Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 10, pp 794-799, October, 2026

MediTrack: Smart Medicine Tracking and Pharmacy Locator System

Bhuvaneswaran K G[1*], Baranidaran K[1], Arun Raja J[1], Hariprasaath S[1], Ms. Dharaniya N G[2]

[1]*Student, Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India*

[2]*Assistant Professor, Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore-641062, Tamil Nadu, India*

bhuvaneswarankg25it@srishakthi.ac.in

ABSTRACT

Unsafe medicine use is often caused by forgotten doses, unfinished courses, and the use of expired packages, while finding an available medicine requires visits or calls to several pharmacies. This paper presents MediTrack, a full-stack web application that brings medicine search, barcode and QR code identification, a personal medicine cabinet with automatic expiry classification, dose reminders, a stock-aware pharmacy map, and verified order handling into one system. The application serves three roles, namely patient, pharmacy owner, and administrator, and controls access using JSON Web Tokens, salted password hashing, and role-based authorization on both server and client. It is implemented with React, Node.js, Express, and MongoDB, and a dual storage layer switches to a persistent file store when the database is unreachable. A functional test of twenty API scenarios covering authentication, search, scanning, personal data, authorization, ordering, and administration produced the expected response in every case. The work shows that a small open-source stack can deliver an integrated and easily deployable medication management system.

Keywords: Medicine Tracking, Expiry Monitoring, Pharmacy Locator, Role-Based Access Control, JSON Web Token, MERN Stack

1. Introduction

1.1 Background

Medicines are used daily by people of every age, and the safety of their use depends on small routines: taking each dose at the right time, completing a prescribed course, and discarding packages that have passed their expiry date. Patients with long-term conditions often take several medicines at fixed times, and the printed expiry date on a strip is rarely checked. A second difficulty is availability. A patient may have to visit or telephone several shops to learn whether a medicine is in stock and at what price, while pharmacies record stock on paper or in software that customers cannot see.

1.2 Problem statement

General reminder applications know nothing about medicines, and map applications know nothing about what a pharmacy holds. Users therefore move between separate tools for information, tracking, and purchase. A single application that connects medicine information, personal tracking, and pharmacy stock can reduce this effort and improve safe use of medicines (Sommerville, 2016).

1.3 Objectives

The objectives of this work are to design and implement a web application that can:

i. search medicines by name, brand, generic name, or barcode;

ii. identify a medicine by scanning a barcode or QR code with the device camera;

iii. maintain a personal medicine cabinet and classify each item as Safe, Expiring Soon, or Expired;

iv. create and manage dose reminders;

v. show verified pharmacies on a map with live stock and price;

vi. reserve medicines from a pharmacy with verified delivery;

vii. separate patient, pharmacy, and administrator functions using role-based access control; and

viii. continue to operate without a database server through a dual storage strategy.

2. Review of Literature

Medication reminder applications help patients follow a schedule, but they usually treat a medicine as free text and provide no safety information or expiry tracking of the physical package. Electronic pharmacy platforms allow ordering and delivery but are operated mainly by large chains. Machine-readable codes printed on packages are widely used for identification, and browser libraries can now read them with a phone camera (html5-qrcode, n.d.). Token-based authentication (Jones et al., 2015), salted password hashing (Provos & Mazières, 1999), and REST (Fielding, 2000) are established practices for securing and structuring web services.

3. Methodology

3.1 System architecture

MediTrack follows a layered architecture, shown in Fig. 1. The presentation layer is a single-page application built with React (React, n.d.) and routed with React Router, and it renders different pages and navigation for each role. The application layer is a REST interface built with Node.js and Express (Express.js, n.d.). Every protected request passes through a protect middleware that verifies the signed token and loads the user, and an authorize middleware that compares the role of the user with the roles permitted for the route. The data layer stores seven collections, namely users, medicines, pharmacies, inventory, personal medicines, reminders, and orders, in MongoDB through Mongoose (Mongoose, n.d.).

