

An Intelligent Mobile Framework for OCR-Based Medicine Recognition and Voice-Assisted Medication Adherence

Nisha Pandey

Department of Electronics and
Computer Engineering,
Pashchimanchal Campus, IOE, TU.
Pokhara, Nepal
pandeynisha113@gmail.com

Smriti Rana

Department of Electronics and
Computer Engineering,
Pashchimanchal Campus, IOE, TU.
Pokhara, Nepal
ranasmriti30@gmail.com

Trishna Lamsal

Department of Electronics and
Computer Engineering,
Pashchimanchal Campus, IOE, TU.
Pokhara, Nepal
lamsaltrishna@gmail.com

Nabin Lamichhane

Department of Electronics and
Computer Engineering,
Pashchimanchal Campus, IOE, TU.
babunabin@gmail.com

Bidhya Koirala

University of the Sunshine Coast
Queensland, Australia.
bidhya.koirala@research.usc.edu.au

Abstract - Medication negligence is a major problem in long-term illness management. Nearly half of patients with long-term conditions do not take their medicines as prescribed. This problem is worse in low-resource settings, where poor label readability and limited pharmacist access make adherence harder. Most mobile reminder apps rely on manual data entry and generic alarms. They provide small help to the users who are unable to read small print or manage complex drug schedules. This paper presents a mobile health framework for medication management. An on-device text recognition (OCR), confidence-based quality checking, fuzzy medicine matching, and voice-assisted reminders are integrated based on the paper. Flutter, Firebase, and a three-layer Clean Architecture are used for building this system. An on-device OCR engine is used to read the medicine labels, and a weighted confidence score is used to check the recognition quality. It then matches the extracted text against an 11,825-record medicine directory using a six-rule fuzzy-matching engine. Reminders are delivered through native Android alarms paired with text-to-speech playback, supporting elderly and visually impaired users. Testing across six image conditions shows recognition reaching 98% success under favourable conditions, with predictable performance drops in poor conditions. A local caching layer cuts medicine-search time from 10–30 seconds to under 100 milliseconds. Reminder notifications reached 100% delivery reliability across 200 test alarms. These results show that combining lightweight on-device processing with accessible design can support faster and more reliable medicine identification and reminder delivery in resource-constrained settings; the framework's effect on real-world medication-taking behaviour was not evaluated in this study and is identified as a direction for future clinical or user research.

Keywords - Medication Adherence, Mobile Health, Optical Character Recognition, Fuzzy String Matching, Voice Interaction, Clean Architecture, Accessibility, Text-to-Speech

I. INTRODUCTION

Medication adherence means taking medicines exactly as prescribed. It remains one of the hardest problems to solve in chronic disease care. The World Health Organization estimates

that only about half of patients in developed countries take their medicines correctly [1]. The rate is likely lower in developing countries, where health-system support is weaker [1]. Recent studies confirm this problem has not improved much over the past two decades [2]. Missed, duplicated, or mistimed doses can reduce treatment benefits and cause harmful side effects. Over time, poor adherence also weakens patient trust in treatment. Mobile health (mHealth) tools are a common response to this problem. Smartphones are now widely available, even in low-income regions. Systematic reviews show that mobile adherence apps generally improve outcomes, especially when they combine reminders with tracking and education [3], [4]. A twelve-month trial of a commercial reminder app reported real gains in both adherence and clinical outcomes [9]. Similar benefits appear in interventions for diabetes and heart disease [6], [7]. However, broader reviews caution that much of this evidence comes from small, short-term trials [3], [8]. Reminders alone are not enough. How easily a patient can identify their medicine, and act on a reminder, also matters.

Most reminder apps assume users can accurately type in a medicine's name, strength, and schedule. This assumption often fails for elderly or visually impaired users, who struggle to read small text on medicine labels. Optical character recognition (OCR) solves this by letting a camera read the label instead of the user. Automated label reading works well, even on real-world pharmacy labels, according to recent deep learning OCR systems [15]. Engines like Google's ML Kit, an on-device engine, can avoid the need for a constant internet connection [20], which matters for users with limited connectivity.

Another dimension of accessibility is added through voice interaction. Studies show that voice assistants lower the effort needed to access health information, especially for elderly and visually impaired users [14]. Yet few mobile apps combine OCR-based label reading with voice-based reminders in a single, connected workflow.

This gap is larger in low-resource settings. A survey of healthcare workers in Nepal found that data cost and limited

digital literacy are real barriers to adopting new mobile tools [13]. Nepal's medication landscape reflects this challenge, since many patients manage multiple long-term prescriptions with limited support tools. To address this gap, this paper proposes a mobile framework, implemented as the MediMate application. It combines on-device OCR, a confidence-based quality check, fuzzy medicine matching, and voice-assisted reminders in one workflow. A single camera scan flows through recognition, matching, and reminder creation with minimal user input. The system also remains functional without a constant internet connection.

The main contributions of this paper are:

- A confidence-aware OCR pipeline that combines image quality, text quality, and orientation signals into one score, used to flag unreliable scans.
- A six-rule fuzzy-matching engine that resolves OCR errors against an 11,825-record medicine directory.
- A two-level local caching system that cuts medicine-search time from seconds to milliseconds.
- A native Android alarm and voice-reminder system that stays reliable under battery-saving restrictions, tested across 200 alarms.
- An evaluation of the complete system across varied scan conditions, search performance, and structured end-to-end workflows.

