Backend Technical Spec – Spring Boot + Bitbucket + DigitalOcean (with Local SMS OTP)

1) Scope & Architecture

Stack: Java 21, Spring Boot 3.3+, PostgreSQL (DigitalOcean Managed), Redis (Managed), Quartz (JDBC store),
springdoc-openapi, Spring Security 6 (JWT), Flyway, Micrometer/Prometheus, Sentry, DigitalOcean Spaces (S3), FCM.

Modules (monolith-first): auth, farms, animals, breeding, health, feeding, milking, sales, inventory, insights, reports,
notifications, sync, common, infra. Data is per-farm scoped.

2) Environments & Infra on DigitalOcean

Environments: dev, staging, prod. Deploy as a container on DO App Platform (initial) with a managed Postgres & Redis.
Private networking via VPC. DO Spaces for files (private bucket + pre-signed URLs). Cron-like via Quartz clustered jobs.

3) Security & Multi-Farm Scoping

Auth: phone OTP → JWT (access/refresh). Every request includes X-Farm-Id; a filter asserts membership before
controller logic. All aggregates carry farmId. Use optimistic locking (@Version) where necessary.
PII: Consider pgcrypto for phone column. Enforce HTTPS. Use presigned URLs for object storage uploads.

4) Local SMS OTP Design

OTPService: requestOtp(phone,purpose), verifyOtp(phone,code). Hash OTP at rest (Argon2id/bcrypt) in Redis with TTL.
Rate limits via Redis token bucket (per phone, per IP). Generic responses to prevent enumeration.

Redis key: otp:{phone}:{purpose} -> {hash, attempts, expiresAt, resendAfter}. TTL 5–10 minutes.
Resend: after 45–60s, max 2 resends per window. Lock after 5 failed attempts until TTL expiry.

SmsGatewayClient: HTTP or SMPP adapter to local provider. Circuit breaker + retries (network errors only). Optional DLR webhook /sms/dlr.
Arabic template: "[illegible]: {123456}. [illegible] {5} [illegible]. [illegible]."

5) Offline Sync Contract

Pull: GET /farms/{farmId}/sync/pull?since=ISO returns changed entities and tombstones.
Push: POST /farms/{farmId}/sync/push receives batched outbox changes; returns id mappings and server snapshots on conflict.

Conflict policy: Last-write-wins using updatedAt; return 409 with latest snapshot when client stale.
Tombstones: tombstone(entity, entityId, deletedAt). Client cleans up on receipt.

6) Domain Highlights & Entities

Core tables (high level): User, Farm, FarmMember(role); Animal; BreedingCycle + EventLedger; HealthEvent, Treatment, Dose, WithdrawalTrack;
Weight, FeedPlan, FeedUsage, LambFeeding; MilkYield, MilkSale, SaleAnimal; Item, Batch, Grn, Usage (optional inventory);
Reminder, NotificationInbox; MetricSnapshot, ActionEvent, InsightCard; Tombstone.

Design pattern: write detailed events to EventLedger (auditable), maintain current state in aggregate rows for fast reads.

7) Best-Effort Costing (Inventory Optional)

Cost source hierarchy: Batch/FIFO → Last Known Price → Catalog default → Derived (unit) → Fuzzy name match → Reference price → Zero.
Persist costValue, costSource, confidence. Nightly backfill upgrades confidence when new GRNs appear.

8) Reminders, Notifications & Jobs

Quartz (JDBC store, clustered-safe). Jobs:
- reminders:generate (on domain events) and reminders:dispatch (*/10m)
- etl:metric_snapshots (00:05 UTC)
- insights:impact_scan (00:45 UTC)
- costing:backfill (nightly)

Notifications: domain event → queue → FCM push + in-app inbox. Per-member subscriptions (bitmask/JSON) by event type.

9) Insights & Causality (MVP)

Nightly ETL populates MetricSnapshot per farm/group. Methods: pre/post uplift, difference-in-differences,
change-point detection, lagged cross-correlation. Output: InsightCard with effect size (%), lag, window, confidence (High/Medium/Low).

