conflicts and recalculates totals deterministically.
- Idempotent cart mutations and consistent pricing across sessions.

9.1. [User Story] - Cart Service

As a shopper, I want my cart to reflect items and totals accurately so that I always know what I'll pay.

Acceptance Criteria:

- Add/update/remove recalculates subtotal, tax, shipping, and discounts deterministically.
- Server cart persists for auth users; guest cart persists locally; merge-on-login resolves conflicts.
- Cart mutations are idempotent and pricing is consistent across sessions.

9.1.1. [Task] - Implement add/update/remove items and totals (subtotal, tax, shipping, discounts)

Description:

- Server cart persistence; guest cart local persistence

9.1.1.1. [Subtask] - Server cart persistence; guest cart local persistence

Description:

- Server cart persistence
- Guest cart local persistence

9.1.2. [Task] - Implement merge-on-login

Description:

- Conflict resolution and price recalc

9.1.2.1. [Subtask] - Conflict resolution and price recalc

Description:

- Conflict resolution
- Price recalc

9.2. [User Story] - Inventory Reservations

As a shopper, I want items reserved briefly when added to cart so that availability doesn’t vanish unexpectedly at checkout.

Acceptance Criteria:

- Reservation occurs on add-to-cart with a clear TTL and release on abandon.
- Oversell guardrails and alerts prevent negative stock situations.
- Reservations are reflected in availability checks during checkout.

9.2.1. [Task] - Implement reservation on add-to-cart

Description:

- TTL and release on abandon
- Oversell guardrails and alerts

9.2.1.1. [Subtask] - TTL and release on abandon

Description:

- TTL
- Release on abandon

9.2.1.2. [Subtask] - Oversell guardrails and alerts

Description:

- Oversell guardrails
- Alerts

9.3. [User Story] - Estimators & Gifting

As a shopper, I want pre-login shipping/tax estimates and gifting options so that I can decide before checking out.

Acceptance Criteria:

- PIN estimator shows shipping/tax estimates prior to login.
- Gift wrap and message options include fees and a preview.
- Packing slip/receipt handles gift messaging correctly.

9.3.1. [Task] - Implement shipping/tax estimate pre-login

Description:

- Pin code estimator on cart

9.3.1.1. [Subtask] - Pin code estimator on cart

Description:

- Pin code estimator on cart

9.3.2. [Task] - Implement gift wrap and messages

Description:

- Fees, preview, and packing slip handling

9.3.2.1. [Subtask] - Fees, preview, and packing slip handling

Description:

- Fees
- Preview
- Packing slip handling

10. [Feature] - Checkout & Payments (Razorpay/Stripe)

Description: Deliver a secure, reliable checkout with address/shipping, payment intents, and robust webhooks.

Acceptance Criteria:

- Address capture validates PIN/phone; shipping methods selectable and priced.
- Payment intents created with idempotency; success/failure states handled.
- Webhook signatures verified; retries with DLQ; orders reflect payment state.
- PCI scope limited (no PAN storage); sensitive data never logged.
- Happy-path checkout completes under target latency; failure paths recover gracefully.

10.1. [User Story] - Address & Shipping

As a shopper, I want to enter my address and choose shipping so that delivery is accurate and priced clearly.

Acceptance Criteria:

- Address capture validates PIN/phone with clear, localized errors.
- Shipping methods are selectable with prices and delivery windows.
- Reused address validation engine; data persists to profile where applicable.

10.1.1. [Task] - Implement address capture and validation; shipping methods

Description:

- Pin code and phone validation
- Reuse address validation engine (see 7.1.1.1)

10.1.1.1. [Subtask] - Pin code and phone validation

Description:

- Pin code
- Phone validation

10.1.1.2. [Subtask] - Reuse address validation engine (see 7.1.1.1)

Description:

- Reuse address validation engine (see 7.1.1.1)

10.2. [User Story] - Payment Intent & Confirmation

As a shopper, I want secure payment flows so that I can complete my order reliably.

Acceptance Criteria:

- Payment intent is created with idempotency; success/failure states are handled.
- Webhook signatures are verified; retries with DLQ keep orders in sync.
- Clear error states and recovery paths are presented to users.

10.2.1. [Task] - Create payment intent; handle success/failure

Description:

- Idempotency keys on writes

10.2.1.1. [Subtask] - Idempotency keys on writes

Description:

- Idempotency keys on writes

10.2.2. [Task] - Implement payment webhooks

Description:

- Signature validation; retries and DLQ

10.2.2.1. [Subtask] - Signature validation; retries and DLQ

Description:

- Signature validation
- Retries and DLQ

10.3. [User Story] - Regulatory & Risk Controls

As a compliance officer, I want regulatory controls in checkout so that we meet legal and risk requirements.

Acceptance Criteria:

- 3DS/SCA challenges function; terms/return policy acceptance is captured and logged.
- COD eligibility is enforced by PIN serviceability, order value caps, and fee toggles.
- Risk actions are logged with an auditable trail.

10.3.1. [Task] - Implement 3DS/SCA and terms acceptance

Description:

- Provider challenge UI; error states
- Terms/return policy checkbox and logging

10.3.1.1. [Subtask] - Provider challenge UI; error states

Description:

- Provider challenge UI
- Error states

10.3.1.2. [Subtask] - Terms/return policy checkbox and logging

Description:

- Terms/return policy checkbox
- Logging

10.3.2. [Task] - Implement COD eligibility and fees

Description:

- PIN serviceability, order value caps, fee toggles

10.3.2.1. [Subtask] - PIN serviceability, order value caps, fee toggles

Description:

- PIN serviceability
- Order value caps
- Fee toggles

10.4. [User Story] - Provider Resilience & Retries

As a shopper, I want resilient payments so that provider hiccups don’t block my purchase.

Acceptance Criteria:

- Retry/backoff strategy handles transient failures; fallback provider routing is available.
- Payment attempts are idempotent to avoid double charges.
- Users see accurate payment status with guidance on next steps.

10.4.1. [Task] - Implement payment provider failover and retry logic

Description:

- Retry/backoff; fallback provider routing

10.4.1.1. [Subtask] - Retry/backoff; fallback provider routing

Description:

- Retry/backoff
- Fallback provider routing

11. [Feature] - Orders, Fulfillment & Returns

Description: Manage order lifecycle, shipments, returns, and refunds with full auditability.

Acceptance Criteria:

- Order state transitions tracked with emitted domain events.
- Shipment tracking synced from carrier webhooks; status visible to users.
- Returns created with eligibility checks; refunds posted to ledger.

- Full timeline and notes maintained for each order.

11.1. [User Story] - Order Lifecycle

As a customer, I want my order to progress transparently so that I always know its status.

Acceptance Criteria:

- Order state transitions are tracked with domain events and timestamps.
- Shipment tracking syncs from carrier and is visible in account.
- Notifications are sent on key transitions (created, paid, shipped, delivered).

11.1.1. [Task] - Implement order states and transitions (created, paid, fulfilled, refunded)

Description:

- Event emission for state changes

11.1.1.1. [Subtask] - Event emission for state changes

Description:

- Event emission for state changes

11.1.2. [Task] - Implement shipment tracking

Description:

- Carrier tracking link and status sync

11.1.2.1. [Subtask] - Carrier tracking link and status sync

Description:

- Carrier tracking link

- Status sync

11.2. [User Story] - Returns & Refunds

As a customer, I want easy returns and prompt refunds so that I trust the store.

Acceptance Criteria:

- RMA creation checks eligibility and provides status updates.
- Refunds are processed and posted to the ledger; customer is notified.
- SLA targets are communicated and met for common cases.

11.2.1. [Task] - Implement RMA creation and eligibility checks

Description:

- Refund processing and ledger updates

11.2.1.1. [Subtask] - Refund processing and ledger updates

Description:

- Refund processing
- Ledger updates

11.3. [User Story] - Split Shipments & Partial Fulfillment

As a customer, I want partial fulfillment when needed so that available items ship without waiting.

Acceptance Criteria:

- Split shipments are supported with multiple tracking numbers.
- Shipment items and aggregated status are clear in the order timeline.

- Price/tax adjustments for partials are handled correctly.

11.3.1. [Task] - Implement partial fulfillment and multi-tracking

Description:

- Shipment items and status aggregation

11.3.1.1. [Subtask] - Shipment items and status aggregation

Description:

- Shipment items
- Status aggregation

11.4. [User Story] - Cancellations & Exchanges

As a customer, I want to cancel or exchange orders within policy so that I can fix mistakes.

Acceptance Criteria:

- Cancellation windows and fees are enforced by pre/after-ship rules.
- Exchange flow creates the new order and validates stock.
- Notifications confirm actions and next steps.

11.4.1. [Task] - Implement cancellation windows and fees

Description:

- Pre-ship/after-ship rules; notifications

11.4.1.1. [Subtask] - Pre-ship/after-ship rules; notifications

Description:

- Pre-ship/after-ship rules
- Notifications

11.4.2. [Task] - Implement exchange flow

Description:

- Exchange order creation; stock checks

11.4.2.1. [Subtask] - Exchange order creation; stock checks

Description:

- Exchange order creation
- Stock checks

11.5. [User Story] - Guest Tracking & Invoices

As a guest, I want to track my order and download invoices so that I don’t need an account.

Acceptance Criteria:

- Guest lookup via email/phone with OTP verification.
- Invoice PDF includes GST details and is downloadable.
- Rate limits and abuse protections apply to guest lookup.

11.5.1. [Task] - Implement guest order lookup via email/phone + OTP

Description:

- Invoice PDF download with GST details

11.5.1.1. [Subtask] - Invoice PDF download with GST details

Description:

- Invoice PDF download with GST details

12. [Feature] - Reviews & UGC

Description: Collect verified product reviews and media with moderation and anti-abuse.

Acceptance Criteria:

- Review CRUD restricted to verified purchasers; media uploads via presigned URLs with AV scanning.
- Anti-spam enforced leveraging global rate limiting (see 6.4); moderation queue operational.
- Published reviews meet content policy; takedown workflow exists.
- Review events tracked for analytics.

12.1. [User Story] - Verified Reviews

As a customer, I want to write reviews only when I've purchased so that content is trustworthy.

Acceptance Criteria:

- Review CRUD is restricted to verified purchasers with moderation queue.
- Media uploads use presigned URLs and pass AV scanning.
- Anti-spam is enforced leveraging global rate limits.

12.1.1. [Task] - Implement review CRUD with verification and moderation

Description:

- Media uploads via presigned URLs; virus scan

12.1.1.1. [Subtask] - Media uploads via presigned URLs; virus scan

Description:

- Media uploads via presigned URLs
- Virus scan

12.1.2. [Task] - Implement anti-spam (uses global rate limiting; see 6.4)

Description:

- Implement anti-spam (uses global rate limiting
- See 6.4)

12.2. [User Story] - Quality Signals & Merchant Replies

As a customer, I want helpful review signals and merchant replies so that I can evaluate products better.

Acceptance Criteria:

- Helpful votes are one-per-user; reply moderation is enforced.
- Flagged content workflow includes reasons, SLAs, and takedown audit.
- Merchant replies display clearly on PDP.

12.2.1. [Task] - Implement helpful votes and merchant replies

Description:

- One-vote-per-user; reply moderation

12.2.1.1. [Subtask] - One-vote-per-user; reply moderation

Description:

- One-vote-per-user
- Reply moderation

12.2.2. [Task] - Implement flagged content workflow

Description:

- Reasons, SLAs, takedown audit

12.2.2.1. [Subtask] - Reasons, SLAs, takedown audit

Description:

- Reasons
- SLAs
- Takedown audit

12.3. [User Story] - Media Handling

As a customer, I want safe media handling so that uploads don’t expose me or others.

