International Journal of Computer Science and Engineering Development
Volume 1, Issue 5 | November - December 2026 | www.ijcsed.com
ISSN: **3139-0862**

Medicine Inventory and Availability Tracking System

A MySQL Based Prototype for Locating Medicines Across Multiple Pharmacies

Sanskar Kadam[1], Shravan Kedari[1]*, Mohan Jadhav[1]

[1] *Department of Information Technology, AISSMS Institute of Information Technology, Pune, Maharashtra, India*

* shravankedari15@gmail.com

***Abstract*—**

This paper presents the Medicine Inventory and Availability Tracking System, a MySQL 8.0 prototype for locating medicines across multiple pharmacies. The implementation uses 12 tables, 6 views, 5 stored procedures, 7 triggers, and 42 example queries. Availability is calculated from batch-level stock after excluding expired batches, inactive pharmacies, and units held by active reservations. Users can search by brand name, generic name, or ingredient. When a brand is unavailable, the prototype can return alternatives from a predefined list or identify medicines with matching ingredients and strengths. Batch tracking, expiry checks, reservations, and stock verification support these availability rules. The system was tested using fictional data representing five pharmacies, 15 medicines, and 30 batches containing 603 units. All batch quantities matched the stock ledger, and the attempted rule violations were rejected with the expected errors. The prototype uses sample data only: it is not connected to real pharmacies and does not provide live availability.

***Keywords*—**medicine availability; pharmacy inventory; batch tracking; expiry monitoring; reservations; MySQL

I. INTRODUCTION

It is not always easy to find a particular medicine. No single pharmacy stocks every brand, so a patient may have to visit several before finding it. Wondmagegn et al. [1] describe patients who go from one pharmacy to the next because they have no way of knowing where a prescribed medicine is available. A survey described in [2] found something similar: 100 of its 271 respondents had used apps to locate pharmacies, and 51 of them said it was hard to find all their prescribed drugs in one pharmacy.

An availability answer is only as good as the stock records behind it. Pharmacies keep medicines in batches, and each batch has its own batch number and expiry date. Stock that has expired, been reserved or been counted wrongly should not be shown as available. Inventory systems for pharmacies usually look after stock, invoices and expiry inside one organisation [3], [4]. Systems made for patients, on the other hand, focus on finding a pharmacy that has the medicine [1], [2]. This work asks how far a database design alone can bring the two together.

We built a prototype that does this inside the database. Its main purpose is to let a user find a medicine across several pharmacies. Inventory management, batch tracking, expiry monitoring, reservations and stock verification are included because the availability answer depends on them. The prototype is written for MySQL 8.0 and tested on fictional sample data. It is not linked to any real pharmacy, and we do not claim it shows real-time availability. Our contributions are:

1) a normalised 12-table schema that records stock batch by batch and keeps each medicine's ingredients in separate tables;
2) an availability method that reports stock for each pharmacy after leaving out expired batches, inactive pharmacies and reserved units, with search by brand, generic name or ingredient;
3) two ways of finding alternative medicines: a prepared list, and an exact ingredient match; and
4) stored procedures and triggers that keep stock, reservations and stock-count differences consistent, together with test results.

The rest of the paper is organised as follows. Section II reviews related work, Section III explains the design and Section IV the implementation. Section V presents the results. Section VI covers limitations and future work, and Section VII concludes.

II. RELATED WORK

Earlier work on pharmacy information systems falls mostly into two groups: systems that manage inventory and systems that help people find medicines. Samaan [3] designed an inventory database for a pharmaceutical trading company. A C# application on top of it handles sale and purchase invoices, customers, suppliers, payments and reports, and it checks for expired medicines that must be thrown away. Budianto et al. [4] built a web-based drug inventory system for small pharmacies, following the Waterfall model. It tracks stock by lot, watches expiry dates automatically and produces reports. The authors tested it with black-box testing and user acceptance testing. Both systems manage stock inside one organisation, not across pharmacies.

On the locating side, Wondmagegn et al. [1] propose a framework and a web application. They use the coordinates of the user and the pharmacies, together with factors such as distance, drug cost, travel time and opening hours, to find the nearest pharmacies that have a prescribed medicine in stock. They tested it on simulated patients and pharmacies. Sarmento et al. [2] carried out the survey mentioned above and proposed MEDLOC, an application that sends a patient's availability request to registered pharmacies nearby, where a pharmacist replies. Chalmeta et al. [5] work in hospitals. They propose a computer architecture that combines blockchain, radio-frequency identification (RFID), quick response (QR) codes, the Internet of Things (IoT), artificial intelligence and big data to trace drugs, and they checked it through expert opinion.

Table I compares these works. Our prototype has a different focus. It has no user interface and does not sort pharmacies by

ISSN: **3139-0862**

distance. What it does offer, inside the database, are the rules that make an availability answer believable: stock recorded by batch, expired stock filtered out, reserved units subtracted, and every stock change written to a ledger.

TABLE I. Comparison with Related Work

Work	Scope	Batch / expiry	Cross-pharmacy availability
Samaan [3]	Inventory database management system for a trading company (C#)	Expired-medicine checking	No (single company)
Budianto et al. [4]	Web-based pharmacy inventory	Lot-level batches; expiry alerts	No (single pharmacy)
Wondmagegn et al. [1]	Locate nearest pharmacy with stock	n/r	Yes (simulated data)
Sarmento et al. [2]	Availability requests to nearby pharmacies	n/r	Yes (prototype)
Chalmeta et al. [5]	Hospital drug-management architecture	Medicine traceability	No (hospital)
This work	Database prototype for availability search	Batch, expiry status, reservations	Yes (sample data)

n/r: not reported in the sources consulted.

