Lab Operations Software – Design Doc (MVP)

This document captures the updated design for a **lab operations software MVP** running on a self-hosted server. It is designed for **visit-centric operations**, allowing future patient accounts, culture reports, billing with discount approval, and PDF report generation.

---

1. System Overview

- Self-hosted lab management software.
- Each **visit** is unique; patient details stored as mandatory fields in visits table.
- Admin can define test templates (parameters, fields, price).
- Reception creates visits; phlebotomy collects samples; lab enters results; approver validates; billing completes the cycle.
- Supports Culture & Sensitivity (C/S) tests, discount approvals, and report PDF generation.
- Lightweight deployment on local CPU.

---

2. Data Model (Postgres)

referral_doctors

```
CREATE TABLE referral_doctors (
    doctor_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    clinic_hospital VARCHAR(255),
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);
```

visits

```
CREATE TABLE visits (
    visit_id SERIAL PRIMARY KEY,
    patient_id INT REFERENCES patients(patient_id), -- optional for future
accounts
    salutation VARCHAR(20) NOT NULL,        -- Mr, Mrs, Baby, Master, etc.
    name VARCHAR(255) NOT NULL,
    age_years INT NOT NULL,
    age_months INT,                         -- optional, for pediatrics
    age_days INT,                           -- optional, for newborns
    sex VARCHAR(10) NOT NULL,               -- Male/Female/Other
    phone VARCHAR(20),                      -- optional for contact
    address TEXT,                           -- optional
    date_of_visit DATE DEFAULT CURRENT_DATE,
    referred_doctor_id INT REFERENCES referral_doctors(doctor_id),
    visit_code VARCHAR(50) UNIQUE,         -- optional short code for printing
    created_at TIMESTAMP DEFAULT NOW(),
    status VARCHAR(50) DEFAULT 'pending'   -- pending, in-progress, awaiting-
approval, approved, billed, completed
);
```

test_templates

```
CREATE TABLE test_templates (
    template_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    parameters JSONB NOT NULL, -- dynamic fields, reference ranges, types
    base_price DECIMAL(10,2) NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

lab_tests

```
CREATE TABLE lab_tests (
    test_id SERIAL PRIMARY KEY,
    visit_id INT NOT NULL REFERENCES visits(visit_id),
    test_template_id INT NOT NULL REFERENCES test_templates(template_id),
    status VARCHAR(50) DEFAULT 'pending',
    price DECIMAL(10,2) NOT NULL,
    results JSONB,                         -- normal test results
    approved BOOLEAN DEFAULT FALSE,
    approved_by VARCHAR(255),
    approved_at TIMESTAMP
);
```

antibiotics (master list)

```
CREATE TABLE antibiotics (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL
);
```

culture_sensitivity_results

```sql
CREATE TABLE culture_sensitivity_results (
    id SERIAL PRIMARY KEY,
    visit_id INT NOT NULL REFERENCES visits(visit_id),
    test_id INT NOT NULL REFERENCES lab_tests(test_id),
    antibiotic_id INT NOT NULL REFERENCES antibiotics(id),
    sensitivity VARCHAR(20) CHECK (sensitivity IN
('Resistant','Sensitive','Moderate'))
);
```

invoices / billing

```sql
CREATE TABLE invoices (
    invoice_id SERIAL PRIMARY KEY,
    visit_id INT NOT NULL REFERENCES visits(visit_id),
    total_amount DECIMAL(10,2) NOT NULL,
    discount_request NUMERIC(10,2),
    discount_approved NUMERIC(10,2),
    discount_status VARCHAR(20) CHECK (discount_status IN
('None','Requested','Approved','Rejected')) DEFAULT 'None',
    approved_by INT REFERENCES users(id),
    payment_mode VARCHAR(20),  -- cash, card, UPI
    paid BOOLEAN DEFAULT FALSE,
    report_pdf_path TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);
```

patients (optional, future use)

```sql
CREATE TABLE patients (
    patient_id SERIAL PRIMARY KEY,
    salutation VARCHAR(20) NOT NULL,
    name VARCHAR(255) NOT NULL,
    age_years INT NOT NULL,
    age_months INT,
    age_days INT,
    sex VARCHAR(10) NOT NULL,
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);
```

---

3. API Specification

Visits

- **Create Visit**: `POST /visits` → patient details inline, optional `patient_id` for future accounts.
- **Get Visit**: `GET /visits/{id}`
- **List Visits**: `GET /visits?status=pending`

Test Templates

- **Add Template**: `POST /test-templates`
- **List Templates**: `GET /test-templates`

Lab Tests

- **Add Test to Visit**: `POST /visits/{visitId}/tests`
- **Update Results**: `PATCH /visits/{visitId}/tests/{testId}/results`
- **Approve Results**: `PATCH /visits/{visitId}/tests/{testId}/approve`

Culture & Sensitivity

- **Add/Update Antibiotic Result**: `POST /visits/{visitId}/tests/{testId}/c_s_results`
- **List Results**: `GET /visits/{visitId}/tests/{testId}/c_s_results`

Billing / Invoices

- **Generate Bill**: `GET /visits/{visitId}/bill`
- **Request Discount**: `POST /invoices/{invoiceId}/discount_request`
- **Approve Discount**: `PATCH /invoices/{invoiceId}/approve_discount` (admin only)
- **Download Report PDF**: `GET /invoices/{invoiceId}/report`

---

4. Visit Lifecycle

1. **Reception** → Create visit with patient details.
2. **Phlebotomy** → Sample collection tracking.
3. **Lab Processing** → Add tests, enter normal or C/S results.
4. **Approval** → Supervisor/doctor approves results.
5. **Billing** → Bill created, discounts requested/approved, payment recorded.
6. **Report Generation** → PDF generated with results.
7. **Completion** → Visit closed.

---

5. Frontend Behavior

- Dynamic forms for normal tests (from JSONB parameters).

- C/S tests: grid with antibiotic dropdowns (Sensitive/Moderate/Resistant).
- Discount requests visible to admin only for approval.
- Report PDF download and print.

---

6. Testing Strategy

- Unit tests for services and validators.
- Integration tests with Postgres (Testcontainers).
- API tests using RestAssured/Postman.
- End-to-end lifecycle tests for visit → result → approval → billing → PDF.

---

7. Deployment Notes

- Docker Compose: Spring Boot + Postgres.
- Cloudflare Tunnel for fixed domain mapping.
- Lightweight deployment: i5 CPU, 8GB RAM, 256GB SSD.

---

8. Future Extension (Patient Accounts)

- Visits can link to `patients` table (`patient_id`).
- Migration: extract unique patients by phone/name → link visits → drop inline visit patient columns if needed.
- APIs support `patient_id` but do not enforce it, keeping MVP functional without accounts.

---

9. Acceptance Criteria

- Visit lifecycle works end-to-end.
- Mandatory patient details included in visits.
- Admin-defined test templates with JSONB parameters.
- Culture & Sensitivity test handling with structured antibiotic results.
- Discount approval workflow enforced for admins.
- PDF report generation for each visit with all results.
- API coverage with tests, deployable on local server.