Acceptance Criteria:

- EXIF stripping, size limits, and safety checks are applied to uploads.
- Image processing and virus scanning occur before publish.
- Non-compliant media is rejected with clear error messages.

12.3.1. [Task] - Implement EXIF stripping, size limits, and safety

Description:

- Image processing and AV scanning

12.3.1.1. [Subtask] - Image processing and AV scanning

Description:

- Image processing
- AV scanning

13. [Feature] - Subscriptions, Loyalty & Referrals

Description: Drive retention via subscription billing, loyalty points, and referrals with fraud checks.

Acceptance Criteria:

- Subscription create/pause/resume/cancel with scheduler; provider reconciliation in place.
- Loyalty accrual/redeem rules persisted and displayed to users.
- Referral links generate attribution; fraud heuristics applied to block abuse.
- Churn and dunning outcomes tracked.

13.1. [User Story] - Subscription Lifecycle

As a subscriber, I want to manage my subscription so that billing happens on my terms.

Acceptance Criteria:

- Create/pause/resume/cancel supported with a reliable billing scheduler.
- Provider webhook reconciliation keeps subscription state accurate.
- Upcoming charges and changes are communicated proactively.

13.1.1. [Task] - Implement subscription create/pause/resume/cancel; billing scheduler

Description:

- Provider webhook reconciliation

13.1.1.1. [Subtask] - Provider webhook reconciliation

Description:

- Provider webhook reconciliation

13.2. [User Story] - Loyalty & Referrals

As a customer, I want loyalty points and referrals so that I’m rewarded for engagement.

Acceptance Criteria:

- Referral links generate attribution; fraud checks block abuse.
- Points accrual/redeem rules are persisted and visible in UI.
- Suspicious activity is detected and prevented.

13.2.1. [Task] - Implement referral link generation and attribution

Description:

- Fraud checks (device/IP/email heuristics)

13.2.1.1. [Subtask] - Fraud checks (device/IP/email heuristics)

Description:

- Fraud checks (device/IP/email heuristics)

14. [Feature] - Web Frontend (React/Next.js)

Description: Ship an accessible, SEO-friendly Next.js storefront with core shopping flows.

Acceptance Criteria:

- App shell, routing, and error boundaries implemented; base i18n (en-IN) configured.
- Design system components meet WCAG AA; focus and keyboard navigation verified.
- PLP, PDP, Cart, Checkout, and Account pages function end-to-end.
- Core Web Vitals budgets monitored; image optimization and code-splitting in place.
- Sitemaps, robots.txt, canonical URLs, and structured data configured.

14.1. [User Story] - App Shell & Routing

As a shopper, I want a fast, accessible storefront so that I can browse and buy smoothly.

Acceptance Criteria:

- Next.js app shell/routing works with error boundaries and i18n baseline.
- Core pages load within performance budgets; SEO basics configured.
- Accessibility checks pass on core navigations.

14.1.1. [Task] - Implement Next.js app with SSR/SSG where applicable

Description:

- Layouts, error boundaries, i18n baseline (en-IN)

14.1.1.1. [Subtask] - Layouts, error boundaries, i18n baseline (en-IN)

Description:

- Layouts
- Error boundaries

- I18n baseline (en-IN)

14.2. [User Story] - Design System

As a designer/developer, I want a reusable design system so that UI is consistent and accessible.

Acceptance Criteria:

- Tokens, theming, and core components meet WCAG AA.
- Keyboard focus states and semantics are verified.
- Components are adopted across pages and documented.

14.2.1. [Task] - Implement UI tokens, theming, and components (buttons, cards, inputs, modals, nav)

Description:

- Accessibility-first (WCAG AA), keyboard focus states

14.2.1.1. [Subtask] - Accessibility-first (WCAG AA), keyboard focus states

Description:

- Accessibility-first (WCAG AA)
- Keyboard focus states

14.3. [User Story] - Core Pages

As a shopper, I want core shopping pages to be complete so that I can discover, evaluate, and purchase without friction.

Acceptance Criteria:

- PLP supports facets/sort with responsive grid and skeleton loaders.

- PDP includes gallery, sourcing block, and required structured data.
- Cart/Checkout flows cover address, shipping, and payment steps end-to-end.

14.3.1. [Task] - Implement PLP with facets and sort

Description:

- Responsive grid and skeleton loaders

14.3.1.1. [Subtask] - Responsive grid and skeleton loaders

Description:

- Responsive grid
- Skeleton loaders

14.3.2. [Task] - Implement PDP with gallery and sourcing block

Description:

- JSON-LD Product and Breadcrumb

14.3.2.1. [Subtask] - JSON-LD Product and Breadcrumb

Description:

- JSON-LD Product
- Breadcrumb

14.3.3. [Task] - Implement cart and checkout flows

Description:

- Address, shipping, payment steps

14.3.3.1. [Subtask] - Address, shipping, payment steps

Description:

- Address
- Shipping
- Payment steps

14.3.4. [Task] - Implement account pages (orders, addresses, returns)

Description:

- Implement account pages (orders
- Addresses
- Returns)

14.4. [User Story] - Performance & SEO

As a shopper, I want fast pages and clear search results so that I can find and complete purchases easily.

Acceptance Criteria:

- Image optimization, prefetching, and code splitting meet Core Web Vitals budgets.
- Sitemaps, robots.txt, canonical URLs are present and correct.
- Structured data (Article, FAQ, Organization) validates without errors.

14.4.1. [Task] - Implement image optimization (AVIF/WebP), prefetching, and code splitting

Description:

- Core Web Vitals budgets and monitoring

14.4.1.1. [Subtask] - Core Web Vitals budgets and monitoring

Description:

- Core Web Vitals budgets
- Monitoring

14.4.2. [Task] - Implement sitemaps, robots.txt, and canonical URLs

Description:

- Structured data (Article, FAQ, Organization)

14.4.2.1. [Subtask] - Structured data (Article, FAQ, Organization)

Description:

- Structured data (Article
- FAQ
- Organization)

15. [Feature] - Analytics & Telemetry

Description: Standardize event tracking, pipeline, alerts, and experimentation for data-driven decisions.

Acceptance Criteria:

- Event taxonomy defined; SDK wrappers implemented with consistent IDs and timestamps.
- ETL to warehouse operational; dbt models apply consent/retention filters.
- Alerts for checkout CR dip, payment failures, and LCP breaches notify Slack/webhook.
- Feature flags with deterministic bucketing and exposure events; holdouts supported.

- KPI dashboards available to stakeholders.

15.1. [User Story] - Event Taxonomy & SDKs

As a product analyst/engineer, I want a clear event taxonomy and SDKs so that analytics are consistent and trustworthy across platforms.

Acceptance Criteria:

- Events are defined with stable names, IDs, and timestamps; wrappers exist on web and server.
- Context (user/session/device) is consistent; PII handling follows policy.
- QA checklist and sample payloads validate event correctness.

15.1.1. [Task] - Define events (view_item, add_to_cart, begin_checkout, purchase, subscribe, refund_initiated, search, view_article)

Description:

- Implement wrappers on web and server; consistent ids and timestamps

15.1.1.1. [Subtask] - Implement wrappers on web and server; consistent ids and timestamps

Description:

- Implement wrappers on web and server
- Consistent ids and timestamps

15.2. [User Story] - Data Pipeline & Dashboards

As a business stakeholder, I want reliable dashboards so that I can track KPIs and catch regressions.

Acceptance Criteria:

- ETL to the warehouse is operational; dbt models apply consent/retention filters.

- Alerts exist for checkout CR dip, payment failure rate, and LCP breaches.
- KPI dashboards are available with documented refresh cadence.

15.2.1. [Task] - Implement ETL to warehouse and BI

Description:

- dbt models; retention and consent filters

15.2.1.1. [Subtask] - dbt models; retention and consent filters

Description:

- Dbt models
- Retention and consent filters

15.2.2. [Task] - Configure alerts

Description:

- Checkout CR dip > 20%, payment failure > 2%, LCP > 3s

15.2.2.1. [Subtask] - Checkout CR dip > 20%, payment failure > 2%, LCP > 3s

Description:

- Checkout CR dip > 20%
- Payment failure > 2%
- LCP > 3s

15.3. [User Story] - Experimentation & Feature Flags

As a product manager, I want experimentation and feature flags so that we can ship safely and learn what works.

Acceptance Criteria:

- Deterministic bucketing with exposure events and holdouts is implemented.
- A/B test lifecycle includes hypothesis, guardrails, rollout/rollback.
- Analysis and governance rules are defined and documented.

15.3.1. [Task] - Implement feature flag SDK and bucketing

Description:

- Deterministic assignment; exposure events and holdouts

15.3.1.1. [Subtask] - Deterministic assignment; exposure events and holdouts

Description:

- Deterministic assignment
- Exposure events and holdouts

15.3.2. [Task] - Define A/B test lifecycle

Description:

- Hypothesis, guardrails, rollout/rollback procedures

15.3.2.1. [Subtask] - Hypothesis, guardrails, rollout/rollback procedures

Description:

- Hypothesis
- Guardrails
- Rollout/rollback procedures

15.3.3. [Task] - Define analysis and governance

Description:

- Segmentation and stats checks; archive of results

15.3.3.1. [Subtask] - Segmentation and stats checks; archive of results

Description:

- Segmentation and stats checks
- Archive of results

16. [Feature] - Notifications & Communications

Description: Deliver transactional and lifecycle communications across email, push, and SMS with preferences.

Acceptance Criteria:

- Transactional email templates built and localized; rendered correctly on major clients.
- Push/SMS set up for key events with user preferences and quiet hours.
- In-app notification center shows categorized messages with deep links.
- Delivery orchestration handles de-duplication and channel failover.

16.1. [User Story] - Email & Templates

As a customer, I want clear transactional emails so that I’m informed about my orders.

Acceptance Criteria:

- Templates exist for order confirmation, shipping, and refund; localized where needed.
- Previews render correctly on major clients and devices.

- Links deep link into app/web where applicable.

16.1.1. [Task] - Build transactional emails (order confirmation, shipping, refund)

Description:

- Template system and localization hooks

16.1.1.1. [Subtask] - Template system and localization hooks

Description:

- Template system
- Localization hooks

16.2. [User Story] - Push & SMS

As a customer, I want timely push/SMS notifications so that I don’t miss important updates.

Acceptance Criteria:

- Topics are configured for orders/content/offers; preferences and consent respected with quiet hours.
- Delivery metrics are tracked; graceful fallbacks exist.
- Opt-out flows work across channels.

16.2.1. [Task] - Configure push topics (orders, content, offers) and SMS for order events (where applicable)

Description:

- Preferences and consent toggles

16.2.1.1. [Subtask] - Preferences and consent toggles

Description:

- Preferences
- Consent toggles

16.3. [User Story] - Notifications Center

As a customer, I want an in-app notification center so that I can see and manage messages.

Acceptance Criteria:

- Unified inbox shows read/unread status, categories, and deep links.
- Preferences and digests support quiet hours and batching.
- De-duplication and channel failover orchestrate delivery across push/email/SMS.

