International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

MEDBRIEF

Sanjay Krishna 1, Logesh R 2, Nikhil Karthik S A 3, Nithiya Sri K 4, Oviya Rethanya S 5

1 Assistant Professor, Dept. of Computer Science and Engineering,
Sri Shakthi Institute of Engineering and Technology, India. Email: sanjaykrishnacse@siet.ac.in
2 Student, Department of Computer Science and Engineering,
Sri Shakthi Institute of Engineering and Technology, India. Email: logeshr25cs@srishakthi.ac.in
3 Student, Department of Computer Science and Engineering,
Sri Shakthi Institute of Engineering and Technology, India.Email: nikilkarthiksa25cs@srishakthi.ac.in
4 Student, Department of Computer Science and Engineering,
Sri Shakthi Institute of Engineering and Technology, India.Email: nithyasrik25cs@srishakthi.ac.in
5 Student, Department of Computer Science and Engineering,
Sri Shakthi Institute of Engineering and Technology, India.Email: oviyarethanyas25cs@srishakthi.ac.in

ABSTRACT

MEDBRIEF is an AI-assisted healthcare document management and clinical support system for outpatient pre-consultation. A kiosk interface registers or looks up patients, records consent, collects medical reports and supports English and Tamil interaction. A separate doctor portal provides a review queue, source-linked draft summaries, emergency red-flag alerts and physician attestation. Each extracted field shows its source, confidence and verification state; unreadable content is labelled and directed to the original document. AI output remains a draft and the physician retains final authority. React, Vite, TypeScript, Node.js, Prisma and PostgreSQL support the application, with OCR and local language-model adapters. An audit reports 301 passing tests in 25 files and successful builds for all three workspaces. The same audit finds a disconnected database, unpersisted upload binaries, an unregistered document route and template-based OCR. These results do not establish clinical accuracy or deployment readiness.

Keywords: healthcare document management, clinical decision support, optical character recognition, grounded summarization, emergency red flags, consent management, physician review, audit logging, kiosk accessibility.

I. INTRODUCTION

Outpatient departments depend on patient history, previous prescriptions and laboratory reports, yet this information often reaches the physician as paper files, photographs and verbal accounts. Registration is repetitive, documents are scattered, and a doctor with only a few minutes per patient must locate relevant facts inside lengthy reports. Important warnings can be missed, and the origin of a recorded value may be unclear. These problems require more than a form or a text extractor. Clinicians need to know what the patient provided, what the system extracted, how reliable that extraction was and which items remain uncertain before relying on a summary.

The objective of MEDBRIEF is to provide a traceable pre-consultation workflow that connects patient registration, consent, document collection, text extraction, draft summarization, emergency alerts and physician review. A patient or an assisted companion completes intake at a hospital kiosk, and the doctor reviews a structured record in a separate portal. The system is intended to support doctors and healthcare staff. It does not diagnose, prescribe or replace clinical judgment, and the treating physician reviews, edits and confirms the final record.

MEDBRIEF separates three ideas that are easily confused: a document that was uploaded, a document that was read, and a record that was reviewed. A successful upload does not prove that the original file was stored, and extracted text does not prove that handwriting or a decimal value was read correctly. A summary without a warning is not a guarantee that the patient is safe. The application exposes these distinctions through field-level source, confidence and verification labels instead of combining them into one confident-looking statement.

This distinction is particularly important for poorly legible documents. A handwritten prescription photographed at an angle can be uploaded successfully yet remain unreadable. In such a case the portal states that the document could not be reliably digitized, lists the missing fields, such as medication name, dosage and allergies, and asks the physician to inspect the original. The case cannot be treated as complete until the critical item has been resolved. This prevents unavailable information from being mistaken for a negative finding.

The central workflow is Registration, Consent, Document, Extraction, Draft Summary and Physician Confirmation. A patient identifies a new or returning record, chooses a language, gives consent and answers intake questions by touch or voice. Processing is blocked when required consent is absent. Extracted fields carry provenance, and the physician can confirm, edit or reject each draft. Emergency red-flag rules and a manual help button route urgent cases to a dedicated alert view with a recommended triage action.

Accessibility and language support are central design choices. The kiosk offers English and Tamil, voice instructions, accessibility modes, a guide and a persistent help button. Self intake and companion or staff-assisted intake are separate entry points, because many outpatients are elderly or unfamiliar with digital devices. The interface is intended to remain simple for patients while the doctor portal supports detailed clinical work.