II. RELATED WORK

Several studies have confirmed that mobile apps can improve medication adherence. Systematic reviews report modest but consistent improvements across chronic-disease populations [3], [4]. Combining reminders with dose tracking, rather than offering reminders alone, is believed to create a bigger difference. This holds across diabetes [6], heart disease [7], and broader cardiovascular disease [11], where reviews report adherence gains when reminders pair with structured monitoring. A twelve-month randomized trial of a commercial app found these gains can be sustained over time, not just short-term [9]. COPD-focused reviews report similar results when tracking includes symptom monitoring [12]. Although large, well-designed trials remain rare, AI-enabled adherence tools show early promise for personalized risk prediction [10].

Despite this progress, reviews consistently flag two gaps. Most apps depend on manual data entry, with no way to populate records from a printed label [4], [5]. Apps are also rarely designed for elderly or visually impaired users, despite this group forming a large share of long-term medicine users. Reviews from low-resource settings further highlight cost and connectivity barriers that high-income studies often overlook [13].

Several studies apply computer vision to automate label reading. Object-detection models such as YOLO, combined with text recognizers, have been used to locate medicine names on prescriptions before recognition, improving accuracy over unsegmented images [15]. Comparative studies of detection models, including RetinaNet and SSD, report meaningful accuracy and speed differences depending on the chosen architecture [15]. In a related clinical study, a smartphone-based deep-learning model helped visually impaired patients identify medication bottles more accurately than they could unaided [16]. Especially for elderly and visually impaired

users, these findings back up using camera-based recognition instead of manual transcription. This framework uses Google's ML Kit, an on-device recognition engine that runs without a network connection [20]. A known limitation of such general-purpose engines is that they return raw text without a confidence score, leaving each application to build its own reliability check. Recognition noise is also common, since OCR often misreads similar-looking characters. Classical edit-distance matching remains a standard method for correcting this kind of error [19]. The fuzzy-matching engine described in Section III extends this idea with rules suited to medicine names.

Voice interfaces are a well-studied accessibility tool for users who find text-based interaction difficult. Reviews of voice assistants in healthcare report that they reduce the effort needed to access medication information and support hands-free use, particularly for older adults [14]. This underscores the value of pairing a screen-free, spoken output channel with a system whose core task, reading a label, depends on vision the target users may not reliably have.

Cross-platform frameworks like Flutter make it cheaper to build accessible healthcare apps for both major mobile platforms from one codebase [17]. Backend platforms such as Firebase provide authentication and secure, structured data storage without needing custom server infrastructure [21]. Reliable background execution is a further Android-specific challenge: newer Android versions restrict background alarms to save battery, and official guidance recommends exact-alarm scheduling for time-sensitive reminders such as medication alerts [22].

Together, this literature points to a clear gap. Adherence apps that combine reminders with tracking outperform bare alarms, but almost none include camera-based label recognition. On-device OCR and fuzzy matching are mature individually, yet their combination into one confidence-gated, offline-capable pipeline for medicine labels is rarely reported. Accessibility for elderly, visually impaired, and low-connectivity users, especially in settings like Nepal, remains a secondary concern rather than a core design goal.

Positioning this contribution against the most closely related recent systems clarifies what is new here. On-device text recognition [15], [20], edit-distance-based text correction [19], and voice-based health interfaces [14] are each independently well established. Recent OCR-based medicine-identification apps illustrate one way to combine them: a 2025 system for visually impaired users integrates a cloud OCR service with a large language model to interpret label text and read it back, reporting high recognition precision on a small evaluation set together with a structured usability study [24]. That design depends on network access at the point of use, and treats matching as open-ended language interpretation rather than lookup against a fixed, auditable directory. The present framework instead targets low-connectivity settings directly: recognition, confidence estimation, and matching all run on-device against a versioned local medicine directory, and a match is only accepted once a candidate clears an explicit, adjustable acceptance threshold (Eq. 6), rather than being generated by a language model. The specific technical contribution is therefore not OCR, fuzzy matching, or voice output in isolation, but their integration into one confidence-gated, offline-capable pipeline purpose-built for a bounded medicine directory rather than open-ended text interpretation.

III. METHODOLOGY

A. System Overview and Architecture

The proposed framework is a cross-platform mobile application which is built with Flutter [23]. It follows a three-layer Clean Architecture model [18]. The presentation layer uses Riverpod-managed widgets and controllers. The domain layer holds framework-independent business rules and entities. The data layer connects to Firebase-backed repositories and local storage. Each layer depends only on the layer beneath it, so domain logic can be tested independently of any external service. Fig 1 shows the overall system architecture. User interaction flows from the presentation layer, through state management, into a backend built from Firebase Authentication, Cloud Firestore, and Cloud Functions. External services, the OCR engine and the text-to-speech engine, are called directly from the client. Fig 2 summarises the full workflow, from authentication through label capture, medicine matching, reminder scheduling, and dose logging.