16.3.1. [Task] - Build unified in-app inbox (web/mobile)

Description:

- Read/unread, categories, deep links

16.3.1.1. [Subtask] - Read/unread, categories, deep links

Description:

- Read/unread
- Categories
- Deep links

16.3.2. [Task] - Implement preferences and digests

Description:

- Per-channel/topic; quiet hours and batching

16.3.2.1. [Subtask] - Per-channel/topic; quiet hours and batching

Description:

- Per-channel/topic
- Quiet hours and batching

16.3.3. [Task] - Implement delivery orchestration

Description:

- De-duplication and channel failover (push/email/SMS)

16.3.3.1. [Subtask] - De-duplication and channel failover (push/email/SMS)

Description:

- De-duplication
- Channel failover (push/email/SMS)

16.4. [User Story] - India DLT & Templates

As a compliance owner, I want registered sender IDs and templates so that SMS is compliant in India.

Acceptance Criteria:

- Sender IDs and templates are registered and mapped to events.
- Compliance logs are retained for audits and reconciliations.
- Failures surface with actionable error messages.

16.4.1. [Task] - Register sender IDs and templates

Description:

- Map events to DLT templates; compliance logs

16.4.1.1. [Subtask] - Map events to DLT templates; compliance logs

Description:

- Map events to DLT templates
- Compliance logs

16.5. [User Story] - WhatsApp Business & Routing

As a customer, I want WhatsApp notifications where I've opted in so that I receive updates on my preferred channel.

Acceptance Criteria:

- WhatsApp templates are configured for key events; opt-in is respected.
- Fallback to SMS/email occurs when WhatsApp fails or is unavailable.
- Quiet hours and preferences are enforced across channels.

16.5.1. [Task] - Implement WhatsApp notifications for key events

Description:

- Fallback to SMS/email; opt-in and quiet hours

16.5.1.1. [Subtask] - Fallback to SMS/email; opt-in and quiet hours

Description:

- Fallback to SMS/email
- Opt-in and quiet hours

17. [Feature] - Support & Help Center

Description: Provide self-serve help content and real-time chat with escalation and CSAT collection.

Acceptance Criteria:

- CMS-driven help center with categories, search, and article feedback.
- Chat widget integrated with ticketing; escalation rules enforced; CSAT captured.
- SLAs and first-response times visible to support leads.
- Content management roles and workflow defined.

17.1. [User Story] - Help Center

As a customer, I want a self-serve help center so that I can resolve common issues quickly without waiting for an agent.

Acceptance Criteria:

- CMS-driven articles support categories, search, and feedback.
- Editorial workflow supports preview and publish with versioning.
- Article helpfulness is tracked for continuous improvement.

17.1.1. [Task] - Build CMS-driven articles and categories

Description:

- Search and feedback on articles

17.1.1.1. [Subtask] - Search and feedback on articles

Description:

- Search
- Feedback on articles

17.2. [User Story] - Chat & Escalation

As a customer, I want real-time chat that escalates when needed so that I get timely help.

Acceptance Criteria:

- Chat integrates with ticketing; escalation rules and SLAs are enforced.
- CSAT survey is captured after resolution.
- Transcripts are attached to tickets for context.

17.2.1. [Task] - Integrate chat widget and ticketing

Description:

- Escalation rules and CSAT survey

17.2.1.1. [Subtask] - Escalation rules and CSAT survey

Description:

- Escalation rules
- CSAT survey

18. [Feature] - Reliability & SRE Operations

Description: Define SLOs, cost guardrails, DR, and automation to maintain reliable operations.

Acceptance Criteria:

- SLOs and error budgets defined per service with dashboards.
- DR plan with RPO/RTO targets; cross-region DR strategy; quarterly drills executed.
- Cost budgets, anomaly alerts, and allocation tagging in place; monthly review.
- Automated runbooks for common incidents; ChatOps triggers and approvals.
- Blameless postmortems with action items tracked to closure.

18.1. [User Story] - SLOs & Runbooks

As an SRE, I want clear SLOs and runbooks so that incidents are prevented or resolved quickly.

Acceptance Criteria:

- SLOs for uptime, latency, and errors are defined with dashboards.
- Error budget policy is documented and enforced.
- Runbooks exist for top incidents with paging and ownership.

18.1.1. [Task] - Define SLOs for uptime, latency, errors

Description:

- Error budget policy and dashboards

18.1.1.1. [Subtask] - Error budget policy and dashboards

Description:

- Error budget policy
- Dashboards

18.2. [User Story] - DR & Backups

As an operations owner, I want DR and backups so that we can recover from failures.

Acceptance Criteria:

- RPO/RTO targets are defined; snapshots and cross-region read replica plan exist.
- Quarterly restore and failover drills are executed and documented.
- Backup failures trigger alerts and are remediated.

18.2.1. [Task] - Define RPO/RTO targets, snapshots, and cross-region read replica plan

Description:

- Quarterly restore and failover drills

18.2.1.1. [Subtask] - Quarterly restore and failover drills

Description:

- Quarterly restore
- Failover drills

18.3. [User Story] - Cost Guardrails & FinOps

As a finance/ops owner, I want cost guardrails so that spend stays under control.

Acceptance Criteria:

- Budgets and anomaly alerts are set; per-service cost allocation uses tags/accounts.
- Dashboards show unit economics KPIs for monthly reviews.
- Review outcomes produce cost actions tracked to closure.

18.3.1. [Task] - Configure budgets and anomaly alerts

Description:

- Per-service cost allocation via tags and accounts

18.3.1.1. [Subtask] - Per-service cost allocation via tags and accounts

Description:

- Per-service cost allocation via tags
- Accounts

18.3.2. [Task] - Create dashboards and reports

Description:

- Unit economics KPIs; monthly reviews and actions

18.3.2.1. [Subtask] - Unit economics KPIs; monthly reviews and actions

Description:

- Unit economics KPIs
- Monthly reviews and actions

18.4. [User Story] - Incident Automation

As an SRE, I want automated runbooks so that common failures are remediated quickly.

Acceptance Criteria:

- Auto-remediation playbooks trigger via ChatOps with approvals.
- Post-incident reviews are blameless; action items tracked to closure.
- MTTR trends improve and are reported.

18.4.1. [Task] - Automate runbooks for common failures

Description:

- Auto-remediation playbooks; ChatOps triggers and approvals

18.4.1.1. [Subtask] - Auto-remediation playbooks; ChatOps triggers and approvals

Description:

- Auto-remediation playbooks
- ChatOps triggers and approvals

18.4.2. [Task] - Conduct post-incident follow-ups

Description:

- Blameless postmortems; action items tracked to closure

18.4.2.1. [Subtask] - Blameless postmortems; action items tracked to closure

Description:

- Blameless postmortems
- Action items tracked to closure

19. [Feature] - Content & CMS

Description: Power the Learn Hub with a flexible content model, webhooks, and SEO-friendly templates.

Acceptance Criteria:

- Article/ritual/glossary types defined and rendered; preview flows supported.
- Webhook ingestion triggers cache invalidation; content publishes within seconds.

- SEO pillar page templates shipped with internal linking.
- Role-based editorial permissions configured in CMS.

19.1. [User Story] - Learn Hub CMS Integration

As a content editor, I want flexible content types and fast publish so that Learn Hub stays fresh and finds an audience.

Acceptance Criteria:

- Article/ritual/glossary types render with preview flows.
- Webhook ingestion triggers cache invalidation within seconds.
- SEO pillar templates support internal linking and metadata.

19.1.1. [Task] - Implement article, ritual, and glossary types and rendering

Description:

- Webhook ingestion and cache invalidation
- SEO pillar pages templates and internal linking

19.1.1.1. [Subtask] - Webhook ingestion and cache invalidation

Description:

- Webhook ingestion
- Cache invalidation

19.1.1.2. [Subtask] - SEO pillar pages templates and internal linking

Description:

- SEO pillar pages templates

- Internal linking

20. [Feature] - Governance & Quality

Description: Enforce quality gates and versioned releases with clear changelogs and test strategy.

Acceptance Criteria:

- Unit/integration/e2e tests cover critical paths to agreed thresholds.
- Semantic versioning and automated changelog generation in CI.
- Release notes produced and distributed each release; rollback procedure documented.
- CI blocks merges when quality gates fail.

20.1. [User Story] - Testing Strategy

As a QA lead, I want a clear testing strategy so that critical paths are protected from regressions.

Acceptance Criteria:

- Unit/integration/e2e tests meet agreed thresholds for critical paths.
- Factories and fixtures provide reliable test data.
- CI blocks merges when quality gates fail.

20.1.1. [Task] - Implement unit tests (critical paths ≥ 80%), integration (API), and e2e (checkout smoke)

Description:

- Test data factories and fixtures

20.1.1.1. [Subtask] - Test data factories and fixtures

Description:

- Test data factories
- Fixtures

20.2. [User Story] - Release & Versioning

As a release manager, I want versioned releases with changelogs so that deployments are traceable and reversible.

Acceptance Criteria:

- Semantic versioning and automated changelog generation run in CI.
- Release notes are produced and distributed each release.
- Rollback procedure is documented and verified.

20.2.1. [Task] - Implement semantic versioning and changelog automation

Description:

- Release notes template and distribution

20.2.1.1. [Subtask] - Release notes template and distribution

Description:

- Release notes template
- Distribution

21. [Feature] - Mobile App (React Native)

Description: Deliver a performant RN app with core shopping flows, push, deep linking, and store readiness.

Acceptance Criteria:

- App boots with navigation/theming; icons/splash configured; env handling works.
- Onboarding variants functional; deep links route correctly; completion state persists.
- PLP, PDP, cart, and checkout screens operate end-to-end.
- Push notifications and deep linking integrated; performance metrics tracked with alerts.
- Store listings prepared; staged rollout configured; crash-free > 99.5%, ANR ≤ 0.3%, cold start ≤ 2.5s.

21.1. [User Story] - React Native Bootstrap

As a mobile developer, I want a bootstrapped RN app so that we can build features quickly.

Acceptance Criteria:

- apps/mobile package has TS/ESLint/Prettier, navigation, theming, and env handling.
- Native modules are configured; icons/splash set up.
- App boots on Android/iOS with dev builds.

21.1.1. [Task] - Create apps/mobile package with TS, ESLint, Prettier

Description:

- Configure navigation (React Navigation), theming, env handling
- Set up native modules config (Android/iOS), app icons/splash

21.1.1.1. [Subtask] - Configure navigation (React Navigation), theming, env handling

Description:

- Configure navigation (React Navigation)
- Theming
- Env handling

21.1.1.2. [Subtask] - Set up native modules config (Android/iOS), app icons/splash

Description:

- Set up native modules config (Android/iOS)
- App icons/splash

21.2. [User Story] - Onboarding & Navigation

As a mobile user, I want a smooth onboarding and clear navigation so that I can explore the app easily.

Acceptance Criteria:

- Onboarding variants work and completion state persists across sessions.
- Deep links route correctly to target screens.
- Tab/stack navigation structure implemented with back behavior defined.

21.2.1. [Task] - Build onboarding screens with variant flags (education-first/quick-start)

Description:

- Persist onboarding completion; deep link routing

21.2.1.1. [Subtask] - Persist onboarding completion; deep link routing

Description:

- Persist onboarding completion

- Deep link routing

21.2.2. [Task] - Implement tab/stack structure (Home, Shop, Learn, Account, Cart)

Description:

- Implement tab/stack structure (Home
- Shop
- Learn

21.3. [User Story] - Mobile Catalog & PDP

As a mobile shopper, I want performant PLP/PDP so that shopping feels native and responsive.

Acceptance Criteria:

- PLP supports facets/sort and infinite scroll with skeletons and empty states.
- PDP includes gallery, energy tags, origin/certification, share/wishlist/related.
- Performance meets mobile budgets on low-end devices.

