SalonOS – Software Development Plan (Local-First Salon Management)

Owner: Angel
Context: Unisex beauty salon, local-first backend on a reception laptop, mobile clients on same Wi-Fi.
Purpose: Master plan for building and operating SalonOS. Serves as source of truth across requirements, architecture, delivery phases, and SOPs.

---

1. Executive Summary

Goal: Replace Invoay with a local-first POS + Scheduling + Inventory + Accounting system that is fast on LAN, low-cost, and automation-ready. Phase delivery prioritizes **Billing**, then **Scheduling**, **Inventory (basic)**, **Accounting dashboards**, and **Staff access control**. Future phases add CRM/loyalty, WhatsApp messaging, self-booking, and CCTV analytics.

---

2. Objectives and Non-Goals

2.1 Objectives

- Fast, reliable **billing** with printed receipts and manual digital payment confirmation.
- Reception-driven **appointments** for walk-ins and call-ins; static calendar view.
- **Inventory** with SKU + UOM + reorder point, owner-approved adjustments, supplier data.
- **Accounting dashboards** with daily-first, real-time charts, GST split (CGST/SGST), monthly tax exports.
- **Access control**: roles for Owner, Receptionist, Staff. Privacy: staff see only first name + ticket.
- Local-first deployment via **Docker Compose** on reception laptop with **Nginx** reverse proxy.

2.2 Non-Goals (for now)

- Customer self-booking via web/WhatsApp.
- Automated WhatsApp/SMS reminders and campaigns.
- Inventory auto-consumption on services.
- Staff commissions.
- CCTV analytics.
- Cloud multi-site support.

---

3. Personas & Roles

- **Owner**: Full access, accounting dashboards, approvals, settings, purge toggle, profit estimate.

- **Receptionist/Cashier**: POS, appointments, daily open/close, manual payment marking, print/send receipts.
- **Staff (Stylists)**: View everyone's schedules; mark assigned service as completed; add optional service notes; no billing or totals display.

---

4. User Flows (Prioritized)

1) **Billing & Payment**: POS cart → discount (bill-level) → manual payment record (cash/UPI/card) → print 80mm receipt → optional send.
2) **Appointments (Reception)**: Create/modify via separate edit screen; mandatory name + phone + ≥1 service; no staff at booking time.
3) **Inventory**: Add SKUs with UOM and reorder point; manual receive/adjust with **reason codes** → owner approval → stock ledger update; suppliers attached to SKUs; visual low-stock filter.
4) **Accounting**: Real-time dashboard; GST split; daily summaries (manual generate + auto job with catch-up); monthly tax reports on demand; refunds/voids shown **separately**; cash drawer flow.
5) **Access & Privacy**: Role-based; staff see only first name + ticket number; staff notes allowed; staff cannot see revenue counts.

---

5. Functional Requirements

5.1 POS & Billing

- Create bill from cart; **bill-level discounts only** with logging (user/device/timestamp/amount, reason optional).
- Taxes: **inclusive pricing** at catalog level; compute CGST/SGST at print/export time.
- Invoice numbering: **yearly reset** `SAL-<YY>-<NNNN>`.
- Rounding: nearest ₹1.
- Payments: manual confirmation; split supported (cash/digital).
- Printing: browser-rendered 80mm; **confirmation step** prompts: Print / Send / Both.
- Receipts show total; internal exports include tax breakdown for compliance.

5.2 Appointments & Scheduling

- Reception creates/edit appointments; **separate edit screen**; statuses optional later.
- Required fields: name, phone, one service, time slot.
- Multiple services for a visit are **separate appointments** linked by `visit_id` (system-generated).
- Staff assignment done at check-in or completion; **manual** only.

5.3 Inventory

- Entities: categories, SKUs (UOM, reorder point, avg_cost), suppliers.
- Operations: **receive** and **adjust** via change requests → **owner approval** → stock ledger.
- Reason codes required; visual low-stock filter; **no hard stop** on zero stock (warning only).
- Cost model: **weighted average** maintained in SKU.

5.4 Accounting & Dashboards

- Daily-first charts with **auto-refresh 150s**; date-range filters.
- Metrics: cash vs digital, bill count, CGST, SGST, discounts, refunds; **adjustments** in separate section.
- **Profit estimate** visible only to owner (Revenue − COGS using avg_cost).
- **Cash drawer**: single drawer/day; open float, manual close with counted cash; allow reopen without approval; log reopen reason automatically.
- Reports: daily and monthly summaries, downloadable **PDF/XLS**, auto-named and stored under `/salon-data/exports/`.
- Snapshots: **auto-saved** daily/monthly; manual regenerate permitted.
- Monthly tax reports available on demand.

