International Journal of
Science, Architecture, Technology and Environment

Design and Implementation of a Web-Based Medicine Reminder System Using Spring Boot, JPA/Hibernate and MySQL for Medication Adherence Support

M. Elakkiya[1*], S. Bhuvaneshwari[2], D. Gopika[3], V. Bharathapriya[4]

[1]*Assistant Professor, Department of Computer Science and Applications, Vivekanandha College of Arts and Sciences for Women (Autonomous)*

[2,3,4]*B.Sc Computer Science, Department of Computer Science and Applications, Vivekanandha College of Arts and Sciences for Women (Autonomous)*

[*]*Corresponding author: elakkiyase@gmail.com*

DOI: https://doi.org/10.63680/ijsate112607.12

Abstract

Missed and delayed doses are a leading cause of reduced treatment effectiveness in long-term therapy. This paper presents the design, implementation and functional verification of a web-based Medicine Reminder System that helps users register, record medicine details (name, disease, dosage, type, start/end date, daily time and expiry date), receive time-triggered reminders with AM/PM scheduling, and keep a history of doses marked as taken, skipped or missed. The system follows a layered client–server architecture: an HTML/CSS/JavaScript front end communicates through REST APIs with a Spring Boot back end, which persists data in MySQL through Spring Data JPA and Hibernate. A four-table relational schema (User, Medicine, Schedule, Medication History) and a three-level data flow model are described. The implementation was exercised with unit, integration, functional, validation and user-acceptance test designs, and its working screens and start-up behaviour are reported. The dashboard computes per-user indicators such as total medicines, doses taken, doses missed and an adherence percentage, and flags medicines approaching expiry. Limitations of the current browser-based reminder mechanism are discussed, together with a roadmap towards push notifications, automatic missed-dose marking, caregiver alerts and security hardening.

Keywords: Medication adherence; Medicine reminder; mHealth; Spring Boot; REST API; JPA/Hibernate; MySQL; Web application

1. Introduction

Adherence to prescribed therapy is a central determinant of clinical outcomes. The World Health Organization reports that adherence to long-term therapy for chronic illness averages only about 50% in developed countries and is lower in developing ones, and that poor adherence undermines the effectiveness of health systems [1]. Patients who take several medicines at different times of the day must remember the name, dosage, schedule and expiry of each, which is difficult to sustain by memory, handwritten notes or a generic phone alarm.

Mobile and web applications are a low-cost way to provide reminders and self-monitoring. Systematic reviews report that apps for medication management generally improve adherence and are found useful by

patients [3], and a review of randomised trials has examined their efficacy across conditions [4]. At the same time, a large store-level review found that only a small fraction of adherence apps document an evidence base or involvement of health professionals [2], so well-structured, transparent implementations remain valuable, particularly for educational and regional deployments.

This paper describes a Medicine Reminder System developed as a web application. The contributions are: (i) a layered architecture that separates presentation, application logic, persistence and data; (ii) a relational model that links users, medicines, schedules and dose history; (iii) a reminder workflow with AM/PM scheduling, repeated prompting until a dose status is recorded, and Taken/Skipped/Missed history; (iv) expiry-date monitoring and dashboard adherence indicators; and (v) a description of the testing strategy, observed behaviour and current limitations.

2. Related Work

Evidence on adherence. The WHO review frames non-adherence as a major global problem requiring multi-component interventions [1]. Pérez-Jover et al. reviewed studies in which mobile apps supported home medication management and concluded that apps help increase adherence; seven of the eleven included studies confirmed an increase [3]. Armitage et al. conducted a systematic review and meta-analysis of randomised trials of adherence apps [4], and the MedISAFE-BP randomised trial examined a smartphone app in relation to adherence and blood pressure control [5].

Quality of existing apps. Ahmed et al. screened 5,881 apps in two app stores, tested 805, and found that only four of the 420 free apps had a documented evidence base [2]. Santo et al. assessed reminder apps with the Mobile App Rating Scale and identified 272 candidate apps, concluding that proven-effective apps could influence clinical practice [6]. These findings motivate transparent designs that expose their functions and data model.

Reminder system designs. Simalatsar et al. presented the safe embedded implementation of a portable reminder device with personalised regimens for a cancer drug [7]. Ronglong et al. proposed a reminder and feedback architecture for elderly users combining voice over IP, SMS and web services [8]. Catindoy et al. built an Android reminder and tracker in which users enter medicine, dosage and time and receive health tips [9]. Table 1 summarises these works against the present system.