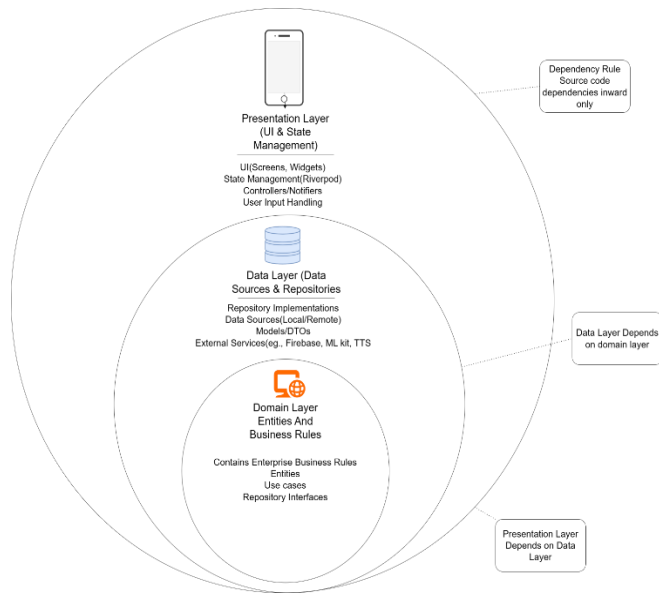


Fig. 1. System architecture

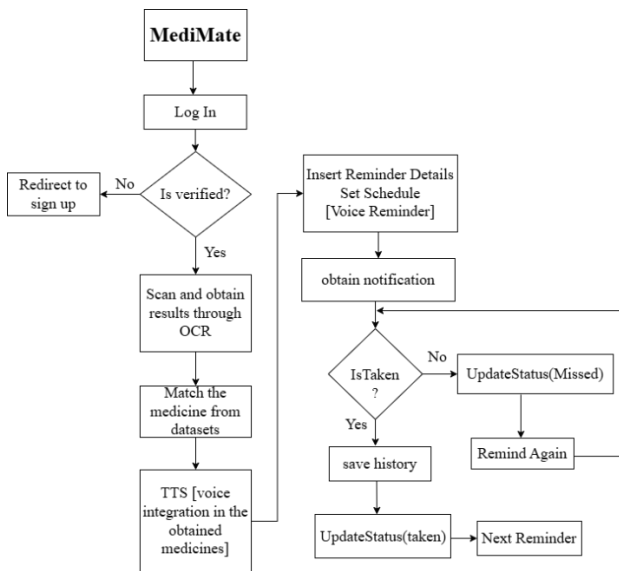


Fig. 2. End-to-end system workflow, from authentication through scan, match, reminder, and dose logging

B. Medicine Directory and Dataset

The recognition pipeline matches extracted text against a local medicine directory of 11,825 records. The records were obtained from a Kaggle dataset and cover commonly available medicines. A subset of 200 records was verified with the assistance of a pharmacist. Each record follows a fourteen-field schema, including a list of search-term aliases. These aliases support partial and colloquial matching, such as local abbreviations for a single medicine. A dedicated CSV-import tool converts each row into a structured medicine entity and logs any import failures for review.

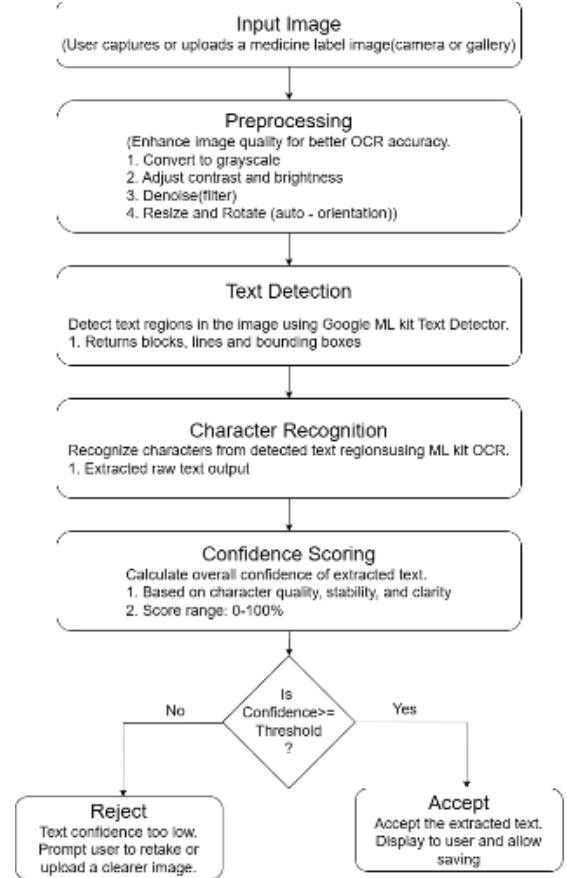


Fig. 3. Flow of OCR (recognition pipeline)

C. On-Device OCR Pipeline and Confidence Estimation

Fig 3 shows the recognition pipeline. A captured or gallery-selected image is pre-processed first. This includes converting to greyscale, adjusting contrast and brightness, denoising, and auto-rotating the image. The pre-processed image then passes to an on-device text-recognition engine, Google ML Kit Text Recognition [20]. This engine performs detection and recognition entirely on the device, without needing a network connection. On-device recognisers of this type typically follow three stages. The first is convolutional feature extraction, where learned filters produce activation maps:

$$y(i,j) = \sum_u \sum_v w(u,v) \cdot x(i+u, j+v) + b, 0 \leq u,v < K \quad (1)$$