III. SYSTEM DESIGN

A. Objectives and Scope

The main objective is simple. A user should be able to find which pharmacies in the network have a given medicine right now, and how many units. Several supporting features keep that answer correct: stock managed batch by batch, with manufacturing, purchase and expiry dates; checks for expiring and low stock; reservations that hold stock without removing it; and regular stock counts with follow-up of any differences.

We kept the scope small on purpose. The system is a MySQL prototype with fictional data. It has no user interface, no connection to real pharmacy systems and no live data feed. Sales, billing, prescriptions and customers are not modelled, so a reservation is recorded against a staff account instead of a customer.

B. Architecture

Fig. 1 shows the architecture in logical terms. A staff member, administrator or project user types Structured Query Language (SQL) statements into MySQL Workbench. For reading data, there are 42 example queries, six views and five stored procedures. For changing data, the intended route is two procedures that run each stock change or reservation as a single transaction. Underneath, constraints, seven triggers and InnoDB transactions enforce the rules, and twelve tables store the data as master data, inventory and transaction records. These layers are logical only. The prototype is not deployed as a multi-tier application.

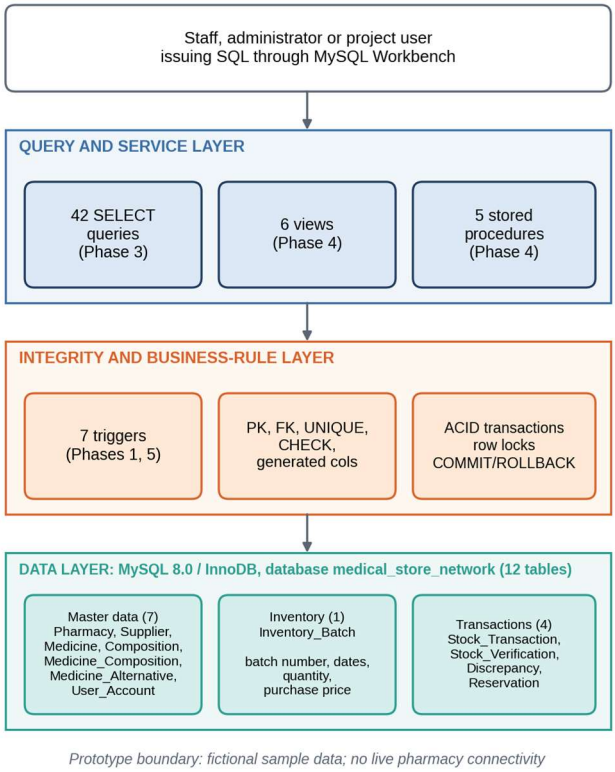


Fig. 1. Logical architecture of the prototype.

C. Database Design

The schema follows the relational model [6] and was planned using entity–relationship modelling [7]. We drew the enhanced entity–relationship (EER) model in MySQL Workbench. Fig. 2 and Table II show the 12 entities and 17 foreign keys. Medicine, Pharmacy and Supplier each link one-to-many to Inventory_Batch, so every stock row belongs to exactly one pharmacy, one medicine and one supplier. A medicine can appear in several batches at several pharmacies. For this reason availability has to be added up over batches; it cannot be read from a single quantity column.

Ingredients are stored in Composition and linked to Medicine through the junction table Medicine_Composition, which also records the strength of each ingredient. Medicine_Alternative is a many-to-many table that points back to Medicine and holds the prepared list of alternatives. Stock_Transaction, Stock_Verification and Reservation all belong to a batch in Inventory_Batch, and each Discrepancy belongs to at most one Stock_Verification. User_Account is referenced by every table that records who did something.