Table 1: Comparison with related reminder systems

Work	Platform	Main features	Gap
[7]	Portable device	Personalised regimen, embedded safety	Single drug
[8]	VoIP/SMS/web	Reminder and feedback for elders	Design only
[9]	Android	Manual entry, reminders, tips	No expiry or history analytics
Proposed	Web (Spring Boot)	Reminders, AM/PM, history, expiry, dashboard	Browser must stay open

Back-end technology. The choice of Spring Boot for REST services is supported by performance studies: Shyam Mohan and Goswami found that Spring Boot outperformed Node.js for POST, PUT and DELETE operations under higher user loads, while Node.js was faster for GET [11], and Di Meglio and Starace compared the performance and resource consumption of REST frameworks and execution environments [10]. Create, update and delete operations dominate a medicine-management workload, which suits this choice.

3. System Requirements and Environment

The system was developed and run on a conventional desktop: an Intel Core i3-class or better processor, 4 GB RAM (8 GB preferred), 10 GB free disk and an HD display, running Windows 10/11. The software stack is summarised in Table 2.

Table 2: Software stack

Layer	Technology
Front end	HTML, CSS, JavaScript (Fetch API for REST calls)
Back end	Java, Spring Boot (embedded Tomcat, port 8080)
Persistence	Spring Data JPA with Hibernate ORM
Database	MySQL; MySQL Workbench for administration
Tools	Visual Studio Code, Maven, Chrome/Edge, JUnit, Mockito

4. System Analysis and Design

4.1. Problem Analysis

Manual approaches such as notes, calendars and basic alarms suffer from five limitations: dependence on memory, no consolidated medicine record, difficulty handling several schedules, no systematic record of taken, skipped or missed doses, and manual expiry checking. The proposed system addresses each by centralising medicine data, automating reminders and persisting dose history.

4.2. Architecture

The system follows a client–server, four-layer architecture (Figure 1). The browser front end issues REST requests; Spring Boot controllers receive them, services apply validation and business rules, and repositories built on Spring Data JPA translate operations to SQL through Hibernate [12], [13]. Data are stored in MySQL [14]. The request path is: User → Front end → REST API → Spring Boot → JPA/Hibernate → MySQL.

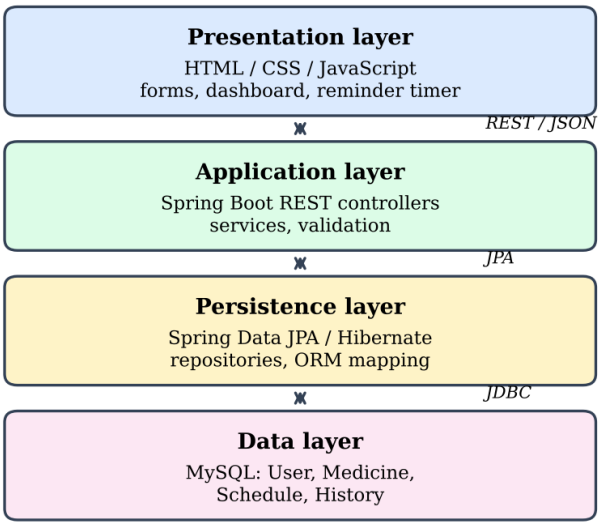


Figure 1: Layered architecture of the Medicine Reminder System

4.3. Data Flow Model

The Level-0 context diagram contains one external entity (User), one process (Medicine Reminder App) and the MySQL data store. Level 1 decomposes the process into User Management, Medicine Management, Reminder Management and History Management, each backed by its own data store (D1–D4); Reminder Management additionally emits alerts to a notification service. Level 2 expands Medicine Management into add, view, update and delete operations on the medicine store.

4.4. Functional Workflow

After registration or login the user reaches a dashboard, adds a medicine, and selects its time using AM/PM. The application continuously compares the current time with scheduled times and raises an alarm or notification when they match. If the dose is not acknowledged the reminder is shown again until a status is recorded; once the user marks the dose Taken, Skipped or Missed, the record is saved to history and can later be viewed. The user may continue managing medicines or log out.