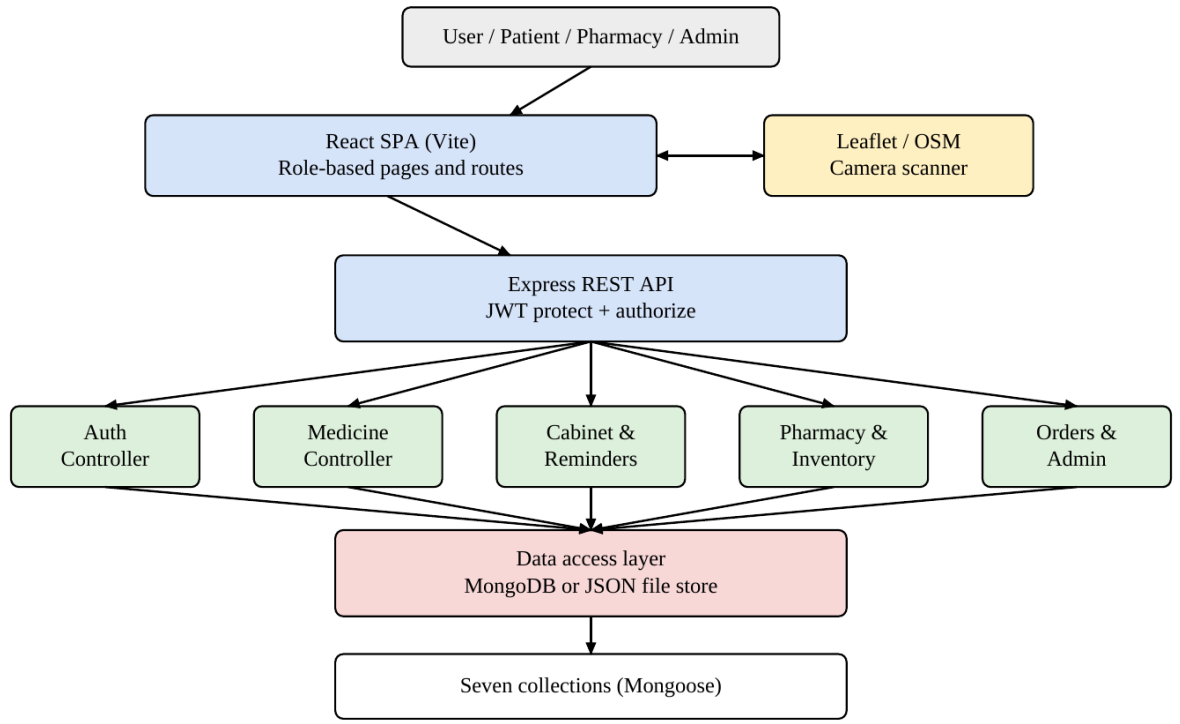


Fig. 1 - System Architecture of the Proposed MediTrack System.

3.2 Technology stack and development environment

The frontend uses React 18 with Vite 5, Axios for HTTP calls, Leaflet with OpenStreetMap tiles (Leaflet, n.d.) for the pharmacy map, and html5-qrcode

(html5-qrcode, n.d.) for scanning. The backend uses Node.js with Express 4 on port 5000 and Mongoose 8 with MongoDB (MongoDB, n.d.). During development the Vite server runs on port 3000 and proxies API calls to the backend. Testing was performed using direct HTTP requests and a supported web browser.

3.3 Roles and access control

A patient searches medicines, scans codes, manages a personal cabinet, reminders, and expiry status, views pharmacies, and places orders. A pharmacy owner registers a shop, maintains the quantity and price of stock, and processes received orders. An administrator manages the master catalogue, views users and platform statistics, verifies pharmacies, and resolves disputes.

Passwords are hashed with bcrypt using a ten-round salt and are excluded from query results by default. A successful login returns a token that expires after thirty days, and the client attaches it to every request through an Axios interceptor. The login request can carry an expected role so that an account cannot enter the wrong portal. In the browser, a ProtectedRoute component redirects unauthenticated visitors and shows an access notice to users whose role is not permitted.

3.4 Database design

The catalogue holds the name, brand, generic name, category, strength, dosage form, description, and warning of each medicine, with a unique sparse barcode field. Personal medicine records store quantity, dosage, frequency, and expiry date. A reminder stores the medicine, dosage instruction, time, frequency, and an enabled flag. An order stores items, total, status, payment method and status, delivery type, a four-digit delivery verification code, a cancellation reason, and a dispute record.

Every controller first tests the Mongoose connection state. When connected, it uses the models; otherwise, it reads and writes a persistent JSON file through two helper functions that use the same field names, so that responses are identical. The database module connects with a two-second selection timeout so that the server starts promptly without MongoDB.

3.5 Expiry and stock classification

The expiry status is not stored but is computed on every read from the number of days remaining, d = ceil((expiry date - current date) / 1 day). A medicine is Expired if $d < 0$, Expiring Soon if $0 \leq d \leq 30$, and Safe otherwise. The availability of a pharmacy item is derived in the same way: Out of Stock if the quantity is zero, Low Stock if it is below ten, and Available otherwise. Computing the status at read time ensures that it can never become stale.

Orders move through the statuses Order Placed, Confirmed, Out for Delivery, Delivered, and Cancelled. The server computes the order total as the sum of price multiplied by quantity, and the customer must give the four-digit code to the delivery person to complete delivery.

Table 1 - REST API endpoint groups of the MediTrack system.