ISSN: **3139-0862**

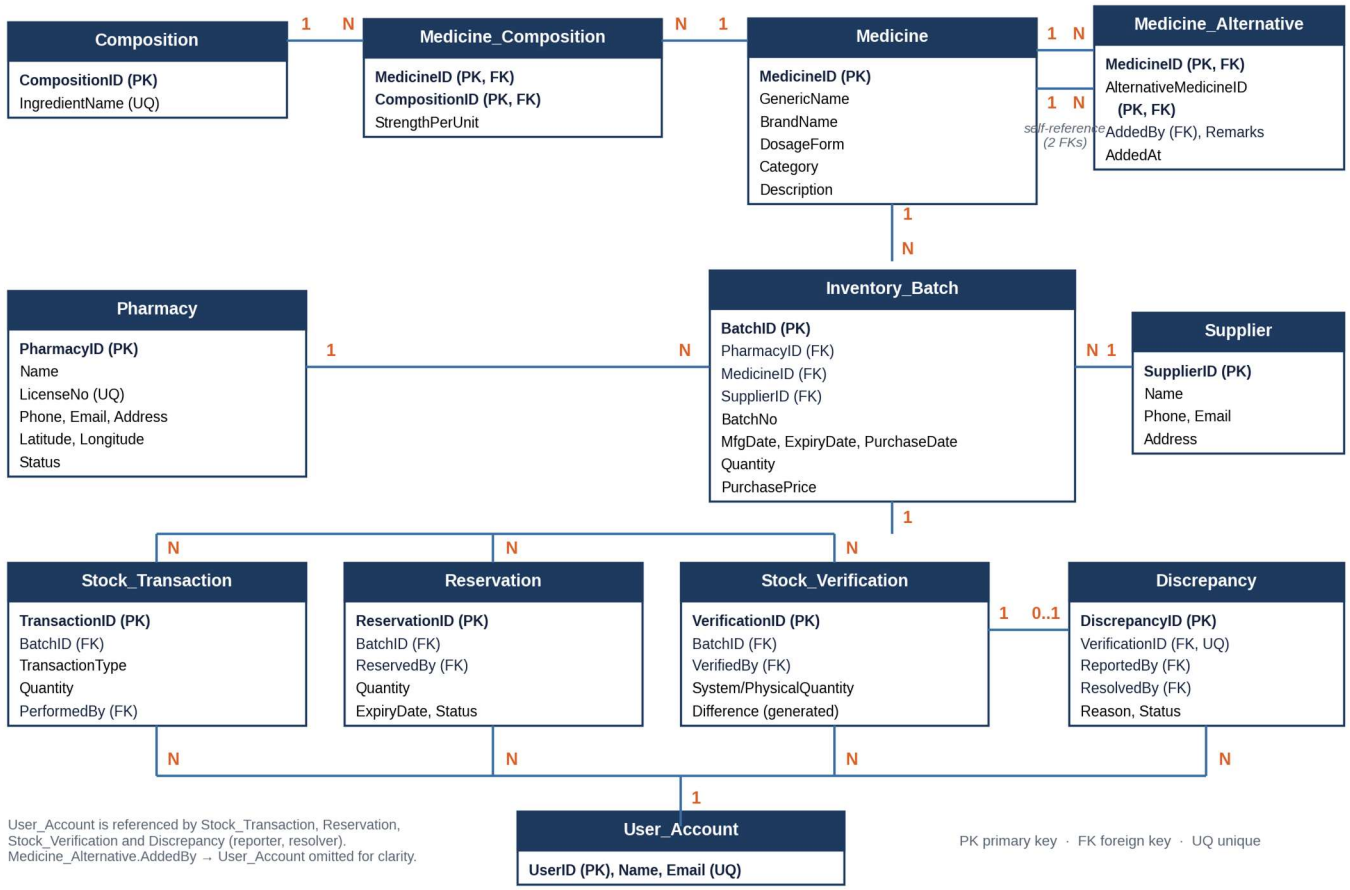


Fig. 2. Entity–relationship diagram of the 12-table schema (drawn from the MySQL Workbench model; Medicine_Alternative.AddedBy → User_Account omitted for clarity).

The referential actions depend on what each table is for. RESTRICT prevents batches, the ledger and the verification history from being deleted. CASCADE is used on the junction tables, and SET NULL on the optional user references (AddedBy and ResolvedBy).

TABLE II. Database Tables and Key Constraints

Table	Purpose	Key constraints
Pharmacy	Store master with location	UNIQUE licence, email; latitude/longitude CHECK
Supplier	Supplier master	UNIQUE email
Medicine	Brand and generic catalogue	Indexes on generic and brand name
Composition	Ingredient master	UNIQUE ingredient name
Medicine_Composition	Medicine–ingredient (M:N)	Composite PK; strength per unit
Medicine_Alternative	Curated alternatives (self M:N)	Triggers block self-reference
User_Account	Staff and administrator accounts	Role ADMIN/STAFF; UNIQUE email
Inventory_Batch	Batch stock per pharmacy	UNIQUE (pharmacy, medicine, batch no.); date and price CHECKs
Stock_Transaction	Stock ledger, six movement types	CHECK quantity > 0
Stock_Verification	System vs physical count	Generated difference and status
Discrepancy	Follow-up of count mismatches	UNIQUE verification; date CHECK
Reservation	Holds on a batch	CHECK quantity > 0; expiry after reservation

D. Integrity Rules and Normalisation

Most integrity rules are declared directly in the schema. CHECK constraints cover coordinates, batch dates (expiry must come after both the manufacturing and purchase dates), purchase price (never negative), transaction and reservation quantities (always positive), reservation expiry and resolution date. A unique key on (PharmacyID, MedicineID, BatchNo) prevents duplicate batches. Stock_Verification also has two stored generated columns, Difference and VerificationStatus, which MySQL calculates from the system and physical counts.

MySQL does not accept a CHECK on a column that also has a cascading foreign key. For that reason, the rule that a medicine cannot be its own alternative is enforced by two triggers. Keeping ingredients in separate tables avoids repeated data and update problems, in line with normalisation principles [8]. Two attributes are still kept in a simpler form: Medicine.GenericName holds the combined ingredient names as text, and Category is free text.

ISSN: 3139-0862

IV. IMPLEMENTATION

A. Environment and Script Organisation

We implemented and tested the prototype on MySQL Server 8.0.46 (InnoDB, utf8mb4) using MySQL Workbench. It is divided into five scripts that are run in order (Table III). The Workbench enhanced entity–relationship (EER) model is kept in a separate file.