21.3.1. [Task] - Implement PLP with facets/sort and infinite scroll

Description:

- Facet drawer UX, skeletons, empty state

21.3.1.1. [Subtask] - Facet drawer UX, skeletons, empty state

Description:

- Facet drawer UX
- Skeletons
- Empty state

21.3.2. [Task] - Implement PDP with gallery, energy tags, and origin/certification

Description:

- Share, wishlist, related products modules

21.3.2.1. [Subtask] - Share, wishlist, related products modules

Description:

- Share
- Wishlist
- Related products modules

21.4. [User Story] - Mobile Cart & Checkout

As a mobile shopper, I want end-to-end cart and checkout so that I can buy in-app.

Acceptance Criteria:

- Cart screens show items, totals, and coupons; persisted securely and merge-on-login.
- Checkout integrates payment SDKs and handles success/failure callbacks.
- Errors are recoverable and status is visible to the user.

21.4.1. [Task] - Build cart screens (items, totals, coupons)

Description:

- Persist cart securely (async storage) and merge-on-login

21.4.1.1. [Subtask] - Persist cart securely (async storage) and merge-on-login

Description:

- Persist cart securely (async storage)
- Merge-on-login

21.4.2. [Task] - Build checkout flow (address, shipping, payment)

Description:

- Integrate Razorpay/Stripe RN SDKs; handle success/failure callbacks

21.4.2.1. [Subtask] - Integrate Razorpay/Stripe RN SDKs; handle success/failure callbacks

Description:

- Integrate Razorpay/Stripe RN SDKs
- Handle success/failure callbacks

21.5. [User Story] - Push, Deep Links, and Device Metrics

As a mobile user, I want relevant push and deep links so that I return to the right place quickly.

Acceptance Criteria:

- Permission prompts include a value proposition; topic subscriptions are configured.
- Deep links/universal links route correctly on cold and warm starts.
- Device performance metrics are tracked with alerts.

21.5.1. [Task] - Implement push notifications (orders, content, offers)

Description:

- Permission prompts with value proposition; topic subscriptions

21.5.1.1. [Subtask] - Permission prompts with value proposition; topic subscriptions

Description:

- Permission prompts with value proposition
- Topic subscriptions

21.5.2. [Task] - Implement deep links and universal/app links

Description:

- Cold/warm start routing tests

21.5.2.1. [Subtask] - Cold/warm start routing tests

Description:

- Cold/warm start routing tests

21.6. [User Story] - Store Readiness & Compliance

As a release manager, I want store readiness tasks complete so that approvals go smoothly.

Acceptance Criteria:

- Device QA and performance targets met (crash-free > 99.5%, ANR ≤ 0.3%, cold start ≤ 2.5s).
- Store listings and release pipelines are configured for staged rollout.
- Post-release monitoring is in place with alerting.

21.6.1. [Task] - Perform device matrix QA (Android 8–14, iOS 15+); meet performance targets

Description:

- Crash-free > 99.5%, ANR ≤ 0.3%, cold start ≤ 2.5s

21.6.1.1. [Subtask] - Crash-free > 99.5%, ANR ≤ 0.3%, cold start ≤ 2.5s

Description:

- Crash-free > 99.5%
- ANR ≤ 0.3%
- Cold start ≤ 2.5s

21.6.2. [Task] - Prepare App Store/Play listings and release pipeline

Description:

- TestFlight/Play internal tracks; staged rollout and monitoring

21.6.2.1. [Subtask] - TestFlight/Play internal tracks; staged rollout and monitoring

Description:

- TestFlight/Play internal tracks
- Staged rollout and monitoring

22. [Feature] - Admin Portal & Backoffice

Description: Equip staff with secure tools to manage catalog, inventory, orders/returns, and promotions.

Acceptance Criteria:

- RBAC with audit logs on admin actions; permission checks enforced server-side.
- Catalog CRUD and bulk import with validation and preview.
- Inventory adjustments/reservations tracked; stock alerts generated.
- Order ops supports status changes, refund approvals, and finance exports.

22.1. [User Story] - RBAC & Audit

As an admin, I want RBAC and audit so that staff actions are controlled and traceable.

Acceptance Criteria:

- Roles/scopes and permissions enforced server-side for admin actions.
- Audit logs capture who/what/when with immutable storage.
- Permission checks are validated with tests.

22.1.1. [Task] - Implement roles/scopes (staff, admin) and permissions

Description:

- API guard middleware; audit logs for admin actions

22.1.1.1. [Subtask] - API guard middleware; audit logs for admin actions

Description:

- API guard middleware
- Audit logs for admin actions

22.2. [User Story] - Catalog Management

As a catalog manager, I want to manage products so that the catalog stays accurate and fresh.

Acceptance Criteria:

- Product/variant/pricing/collections CRUD with validation and preview.
- Bulk CSV upload validates and provides a preview before publish.
- Publish workflow and versioning are documented.

22.2.1. [Task] - Implement product CRUD, variants, pricing, and collections

Description:

- Bulk upload (CSV) and validation; preview before publish

22.2.1.1. [Subtask] - Bulk upload (CSV) and validation; preview before publish

Description:

- Bulk upload (CSV) and validation
- Preview before publish

22.3. [User Story] - Inventory & Fulfillment

As an operations manager, I want inventory tools so that stock levels remain accurate and actionable.

Acceptance Criteria:

- Inventory adjustments and reservations are supported with audit.
- Stock thresholds and alerts notify when levels dip.
- Ops views support quick corrections and exports.

22.3.1. [Task] - Implement inventory adjustments and reservations

Description:

- Stock thresholds and alerts

22.3.1.1. [Subtask] - Stock thresholds and alerts

Description:

- Stock thresholds

- Alerts

22.4. [User Story] - Orders & Returns Ops

As a support agent, I want order ops tools so that I can resolve customer requests efficiently.

Acceptance Criteria:

- Order search, status updates, and refunds/RMA approvals are available with permission checks.
- Each order shows a notes/timeline for context.
- Refund exports (CSV) exist for finance reconciliation.

22.4.1. [Task] - Implement order search, status updates, and refunds/RMA approvals

Description:

- Notes/timeline per order; permission checks
- Refunds export (CSV) for finance reconciliation

22.4.1.1. [Subtask] - Notes/timeline per order; permission checks

Description:

- Notes/timeline per order
- Permission checks

22.4.1.2. [Subtask] - Refunds export (CSV) for finance reconciliation

Description:

- Refunds export (CSV) for finance reconciliation

22.5. [User Story] - Promotions & Coupons

As a marketer, I want to create and control promotions so that campaigns can run safely.

Acceptance Criteria:

- Create/disable coupons with stacking and eligibility rules.
- Limits and schedules are enforced; misuse prevented.
- Audit trail exists for all changes.

22.5.1. [Task] - Implement create/disable coupons, rules, and limits

Description:

- Stacking rules and eligibility conditions

22.5.1.1. [Subtask] - Stacking rules and eligibility conditions

Description:

- Stacking rules
- Eligibility conditions

22.6. [User Story] - Reviews Moderation

As a moderator, I want a reviews queue so that policy-violating content is handled quickly.

Acceptance Criteria:

- Moderation queue supports approve/reject with reasons; media review workflow.
- Policy enforcement and takedowns are audited.
- SLAs exist for response times.

22.6.1. [Task] - Implement queue with approve/reject (reasons) and policy enforcement

Description:

- Media review and takedown workflow

22.6.1.1. [Subtask] - Media review and takedown workflow

Description:

- Media review
- Takedown workflow

22.7. [User Story] - Customer Support Tools

As a support agent, I want customer tools so that I can look up and assist users securely.

Acceptance Criteria:

- Customer search and impersonation available with consent banner.
- All actions are audited and permission-checked.
- Sensitive data views are gated.

22.7.1. [Task] - Implement customer search and impersonation

Description:

- Audit logs; impersonation consent banner

22.7.1.1. [Subtask] - Audit logs; impersonation consent banner

Description:

- Audit logs
- Impersonation consent banner

22.8. [User Story] - Fraud Review Queue

As a risk analyst, I want a manual review queue so that risky orders are handled with consistent decisions.

Acceptance Criteria:

- Queue supports approve/hold/cancel with notes and SLAs.
- Decisions integrate with order state and notifications.
- Metrics track hit rates and outcomes.

22.8.1. [Task] - Implement manual review for risky orders

Description:

- Decisions (approve/hold/cancel); notes and SLAs

22.8.1.1. [Subtask] - Decisions (approve/hold/cancel); notes and SLAs

Description:

- Decisions (approve/hold/cancel)
- Notes and SLAs

22.9. [User Story] - Bulk Operations

As an operator, I want bulk updates so that large changes are safe and efficient.

Acceptance Criteria:

- Bulk price/inventory updates via CSV with validation and preview.
- Rollback or correction flow exists for mistakes.
- Audit and notifications for bulk jobs.

22.9.1. [Task] - Implement bulk price/inventory updates

Description:

- CSV upload with validation; preview + rollback

22.9.1.1. [Subtask] - CSV upload with validation; preview + rollback

Description:

- CSV upload with validation
- Preview + rollback

22.10. [User Story] - Feature Flags & Toggles

As an admin, I want to manage feature flags so that we can control rollouts across audiences.

Acceptance Criteria:

- Admin UI controls flags with targeting rules and approvals.
- Changes are audited; server-side enforcement is authoritative.
- Safety checks prevent locking out critical paths.

22.10.1. [Task] - Implement admin control of flags

Description:

- Targeting rules; audit and approvals

22.10.1.1. [Subtask] - Targeting rules; audit and approvals

Description:

- Targeting rules

- Audit and approvals

23. [Feature] - Tax & Invoicing (India GST)

Description: Comply with Indian GST for pricing, invoices, and reporting; prepare for cross-border duties.

Acceptance Criteria:

- GST rates applied by HSN and destination; HSN tables maintained.
- GSTIN capture/validation (B2B); invoice numbering scheme defined; PDFs generated and stored.
- Tax summary reports exportable for filing with audit trails.
- Cross-border landed cost and compliance document generation supported.

23.1. [User Story] - GST Tax Calculation

As a finance owner, I want correct GST calculation so that invoices and prices comply with law.

Acceptance Criteria:

- GST rates are applied by HSN and destination with maintained tables.
- Price components and tax breakdowns are visible and correct.
- Edge cases (exempt/zero-rated) are handled.

23.1.1. [Task] - Apply GST rates by HSN and destination

Description:

- Maintain HSN catalog and rate tables

23.1.1.1. [Subtask] - Maintain HSN catalog and rate tables

Description:

- Maintain HSN catalog
- Rate tables

23.2. [User Story] - Invoices & GSTIN

As a business customer, I want valid tax invoices so that I can claim input credit.

Acceptance Criteria:

- GSTIN capture/validation (B2B) and invoice numbering scheme are enforced.
- PDF invoices are generated, emailed, and stored for download.
- Invoice data is searchable and auditable.

23.2.1. [Task] - Capture GSTIN (B2B), validate formats; define invoice numbering scheme

Description:

- Generate PDF invoices; email to customer; store for download

23.2.1.1. [Subtask] - Generate PDF invoices; email to customer; store for download

Description:

- Generate PDF invoices
- Email to customer
- Store for download

23.3. [User Story] - Reports & Reconciliation

As a finance owner, I want tax reports so that monthly filing and reconciliation are straightforward.

Acceptance Criteria:

- Tax summary reports by period are exportable (CSV) with audit trail.
- Filters for B2B/B2C and tax categories exist.
- Data matches ledger records.

23.3.1. [Task] - Generate tax summary reports by period

Description:

- Export CSV for filing and audit trail

23.3.1.1. [Subtask] - Export CSV for filing and audit trail

Description:

- Export CSV for filing
- Audit trail

23.4. [User Story] - Cross-border Duties & Taxes

As a cross-border ops owner, I want landed cost and compliance docs so that international orders clear smoothly.