This is followed by a rectified-linear activation, $\text{ReLU}(z) = \max(0, z)$, and spatial pooling. The second stage is sequence modelling, commonly a bidirectional recurrent network. Its forward and backward hidden states combine as:

$$h_t = [h \rightarrow; h \leftarrow] \quad (2)$$

Transformer-based variants instead use scaled dot-product attention:

$$Attention(Q, K, V) = softmax(QK^T / \sqrt{d_k}) \cdot V \quad (3)$$

The third stage is Connectionist Temporal Classification (CTC) decoding. This stage sums over all label alignments π consistent with a target sequence y :

$$L_CTC = -\log \sum_{\pi \in B^{-1}(y)} \prod_{t=1}^T p(t, \pi) \quad (4)$$

This lets the recogniser output variable-length text without needing pre-segmented characters. The description above covers the general architecture family for on-device recognisers of this type. This study treats the recogniser as a fixed, pre-trained component; it is not re-trained or fine-tuned. Instead, the study focuses on the reliability layer built around its output. ML Kit does not return a per-field confidence score for recognised text. To address this, the framework computes its own composite confidence estimate directly from the extracted text and image. Three normalised sub-scores combine into one weighted score. Image quality (Q_img) comes from variance-based blur detection, plus brightness and contrast statistics. Text quality (Q_txt) comes from the ratio of recognised, medicine-relevant tokens to noise characters and symbols. Orientation consistency (Q_orient) comes from line-alignment regularity. These combine as:

$$C_ocr = 0.40 \cdot Q_img + 0.35 \cdot Q_txt + 0.25 \cdot Q_orient \quad (5)$$

Scans scoring above an operating threshold are accepted and passed to the matching engine in Section III-D. Scans below the threshold prompt the user to recapture the image, with a categorical high / medium / low / insufficient label shown in the interface. Table I in Section IV reports observed accuracy across six representative image conditions.

For the controlled OCR evaluation, image-capture and scanning were performed on an OPPO A78 smartphone (model CPH2565) running ColorOS 15.0. This device uses a Qualcomm Snapdragon 680 octa-core processor, 8GB RAM with an additional 4GB extended memory configuration, and 256 GB internal storage.

D. Fuzzy Medicine-Matching Engine

Accepted OCR text is normalised first: lower-cased, corrected for common single-character OCR confusions (e.g., '0'↔'o', '1'↔'l', '5'↔'S'), and standardised for dosage formatting (e.g., '500mg' → '500mg'). The normalised text is then scored against every candidate medicine record using six weighted rules, applied from highest to lowest specificity: exact brand-name match (weight 1.00), generic-name match (weight 0.95), search-term/alias match (weight 0.85), partial word-prefix match (weight 0.70), character-level fuzzy similarity based on shared character sequences (weight 0.60) [19], and manufacturer-name match (weight 0.50). Each candidate's aggregate score is normalised against the maximum possible combined weight:

$$S_match = (\sum_{i=1}^6 w_i \cdot r_i) / W_max, W_max = 4.60 \quad (6)$$

Here, $r_i \in [0,1]$ indicates how well rule i is satisfied; this value can be fractional for the fuzzy-similarity rule. Candidates scoring $S_match \geq 0.20$ are returned (the weights were set during development and were not subjected to formal sensitivity analysis), ranked by descending score. If exactly one candidate exceeds this threshold, the interface goes directly to its detail view instead of showing a result list.

E. Local Caching Architecture

During development, resolving every scan directly against Cloud Firestore introduced search delays of several seconds to tens of seconds on typical mobile networks. This delay was unacceptable right after a time-sensitive camera interaction. The framework instead uses a two-level local cache. On first use, the full medicine directory is downloaded once and saved as a local JSON file. On later scans, matching happens entirely against this local copy. An in-memory inverted index, built over medicine names, generic names, and search-term aliases, narrows the candidate set before the weighted-scoring pass in Section III-D runs. Cache population happens in the background and does not block OCR scanning, which continues in parallel. Section IV-B reports how this design affects search latency.

F. Reminder Scheduling and Voice-Assisted Notification

Reminder delivery uses the native Android AlarmManager, not Flutter-level timers. Android's background-execution policies, introduced from Android 6.0 (Doze mode) onward, defer or suppress non-exact alarms. This behaviour is unacceptable for a medication reminder [22]. The implementation requests the SCHEDULE_EXACT_ALARM and USE_EXACT_ALARM permissions required on Android 12 and above. A Kotlin BroadcastReceiver manages alarm delivery, and a BOOT_COMPLETED receiver re-registers all active alarms after a device restart. All scheduling uses a fixed Asia/Kathmandu timezone, avoiding errors from device locale settings. When a reminder fires, the notification shows two direct actions, Taken and Missed, each logged with an exact timestamp to Cloud Firestore. The dose-tracking workflow records whether a scheduled dose is registered as a taken or a missed. When a dose is recorded as a taken, the corresponding reminder cycle is completed. When a dose is recorded as a missed, the scheduler creates subsequent reminder attempts for the next five scheduled opportunities after every three minutes, continuing until a taken is recorded or the retry limit is reached. Each notification pairs with on-device text-to-speech playback. Dosage abbreviations are expanded (e.g., 'mg' → 'milligrams'), and speech plays at a reduced rate to improve intelligibility for elderly listeners. This lets a user register the reminder's content without needing to view or read the screen.