TABLE III. Database Scripts and Their Contents

No.	Script	Contents
01	Full schema	Database, 12 tables, indexes, constraints, 2 validation triggers
02	Sample data	Fictional rows for all 12 tables, re-runnable inserts
03	SQL queries	42 numbered SELECT queries in ten groups
04	Views and procedures	6 views and 5 stored procedures
05	Triggers and transactions	5 triggers; commented commit, rollback and savepoint demonstrations; integrity check

B. Sample Dataset

All the data are fictional. The dataset has five pharmacies in Pune localities (Kothrud, Deccan, Aundh, Viman Nagar and Karve Nagar), four suppliers, four user accounts (one administrator and three staff) and 15 medicine brands in five categories: analgesic/antipyretic, NSAID, antihistamine, gastrointestinal and antibiotic. The brands are made from eight ingredients (paracetamol, ibuprofen, cetirizine, pantoprazole, omeprazole, azithromycin, amoxicillin and clavulanic acid), and two of the brands contain two ingredients. There are 18 prepared alternative mappings and 30 inventory batches holding 603 units. Every batch has one stock-in and one stock-out entry in the ledger, giving 60 transactions (903 units in and 300 out). Six stock verifications, three discrepancies and five reservations exercise the other features. The licence numbers and e-mail addresses are demonstration values only.

C. Medicine Availability Search

We define availability batch by batch. For a batch, the reserved quantity is the total of reservations that are ACTIVE and not yet expired. The available quantity is the recorded quantity minus the reserved quantity, and it never goes below zero. The view vw_active_medicine_availability adds this up for each medicine and pharmacy. It only includes ACTIVE pharmacies and unexpired batches, leaves out anything with no available units, and returns the pharmacy name, address and phone number, the number of available batches, the available quantity and the nearest expiry date.

To search, a user calls sp_search_medicine, which matches a search term against brand names, generic names and ingredient names. The user then reads the view for the medicine they chose (Fig. 3). Results are listed by pharmacy. They are not sorted by distance, because latitude and longitude are stored but no query uses them.

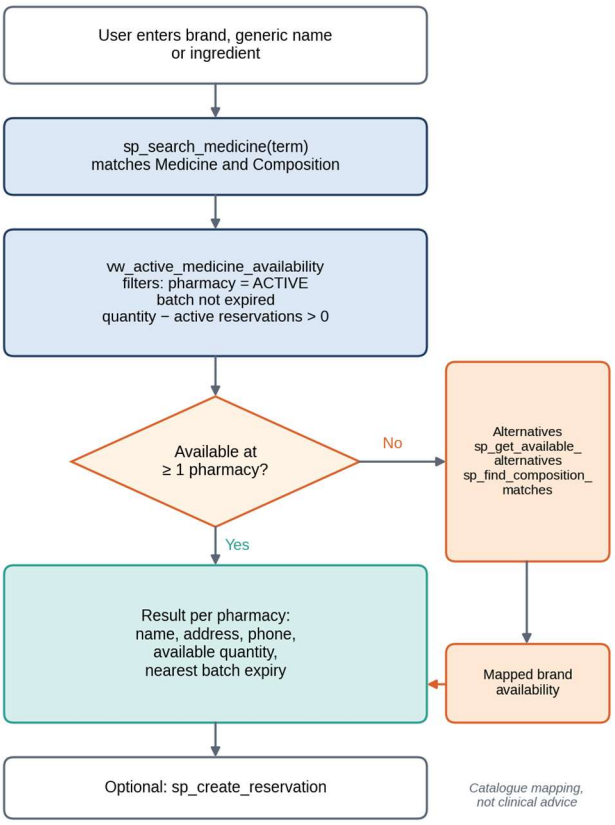


Fig. 3. Medicine availability search workflow.

D. Alternative Medicines

There are two ways to look for alternatives when a brand is not available. The first is sp_get_available_alternatives. It joins the prepared Medicine_Alternative list with the availability view and returns each listed alternative together with the pharmacies that have it; an alternative with no stock is returned with quantity zero. The second is sp_find_composition_matches. It looks for medicines in the catalogue with the same ingredients, the same strengths and the same dosage form as the selected medicine. To do this it nests NOT EXISTS conditions in both directions, which is the usual SQL way of expressing relational division [9]. A match therefore has to contain every ingredient of the original medicine and nothing extra. Both methods work only with catalogue data and are not clinical advice about substitution.

E. Batch Tracking and Expiry Monitoring

Each Inventory_Batch row stores the batch number, the manufacturing, purchase and expiry dates, the quantity and the purchase price. The view vw_batch_stock_status gives every batch one status. It checks them in a fixed order: EXPIRED (expiry date has passed), OUT_OF_STOCK (quantity is zero), FULLY_RESERVED (nothing is available), EXPIRING_SOON (expires within 90 days), LOW_STOCK (ten or fewer units available) and NORMAL. Expiry is therefore monitored through this status and through queries that look at an expiry window. There is no scheduled job and no automatic alert. Only the purchase price is stored; the schema has no selling price.

ISSN: **3139-0862**

F. Reservations and Stock Transactions

Stock changes and reservations go through two stored procedures (Fig. 4). Both check their inputs, start a transaction and lock the batch row with SELECT … FOR UPDATE, so that two sessions cannot use the same stock at the same time [10], [11]. sp_record_stock_transaction accepts six movement types and refuses an outgoing quantity larger than the stock not held by active reservations. It writes the ledger row and updates the batch quantity inside the same transaction. sp_create_reservation needs an active pharmacy, an unexpired batch, a reservation expiry no later than the batch expiry date, and enough unreserved stock. It records the hold but does not reduce the batch quantity. If an SQL error occurs, an exit handler rolls the transaction back and raises the error again, so a failed operation leaves nothing half-finished. This follows the transaction properties described in [10].