Acceptance Criteria:

- Landed cost estimates use HS codes, duties, and VAT with provider fallback.
- Commercial invoice and customs declarations are generated when required.
- Region-specific rules are documented and enforced.

23.4.1. [Task] - Implement landed cost estimation (HS codes, duties, VAT)

Description:

- Provider integration and fallback rate tables

23.4.1.1. [Subtask] - Provider integration and fallback rate tables

Description:

- Provider integration
- Fallback rate tables

23.4.2. [Task] - Generate compliance documents

Description:

- Commercial invoice and customs declarations

23.4.2.1. [Subtask] - Commercial invoice and customs declarations

Description:

- Commercial invoice
- Customs declarations

23.5. [User Story] - Rounding & Labels

As a shopper, I want clear tax-inclusive pricing so that I understand what I'll pay.

Acceptance Criteria:

- India rounding policies and GST labels are applied consistently.
- MRP/compare-at display adheres to Indian guidelines.
- SEO and UX reflect inclusive/exclusive tax notes appropriately.

23.5.1. [Task] - Implement India rounding policies and GST labels

Description:

- MRP/compare-at display with GST notes

23.5.1.1. [Subtask] - MRP/compare-at display with GST notes

Description:

- MRP/compare-at display with GST notes

23.6. [User Story] - GST Filing Exports

As a finance owner, I want filing exports so that GST submissions are accurate and timely.

Acceptance Criteria:

- GSTR-1/3B exports include required fields with period selection.
- CSV exports have an audit trail and integrity checks.
- Reconciliation notes can be attached.

23.6.1. [Task] - Generate GSTR-1/3B exports

Description:

- Period selection; CSV with audit trail

23.6.1.1. [Subtask] - Period selection; CSV with audit trail

Description:

- Period selection
- CSV with audit trail

23.7. [User Story] - TCS/TDS & COD Nuances

As a finance owner, I want TCS/TDS and COD nuances handled so that compliance and accounting remain correct.

Acceptance Criteria:

- TCS/TDS logic is applied where applicable with correct reporting.
- COD-specific tax reporting adjustments are implemented and tested.
- Documentation covers scenarios and thresholds.

23.7.1. [Task] - Implement TCS/TDS handling where applicable

Description:

- COD-specific tax reporting adjustments

23.7.1.1. [Subtask] - COD-specific tax reporting adjustments

Description:

- COD-specific tax reporting adjustments

24. [Feature] - Shipping & Logistics Integration

Description: Integrate carriers for rates, labels, tracking, and exception handling, including international.

Acceptance Criteria:

- Aggregator integration for rate shopping, serviceability, pickups.
- Label generation and manifesting operational; tracking webhooks update orders.
- Exceptions (RTO/failed delivery) flows create tickets and customer comms.

- International shipping pilot supports region restrictions and address validation.

24.1. [User Story] - Carrier Aggregator Integration

As an operations manager, I want carrier aggregator integration so that shipping is reliable and cost-effective.

Acceptance Criteria:

- Rate shopping, serviceability checks, and pickup scheduling work end-to-end.
- Errors and fallbacks are logged with alerts.
- Credentials and webhooks are secured and monitored.

24.1.1. [Task] - Integrate Shiprocket/Delhivery (or chosen partner)

Description:

- Rate shopping, serviceability, pickup scheduling

24.1.1.1. [Subtask] - Rate shopping, serviceability, pickup scheduling

Description:

- Rate shopping
- Serviceability
- Pickup scheduling

24.2. [User Story] - Labels & Tracking

As a warehouse operator, I want easy label generation and tracking so that fulfillment is smooth.

Acceptance Criteria:

- Label generation and manifesting function reliably; reprints are supported.

- Tracking webhooks update orders and notify customers.
- Exceptions are visible in ops views.

24.2.1. [Task] - Implement label generation and manifesting

Description:

- Tracking webhooks → order updates and notifications

24.2.1.1. [Subtask] - Tracking webhooks → order updates and notifications

Description:

- Tracking webhooks → order updates
- Notifications

24.3. [User Story] - Exceptions Handling

As a support agent, I want exceptions workflows so that RTO/failed deliveries are handled promptly.

Acceptance Criteria:

- RTO/failed delivery flows create tickets and customer communications.
- Refund/Reship options are presented and tracked.
- SLAs are defined and monitored.

24.3.1. [Task] - Implement RTO/failed delivery flows and customer communications

Description:

- Automatic ticket creation and refund/Reship options

24.3.1.1. [Subtask] - Automatic ticket creation and refund/Reship options

Description:

- Automatic ticket creation
- Refund/Reship options

24.4. [User Story] - International Shipping Pilot

As a cross-border ops owner, I want to pilot international shipping so that we can expand sales safely.

Acceptance Criteria:

- Country/zone serviceability and carriers are configured with region restrictions.
- International address validation and postal verification are integrated.
- Cross-border returns flow and customer comms are defined.

24.4.1. [Task] - Configure country/zone serviceability and carriers

Description:

- Region-based SKU restrictions and lead times

24.4.1.1. [Subtask] - Region-based SKU restrictions and lead times

Description:

- Region-based SKU restrictions
- Lead times

24.4.2. [Task] - Implement address validation (international formats)

Description:

- Postal code and address verification APIs

24.4.2.1. [Subtask] - Postal code and address verification APIs

Description:

- Postal code
- Address verification APIs

24.4.3. [Task] - Define cross-border returns and CX

Description:

- RTO and refund SLAs; customer communications

24.4.3.1. [Subtask] - RTO and refund SLAs; customer communications

Description:

- RTO and refund SLAs
- Customer communications

24.5. [User Story] - Packaging & DIM Weight

As a warehouse operator, I want packaging rules so that rates reflect DIM weight accurately.

Acceptance Criteria:

- Packaging rules and DIM calculations are applied with auto-boxing.
- Rate selection considers dimensional weight vs actual weight.
- Exceptions are logged for tuning.

24.5.1. [Task] - Implement packaging rules and DIM calculations

Description:

- Auto-boxing and rate selection impact

24.5.1.1. [Subtask] - Auto-boxing and rate selection impact

Description:

- Auto-boxing
- Rate selection impact

24.6. [User Story] - NDR & Delivery Retries

As a support agent, I want NDR workflows so that delivery issues are resolved with minimal churn.

Acceptance Criteria:

- NDR workflows define retry attempts and RTO handling with notifications.
- Customer communications templates are localized and tracked.
- Outcomes feed into risk and carrier performance dashboards.

24.6.1. [Task] - Implement NDR workflows and customer communications

Description:

- Retry attempts; return-to-origin handling

24.6.1.1. [Subtask] - Retry attempts; return-to-origin handling

Description:

- Retry attempts
- Return-to-origin handling

24.7. [User Story] - COD Remittance Tracking

As a finance owner, I want COD remittance tracking so that settlements reconcile correctly.

Acceptance Criteria:

- Expected vs received amounts are tracked with aging reports.
- Discrepancies trigger alerts and follow-up tasks.
- Exports support reconciliation with bank statements.

24.7.1. [Task] - Implement tracking for COD settlements and reconciliation

Description:

- Expected vs received; aging reports

24.7.1.1. [Subtask] - Expected vs received; aging reports

Description:

- Expected vs received
- Aging reports

24.8. [User Story] - Aggregator Failover

As an operations owner, I want aggregator failover so that shipments continue during provider outages.

Acceptance Criteria:

- Health-based routing switches providers automatically with alerts.
- Credentials and service mappings are maintained for alternates.
- Post-failover reconciliation is documented.

24.8.1. [Task] - Configure provider failover rules

Description:

- Health-based routing and alerts

24.8.1.1. [Subtask] - Health-based routing and alerts

Description:

- Health-based routing
- Alerts

25. [Feature] - Media & CDN Pipeline

Description: Provide secure media uploads, derivatives, and global delivery with smart caching.

Acceptance Criteria:

- Signed uploads to S3; image variants generated via Lambda/CDN (AVIF/WebP).
- CloudFront caching strategy with invalidation hooks; cache keys vary by DPR/format.
- Safety pipeline scans uploads; quarantine and review workflow available.
- Access URLs are time-bound and least-privileged.

25.1. [User Story] - Uploads & Derivatives

As a content manager, I want secure uploads and responsive derivatives so that media looks great and loads fast.

Acceptance Criteria:

- Signed uploads to S3 use role-limited access with time-bound URLs.
- Image variants (AVIF/WebP) are generated and cached for thumb/PDP/zoom.

- Access is least-privileged; unauthorized access is blocked.

25.1.1. [Task] - Implement signed uploads to S3 and image variants (thumb, PDP, zoom)

Description:

- Implement image transformation via Lambda/Image CDN.
- Support AVIF/WebP responsive variants.

25.1.1.1. [Subtask] - Implement image transformation via Lambda/Image CDN; support AVIF/WebP

Description:

- Implement image transformation via Lambda/Image CDN.
- Support AVIF/WebP responsive variants.

25.2. [User Story] - Caching & Invalidations

As a performance engineer, I want effective CDN caching so that media loads quickly without stale content.

Acceptance Criteria:

- CloudFront cache keys vary by DPR/format; hit/miss metrics tracked.
- Invalidation hooks fire on publish/update for affected paths only.
- Tuning decisions are informed by observability dashboards.

25.2.1. [Task] - Implement CloudFront caching strategy and invalidation hooks

Description:

- Configure cache keys to vary by device DPR.
- Configure cache keys to vary by image format (AVIF/WebP/JPEG).

25.2.1.1. [Subtask] - Configure cache keys by device DPR and format

Description:

- Configure cache keys to vary by device DPR.
- Configure cache keys to vary by image format (AVIF/WebP/JPEG).

25.3. [User Story] - Safety Pipeline

As a compliance owner, I want a safety pipeline for uploads so that harmful content is blocked.

Acceptance Criteria:

- Malware/virus scan and content-type validation run on upload.
- Quarantine bucket and review workflow handle flagged assets.
- Moderation decisions are audited with reasons.

25.3.1. [Task] - Implement malware scan and content-type validation for uploads

Description:

- Create quarantine bucket for flagged assets.
- Implement review workflow for quarantined assets.

25.3.1.1. [Subtask] - Set up quarantine bucket and review workflow

Description:

- Create quarantine bucket for flagged assets.
- Implement review workflow for quarantined assets.

26. [Feature] - Performance & Load Testing

Description: Validate performance budgets and scale under load with synthetic checks and alerts.

Acceptance Criteria:

- k6/Artillery scenarios hit targets (p95, concurrencies) for critical journeys.
- Lighthouse budgets enforced in CI (LCP ≤ 2.5s P75; JS ≤ 200KB gz critical path).
- Synthetic monitoring covers uptime/transactions from multiple regions with alert thresholds.
- Performance regressions block releases until resolved or waived.

26.1. [User Story] - API Load Tests

As an SRE, I want API load tests so that hot paths scale to targets.

Acceptance Criteria:

- k6/Artillery scenarios cover browse, PDP, and checkout with thresholds.
- Scripts and pass/fail thresholds run in CI; reports are archived.
- Bottlenecks and recommendations are documented per run.

26.1.1. [Task] - Create k6/Artillery scenarios for hot paths (browse, PDP, checkout)

Description:

- Targets: p95 < 200ms cached reads; 5k concurrent users baseline

26.1.1.1. [Subtask] - Targets: p95 < 200ms cached reads; 5k concurrent users baseline

Description:

- Targets: p95 < 200ms cached reads
- 5k concurrent users baseline

26.2. [User Story] - Web Performance Budgets

As a frontend lead, I want performance budgets so that the site stays fast.