API group	Main operations	Access
/api/auth	Register, login, current user	Public / logged in
/api/medicines	Search, by id, by barcode, create	Public; create: admin
/api/user-medicines	List, add, delete	Logged in
/api/reminders	List, add, toggle, delete	Logged in
/api/pharmacies	List, details with stock, update stock	Public; stock: pharmacy, admin
/api/orders	Create, my orders, shop orders, status	Logged in
/api/admin	Statistics, users, delete medicine	Admin

3.6 Application workflow

The workflow begins when the user logs in or registers and receives a token that carries the role. The patient then searches for or scans a medicine, adds it to the cabinet, and sets reminders. The expiry tracker groups the cabinet into Safe, Expiring Soon, and Expired items. When a medicine is needed, the patient opens the pharmacy map, checks live stock and price, and reserves the medicine. The pharmacy confirms the order and the delivery is completed with the verification code, as shown in Fig. 2.

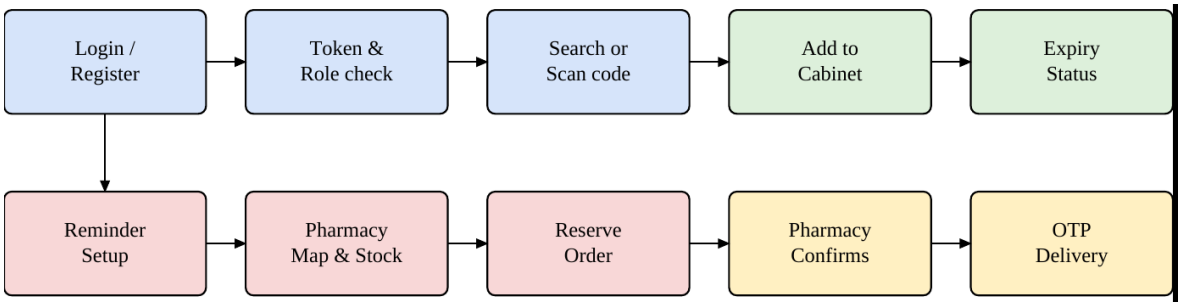


Fig. 2 - Workflow of the Proposed MediTrack System.

3.7 Testing and evaluation

The server was started on a clean copy of the default data with MongoDB not running, so the tests exercise the file-store mode. Twenty requests were issued with direct HTTP calls using demo accounts for the three roles. Unit-level checks covered the expiry and stock rules, and integration-level checks covered the communication between routes, middleware, and storage. Error conditions such as wrong passwords, missing tokens, wrong roles, empty orders, and unknown barcodes were included.

4. Results

4.1 System functionality

The application brings medicine search, scanning, personal tracking, reminders, pharmacy stock, and ordering into a centralized interface. Fig. 3 shows the patient dashboard and the expiry tracker. The dashboard summarizes active medicines, scheduled reminders, expiry warnings, and a link to the pharmacy map, while the tracker reports the number of Safe, Expiring Soon, and Expired medicines.