G. Security, Privacy, and Offline Support

User data is scoped through Firebase Authentication combined with Cloud Firestore security rules. These rules restrict access to each document to requests from its owning user [21]. Text-to-speech playback and OCR inference both run entirely on-device, so label images and their extracted text are never sent to a third-party inference service. The application remains partially functional offline: OCR scanning and the local medicine cache work without connectivity, and previously synced reminders continue to display and fire while offline. Only scan and dose-log records require an eventual sync back to Firestore.

H. Failure-Mode Handling and Parameter Selection

The pipeline is designed to fail safely rather than silently, and three categories of failure are handled explicitly. First, when the composite confidence score C_ocr (Eq. 5) falls below the operating threshold, the scan is rejected before it reaches the matching stage in Section III-D, and the user is prompted to recapture the image rather than being shown a low-confidence result (Fig. 3). Second, when no candidate medicine record reaches the $S_match \geq 0.20$ acceptance threshold (Eq. 6), the system reports no match rather than returning the closest-scoring candidate; within this design, a missed match is treated

as a safer outcome than a wrong one. Third, when more than one candidate clears the threshold, for example, when two products share a brand or generic name at different strengths, the interface presents the ranked candidate list for the user to confirm, rather than auto-selecting the top-scoring entry; only a single surviving candidate is opened directly. The acceptance threshold in Eq. 6 and the sub-score weights in Eq. 5 and Section III-D were set to the values reported here based on development-time testing; neither set of values has yet been validated through a dedicated sensitivity analysis that sweeps the threshold and rule weights independently and reports the resulting precision-recall trade-off. Section IV-F identifies this analysis as a direct next step.

This section reports results from four testing approaches: controlled OCR-condition testing, search-latency benchmarking, notification-reliability testing, and structured end-to-end workflow testing. These follow the four-tier strategy: unit, integration, performance, and manual end-to-end testing, used during development.

A. OCR Recognition Performance Across Image Conditions

Recognition performance was tested across six image conditions, representing the range a user is likely to meet during unsupervised scanning.

For each of the six image conditions, more than 20 medicine images were used, and the evaluation was repeated 20 times. The reported values represent the average result across the repetitions. The test images were obtained from medicines available at the nearby pharmacy, and all physical OCR scans performed using the OPPO A78(CPH2565) smartphone. Table I reports the confidence score and success rate for each condition. The six conditions included clear images under good lighting, slightly blurred image, poor lighting, upside-down orientation, handwritten text, and severely blurred images.

TABLE I. OCR RECOGNITION PERFORMANCE BY IMAGE CONDITION

Image Condition	Confidence Score	Success Rate	Notes
Clear image, good lighting	95–100%	98%	Ideal conditions
Slightly blurred	70–85%	75%	Some character errors
Poor lighting	60–75%	65%	Shadow / underexposure
Upside down	30–50%	40%	Orientation not corrected
Handwritten text	20–40%	25%	Outside intended use case
Severely blurred	10–30%	15%	Largely unreadable

Recognition success is treated here as the operational accuracy of the full capture-to-match pipeline, not a component-level OCR benchmark. Success reaches 98% under favourable conditions and degrades in a graded, predictable way as conditions worsen; this figure should not be interpreted as standalone OCR precision or medicine-matching accuracy. This pattern is broadly consistent with results reported for comparable OCR-based recognition systems tested under

varied capture conditions [15], [16]. The sample size and independent annotation procedure for the manual confidence-score review were not formally recorded. This categorisation step is what most directly protects users from acting on a poorly scanned label. This differs from a formal precision–recall–F1 evaluation against an independently annotated ground-truth corpus. For a binary correct/incorrect task, these metrics are defined as

$$\text{Precision} = TP / (TP + FP), \text{Recall} = TP / (TP + FN) \quad (7)$$

$$F1 = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}) \quad (8)$$

Computing these metrics requires confusion-matrix data, a labelled corpus large enough to estimate false-positive and false-negative rates for each condition. Building this dataset was outside the scope of the current evaluation, which recorded aggregate success/failure counts rather than field-level ground truth. This operational figure also does not separate matching accuracy from recognition accuracy: a case counted as successful requires both that the OCR engine produced a usable text string and that the six-rule matching engine in Section III-D then resolved it to the correct directory entry. The two error sources, and metrics specific to the matching stage; precision, recall, F1-score, false-match rate, and top-k accuracy are reported in Section IV-F as a direct next step rather than in this section.

B. Search Latency and Caching Performance

Table II compares medicine-search latency before and after the local caching architecture from Section III-E was introduced. Latency was measured from OCR completion to result display, across devices, using the full 11,825-record directory.