Acceptance Criteria:

- Lighthouse CI budgets are enforced; regressions trigger alerts.
- JS size budgets for the critical path are monitored in CI.
- PRs surface performance deltas for review.

26.2.1. [Task] - Enforce Lighthouse CI and budgets in CI

Description:

- LCP ≤ 2.5s P75; JS ≤ 200KB gz critical path

26.2.1.1. [Subtask] - LCP ≤ 2.5s P75; JS ≤ 200KB gz critical path

Description:

- LCP ≤ 2.5s P75
- JS ≤ 200KB gz critical path

26.3. [User Story] - Synthetic Monitoring

As an SRE, I want synthetic checks so that uptime and journeys are verified continuously.

Acceptance Criteria:

- Uptime and transaction scripts run from multiple regions with alert thresholds.
- On-call rotation integration pages the right team.
- Incidents auto-create tickets with context for triage.

26.3.1. [Task] - Configure uptime and transaction checks from multiple regions

Description:

- Alert thresholds and on-call rotation

26.3.1.1. [Subtask] - Alert thresholds and on-call rotation

Description:

- Alert thresholds
- On-call rotation

27. [Feature] - Mobile Crash & Performance Analytics

Description: Monitor app stability and performance to ensure store-readiness and user experience.

Acceptance Criteria:

- Crashlytics/Sentry integrated with symbol uploads and release versions.
- Cold start, TTI, memory, and ANR metrics tracked with dashboards and alerts.
- Crash-free sessions > 99.5% maintained; regression alarms configured.
- Triage and ownership for crash groups defined.

27.1. [User Story] - Crash Reporting

As a mobile developer, I want crash reporting so that I can fix stability issues quickly.

Acceptance Criteria:

- Sentry/Crashlytics integrated; dSYM/ProGuard mappings uploaded; release tagging enabled.

- Crash groups are assigned and triaged with ownership.
- Crash-free sessions > 99.5% maintained.

27.1.1. [Task] - Integrate Sentry/Crashlytics for iOS/Android

Description:

- Symbol upload (dSYM/ProGuard mappings); release tagging

27.1.1.1. [Subtask] - Symbol upload (dSYM/ProGuard mappings); release tagging

Description:

- Symbol upload (dSYM/ProGuard mappings)
- Release tagging

27.2. [User Story] - App Performance

As a mobile PM, I want performance metrics so that app experience remains smooth.

Acceptance Criteria:

- Cold start, TTI, memory, and ANR metrics have dashboards and alerts.
- Regression alarms are configured; thresholds are documented.
- Remediation work is tracked to completion.

27.2.1. [Task] - Track cold start, TTI, memory, and ANR

Description:

- Dashboards and alerts for regressions

27.2.1.1. [Subtask] - Dashboards and alerts for regressions

Description:

- Dashboards
- Alerts for regressions

28. [Feature] - Creator/Influencer Marketplace

Description: Manage creator listings, briefs, deliverables, and compliance with attribution.

Acceptance Criteria:

- Creator profiles/listings and campaign briefs support availability and rates.
- Deliverables tracked with attribution parameters; analytics events emitted.
- Contract terms, approvals, and compliance checks enforced with audit trails.
- Dispute workflow documented with service-level targets.

28.1. [User Story] - Listings & Briefs

As a marketer, I want creator listings and briefs so that campaigns are predictable and efficient.

Acceptance Criteria:

- Profiles/listings support availability, rates, and content formats.
- Briefs define deliverables, timelines, and approvals.
- Changes are tracked with an audit trail.

28.1.1. [Task] - Build creator profiles, listings, and campaign briefs

Description:

- Availability, rates, and supported content formats

28.1.1.1. [Subtask] - Availability, rates, and supported content formats

Description:

- Availability
- Rates
- Supported content formats

28.2. [User Story] - Deliverables & Attribution

As a marketer, I want deliverables tracked with attribution so that performance is measurable.

Acceptance Criteria:

- Deliverables link to content with attribution parameters.
- Analytics events emit for clicks/views; reporting is available.
- SLA breaches are flagged for follow-up.

28.2.1. [Task] - Track deliverables, links, and performance

Description:

- Attribution parameters and analytics events

28.2.1.1. [Subtask] - Attribution parameters and analytics events

Description:

- Attribution parameters
- Analytics events

28.3. [User Story] - Reviews & Compliance

As a compliance owner, I want contract and policy checks so that content stays on-brand and legal.

Acceptance Criteria:

- Contract terms, approvals, and policy checks are enforced and recorded.
- Audit trail and dispute workflow are defined.
- Takedown process is documented and exercised.

28.3.1. [Task] - Implement contract terms, approvals, and policy checks

Description:

- Audit trail and dispute workflow

28.3.1.1. [Subtask] - Audit trail and dispute workflow

Description:

- Audit trail
- Dispute workflow

29. [Feature] - Community & Gamification

Description: Build community engagement and retention through forums and gamified progression.

Acceptance Criteria:

- Topics/threads/comments/reactions shipped with moderation tools and reports.
- Points, badges, levels, and streaks granted by rules engine with abuse checks.
- Community notifications delivered; role-based moderation available.
- Escalation workflows measured with CSAT.

29.1. [User Story] - Forums & Threads

As a community member, I want forums to discuss so that I can connect and learn.

Acceptance Criteria:

- Topics/threads/comments/reactions exist with moderation tools.
- Abuse checks and reporting with rate limits protect the community.
- Notifications for mentions and replies are delivered.

29.1.1. [Task] - Build topics, threads, comments, and reactions

Description:

- Moderation tools and community reports

29.1.1.1. [Subtask] - Moderation tools and community reports

Description:

- Moderation tools
- Community reports

29.2. [User Story] - Gamification

As a community member, I want points and badges so that engagement is rewarded.

Acceptance Criteria:

- Points, badges, levels, and streaks are granted by a rules engine.
- Anti-abuse checks, caps, and cooldowns prevent gaming the system.
- Profiles show progress; users can opt out.

29.2.1. [Task] - Implement points, badges, levels, and streaks

Description:

- Rules engine and abuse checks

29.2.1.1. [Subtask] - Rules engine and abuse checks

Description:

- Rules engine
- Abuse checks

29.3. [User Story] - Notifications & Moderation

As a moderator, I want community notifications and tools so that I can manage at scale.

Acceptance Criteria:

- Role-based moderation actions and escalation workflows exist.
- Community notifications are delivered with user preferences.
- CSAT on resolutions is measured.

29.3.1. [Task] - Implement community notifications and role-based moderation

Description:

- Escalation workflows; CSAT on resolutions

29.3.1.1. [Subtask] - Escalation workflows; CSAT on resolutions

Description:

- Escalation workflows

- CSAT on resolutions

30. [Feature] - Internationalization & Localization

Description: Prepare the platform for multilingual content and locale-aware rendering.

Acceptance Criteria:

- Locale negotiation on server; consistent message/date/number formatting.
- Translation pipeline supports export/import and fallbacks; missing-keys monitored.
- SEO signals (hreflang/canonicals) compatible with localized pages.
- Key flows localized to target languages as prioritized.

30.1. [User Story] - Service-wide i18n Scaffolding

As a global user, I want localized content so that the site feels native.

Acceptance Criteria:

- Server-side locale negotiation and message/date/number formatting work consistently.
- Missing-key monitoring and fallback behavior are defined.
- Localized routes and SEO signals are honored.

30.1.1. [Task] - Implement locale-aware messages, dates, and numbers

Description:

- Server-side locale negotiation and formatting

30.1.1.1. [Subtask] - Server-side locale negotiation and formatting

Description:

- Server-side locale negotiation
- Formatting

30.2. [User Story] - Content & Translation Pipeline

As a content manager, I want a translation pipeline so that content ships in multiple languages.

Acceptance Criteria:

- Export/import tooling and translation status tracking exist.
- Fallbacks are configured; missing keys are monitored.
- QA checks validate translation quality.

30.2.1. [Task] - Implement translation management and fallbacks

Description:

- Missing-key monitoring; export/import tooling

30.2.1.1. [Subtask] - Missing-key monitoring; export/import tooling

Description:

- Missing-key monitoring
- Export/import tooling

31. [Feature] - Identity Enhancements

Description: Strengthen identity with MFA, SSO, passwordless, and device trust for higher security.

Acceptance Criteria:

- TOTP/SMS MFA enrollment and recovery codes; enforced for staff, optional for customers.
- SSO for staff (SAML/OIDC) with JIT provisioning and role mapping.
- Passwordless login via OTP/magic link; biometric unlock on mobile where supported.
- Step-up re-auth for sensitive actions; anomalous session detection and device trust.

31.1. [User Story] - Multi-Factor Authentication (MFA)

As a staff user, I want MFA so that my account is protected.

Acceptance Criteria:

- TOTP/SMS enrollment with recovery codes is available; enforced for staff.
- Step-up re-auth is required for sensitive actions.
- Backup/disable flows exist with audit.

31.1.1. [Task] - Add TOTP/SMS MFA for customers; enforce for staff

Description:

- Enrollment, recovery codes, step-up flows

31.1.1.1. [Subtask] - Enrollment, recovery codes, step-up flows

Description:

- Enrollment
- Recovery codes
- Step-up flows

31.2. [User Story] - Staff SSO (Google/O365)

As a staff user, I want SSO so that I can log in with corporate identity.

Acceptance Criteria:

- SAML/OIDC with JIT provisioning and role mapping works.
- Optional SCIM sync updates roles; logout/invalidate flows exist.
- Audit and alerts monitor login anomalies.

31.2.1. [Task] - Implement SAML/OIDC SSO with JIT provisioning

Description:

- Role mapping to RBAC; optional SCIM sync

31.2.1.1. [Subtask] - Role mapping to RBAC; optional SCIM sync

Description:

- Role mapping to RBAC
- Optional SCIM sync

31.3. [User Story] - Passwordless & Device Trust

As a customer, I want passwordless and trusted devices so that sign-in is easy and safe.

Acceptance Criteria:

- OTP/magic link flows exist; mobile biometric unlock where supported.
- Trusted devices and anomalous session detection implemented.
- Step-up re-auth required for sensitive actions.

31.3.1. [Task] - Implement OTP/email magic link and mobile biometric unlock

Description:

- Trusted devices, anomalous session detection; re-auth for sensitive actions

31.3.1.1. [Subtask] - Trusted devices, anomalous session detection; re-auth for sensitive actions

Description:

- Trusted devices, anomalous session detection
- Re-auth for sensitive actions

32. [Feature] - Wishlists, Alerts & Recovery

Description: Improve retention with wishlists, stock/price alerts, and abandoned recovery journeys.

Acceptance Criteria:

- Wishlist and recently viewed work for guest and auth users; merge-on-login supported.
- Back-in-stock and price-drop alerts configurable with frequency caps and consent.
- Abandoned cart/checkout journeys trigger across channels with guardrails.
- Event attribution tracked to measure uplift and suppression.

32.1. [User Story] - Wishlist & Recently Viewed (Web)

As a shopper, I want wishlists and history so that I can revisit items.

Acceptance Criteria:

- Guest and auth modes supported with merge-on-login.
- Recently viewed shows last items with privacy controls.
- API and UI meet performance targets.

32.1.1. [Task] - Implement API and UI for wishlist and recently viewed

Description:

- Guest + auth modes; merge-on-login

32.1.1.1. [Subtask] - Guest + auth modes; merge-on-login

Description:

- Guest + auth modes
- Merge-on-login

32.2. [User Story] - Back-in-Stock & Price-Drop Alerts

As a shopper, I want alerts so that I don’t miss desired products.