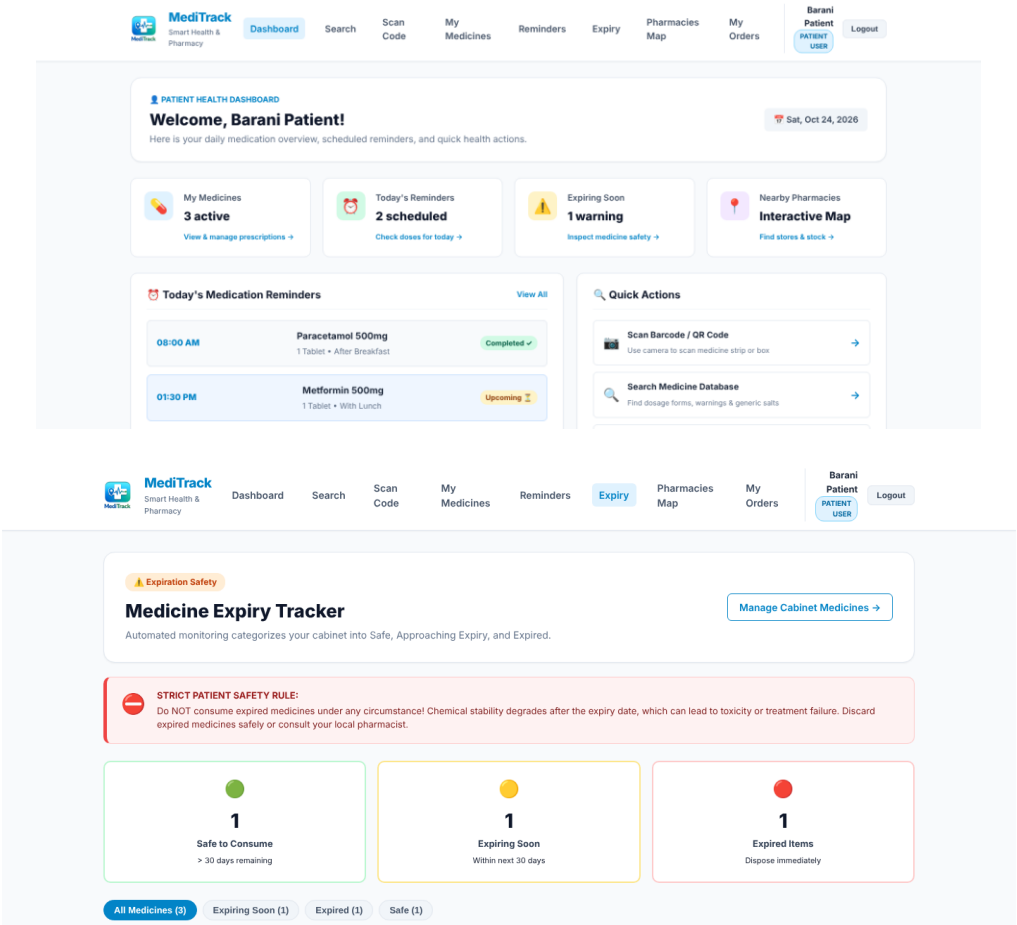


Fig. 3 - Patient dashboard (top) and medicine expiry tracker (bottom).

4.2 Functional evaluation

Representative test scenarios for each module are summarised in Table 2. All twenty requests returned the expected status code and message.

Table 2 - Summary of functional API test results.

Test ID	Module	Test scenario	Observed result
T1	Authentication	Login valid, wrong password, wrong portal, duplicate, register	200 / 401 / 403 / 400 / 201 as expected
T2	Catalogue	Text search, category, barcode found, barcode missing	Correct medicine; 404 with message for unknown code
T3	Personal data	No token, own medicines, add reminder, missing time	401; own records only; 201; 400
T4	Authorization	Patient on admin route, admin route, patient stock update	403; statistics returned; 403
T5	Orders	Valid order, empty order	201 with total 37 for 2 x 18.5; 400
T6	Administration	Create medicine, service status	201 with defaults applied; 200

5. Discussion

Errors are returned consistently: 400 for validation, 401 for missing or invalid tokens, 403 for role violations, and 404 for missing records, each with a readable message. Because every controller handles both storage modes, the same interface worked without change. Deriving expiry and stock status at read time removed the need to schedule status updates.

The system also has limitations. The file store is meant for demonstration and is not suitable for concurrent production use. Payment is recorded as a method and status only, and reminders depend on the browser being open. The order controller writes the status value Pending, which is not in the order model enumeration, and the browser-side fallback accepts demo passwords when the server is unreachable; both should be corrected before production. The evaluation is functional; no usability study with patients or pharmacists has been carried out, and response time was not benchmarked.

6. Conclusion

MediTrack integrates medicine information, personal tracking, expiry monitoring, reminders, a stock-aware pharmacy map, and verified ordering in a single role-based web application built from open-source components. Token-based authentication, hashed passwords, and server-side role checks protect the data, and the dual storage layer lets the system run with or without a database server. The functional tests confirmed the intended behaviour.

Future development can add a payment gateway, push and SMS notifications, prescription upload, drug-interaction warnings, a mobile application, real-time delivery tracking, multilingual support, and a usability study with patients and pharmacists.

Acknowledgements

The authors would like to thank Sri Shakthi Institute of Engineering and Technology for providing the academic environment and facilities required for this project, and the Head of the Department of Information Technology and the faculty members for their guidance and support.

References

Express.js. (n.d.). *Express: Fast, unopinionated, minimalist web framework for Node.js*. https://expressjs.com/

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Doctoral dissertation, University of California, Irvine].

html5-qrcode. (n.d.). *Cross platform QR code and barcode scanning library*. https://github.com/mebjas/html5-qrcode

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC 7519). Internet Engineering Task Force. https://doi.org/10.17487/RFC7519

Leaflet. (n.d.). *Leaflet: An open-source JavaScript library for interactive maps*. https://leafletjs.com/

MongoDB. (n.d.). *MongoDB manual*. https://www.mongodb.com/docs/

Mongoose. (n.d.). *Mongoose: Elegant MongoDB object modeling for Node.js*. https://mongoosejs.com/

Provos, N., & Mazières, D. (1999). A future-adaptable password scheme. In *Proceedings of the USENIX Annual Technical Conference*.

React. (n.d.). *React: The library for web and native user interfaces*. https://react.dev/

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.