4.5. Input and Output Design

Inputs are registration data (name, email, phone, age, password with a minimum length), login credentials, medicine details (disease, medicine name, dosage, type, start date, end date, daily time, expiry date), dose status, and update/delete requests. Client-side validation checks required fields and formats before the request is sent. Outputs are the dashboard overview, medicine list, today's schedule, reminders, history and expiry information, together with success and error messages.

4.6. Database Design

The relational schema comprises four tables (Table 3). One user owns many medicines; a medicine has one or more schedules and many history records, giving the chain User → Medicine → Schedule → Medication History.

Table 3: Database tables

Table	Principal attributes
User	User ID (PK), name, email, password, age
Medicine	Medicine ID (PK), User ID (FK), name, disease, dosage, expiry date, time
Schedule	Schedule ID (PK), Medicine ID (FK), date, time, AM/PM, reminder status
History	History ID (PK), Medicine ID (FK), date, scheduled time, status, updated time

5. Implementation

Front end. HTML defines the registration, login, dashboard, medicine, schedule and history pages; CSS provides the card-based layout; JavaScript performs validation, asynchronous calls and the reminder timer. Medicine data are sent as JSON with the Fetch API [16], and in-page alerts can be complemented by the browser Notifications API [15].

Back end. A Medicine entity (identifier, medicine name, disease, dosage, expiry date and medicine time) is mapped to a table by JPA. A repository interface extending JpaRepository provides save, find, update and delete operations without hand-written SQL, and a REST controller mapped to /api/medicines exposes a POST endpoint that stores a medicine and a GET endpoint that returns all medicines. Database connection and automatic schema update are configured in application.properties.

Reminder logic. Each stored schedule is compared with the current time at regular intervals on the client. When the time matches, an alarm or notification is raised and the user records Taken, Skipped or Missed;

the status and timestamps are written to the history table. Expiry dates are compared with the current date so that medicines expiring within 30 days appear in an "Expiring soon" panel on the dashboard.

6. Testing Strategy

Five levels of testing were designed (Table 4). Unit tests use JUnit, with Mockito for isolating dependencies, to check registration, login, medicine CRUD, schedule storage, reminder triggering, history status and date handling [12]. Integration tests verify the Front end–REST–Spring Boot–JPA–MySQL path. Functional tests confirm that each requirement behaves as specified, validation tests confirm that invalid input is rejected with clear messages, and acceptance tests assess ease of use and reliability of reminders for end users.

Table 4: Representative test cases

Level	Scenario	Expected outcome
Unit	Register with invalid email	Rejected with message
Unit	Save medicine with expiry date	Date stored correctly
Integration	Submit medicine form	Row present in MySQL
Functional	Schedule time reached	Alarm raised
Functional	Mark dose Skipped	History shows Skipped
Validation	Empty dosage field	Error displayed
Acceptance	New user adds a medicine	Completed unaided

7. Results and Discussion

Start-up behaviour. Running the Maven Spring Boot goal compiled the project and started embedded Tomcat on port 8080; the log shows initialisation of the JPA EntityManagerFactory, registration of the DispatcherServlet and application start-up in about 6.3 seconds on a development laptop. The log also notes that no JTA platform was configured, which is expected for a single-database application.

Working screens. The application, branded MedTrack in its interface, provides a Create Account page (name, email, phone, age, password), a Login page, a Dashboard, an Add Medicine page (disease, medicine name, dosage, type, start and end dates, daily time, expiry date) and a Today's Schedule page with a "Reminders enabled" indicator. The dashboard summarises the user's profile, the total number of medicines, doses taken, doses missed and an adherence percentage; in the captured session it showed five medicines, three doses taken, three missed and 38% adherence, and an "Expiring soon" panel reporting that no medicines expired within 30 days. These values illustrate the indicator set rather than a clinical evaluation.

Discussion. Compared with note-keeping or generic alarms, the system keeps medicine details, schedules, expiry information and dose history together, which addresses the limitations identified in Section 4.1. Its layered design allows each tier to be tested and replaced independently, and its use of Spring Boot is consistent with reported strengths of the framework for write-heavy REST workloads [10], [11]. Because evidence for adherence apps is generally limited [2], a controlled study measuring adherence before and after use would be the logical next evaluation step.

Limitations. (i) Reminders run in the browser and the interface advises keeping the application open in Chrome, so alarms cannot fire when the page is closed; (ii) missed doses are not yet marked automatically; (iii) password handling, authentication and transport security must be hardened before real-world use, and database credentials must be kept out of source control and screenshots; (iv) no clinical outcome study has yet been performed.