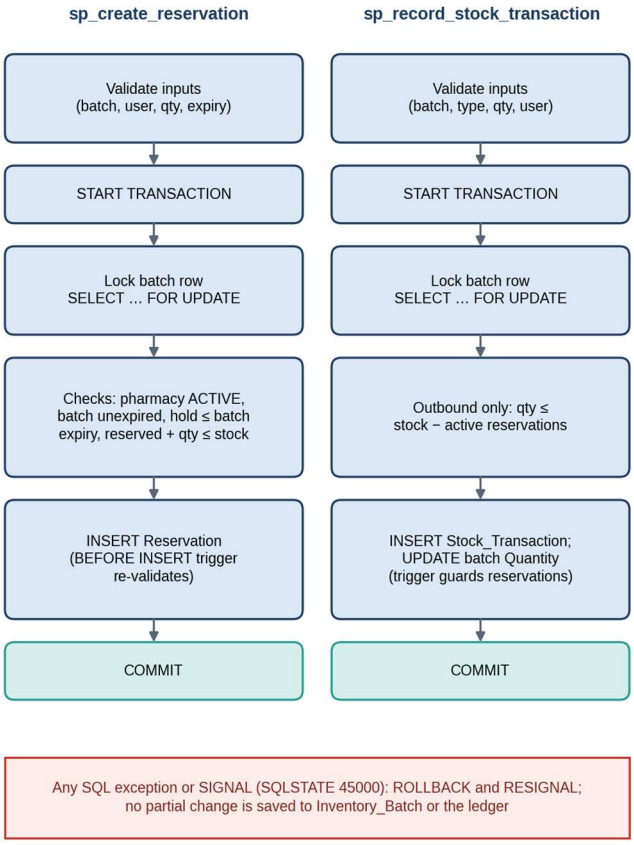


Fig. 4. Reservation and stock-transaction procedures.

Seven triggers (Table IV) enforce related rules on the tables themselves, so they also apply to statements that do not use the procedures.

TABLE IV. Triggers and the Rules They Enforce

Trigger	Table / timing	Rule enforced
trg_ma_before_insert, *trg_ma_before_update*	*Medicine_Alternative* BEFORE	A medicine cannot be its own alternative
trg_inventory_batch_before_update	*Inventory_Batch* BEFORE UPDATE	Quantity cannot fall below stock held by active reservations
trg_reservation_before_insert	*Reservation* BEFORE INSERT	Re-validates pharmacy, batch expiry, hold expiry and stock
trg_reservation_before_update	*Reservation* BEFORE UPDATE	Same checks, excluding the row's own quantity
trg_stock_verification_after_insert	*Stock_Verification* AFTER INSERT	Opens a discrepancy when counts differ
trg_stock_verification_after_update	*Stock_Verification* AFTER UPDATE	Creates, updates, reopens or resolves the discrepancy

G. Stock Verification and Discrepancies

Stock_Verification stores the system count and the physical count. When the two differ, the AFTER INSERT trigger opens a discrepancy automatically. The AFTER UPDATE trigger then creates, updates, reopens or resolves the discrepancy as the counts change, so a recount that matches the system quantity closes it. Closing a discrepancy does not change the batch quantity. Any correction has to be entered separately as a stock transaction.

H. Demonstration Queries

The 42 queries in script 03 only read data. Table V groups them by topic. Query 14 is the most complex: it is the relational-division query that sp_find_composition_matches wraps in a procedure. Some queries depend on today's date, so their output changes over time. Fig. 5 shows Workbench output for two of the views, and we discuss it in Section V.

TABLE V. Groups of Demonstration Queries

	Group	Queries	Main concepts
A	Basic retrieval	1–6	Filtering, sorting, DISTINCT, LIKE
B	Joins and composition	7–10	Multi-table joins, ingredient lists
C	Availability	11–15	Brand availability, alternatives, composition match
D	Aggregates	16–20	COUNT, MIN, MAX, AVG, GROUP BY, HAVING
E	Subqueries	21–24	EXISTS, NOT EXISTS, nested queries
F	Stock and expiry	25–28	Low stock, expiring, expired, suppliers
G	Ledger, verification	29–33	Ledger, monthly totals, discrepancies
H	Reservations	34–36	Listings and status counts
I	Set and functions	37–39	UNION ALL, string and date functions
J	Summaries	40–42	Per-pharmacy availability, value, ledger check

ISSN: **3139-0862**

(a)

```
1 •  SELECT BrandName, PharmacyName, AvailableBatchCount, AvailableQuantity, Ne
2    FROM vw_active_medicine_availability WHERE BrandName = 'Dolo 650';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BrandName	PharmacyName	AvailableBatchCount	AvailableQuantity	NearestBatchExpiry
Dolo 650	Demo Pharmacy - Kothrud	1	35	2027-10-31
Dolo 650	Demo Pharmacy - Deccan	1	16	2027-11-30
Dolo 650	Demo Pharmacy - Viman Nagar	1	50	2028-02-29

(b)

