FindMyMedi - V.1.0.0

FindMyMedi Phase 1 focuses on verified pharmacy onboarding and accurate medicine availability discovery. The system enables pharmacy owners to register pharmacies, allows system administrators to review and approve those registrations, and provides end users with a reliable way to search medicines and view available pharmacies on a map with quantities. Advanced workflows such as reservations, orders, payments, and notifications are intentionally deferred to later phases.

Actors

The system recognizes three primary actors. Pharmacy Owners are responsible for registering pharmacies and managing inventory after approval. System Administrators verify pharmacy registrations and govern master data. End Users search for medicines and view availability across approved pharmacies.

High-Level Workflow

A pharmacy owner creates an account and submits a pharmacy registration request. The pharmacy remains inactive until reviewed. A system administrator verifies the submitted details and either approves or rejects the request. Upon approval, the pharmacy becomes active and visible in search results. The owner can then add medicines to inventory by selecting from master medicine data and updating quantities and prices. End users search for medicines and receive a map-based list of approved pharmacies with available quantities and last updated timestamps.

User and Access Control

A User represents any authenticated account and stores identity and security attributes. Roles define permissions and are assigned through a junction table, allowing users to evolve without schema changes. Pharmacy staff associations are modeled separately to avoid coupling authentication with pharmacy ownership and to support future employee roles.

Pharmacy Registration and Verification

A Pharmacy stores registration details, licensing information, location coordinates, and operational status. Newly registered pharmacies are marked as pending verification. Administrators review details and update verification status to approved or rejected, optionally recording verification metadata and rejection reasons. Only approved pharmacies can publish inventory and appear in search results.

Inventory and Availability

Inventory connects pharmacies and medicines and represents real-time availability. It stores quantity, unit price, stock status, and the last updated timestamp. A unique constraint on pharmacy and medicine prevents duplicates. Stock status can be derived from quantity to reduce manual errors. Inventory update logs capture who changed quantities and when to support auditability.

Search and Map View

Search queries match medicines by name or attributes and return only approved pharmacies with in-stock inventory. Results include quantity and last updated time and are rendered on a map using pharmacy latitude and longitude. This read-heavy flow is optimized through indexing and can later be enhanced with caching.

Data Dictionary

User

Stores authenticated account information. user_id: Primary key, unique identifier for a user. full_name: User’s display name. email: Unique email used for login. password_hash: Hashed password value. phone_number: Contact number. status: Account state such as ACTIVE or SUSPENDED. created_at: Account creation timestamp. updated_at: Last modification timestamp.

Role

Defines system-level permissions. role_id: Primary key. role_name: Role identifier such as CUSTOMER, PHARMACY_OWNER, PHARMACY_STAFF, SYSTEM_ADMIN.

User_Role

Maps users to roles. user_role_id: Primary key. user_id: Foreign key referencing User. role_id: Foreign key referencing Role.

Pharmacy

Represents a registered pharmacy. pharmacy_id: Primary key. name: Pharmacy name. license_number: Government-issued license identifier. address: Physical address. city: City or region. latitude: Geographic latitude. longitude: Geographic longitude. contact_number: Pharmacy contact. verification_status: PENDING, APPROVED, or REJECTED. verified_by: Admin user who verified the pharmacy. verified_at: Verification timestamp. rejection_reason: Reason for rejection if applicable. created_at: Registration timestamp.

Pharmacy_User

Associates users with pharmacies. pharmacy_user_id: Primary key. pharmacy_id: Foreign key referencing Pharmacy. user_id: Foreign key referencing User. staff_role: OWNER or STAFF. created_at: Association creation timestamp.

Medicine

Master data for medicines. medicine_id: Primary key. generic_name: Generic drug name. brand_name: Brand name. dosage_form: Tablet, syrup, capsule, etc. strength: Dosage strength. regulatory_code: Official regulatory identifier. is_active: Indicates whether the medicine is active.

Inventory

Tracks medicine availability per pharmacy. inventory_id: Primary key. pharmacy_id: Foreign key referencing Pharmacy. medicine_id: Foreign key referencing Medicine. quantity_available: Current stock quantity. unit_price: Selling price per unit. status: IN_STOCK or OUT_OF_STOCK. last_updated_at: Last inventory update time.

Inventory_Update_Log

Audit log for inventory changes. log_id: Primary key. inventory_id: Foreign key referencing Inventory. updated_by: User who performed the update. old_quantity: Quantity before update. new_quantity: Quantity after update. updated_at: Update timestamp.

Non-Goals for Phase 1

Phase 1 explicitly excludes reservations, orders, payments, prescriptions, notifications, and POS integrations. These features are planned for subsequent phases and the data model is intentionally designed to accommodate them without breaking changes.

Future Extensions

Planned extensions include reservation and order workflows, pharmacy employee roles and permissions UI, real-time notifications, analytics dashboards, and external system integrations. The current design supports these additions through clean separation of concerns and stable master data.

Identify entities

- **User**
- **Role**
- **User_Role** (junction table for many-to-many)
- **Pharmacy**
- **Pharmacy_User** (links users to pharmacies)
- **Medicine**
- **Inventory**
- **Inventory_Update_Log** (audit table)

Relationships

1. **User** ↔ **Role**: many-to-many via `User_Role`
    - Each user can have multiple roles.
    - Each role can be assigned to multiple users.
2. **User** ↔ **Pharmacy**: many-to-many via `Pharmacy_User`
    - Each pharmacy can have multiple users (owners initially, later staff).
    - Each user can be associated with multiple pharmacies (future-proof design).
3. **Pharmacy** ↔ **Inventory** ↔ **Medicine**
    - Each inventory record connects **one pharmacy** with **one medicine**.
    - Each pharmacy can have multiple inventory entries.
    - Each medicine can appear in inventories of multiple pharmacies.

4. **Inventory ↔ Inventory_Update_Log**: one-to-many
    - Each inventory update can have multiple logs if tracked over time.
    - Logs store who made the update, old/new quantities, and timestamp.
5. Pharmacy `verification_status` indicates whether the pharmacy is approved or pending. Only `ACTIVE` pharmacies appear in search results.