5.5 Access Control & Privacy

- Roles: **Owner**, **Receptionist**, **Staff**.
- Staff landing = **today's schedule**; persistent sessions until manual logout.
- Staff can mark service **completed** and add optional **service notes** (editable for 15 minutes).
- Staff see everyone's schedules but **no** totals/analytics; customer PII withheld (first name + ticket only).

---

6. Non-Functional Requirements

- **Performance**: POS operations < 200 ms API latency on LAN; dashboard charts < 1.5 s.
- **Reliability**: Data durability via Postgres WAL; daily backups; catch-up job for missed daily summary.
- **Security**: JWT auth; role-based ACL; PII fields encrypted; Postgres not exposed to LAN; Nginx single entrypoint.
- **Observability**: structured logs, request IDs, health endpoints, job failure counters.
- **Portability**: Docker Compose on Linux/Windows.

---

7. Architecture

- **Frontend**: React/Next.js app accessed at `http://salon.local` through Nginx.
- **Backend**: FastAPI, **SQLAlchemy + Alembic**.
- **DB**: PostgreSQL.
- **Cache/Queue**: Redis + **RQ worker**; APScheduler inside worker to enqueue cron jobs.
- **Reverse Proxy**: **Nginx**.
- **Printing**: browser 80mm receipt route triggers `window.print()`.
- **Files**: `/salon-data/exports` for reports; nightly DB dumps.

7.1 Container Topology

- `nginx` (ports: 80 to LAN) → `api` (internal) → `postgres` (internal), `redis` (internal), `worker` (internal).
- No direct LAN access to Postgres/Redis.

7.2 Eventing

- Synchronous writes on transactions; enqueue events to Redis:
- `bill.posted`, `bill.refunded`, `payment.captured`, `day.closed`, `inventory.approved`.
- Worker updates rollups, snapshots, and export logs.

---

8. Data Model Summary (canonical)

Refer to ERD (attached). Key tables:
`users, roles, staff, customers, services, service_addons, appointments, walkins, bills, bill_items, payments, inventory_categories, suppliers, skus, stock_ledger, inventory_change_requests, cash_drawer, day_summary, export_log, events, audit`.

Conventions:
- `id` = ULID; times in IST in UI, UTC in DB.
- Monetary fields in paise (int) for accuracy; rounding at presentation layer.
- Invoice numbers: `SAL-YY-NNNN` per fiscal year reset.

---

9. API Contract (v1 overview)

- **Auth**: `POST /auth/login`, `POST /auth/logout`
- **Catalog**: `GET /catalog/services`, `POST /catalog/service`, `PATCH /catalog/service/:id`
- **Appointments**: `POST /appointments`, `GET /appointments?from&to`, `PATCH /appointments/:id`, `POST /appointments/:id/assign`
- **Walk-ins**: `POST /walkins`, `PATCH /walkins/:id/assign`
- **POS**: `POST /pos/bills`, `GET /pos/bills/:id`, `POST /pos/bills/:id/payments`, `POST /pos/bills/:id/refund`
- **Inventory**: `GET /inventory/skus`, `POST /inventory/sku`, `POST /inventory/change-request`, `POST /inventory/approve/:req_id`, `GET /inventory/ledger/:sku_id`
- **Accounting**: `GET /reports/daily?date`, `POST /reports/daily/generate`, `GET /reports/monthly?month`, `GET /reports/tax?month`, `GET /exports`
- **Ops**: `POST /cash/open`, `POST /cash/close`, `POST /cash/reopen`
  All write endpoints accept **Idempotency-Key**; pagination, ETags on reads.

---

10. Background Jobs & Schedules

- **21:45 IST** Daily Summary generation (auto + catch-up on startup).
- **23:30 IST** Nightly backup: `pg_dump` + rclone to Drive/S3 (configurable).
- **Every 2–5 min**: rollup cache TTL refresh; dashboard auto-refresh **150s** in UI.

- **On event**: `bill.posted` → update day totals, update materialized views, create export entries as needed.

---

11. Security, Privacy, Compliance

- LAN-only DB and Redis; Nginx single port to LAN.
- JWT with short access + rotating refresh tokens; device binding for reception terminal.
- Field-level encryption for phone/email; encrypt backups at rest.
- Audit log for price overrides, discounts, voids, inventory adjustments.
- GST compliance: internal ledger stores taxable value + CGST/SGST; monthly tax report export.

---

12. Dev Process & Environments

- **Branching**: trunk with feature branches; PR reviews.
- **Migrations**: Alembic per feature; migration checklist.
- **Testing**: unit (FastAPI routes, domain services), integration (DB), e2e smoke on local Compose.
- **CI (optional later)**: GitHub Actions for lint+tests; image build.