```
3  SELECT BatchID, BatchNo, BrandName, PharmacyName,
4         RecordedQuantity, ReservedQuantity, AvailableQuantity, StockStatus
5  FROM vw_batch_stock_status
6  ORDER BY PharmacyName, BrandName;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BatchID	BatchNo	BrandName	PharmacyName	RecordedQuantity	ReservedQuantity	AvailableQuantity	StockStatus
18	AUN-AZI-2602	Azithral 500	Demo Pharmacy - Aundh	20	0	20	NORMAL
15	AUN-CAL-2603	Calpol 650	Demo Pharmacy - Aundh	14	0	14	NORMAL
16	AUN-IBU-2602	Ibugesic 400	Demo Pharmacy - Aundh	25	0	25	NORMAL
19	AUN-OME-2603	Omez 20	Demo Pharmacy - Aundh	22	0	22	NORMAL
17	AUN-PAN-2603	Pantocid 40	Demo Pharmacy - Aundh	12	0	12	NORMAL
13	DEC-AZE-2603	Azee 500	Demo Pharmacy - Deccan	14	0	14	NORMAL
10	DEC-BRU-2603	Brufen 400	Demo Pharmacy - Deccan	22	0	22	NORMAL
14	DEC-CLA-2602	Clavam 625	Demo Pharmacy - Deccan	10	0	10	LOW_STOCK
9	DEC-CRO-2601	Crocin 650	Demo Pharmacy - Deccan	35	0	35	NORMAL
8	DEC-DOL-2602	Dolo 650	Demo Pharmacy - Deccan	20	4	16	NORMAL
11	DEC-OKA-2602	Okacet 10	Demo Pharmacy - Deccan	16	0	16	NORMAL
12	DEC-PAN-2601	Pan 40	Demo Pharmacy - Deccan	18	0	18	NORMAL
28	KRV-AZI-2604	Azithral 500	Demo Pharmacy - Karve ...	10	0	10	LOW_STOCK

Fig. 5. MySQL Workbench output: (a) availability of Dolo 650 from vw_active_medicine_availability; (b) batch stock status from vw_batch_stock_status.

V. RESULTS AND DISCUSSION

We ran the five scripts on MySQL 8.0.46 with the dataset from Section IV-B on 31 October 2026. All 42 queries ran without errors. Thirty-nine of them returned rows. Queries 21, 26 and 27 returned nothing, and that is the correct result: every medicine has unexpired stock, and no batch is expired or within 90 days of expiry.

A. Availability Across Pharmacies

Fig. 6 shows how many units of each of the 15 brands are available at the five pharmacies, as returned by vw_active_medicine_availability. Dolo 650 is the only brand stocked at three pharmacies (Kothrud 35, Deccan 16, Viman Nagar 50). Thirteen brands are stocked at exactly two pharmacies, and Azee 500 at only one (Deccan, 14 units). The 30 filled cells match the 30 batches, because each pharmacy holds at most one batch of a brand in the sample data.

	Kothrud	Deccan	Aundh	Viman Nagar	Karve Nagar
Dolo 650	35	16	–	50	–
Calpol 650	25	–	14	–	–
Crocin 650	–	35	–	–	18
Ibugesic 400	18	–	25	–	–
Brufen 400	–	22	–	–	14
Cetzine 10	30	–	–	24	–
Okacet 10	–	16	–	–	28
Pantocid 40	20	–	12	–	–
Pan 40	–	18	–	16	–
Azithral 500	–	–	20	–	10
Azee 500	–	14	–	–	–
Augmentin 625	12	–	–	8	–
Clavam 625	–	10	–	–	15
Omez 20	15	–	22	–	–
Ocid 20	–	–	–	20	12

Fig. 6. Available units per brand and pharmacy (– = not stocked), from vw_active_medicine_availability.

Fig. 5(a) shows the query for Dolo 650. Deccan shows 16 although its batch records 20 units, and Kothrud shows 35 although it records 40. The difference of 4 and 5 units is held by active reservations. Viman Nagar has no

ISSN: **3139-0862**

reservation and shows 50. The user is therefore told what can still be taken, not only what is on the shelf.

B. Batch Status and Inventory Distribution

Fig. 5(b) shows the batch status view. On the test date, 27 of the 30 batches are NORMAL and three are LOW_STOCK: Clavam 625 at Deccan (10 units), Augmentin 625 at Viman Nagar (8) and Azithral 500 at Karve Nagar (10). No batch is expired or close to expiry, since the expiry dates in the sample data run from 30 September 2027 to 29 February 2028. The EXPIRING_SOON and EXPIRED statuses are defined and can be queried, but the sample data never reaches them.

The dataset holds 603 recorded units, and 594 of them are available once active reservations are left out (Table VI, Fig. 7). Only Kothrud (160 recorded, 155 available) and Deccan (135 recorded, 131 available) have reserved stock. By category (Fig. 8), analgesic/antipyretic products account for 202 units, roughly one third of the stock.

TABLE VI. Per-Pharmacy Inventory Summary

Pharmacy	Batches	Distinct medicines	Recorded units	Available units	Stock value at purchase price
Kothrud	7	7	160	155	3,755.00
Deccan	7	7	135	131	3,725.75
Aundh	5	5	93	93	2,602.25
Viman Nagar	5	5	118	118	2,625.00
Karve Nagar	6	6	97	97	2,891.20
Total	30	—	603	594	15,599.20

The schema does not specify a currency unit.

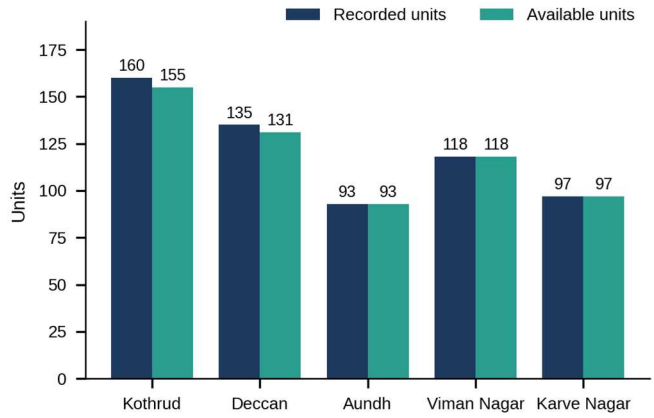


Fig. 7. Recorded and available units per pharmacy.