TABLE II. SEARCH PERFORMANCE BEFORE AND AFTER LOCAL CACHING

Operation	Without Cache (Firestore)	With Local Cache	Improvement
Initial database load	10–30 s	0.5–1.0 s	~20–30×
Subsequent searches (same session)	10–30 s	<50 ms	>400×
Candidate pre-filtering (inverted index)	all records scored	10–200 candidates	up to 1000× fewer
Full search pipeline	10–30 s	20–95 ms	~100–300×

This improvement matters beyond raw responsiveness. A medicine scan is a single, time-sensitive action, and a multi-second delay between capture and result display is enough to disrupt the workflow this system is designed to streamline, especially for the elderly users this framework targets. Reducing the search step to tens of milliseconds removes directory lookup as a source of perceived delay. This leaves on-device OCR inference (1–3 seconds, measured separately) as the main, and largely fixed, cost in the pipeline.

C. Reminder and Notification Reliability

The physical notification and alarm firing test was conducted on an OPPO A78 smartphone (model CPH2565), the same device used for the OCR scanning evaluation. The native AlarmManager-based reminder subsystem was tested across 200 scheduled alarms. Testing spanned four Android releases (Android 12 through 15) and covered foreground, background, Doze-mode, and post-reboot delivery scenarios.

All 200 alarms fired successfully. During the physical test, notifications were observed to trigger within approximately ± 30 seconds of the configured reminder time under the tested conditions and within sixty seconds after device wake from Doze mode. Alarm restoration after a device restart, handled by the boot-time receiver from Section III-F, was verified across all tested restarts. This result matches platform guidance, which identifies exact-alarm scheduling, not default timer-based approaches, as the mechanism needed to guarantee delivery under Android's background restrictions [22]. Text-to-speech playback for each notification was verified to render correctly for English-language medicine names and dosage instructions, across all tested devices.

D. Dose Tracking and End-to-End Workflow

Structured manual end-to-end testing covered fourteen representative user-journey test cases. These spanned registration, clear- and blurred-label scanning, exact and OCR-error medicine lookup, reminder creation, notification delivery under closed-app, post-reboot, and Doze-mode conditions, dose logging, and cache warm-up/hit behaviour. Dose-adherence percentage is computed as

$$A = (D_{\text{taken}} / D_{\text{scheduled}}) \times 100 \quad (9)$$

This value was verified to update immediately on the dashboard after each dose action, whether triggered from a notification or the dashboard itself. Colour-coded indicators (green / yellow / red) reflect the resulting adherence band. During end-to-end testing, the dose-tracking workflow recorded both taken and missed actions. A missed action initiated the configured retry sequence, while a subsequent taken terminated the retry sequence.

In addition to physical-device testing, Android emulators with different screen and device sizes were used to evaluate the user interface and functional behaviour across the application's available features. Physical-device testing was used specifically for OCR scanning and notification/alarm firing, while emulator-based testing was used to examine UI presentation and interaction across different screen configurations.

E. Comparison with Existing Systems

Table III compares the proposed framework against a commercial adherence platform evaluated in a randomised trial [9], and against generic reminder applications more broadly. This comparison shows that the framework's main point of difference is not reminder delivery, which is already well served even by generic apps. Instead, it is the combination of camera-based label recognition, confidence gating, and offline-capable local matching feeding directly into that reminder pipeline. A separate comparison point is a recent mobile application for visually impaired users that combines cloud-based OCR with a large language model to read back medication information, evaluated on a small image set alongside a structured usability study [24]. That system reports high recognition precision but depends on network connectivity for both its OCR and language-interpretation steps, and its scope stops at medicine identification and information playback rather than extending to reminder scheduling or dose tracking. Table III's rightmost column adds this system for reference. The present framework targets the complementary case: bounded, on-device recognition and matching against a fixed directory, integrated with an offline-capable reminder and dose-logging workflow, for settings where network access cannot be assumed.

Table III . COMPARISON WITH EXISTING MEDICATION-MANAGEMENT APPROACHES

Feature	Proposed Framework	RCT-Evaluated Platform [9]	Generic Reminder Apps	VisionCare [24]
OCR-based label scanning	Yes (on device)	No	No	Yes (cloud)
Native exact-alarm scheduling	Yes	Yes	Varies	No
Offline-capable core workflow	Yes (OCR + cache)	Partial	Yes	No
Dose-history tracking	Yes	Yes	Rarely	No
Voice / TTS medicine alerts	Yes	No	No	Yes
Custom OCR confidence scoring	Yes	N/A	N/A	N/A (LLM-Based)
Low-connectivity local directory	Yes (11,825 records)	N/A	N/A	No

F. Discussion of Limitations

Several limitations qualify these results, most of which map directly onto unresolved concerns raised in the first review round rather than to claims this paper makes with confidence.

First, as noted in Section IV-A, OCR performance is reported as an operational success rate for the combined recognition-and-matching pipeline, not as a formal precision-recall-F1 breakdown against an independently annotated ground-truth corpus, and recognition errors are not currently separated from matching errors. Building the confusion-matrix data this requires; per-condition images with verified ground truth, plus a record of which stage failed, is identified as the most immediate next step.

Second, matching-specific safety metrics appropriate for a medicine-identification system; precision, recall, F1-score, false-match rate, top-k accuracy, and rejection accuracy, have not yet been computed. An overall success rate, as reported in Table I, is not a substitute for these metrics, since it does not distinguish a false match, a genuine safety concern, from a correctly rejected low-confidence scan.