---

13. Deployment

- **Docker Compose** services: `nginx`, `api`, `worker`, `postgres`, `redis`.
- `.env` for non-sensitive config; Docker Secrets for tokens/keys.
- Static IP on router; access via `http://salon.local` or `http://192.168.1.50`.
- Printing station uses Chromium kiosk for receipt route.

---

14. Backup & DR

- Nightly `pg_dump` to `/salon-data/backups`; weekly test restore to test DB.
- Optional cloud copy via rclone.
- Recovery SOP: reinstall Compose, restore DB, restore `/exports` folder.

---

15. Observability

- Logs: JSON with request_id, user_id, ip, latency, status.
- Health: `/healthz`, `/readyz`.
- Job metrics: queue depth, failures, last success times.
- Admin UI: event viewer and retry.

---

16. Acceptance Criteria (per module)

POS/Billing (Phase 1)

- Create bill with bill-level discount; manual payment; 80mm receipt print with confirmation step.
- Invoice number format validated; rounding works; refund appears in Adjustments and does not change posted totals.
- Discount logs visible in owner report.

Scheduling (Phase 1)

- Reception creates/edits appointments with required fields; inline conflict check; view day/week.
- Staff assignment optional; live statuses not required.

Inventory (Phase 1)

- Create SKUs with UOM and reorder point; supplier link.
- Change requests require reason codes; owner approval updates stock ledger.
- Low-stock filter works; warnings on zero stock at POS.

Accounting (Phase 1)

- Real-time charts; daily summary manual generate; snapshots saved; monthly tax report downloadable.
- Cash drawer open/close/reopen flows logged; reopen allowed without approval.

Access & Privacy (Phase 1)

- Role permissions enforced; staff default to today’s schedule; PII hidden from staff.

---

17. Delivery Phases & Timeline (effort-based)

Phase 1 – Core Operations (Weeks 1–4)

- POS/Billing, Appointments (reception), Inventory basics with approvals, Accounting dashboards (real-time), Cash drawer, Access control, Exports.
- Deliverables: running Compose stack, schema + migrations, seeded catalog, printable receipts, daily/monthly reports.

Phase 2 – Stability & UX (Weeks 5–6)

- Snapshots + regenerate, export log UI, search/sort inventory, persistent sort preferences, staff notes, owner approvals UX, error handling & retries, backup scripts.

Phase 3 – CRM & Automations (Weeks 7–9)

- Customer profiles maturity, rebook & review templates (manual trigger), WhatsApp provider setup (config only), campaign logs, loyalty ledger skeleton.

Phase 4 – Online & Advanced (Weeks 10–12, optional)

- Self-booking via hosted endpoint, appointment reminders, service-level discounts option, inventory auto-consumption mapping, supplier emails, CCTV analytics (footfall/queue) as feature flags.

---

18. Risks & Mitigations

- **Single-machine dependency** → UPS + nightly backups + catch-up jobs.
- **Data entry discipline** → role perms, reason codes, approval workflow.
- **GST compliance drift** → monthly tax report checks; snapshot immutability.
- **Local network issues** → static IP, router QoS; offline receipt reprint cache.

---

19. SOPs (Operations)

- **Open Day**: login → cash drawer open (float) → verify calendar.
- **Billing**: add services → discount if needed → confirm payments → Print/Send.
- **Close Day**: count cash → close drawer → generate daily summary → optional manual exports.
- **Inventory**: reception/staff submits receive/adjust → owner approves daily.
- **Backups**: verify morning catch-up; weekly restore test.

---

20. Roadmap (Future Accommodations)

- **Automated reminders & campaigns** via WhatsApp/SMS; customer portal links for receipts/points.
- **Service-specific discounts** and promotions engine.
- **Inventory auto-consumption** mapping service→SKU usage; variance analytics.
- **Staff commissions** engine and payouts.
- **Self-booking** via Vercel or VPS; online deposits.
- **CCTV analytics** for footfall, queue, no-show checks (privacy-safe).
- **Cloud sync** for off-site dashboards.

---

21. Appendices

21.1 Naming & IDs

- ULID for `id`; invoice `SAL-YY-NNNN`; ticket `TKT-YYMMDD-###`.

21.2 Cache & Auto-Refresh

- Materialized views for daily/monthly; cache invalidated on `bill.posted` / `bill.refunded`; UI auto-refresh **150s**.

21.3 Config Matrix (defaults)

- Tax inclusive, CGST+SGST shown in exports; bill-level discounts; anonymous bills allowed; staff privacy enforced.

21.4 Backup Paths

- DB: `/salon-data/backups`
- Exports: `/salon-data/exports`
- Logs: `/salon-data/logs`

End of Document