10) Exports/Imports

Exports: CSV/XLSX for Animals, Events (breeding/health/feeding/weights), Sales (animals/milk), KPIs, Costs (with confidence).
Imports: CSV/XLSX templates; preview screen with row-level status; idempotency using (farmId, externalRowId).

11) API Surface (MVP)

/auth/otp/request (POST) – request local SMS OTP
/auth/otp/verify (POST) – verify OTP → tokens

/farms (GET) – list farms for user
/farms/{farmId}/members (GET/POST) – manage farm team
/farms/{farmId}/animals (GET/POST/PUT) – registry & search (tag/rfid)
/farms/{farmId}/breeding/events (POST)
/farms/{farmId}/health/events (POST)
/farms/{farmId}/weights (POST)
/farms/{farmId}/milk/yields (POST)
/farms/{farmId}/sales/animals (POST)
/farms/{farmId}/sales/milk (POST)
/farms/{farmId}/sync/pull (GET) – delta by since
/farms/{farmId}/sync/push (POST) – push outbox
/reports/* (GET) – CSV/XLSX endpoints
/notifications/inbox (GET) – in-app

12) Env Vars & Config (DigitalOcean)

SPRING_PROFILES_ACTIVE (dev|staging|prod)
SPRING_DATASOURCE_URL / USERNAME / PASSWORD
REDIS_URL
JWT_SECRET, JWT_ACCESS_TTL, JWT_REFRESH_TTL
S3_ENDPOINT, S3_BUCKET, S3_ACCESS_KEY, S3_SECRET_KEY
FCM_SERVER_KEY
SMS_BASE_URL, SMS_API_KEY, SMS_SENDER_ID
OTP_TTL_MINUTES (default 5), OTP_RESEND_SECONDS (default 45), OTP_MAX_ATTEMPTS (default 5)
RATE_LIMIT_OTP_PER_10M (default 3), RATE_LIMIT_OTP_PER_DAY (default 10)

13) Bitbucket Pipelines → DigitalOcean App Platform

Pipeline:
1) Build & test (PR)
2) Build JAR → Docker image (main)
3) Push to DO Container Registry
4) Flyway migrate (target DB)
5) Update App Platform app spec

Store secrets in Bitbucket secured variables. App Platform spec (.do/app.yaml) references current image tag and env vars.

14) Observability & Ops

Micrometer → Prometheus; Grafana dashboards for sync latency, job durations, notification success rate, DB slow queries.
JSON logs to stdout (DO collects). Optional OpenTelemetry traces. Postgres automated backups + weekly logical dump to DO Spaces.

15) Validation & Testing

Contract-first via springdoc annotations. Integration tests with Testcontainers (Postgres, Redis). Sync tests: simulate outbox push/pull and conflict 409 flows. OTP tests: rate limit, invalid code, lockout, DLR webhook.

16) Risks & Mitigations

Risk: SMS provider downtime → fall back to generic error; let users resend after cooldown.
Risk: Clock skew on clients → server treats incoming timestamps with tolerance; use server time in responses.
Risk: Large offline backlogs → cap batch size; incremental push with resume tokens.
Risk: Data mix between farms → strict farmId enforcement in filters + tests.

Appendix A – Environment Examples

Key	Example / Notes
SPRING_PROFILES_ACTIVE	prod
SPRING_DATASOURCE_URL	jdbc:postgresql://db:5432/farmapp
REDIS_URL	redis://:pass@redis:6379/0
S3_ENDPOINT	https://<space>.<region>.digitaloceanspaces.com
SMS_BASE_URL	https://local-sms.example/api

Appendix B – Scheduled Jobs

Job	Schedule (UTC)	Purpose
reminders:dispatch	Every 10 min	Send due reminders (push + inbox)
etl:metric_snapshots	00:05 daily	Aggregate metrics for insights
insights:impact_scan	00:45 daily	Correlate actions to outputs
costing:backfill	Nightly	Upgrade cost confidence from new GRNs