8. Conclusion and Future Work

A web-based Medicine Reminder System was designed and implemented with an HTML/CSS/JavaScript front end, a Spring Boot REST back end, JPA/Hibernate persistence and a MySQL database. It centralises medicine records, provides AM/PM-based reminders, records Taken, Skipped and Missed history, tracks expiry dates and summarises adherence on a dashboard. Future work includes Android/iOS clients with push notifications and vibration, server-side scheduling so reminders do not depend on an open browser, automatic missed-dose marking, expiry alerts, daily/weekly/custom schedules, caregiver notification, prescription upload, periodic adherence reports, cloud deployment, stronger authentication and encryption, sharing of records with clinicians, and a controlled study of adherence outcomes.

Acknowledgments

The authors thank the Department of Computer Science and Applications, Vivekanandha College of Arts and Sciences for Women (Autonomous), for providing the facilities used in this work.

Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship and publication of this article.

Funding

The authors received no financial support for the research, authorship and publication of this article.

References

1. World Health Organization, Adherence to Long-Term Therapies: Evidence for Action. Geneva, Switzerland: WHO, 2003.
2. I. Ahmed et al., “Medication adherence apps: Review and content analysis,” JMIR mHealth uHealth, vol. 6, no. 3, p. e62, Mar. 2018.
3. V. Pérez-Jover, M. Sala-González, M. Guilabert, and J. J. Mira, “Mobile apps for increasing treatment adherence: Systematic review,” J. Med. Internet Res., vol. 21, no. 6, p. e12505, Jun. 2019, doi: 10.2196/12505.
4. L. C. Armitage, A. Kassavou, and S. Sutton, “Do mobile device apps designed to support medication adherence demonstrate efficacy? A systematic review of randomised controlled trials, with meta-analysis,” BMJ Open, vol. 10, no. 1, 2020.
5. M. Morawski et al., “Association of a smartphone application with medication adherence and blood pressure control: The MedISAFE-BP randomized clinical trial,” JAMA Intern. Med., vol. 178, no. 6, p. 802, 2018, doi: 10.1001/jamainternmed.2018.0447.
6. K. Santo et al., “Mobile phone apps to improve medication adherence: A systematic stepwise process to identify high-quality apps,” JMIR mHealth uHealth, vol. 4, no. 4, 2016.
7. A. Simalatsar, R. Bornet, W. You, Y. Thoma, and G. De Micheli, “Safe implementation of embedded software for a portable device supporting drug administration,” in Proc. IEEE Int. Conf. Bioinformatics Bioeng. (BIBE), Boca Raton, FL, USA, Nov. 2014, pp. 257–264.

8. S. Ronglong, C. Sookplang, C. Arpnikanondt, and V. Vanijja, “Design of a medication reminder and feedback system for Thai elders,” in Proc. IEEE Int. Conf. Comput. Inf. Sci. (ICCIS), 2012.
9. V. O. Catindoy et al., “Medication reminder and tracker application,” in Proc. 7th Int. Conf. Vocational Educ. Elect. Eng. (ICVEE), Oct. 2024, pp. 341–347.
10. S. Di Meglio and L. L. L. Starace, “Evaluating performance and resource consumption of REST frameworks and execution environments: Insights and guidelines for developers and companies,” IEEE Access, vol. 12, 2024.
11. J. S. Shyam Mohan and K. Goswami, “Performance analysis and comparison of Node.js and Java Spring Boot in implementation of RESTful applications,” Softw. Pract. Exper., vol. 55, no. 7, p. 1209, 2025, doi: 10.1002/spe.3418.
12. VMware, “Spring Boot documentation.” [Online]. Available: https://docs.spring.io/spring-boot/documentation.html (accessed Nov. 3, 2026).
13. VMware, “Spring Data JPA reference.” [Online]. Available: https://docs.spring.io/spring-data/jpa/reference/ (accessed Nov. 3, 2026).
14. Oracle, “MySQL 8.4 reference manual.” [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/ (accessed Nov. 3, 2026).
15. Mozilla, “Notifications API,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API (accessed Nov. 3, 2026).
16. Mozilla, “Fetch API,” MDN Web Docs. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (accessed Nov. 3, 2026).