Third, the acceptance threshold ($S_{\text{match}} \geq 0.20$) and the weights used in the composite OCR-confidence score (Eq. 5) and the six-rule matching engine (Section III-D) were set during development rather than validated through a dedicated sensitivity analysis, as discussed in Section III-H. Whether a higher threshold would materially reduce false matches, and at what cost to recall, remains an open question.

Fourth, no baseline or ablation comparison has been run. The framework has not been benchmarked against a standard OCR-plus-generic-fuzzy-matching pipeline, and the individual contribution of the confidence-gating mechanism, the six matching rules, and the local caching layer has not been isolated.

Fifth, manual end-to-end and notification-reliability testing was conducted with the modest number of physical devices and alarms rather than through a large-scale field deployment. The physical OCR and notification tests reports in Sections IV-A and IV-C were conducted on an OPPO A78 (CPH2565), while

additional UI and functional checks were performed using Android emulator with different screen sizes.

Sixth, the medicine directory was obtained from a Kaggle dataset, with a subset of 200 records verified with the assistance of a pharmacist. However, the duplicate-removal procedure, detailed verification workflow, licensing/provenance documentation, and update cadence of the full directory are not documented in sufficient detail for independent reproduction.

Seventh, no usability evaluation with real users has been conducted. A recent OCR-based medicine-identification app for visually impaired users paired a small technical evaluation with a structured usability study [24]; a comparable small-scale usability study, covering ease of use, OCR-correction frequency, and reminder effectiveness under realistic conditions, would substantially strengthen the practical claims made here and is identified as future work rather than completed evidence.

Eighth, the current implementation has no caregiver-facing access and no explicit confirmation step in which a user affirmatively verifies dosage information before it is accepted; both are listed as planned extensions in Section V rather than implemented safeguards.

Ninth, both the OCR pipeline and the text-to-speech subsystem currently support English only, which limits applicability in multilingual settings such as Nepal, where many elderly users are more comfortable in Nepali than in English.

Finally, and most importantly, this evaluation measures technical performance; recognition success, search latency, and alarm-delivery reliability, and does not measure medication adherence as a patient outcome. The claim that the framework improves medication adherence would require a user or clinical study beyond the technical evaluation reported here; Section V restates the paper's contribution accordingly

V. CONCLUSION

This paper presented a mobile health framework that unifies on-device OCR-based medicine-label recognition, a composite confidence-estimation layer, a six-rule weighted fuzzy-matching engine, a two-level local caching architecture, and a native-alarm-based voice reminder subsystem, implemented as the MediMate application. The framework runs on Flutter and Firebase, organised under Clean Architecture, and is designed so that recognition, matching, and reminder delivery all function without a constant network connection.

The evaluation in Section IV supports three specific, technical claims. Recognition success, the operational accuracy of the full capture-to-match pipeline, reached 98% under favourable image conditions and degraded predictably rather than sharply as lighting, focus, and orientation worsened (Section IV-A). Local caching reduced medicine-search latency by two to three orders of magnitude relative to direct Cloud Firestore lookups (Section IV-B). Reminder delivery was verified as reliable across 200 scheduled alarms spanning four Android releases, including Doze-mode and post-reboot scenarios (Section IV-C). These are measurements of technical performance, not of patient outcomes: this study did not evaluate medication adherence as a clinical or behavioural outcome, and the framework's effect on real-world medicine-taking is not claimed here.

Section IV-F lists this paper's limitations in full; the most significant are that matching accuracy has not yet been isolated from OCR accuracy using precision, recall, F1-score, false-

match rate, and top-k metrics; that the confidence and matching thresholds have not been validated through a formal sensitivity analysis; that no baseline or ablation comparison has been run against simpler OCR-plus-fuzzy-matching pipelines; and that the system has not yet been evaluated with real users. Addressing these is the immediate priority for future work, alongside larger-scale, longer-duration field testing of notification reliability across more Android OEM devices, and multilingual extension of the OCR and text-to-speech subsystems, starting with Nepali. Beyond these immediate items, planned extensions include a structured drug-interaction and duplicate-medication check, a caregiver-facing read-only monitoring mode, an explicit dosage-confirmation step before a match is accepted, and perceptual-hash-based duplicate-scan detection. Taken together, the contribution of this paper is a specific, reproducible combination of confidence-gated on-device recognition and rule-based medicine matching for low-connectivity settings; not yet a validated adherence intervention, with a clearly scoped path toward the additional evidence that claim would require.