Acceptance Criteria:

- Subscriptions and routing exist with frequency caps; consent integrated.
- Opt-out and pause controls available to users.
- Attribution of lift is measured.

32.2.1. [Task] - Implement subscriptions and notifications routing

Description:

- Frequency caps; consent/preferences integration

32.2.1.1. [Subtask] - Frequency caps; consent/preferences integration

Description:

- Frequency caps

- Consent/preferences integration

32.3. [User Story] - Abandoned Cart/Checkout Recovery

As a marketer, I want recovery journeys so that abandoned orders convert.

Acceptance Criteria:

- Triggers across email/SMS/push operate with guardrails.
- Suppression rules for recency and discount limits applied.
- Attribution is tracked in analytics.

32.3.1. [Task] - Implement triggered journeys via email/SMS/push

Description:

- Guardrails: frequency, recency, discount limits

32.3.1.1. [Subtask] - Guardrails: frequency, recency, discount limits

Description:

- Frequency
- Recency
- Discount limits

33. [Feature] - Payments Advanced

Description: Expand payment method coverage and robustness while keeping PCI scope minimal.

Acceptance Criteria:

- UPI, NetBanking, cards, and COD options available with regional toggles.
- Tokenized saved methods via provider vault; SAQ A scope confirmed.
- Partial capture, split refunds, and retry strategies implemented; subscription dunning configured.
- Payment success/failure monitored with alerts and dashboards.

33.1. [User Story] - India Payment Methods Coverage

As a shopper, I want familiar payment options so that I can pay how I prefer.

Acceptance Criteria:

- UPI, NetBanking, cards, and COD options are available with regional toggles.
- Fraud guardrails exist for COD; provider fallback is documented.
- UX shows applicable fees or limitations clearly.

33.1.1. [Task] - Implement UPI, NetBanking, card, and COD toggles

Description:

- COD fraud guardrails; provider fallback

33.1.1.1. [Subtask] - COD fraud guardrails; provider fallback

Description:

- COD fraud guardrails
- Provider fallback

33.2. [User Story] - Saved Methods & PCI Scope

As a customer, I want to save payment methods safely so that future checkout is faster.

Acceptance Criteria:

- Tokenized storage via provider vault keeps us in SAQ A scope.
- Add/remove flows and default method management exist.
- Clear error states and recovery paths are presented.

33.2.1. [Task] - Implement tokenized storage via provider vault

Description:

- SAQ A scope; vault lifecycle (add/remove)

33.2.1.1. [Subtask] - SAQ A scope; vault lifecycle (add/remove)

Description:

- SAQ A scope
- Vault lifecycle (add/remove)

33.3. [User Story] - Capture, Refunds, Retries

As a finance owner, I want robust capture, refunds, and retries so that payment operations are correct.

Acceptance Criteria:

- Partial capture and split refunds implemented with audit.
- Retry strategies configured with idempotency protections.
- Subscription dunning logic and notifications implemented.

33.3.1. [Task] - Implement partial capture, split refunds, and payment retry

Description:

- Subscription dunning logic and notifications

33.3.1.1. [Subtask] - Subscription dunning logic and notifications

Description:

- Subscription dunning logic
- Notifications

34. [Feature] - Marketing & CRM

Description: Enable compliant messaging, segmentation, and attribution for lifecycle marketing.

Acceptance Criteria:

- SPF/DKIM/DMARC configured; DMARC reports monitored; sender warmup plan executed.
- ESP/CRM integrated; segments and consent states synced bidirectionally.
- GTM and server-side tagging honor consent; PII redaction applied.
- Post-purchase automations (reviews/winback/replenishment) deliver with holdouts.

34.1. [User Story] - Deliverability & Domain Auth

As an email operations owner, I want authenticated domains so that emails deliver reliably.

Acceptance Criteria:

- SPF/DKIM/DMARC are configured; DMARC reports monitored; warmup plan executed.
- BIMI (optional) considered; deliverability metrics tracked.
- Policy docs maintained for send practices.

34.1.1. [Task] - Configure SPF, DKIM, DMARC (+BIMI optional)

Description:

- DMARC reporting and warmup plan

34.1.1.1. [Subtask] - DMARC reporting and warmup plan

Description:

- DMARC reporting
- Warmup plan

34.2. [User Story] - CRM/ESP Integration

As a marketer, I want CRM/ESP integrated so that segments and campaigns run with fresh data.

Acceptance Criteria:

- Profiles/events sync bidirectionally; segments and consent states align.
- List hygiene (bounces/unsubscribes) enforced automatically.
- Sync errors are retried and surfaced for resolution.

34.2.1. [Task] - Implement Klaviyo/Mailchimp sync for profiles/events

Description:

- Segments, list hygiene, consent states

34.2.1.1. [Subtask] - Segments, list hygiene, consent states

Description:

- Segments
- List hygiene
- Consent states

34.3. [User Story] - Tag Management & Server-Side Tagging

As a data privacy owner, I want consent-gated tag management so that tracking respects user choices.

Acceptance Criteria:

- GTM setup gates pixels by consent; server-side tagging explored for key tags.
- PII redaction is enforced; tag changes are audited.
- Tag catalog and ownership are documented.

34.3.1. [Task] - Configure GTM and evaluate server-side tagging options

Description:

- Consent-gated pixels; PII redaction

34.3.1.1. [Subtask] - Consent-gated pixels; PII redaction

Description:

- Consent-gated pixels
- PII redaction

34.4. [User Story] - Post-Purchase Automations

As a marketer, I want automations for reviews, winback, and replenishment so that retention improves.

Acceptance Criteria:

- Journeys are configured with holdouts and attribution tagging.
- Frequency caps and suppression rules are enforced.
- Reporting shows uplift and ROI.

34.4.1. [Task] - Implement review requests, winback, and replenishment

Description:

- Holdouts and attribution tagging

34.4.1.1. [Subtask] - Holdouts and attribution tagging

Description:

- Holdouts
- Attribution tagging

34.5. [User Story] - UTM Governance

As a marketer, I want UTM governance so that acquisition reporting is clean.

Acceptance Criteria:

- UTM validation and mapping rules exist; auto-tagging applies where possible.
- PII stripping is applied to inbound links.
- Mis-tag detection alerts are configured.

34.5.1. [Task] - Implement UTM validation and mapping

Description:

- Auto-tagging rules; PII stripping

34.5.1.1. [Subtask] - Auto-tagging rules; PII stripping

Description:

- Auto-tagging rules

- PII stripping

34.6. [User Story] - WhatsApp CRM Integration

As a marketer, I want WhatsApp CRM sync so that profiles and events are consistent across channels.

Acceptance Criteria:

- Profiles and events sync with consent/opt-out handling.
- Channel routing and de-duplication rules enforced.
- Errors are retried and surfaced to ops.

34.6.1. [Task] - Implement syncing of profiles and events

Description:

- Consent sync and opt-out handling (integrates with 4.2.1)

34.6.1.1. [Subtask] - Consent sync and opt-out handling (integrates with 4.2.1)

Description:

- Consent sync
- Opt-out handling (integrates with 4.2.1)

35. [Feature] - Eventing & Tracing

Description: Ensure reliable domain event delivery and end-to-end observability.

Acceptance Criteria:

- Outbox pattern implemented with SNS/SQS; retries and DLQs configured.
- OpenTelemetry tracing across API/web/jobs with propagation headers.
- Web error tracking (Sentry) includes release tagging and source maps.
- Log scrubbing removes PII; retention policies enforced.

35.1. [User Story] - Domain Events & Outbox

As a backend engineer, I want reliable domain events so that downstream systems stay in sync.

Acceptance Criteria:

- Outbox pattern with SNS/SQS is implemented with retries and DLQs.
- Idempotency protects against duplicate processing.
- Monitoring covers lag, retries, and dead letters.

35.1.1. [Task] - Implement outbox pattern and SNS/SQS backbone

Description:

- Idempotency, retries, and DLQs

35.1.1.1. [Subtask] - Idempotency, retries, and DLQs

Description:

- Idempotency
- Retries
- DLQs

35.2. [User Story] - Distributed Tracing

As an SRE, I want distributed tracing so that I can debug cross-service issues.

Acceptance Criteria:

- OpenTelemetry spans exist across API/web/jobs with propagation headers.
- Sampling policy is set; traces visible in APM.
- Key spans instrumented for hot paths.

35.2.1. [Task] - Instrument OpenTelemetry across API/web/jobs

Description:

- Trace propagation headers; sampling policy

35.2.1.1. [Subtask] - Trace propagation headers; sampling policy

Description:

- Trace propagation headers
- Sampling policy

35.3. [User Story] - Web Error Tracking

As a frontend engineer, I want web error tracking so that client errors are visible and actionable.

Acceptance Criteria:

- Sentry instrumentation includes release tagging and source maps.
- PII scrubbing is applied to events.
- Alerting thresholds are configured by severity.

35.3.1. [Task] - Add Sentry instrumentation for web

Description:

- Release tagging and source maps

35.3.1.1. [Subtask] - Release tagging and source maps

Description:

- Release tagging
- Source maps

35.4. [User Story] - Log Scrubbing & Retention

As a security owner, I want scrubbing and retention so that logs are safe and cost-effective.

Acceptance Criteria:

- PII redaction filters applied; access controls enforced.
- Retention policies set by log type; exports controlled.
- Access and changes are audited.

35.4.1. [Task] - Implement PII redaction filters and retention policies

Description:

- Access controls and audit trail

35.4.1.1. [Subtask] - Access controls and audit trail

Description:

- Access controls
- Audit trail

36. [Feature] - Legal & Compliance

Description: Publish policies, handle vuln disclosures, and implement retention aligned to regulation.

Acceptance Criteria:

- Terms/Privacy/Returns/Shipping pages managed via CMS with version history.
- /.well-known/security.txt published; vulnerability disclosure intake defined with SLAs.
- E-invoicing/e-way bill readiness assessed; sandbox integrations validated if applicable.
- Data retention schedule executed with automated purges and legal holds.

36.1. [User Story] - Policy Pages & Disclosure

As a legal/compliance owner, I want managed policy pages and disclosure so that users are informed and researchers can report vulns.

Acceptance Criteria:

- Terms/Privacy/Returns/Shipping pages are managed via CMS with version history.
- /.well-known/security.txt is published; VDP intake defined with SLAs.
- Change approvals and edits have an audit trail.

36.1.1. [Task] - Publish Terms, Privacy, Returns, and Shipping via CMS

Description:

- Footer links, versioning, approval workflow

36.1.1.1. [Subtask] - Footer links, versioning, approval workflow

Description:

- Footer links
- Versioning

- Approval workflow

36.2. [User Story] - Security.txt & VDP

As a security owner, I want a VDP so that vulnerabilities are reported responsibly.

Acceptance Criteria:

- security.txt is published with intake instructions; optional public key provided.
- Triage SLAs and secure inbox are set up.
- Reports are tracked to closure.

36.2.1. [Task] - Publish /.well-known/security.txt and define intake process

Description:

- Triage SLAs; optional public key

36.2.1.1. [Subtask] - Triage SLAs; optional public key

Description:

- Triage SLAs
- Optional public key

36.3. [User Story] - India E-Invoicing/E-Way Bill Readiness

As a finance owner, I want e-invoicing readiness so that we can comply as thresholds are met.

Acceptance Criteria:

- Threshold evaluation is documented; provider integrations sandbox-tested.
- Fallback procedures are defined for outages.