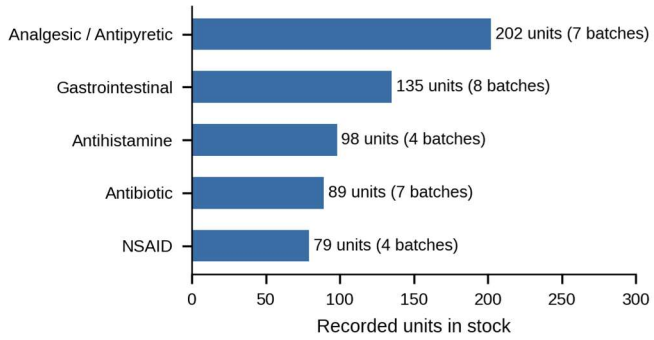


Fig. 8. Recorded units and batches by medicine category.

C. Alternatives

For Dolo 650, sp_get_available_alternatives returns Calpol 650 (Kothrud 25, Aundh 14) and Crocin 650 (Deccan 35, Karve Nagar 18). For Augmentin 625, which has two ingredients, sp_find_composition_matches returns Clavam 625 (Deccan 10, Karve Nagar 15). Dolo 650 is not stocked at Aundh or Karve Nagar, so the alternatives give the user options at exactly those pharmacies. The composition match also shows that matching depends on ingredients, strength and dosage form, not on the brand name.

D. Integrity and Consistency Tests

Table VII lists the functional tests. Before and after the tests, no batch had a recorded quantity that differed from the net of its transactions (903 units in minus 300 out gives the 603 recorded units). Every rule violation we tried was rejected with the expected message, and the rollback demonstration left both the batch quantity and the ledger unchanged. Fig. 9 shows the seven installed triggers and the verification report, which has three discrepancy records: two resolved and one open.

TABLE VII. Functional Test Results

Test	Observed outcome
Recorded quantity vs net ledger, 30 batches	0 mismatched batches
Stock-out of 1000 units from batch 1	Rejected: outbound quantity exceeds the unreserved stock
Reservation of 1000 units	Rejected: insufficient unreserved stock
Medicine inserted as its own alternative	Rejected: a medicine cannot be its own alternative
Batch 8 quantity set to 0 (4 units reserved)	Rejected: stock cannot be reduced below reserved quantity; quantity stays 20
Stock-in of 5 units, then reversal	Batch 1: 40 → 45 → 40, two ledger rows
Reservation of 3 units on batch 20	Available 50 → 47; reservation ACTIVE
Verification 40 vs 37; then recount to 40	Discrepancy opened (difference −3); then resolved automatically
Insert and update, then ROLLBACK	Quantity 40 before and after; no ledger row kept
SAVEPOINT, then ROLLBACK TO SAVEPOINT	Only the first change retained (batch 1: 40 → 42)

All tests were run on MySQL 8.0.46 with the project scripts.

ISSN: **3139-0862**

(a)

```
1 • USE medical_store_network;
2
3   SHOW TRIGGERS FROM medical_store_network;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Trigger	Event	Table	Statement	Timing	Created
trg_inventory_batch_before_update	UPDATE	inventory_batch	BEGIN DECLARE v_reserved_quantity BIGIN...	BEFORE	2026-10-31 08:44:44.08
trg_ma_before_insert	INSERT	medicine_alternative	BEGIN IF NEW.MedicineID = NEW.Alternativ...	BEFORE	2026-10-31 08:07:45.02
trg_ma_before_update	UPDATE	medicine_alternative	BEGIN IF NEW.MedicineID = NEW.Alternativ...	BEFORE	2026-10-31 08:07:45.07
trg_reservation_before_insert	INSERT	reservation	BEGIN DECLARE v_batch_count INT DEFAUL...	BEFORE	2026-10-31 08:44:44.09
trg_reservation_before_update	UPDATE	reservation	BEGIN DECLARE v_batch_count INT DEFAUL...	BEFORE	2026-10-31 08:44:44.10
trg_stock_verification_after_insert	INSERT	stock_verification	BEGIN IF NEW.SystemQuantity <> NEW.Phy...	AFTER	2026-10-31 08:44:44.11
trg_stock_verification_after_update	UPDATE	stock_verification	BEGIN DECLARE v_discrepancy_count INT D...	AFTER	2026-10-31 08:44:44.12

(b)

```
1 • SELECT VerificationID, BatchNo, SystemQuantity, PhysicalQuantity, Difference,
2         VerificationStatus, DiscrepancyID, DiscrepancyStatus, ResolvedByName
3   FROM vw_stock_verification_report
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

VerificationID	BatchNo	SystemQuantity	PhysicalQuantity	Difference	VerificationStatus	DiscrepancyID	DiscrepancyStatus	ResolvedByName
1	KTH-DOL-2601	40	40	0	MATCHED	NULL	NULL	NULL
2	KTH-CAL-2602	25	23	-2	DISCREPANCY	1	RESOLVED	System Administrator
3	KTH-CET-2603	30	32	2	DISCREPANCY	2	OPEN	NULL
4	DEC-DOL-2602	20	20	0	MATCHED	NULL	NULL	NULL
5	DEC-BRU-2603	22	22	0	MATCHED	NULL	NULL	NULL
6	AUN-AZI-2602	20	18	-2	DISCREPANCY	3	RESOLVED	System Administrator

Fig. 9. MySQL Workbench output: (a) the seven triggers installed in the schema (SHOW TRIGGERS); (b) the stock verification report with discrepancy status.