REFERENCES

- [1] World Health Organization, *Adherence to Long-Term Therapies: Evidence for Action*. Geneva, Switzerland: WHO, 2003.
- [2] S. D. Pruitt, R. Khan, N. Chaiyakunapruk, et al., "The silent epidemic of non-adherence - insights from the 2024 a: care congress," *BMC Proc.*, vol. 19, suppl. 10, p. 13, 2025, doi: 10.1186/s12919-025-00326-4.
- [3] V. Lanke, K. Trimm, B. Habib, and R. Tamblyn, "Evaluating the effectiveness of mobile apps on medication adherence for chronic conditions: Systematic review and meta-analysis," *J. Med. Internet Res.*, vol. 27, e60822, 2025, doi: 10.2196/60822.
- [4] Y. Peng et al., "Effectiveness of mobile applications on medication adherence: A systematic review and meta-analysis," *J. Manag. Care Spec. Pharm.*, vol. 26, no. 4, pp. 550–560, 2020, doi: 10.18553/jmcp.2020.26.4.550.
- [5] K. Tabi, A. S. Randhawa, F. Choi, et al., "Medication management and mobile health applications: A systematic review," *JMIR mHealth uHealth*, vol. 7, no. 9, e13608, 2019, doi: 10.2196/13608.
- [6] A. M. Hakami, B. Almutairi, A. S. Alanazi, and M. A. Alzahrani, "Effect of mobile apps on medication adherence of type 2 diabetes mellitus: A systematic review of recent studies," *Cureus*, vol. 16, no. 1, e51791, 2024, doi: 10.7759/cureus.51791.
- [7] Z. Ni et al., "An mHealth intervention to improve medication adherence and health outcomes among patients with coronary heart disease: Randomized controlled trial," *J. Med. Internet Res.*, vol. 24, no. 3, e27202, 2022, doi: 10.2196/27202.
- [8] L. C. Armitage et al., "Do mobile device apps designed to support medication adherence demonstrate efficacy? A systematic review of randomized controlled trials," *BMJ Open*, vol. 10, no. 2, e032045, 2020, doi: 10.1136/bmjopen-2019-032045.
- [9] A. Li et al., "Effect of a smartphone application (Perx) on medication adherence and clinical outcomes: 12-month randomized controlled trial," *BMJ Open*, vol. 11, no. 6, e047041, 2021, doi: 10.1136/bmjopen-2020-047041.
- [10] Z. S. N. Reis, G. M. V. Pereira, C. S. Dias, E. M. Lage, I. J. R. de Oliveira, and A. S. Pagano, "Artificial intelligence-based tools for patient support to enhance medication adherence: A focused review," *Front. Digit. Health*, vol. 7, 1523070, 2025, doi: 10.3389/fdgh.2025.1523070.
- [11] M. Arshed, A. B. Mahmud, H. S. Minhat, L. P. Ying, and M. F. Umer, "Effectiveness of mHealth interventions in medication adherence among patients with cardiovascular diseases: A systematic review," *Diseases*, vol. 11, no. 1, p. 41, 2023, doi: 10.3390/diseases11010041.
- [12] B. Machado, "Assessment of medication adherence using mobile applications in COPD: A scoping review," *Int. J. Environ. Res. Public Health*, vol. 21, no. 10, p. 1265, 2024, doi: 10.3390/ijerph21101265.
- [13] V. LeBaron, A. Adhikari, R. Bennett, et al., "A survey of cancer care institutions in Nepal to inform design of a pain management mobile application," *BMC Palliat. Care*, vol. 20, p. 171, 2021, doi: 10.1186/s12904-021-00824-0.
- [14] E. Sezgin et al., "Voice assistants and health: Review of current evidence and design implications," *NPJ Digit. Med.*, vol. 3, p. 33, 2020, doi: 10.1038/s41746-020-00332-0.

- [15] [15] L. Tan et al., "Comparison of object detection algorithms for pill/packaging recognition: RetinaNet, SSD and YOLO," *BMC Med. Inform. Decis. Mak.*, vol. 21, pp. 1–15, 2021, doi: 10.1186/s12911-021-01691-8.
- [16] [16] A. C. Spina, C. D. Yang, A. Jain, C. Ha, L. E. Chen, P. Yee, and K. Y. Lin, "Deep learning-driven glaucoma medication bottle recognition: A multilingual clinical validation study in patients with impaired vision," *Ophthalmol. Sci.*, 2025, doi: 10.1016/j.xops.2025.100758.
- [17] [17] X.-R. Chen et al., "Developing a mobile application-based system for enhancing medical skill training," *J. Adv. Med. Educ. Prof.*, vol. 13, no. 1, pp. 25–35, 2025, doi: 10.30476/jamp.2024.103650.2019.
- [18] [18] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston, MA, USA: Prentice Hall, 2017.
- [19] [19] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.
- [20] [20] Google, "Text recognition v2 | ML Kit," Google for Developers. [Online]. Available: <https://developers.google.com/ml-kit/vision/text-recognition/v2>. [Accessed: August 20, 2025]
- [21] [21] Google, "Secure data in Cloud Firestore," *Firestore Documentation*. [Online]. Available: <https://firebase.google.com/docs/firestore/security/overview>. [Accessed: Aug 01, 2025]
- [22] [22] Android Developers, "Schedule alarms | Background work," *Android Developer Documentation*. [Online]. Available: <https://developer.android.com/develop/background-work/services/alarms>. [Accessed: October 01, 2025]
- [23] [23] Google, "Flutter-Build apps for any screen," *Flutter Documentation*. [Online]. Available: <https://flutter.dev/>. [Accessed: Aug 01, 2025]
- [24] [24] W. M. Riega Ocharan, B. M. Palomino Santa Cruz, and F. D. Saenz Egusquiza, "Mobile application for identification and information of medications for the visually impaired by OCR and generative AI," in *Int. Conf. Applied Technologies (ICAT 2025)*, *Communications in Computer and Information Science*, vol. 2949, Springer, Cham, 2026, pp. 181-195, doi: 10.1007/978-3-032-22641-9_12.