- Monitoring of thresholds and compliance status exists.

36.3.1. [Task] - Implement threshold evaluation and provider integration

Description:

- Sandbox tests and fallback procedures

36.3.1.1. [Subtask] - Sandbox tests and fallback procedures

Description:

- Sandbox tests
- Fallback procedures

36.4. [User Story] - Data Retention & Purges

As a privacy owner, I want retention schedules so that data is purged appropriately.

Acceptance Criteria:

- Retention schedule implemented with automated purge jobs.
- Legal holds are supported with overrides.
- Reporting on deletions is available to compliance.

36.4.1. [Task] - Implement retention schedule and automated purge jobs

Description:

- Legal holds and reporting

36.4.1.1. [Subtask] - Legal holds and reporting

Description:

- Legal holds
- Reporting

37. [Feature] - Quality Gates & Visual Testing

Description: Prevent regressions via accessibility, visual, and contract testing with release gates.

Acceptance Criteria:

- Accessibility audits (manual + automated) pass on critical flows.
- Visual regression suite runs in CI; baseline management documented.
- Pact contract tests verify client/server agreements.
- UAT sign-off checklist and rollback plan enforced before release.

37.1. [User Story] - Accessibility Audits

As an accessibility lead, I want audits so that the app meets WCAG.

Acceptance Criteria:

- Manual and automated checks pass on critical flows.
- Screen reader and keyboard-only flows are validated.
- Issues are tracked and fixed before release.

37.1.1. [Task] - Run manual and automated (axe/Pa11y) accessibility tests

Description:

- Screen reader checks; keyboard-only flows

37.1.1.1. [Subtask] - Screen reader checks; keyboard-only flows

Description:

- Screen reader checks
- Keyboard-only flows

37.2. [User Story] - Visual Regression Testing

As a frontend lead, I want visual testing so that UI changes don’t introduce regressions.

Acceptance Criteria:

- Storybook/Chromatic or Playwright snapshots are configured.
- Baseline management is documented; PR checks are mandatory.
- Visual diffs are reviewed and approved.

37.2.1. [Task] - Set up Storybook/Chromatic or Playwright snapshots

Description:

- Baseline management and PR checks

37.2.1.1. [Subtask] - Baseline management and PR checks

Description:

- Baseline management
- PR checks

37.3. [User Story] - Contract Testing

As a platform engineer, I want contract tests so that client/server integration remains stable.

Acceptance Criteria:

- Pact provider/consumer tests exist with a contracts registry.
- CI verification pipeline enforces compatibility.
- Backward compatibility checks are in place.

37.3.1. [Task] - Implement Pact provider/consumer tests

Description:

- CI verification and contracts registry

37.3.1.1. [Subtask] - CI verification and contracts registry

Description:

- CI verification
- Contracts registry

37.4. [User Story] - Release/UAT Gates

As a release manager, I want UAT gates so that releases roll out safely.

Acceptance Criteria:

- UAT checklist and sign-off criteria are defined and used.
- Rollback plan is verified for each release.
- Blocking conditions are enforced in CI/CD.

37.4.1. [Task] - Define UAT checklist and sign-off criteria

Description:

- Verified rollback plan

37.4.1.1. [Subtask] - Verified rollback plan

Description:

- Verified rollback plan

38. [Feature] - Catalog Extensions

Description: Enhance catalog with bundles, advanced inventory, PDP Q&A;, and compliant pricing.

Acceptance Criteria:

- Bundles/kits priced and purchasable; cross-sell/upsell placements configured.
- Multi-warehouse/backorder/preorder flows modeled with reservations and lead times.
- PDP Q&A; available with moderation and notifications.
- India pricing fields (MRP, compare-at, GST labels) displayed correctly.

38.1. [User Story] - Bundles/Kits & Merchandising

As a merchandiser, I want bundles so that I can promote value combos.

Acceptance Criteria:

- Bundle SKUs and pricing rules work in cart and checkout.
- Cross-sell and upsell placements are configured.
- Analytics track attach rates and performance.

38.1.1. [Task] - Implement bundle SKUs and pricing rules

Description:

- Cross-sell/upsell placements

38.1.1.1. [Subtask] - Cross-sell/upsell placements

Description:

- Cross-sell/upsell placements

38.2. [User Story] - Inventory Models

As an operations manager, I want advanced inventory models so that preorders and backorders are handled correctly.

Acceptance Criteria:

- Multi-warehouse, backorder, and preorder flows include reservations and lead times.
- UI indicates availability dates for preorders.
- Alerts and exceptions are surfaced to ops.

38.2.1. [Task] - Implement multi-warehouse, backorder, and preorder

Description:

- Reservations and lead times

38.2.1.1. [Subtask] - Reservations and lead times

Description:

- Reservations
- Lead times

38.3. [User Story] - PDP Q&A;

As a shopper, I want Q&A; on PDP so that I can ask and learn from others.

Acceptance Criteria:

- Questions support moderation and notifications for answers.
- Merchant answers are highlighted and trustworthy.
- Abuse controls and rate limits apply.

38.3.1. [Task] - Implement customer questions and moderation

Description:

- Notifications and reporting

38.3.1.1. [Subtask] - Notifications and reporting

Description:

- Notifications
- Reporting

38.4. [User Story] - India Pricing Display

As a shopper, I want correct India pricing labels so that expectations are clear.

Acceptance Criteria:

- MRP, compare-at, and GST inclusive labels display per policy.
- UX toggles and SEO tags are correct.
- Tests verify presentation.

38.4.1. [Task] - Implement MRP, compare-at, and GST inclusive labels

Description:

- UX toggles and SEO tags

38.4.1.1. [Subtask] - UX toggles and SEO tags

Description:

- UX toggles
- SEO tags

39. [Feature] - Mobile Store Compliance

Description: Meet iOS/Android store policies for privacy, ratings, and versioning.

Acceptance Criteria:

- iOS ATT prompt implemented; Google Play Data Safety declarations accurate.
- In-app review prompts deep link to stores with frequency caps.
- Version gating supports force-upgrade and soft prompts; remote kill switches available.
- Compliance items reviewed before every release.

39.1. [User Story] - Privacy Disclosures

As a mobile user, I want clear privacy disclosures so that I understand data use.

Acceptance Criteria:

- iOS ATT prompt implemented; Play Data Safety declarations are accurate.
- Consent logging and audit exist.
- Onboarding messaging is consistent across platforms.

39.1.1. [Task] - Implement iOS ATT and Play Data Safety

Description:

- Consent logging and audit

39.1.1.1. [Subtask] - Consent logging and audit

Description:

- Consent logging
- Audit

39.2. [User Story] - Ratings & Reviews

As a mobile user, I want to rate the app so that feedback is shared.

Acceptance Criteria:

- In-app review prompts and deep links honor frequency caps and locales.
- Prompts avoid poor-experience moments (e.g., after crashes).
- Metrics for ratings are tracked over time.

39.2.1. [Task] - Implement in-app review prompts and deep links

Description:

- Frequency caps and locales

39.2.1.1. [Subtask] - Frequency caps and locales

Description:

- Frequency caps
- Locales

39.3. [User Story] - Version Gating

As a mobile user, I want version gating so that I’m on a compatible, secure version.

Acceptance Criteria:

- Force-upgrade and soft prompts are controlled via remote config.
- Kill switches exist for critical issues.
- Compatibility matrix is documented and enforced.

39.3.1. [Task] - Implement force-upgrade and soft prompts

Description:

- Remote config and kill switches

39.3.1.1. [Subtask] - Remote config and kill switches

Description:

- Remote config
- Kill switches

40. [Feature] - Multi-Currency Pricing

Description: Support FX-driven price display and SEO for international audiences.

Acceptance Criteria:

- FX rates fetched with defined cadence and fallback provider; rounding rules applied.
- Prices render in selected currency consistently across web and mobile.

- SEO canonicals and hreflang configured for price variants where applicable.
- Automated tests verify price math and formatting.

40.1. [User Story] - FX and Rounding

As an international shopper, I want prices in my currency so that I can evaluate cost.

Acceptance Criteria:

- FX rates are fetched with fallback; rounding rules applied.
- Display is consistent across web and mobile.
- Automated tests verify math and formatting.

40.1.1. [Task] - Implement currency rates and rounding rules

Description:

- Cadence, provider fallback

40.1.1.1. [Subtask] - Cadence, provider fallback

Description:

- Cadence
- Provider fallback

40.2. [User Story] - Price Presentation

As an international shopper, I want clear price presentation so that SEO and UX remain correct.

Acceptance Criteria:

- Multi-currency UI and SEO tags (canonicals/hreflang) are correct.

- Localized currency symbols and formats are applied.
- Performance is unaffected by currency switching.

40.2.1. [Task] - Implement multi-currency UI and SEO tags

Description:

- Canonicals and hreflang updates

40.2.1.1. [Subtask] - Canonicals and hreflang updates

Description:

- Canonicals
- Hreflang updates

41. [Feature] - Ethical Sourcing & Provenance

Description: Provide transparent provenance via supplier KYC, certifications, lot-level traceability, and PDP trust signals.

Acceptance Criteria:

- Supplier onboarding with KYC and document verification; expiries tracked with alerts.
- Lot-level chain-of-custody linked to SKUs; PDP shows verified badges and sourcing info.
- Partner profile pages published with attribution; audit trails recorded for changes.
- Compliance checks and audit schedules tracked to closure.

41.1. [User Story] - Supplier Onboarding & KYC

As a procurement manager, I want supplier KYC so that partners are verified.

Acceptance Criteria:

- Collect KYC and certifications; expiries tracked with reminders.
- Upload/store docs securely; access controlled.
- Audit logs maintained for changes.

41.1.1. [Task] - Collect KYC and certifications

Description:

- Upload/store docs; expiry reminders

41.1.1.1. [Subtask] - Upload/store docs; expiry reminders

Description:

- Upload/store docs
- Expiry reminders

41.2. [User Story] - Lot-Level Traceability

As a compliance owner, I want lot-level traceability so that provenance is transparent.

Acceptance Criteria:

- Lots link to inventory and orders; chain-of-custody is recorded.
- Scan/import flows are supported for operations.
- Reports are exportable for audits.

41.2.1. [Task] - Link lots to inventory and orders

Description:

- Scan/import lots; chain-of-custody record

41.2.1.1. [Subtask] - Scan/import lots; chain-of-custody record

Description:

- Scan/import lots
- Chain-of-custody record

41.3. [User Story] - PDP Trust & Partner Pages

As a shopper, I want trust signals so that I can buy confidently.

Acceptance Criteria:

- Verified badges and sourcing info are displayed on PDP.
- Partner pages include certificates and stories.
- Changes are moderated and audited.

41.3.1. [Task] - Display verified badges and sourcing

Description:

- Partner pages with certificates and stories

41.3.1.1. [Subtask] - Partner pages with certificates and stories

Description:

- Partner pages with certificates
- Stories

41.4. [User Story] - Audits & Compliance

As a compliance owner, I want audits tracked so that issues are remediated.

Acceptance Criteria:

- Audit scheduling and outcomes are recorded with SLAs.
- Findings and remediation are tracked; transparency notes maintained.
- Status dashboards are available to stakeholders.

41.4.1. [Task] - Track audit scheduling and outcomes

Description:

- Findings, remediation, and public transparency notes

41.4.1.1. [Subtask] - Findings, remediation, and public transparency notes

Description:

- Findings
- Remediation
- Public transparency notes

Version History

- v1.0 (2023-10-14) — Finalized: added status header, anchors, table of contents, and clarified success criteria wording to align with project.md conventions.