E. Discussion

The results suggest that the availability question can be answered consistently from batch-level data using a fairly small set of views and procedures. Three design choices did most of the work. First, availability is calculated when it is requested and never stored, so it cannot drift away from the underlying records. Second, reservations hold stock without changing it, so when a hold lapses the stock is freed without any update. Third, the ledger and the batch quantity change in the same transaction, which kept all 30 batches consistent in our tests. Compared with the systems in Table I, our prototype is narrower in its interface and in how it handles location [1], [2]. In return, it spells out the batch, expiry and reservation rules that the location-based systems do not describe.

VI. LIMITATIONS AND FUTURE WORK

The prototype has clear limits, and readers should keep them in mind. It is a SQL implementation with fictional data. It has no user interface, no connection to live pharmacies and no real-time availability. Our evaluation consists of functional tests on a small dataset, with no performance measurements and no user study. The design itself also has some limits.

A direct INSERT into Stock_Transaction does not change the batch quantity, so the ledger and the stock agree only when changes go through sp_record_stock_transaction. Closing a discrepancy does not adjust stock. A reservation's status is never changed to EXPIRED automatically. Availability ignores lapsed reservations by comparing times, but a lapsed row stays ACTIVE (sample reservation 5). Users are not linked to a pharmacy, roles are stored but not enforced through database privileges, and there is no customer entity. Latitude and longitude are stored but not used, so results are not sorted by distance, and a LIKE search with a leading wildcard cannot use the name indexes. Finally, the sample data never produces the EXPIRING_SOON and EXPIRED statuses, and it uses only two of the six transaction types.

Future work could add an application layer or application programming interface (API) over the procedures, sorting by distance using the stored coordinates, scheduled events for reservation expiry and expiry alerts, and support for first-expiry-first-out dispensing. Other useful steps are linking users to pharmacies with role-based privileges, testing with larger datasets and indexes, and a usability study. Connecting to real pharmacy systems would be a separate project, and we do not claim it here.

VII. CONCLUSION

This paper presented the Medicine Inventory and Availability Tracking System, a MySQL prototype for finding medicines across several pharmacies. Its 12 tables, six views, five procedures and seven triggers work out availability from batch-level stock, leaving out expired batches, inactive pharmacies and reserved units. They also offer alternatives through a prepared list and an exact ingredient match. On a fictional dataset of five pharmacies and 30 batches, the stock ledger matched the batch quantities in every case, and every rule violation we tested was rejected. The system is a database prototype running on sample data, and we do not claim live pharmacy connectivity. What it offers is a tested and documented set of rules on which an application layer could later be built.

ACKNOWLEDGMENT

The authors acknowledge Dr. Mrunal Pathak as the project guide.

ISSN: **3139-0862**

REFERENCES

[1] M. T. Wondmagegn, E. Nabaasa, E. A. Melesse, and E. A. Kassaw, "A framework for locating prescribed medication at pharmacies," Integr. Pharm. Res. Pract., vol. 12, pp. 127–136, Jun. 2023, doi: 10.2147/IPRP.S415674.

[2] R. P. Sarmento, A. Tarrinho, P. Câmara, and V. Costa, "A system for efficient communication between patients and pharmacies," arXiv:1812.01499, Dec. 2018. [Online]. Available: https://arxiv.org/abs/1812.01499

[3] S. S. Samaan, "Design and implementation of a pharmaceutical inventory database management system," Al-Khwarizmi Eng. J., vol. 13, no. 1, pp. 118–128, Mar. 2017, doi: 10.22153/kej.2017.08.008.

[4] H. Budianto, L. N. Angraeni, N. H. Nabila, and M. Fikri, "Development and quality evaluation of a web-based drug inventory system for pharmacy management," Nusantara J. Artif. Intell. Inf. Syst., vol. 2, no. 1, pp. 27–38, Jun. 2026, doi: 10.47776/nuai.v2i1.1933.

[5] R. Chalmeta, A. Navarro-Ruiz, and L. Soriano-Irigaray, "A computer architecture based on disruptive information technologies for drug management in hospitals," PeerJ Comput. Sci., vol. 9, p. e1455, Jun. 2023, doi: 10.7717/peerj-cs.1455.

[6] E. F. Codd, "A relational model of data for large shared data banks," Commun. ACM, vol. 13, no. 6, pp. 377–387, Jun. 1970, doi: 10.1145/362384.362685.

[7] P. P.-S. Chen, "The entity-relationship model—Toward a unified view of data," ACM Trans. Database Syst., vol. 1, no. 1, pp. 9–36, Mar. 1976, doi: 10.1145/320434.320440.

[8] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.

[9] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. New York, NY, USA: McGraw-Hill, 2019.

[10] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," ACM Comput. Surv., vol. 15, no. 4, pp. 287–317, Dec. 1983, doi: 10.1145/289.291.

[11] Oracle Corporation, "MySQL 8.0 reference manual: InnoDB locking reads." [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/innodb-locking-reads.html (accessed Oct. 31, 2026).