Privacy and security are treated as operational requirements. The backend is responsible for authentication, role-based authorization, consent enforcement, audit logging and persistence, and the frontends communicate with it through API endpoints rather than holding separate data stores. Passwords for seeded development accounts use bcrypt hashing, and authentication bypasses were reported as removed. These controls still require verification through security testing, and their existence does not imply that every deployment is protected.

The system is organized as a patient kiosk frontend on port 5175, a doctor portal on port 5174 and a backend API on port 5001. All three workspaces build successfully and an automated suite of 301 tests passes. An integration audit nevertheless found that the database was disconnected, the uploaded file binary was not written to durable storage, the document-image route was not registered and the OCR provider used template behaviour. These findings are reported in this paper rather than hidden, and they define the remaining development stages.

MEDBRIEF therefore integrates consent-aware intake with source-linked physician review. Missing, unreadable or mock-generated information cannot silently appear as verified clinical data. The following sections describe related approaches, methods, implementation and measured results.

II. EXISTING SYSTEM AND LITERATURE SURVEY

Existing approaches include paper registration, hospital information systems, document extraction services, clinical documentation assistants and manual physician review. Paper and manual entry allow contextual interpretation but are slow, repetitive and difficult to audit. Extraction services identify text and layout, summarization tools generate readable overviews, and electronic record systems store structured data. Each capability is useful, but none should be confused with proof that a patient's documents were fully read, stored and reviewed.

Microsoft Document Intelligence documents extraction of layout information such as text, tables and structural elements [5]. Tesseract provides an open-source OCR engine for converting image text into machine-readable content [4]. These examples show that general extraction already exists. MEDBRIEF does not claim to improve them or to outperform commercial products. Its contribution lies in the handling of extraction uncertainty inside a consent-aware, physician-facing workflow.

Lewis et al. describe retrieval-augmented generation as a way to ground language generation in retrieved information for knowledge-intensive tasks [1]. This motivates MEDBRIEF 's requirement that a summary remain tied to the source document. However, supplying source text does not independently validate every generated statement. Ji et al. survey hallucination in natural language generation and show that fluent output can contain unsupported content [2]. In a clinical setting an invented medication or allergy may be more harmful than an omission, so MEDBRIEF applies grounding validation and labels uncertain content.

Optical character recognition is valuable for scanned reports, but image quality, handwriting, abbreviations, decimal points and layout affect extraction accuracy [4]. A prescription may be partly recognized and still be unsafe to summarize. MEDBRIEF therefore reports failed and low-confidence extraction, retains the original document for comparison and offers manual review rather than generating a confident summary from unreliable text. At present the standard OCR provider uses mock or template behaviour, so this handling has been exercised structurally rather than on arbitrary real documents.

Prescriptions and laboratory reports present an additional difficulty because meaning depends on units and reference ranges. A haemoglobin value is interpretable only with its unit and the range printed on the report. MEDBRIEF therefore treats a value, its unit and its range as related fields and

avoids presenting a number without source context. Where a range is absent from the report, the summary says so and does not insert a typical range from memory.

Clinical alerting systems highlight potentially urgent information, but a rule that fires is not a diagnosis, and silence is not proof of safety. MEDBRIEF defines explicit red-flag rules with tiers, shows the evidence that triggered an alert and recommends a triage action for qualified staff. A manual help request from a kiosk is treated as its own alert type. The rules require clinical review before any real deployment.

Authentication practice also matters for medical data. Provos and Mazieres describe the adaptive bcrypt password scheme [3], which MEDBRIEF uses to hash seeded development credentials. Hashing is only one control. Role-based authorization, server-side route protection, consent enforcement and audit logging are needed in addition, and none of them is established merely by the choice of a hashing algorithm.

Database and file persistence form a separate class of risk. Prisma provides schema management and database access, and PostgreSQL is the intended relational store [6], [7]. Metadata in a table does not contain the original PDF, and an in-memory fallback can make a demonstration work while losing data on restart. MEDBRIEF treats storage references, file integrity and durable writes as separate requirements. A database record should not claim that a document is stored until the file write and the record write have both succeeded.

Many kiosk systems prioritise speed and ignore accessibility. Government hospital outpatient departments serve multilingual and often elderly patients, for whom touch-only forms in a single language are a barrier. MEDBRIEF therefore supports English and Tamil, voice instructions, companion-assisted entry and a visible help button. Language support improves access, but translated questions and spoken prompts must still be checked by clinical and linguistic reviewers before use with patients.

Workflow tools for multi-user review raise concurrency problems. If two physicians open the same case, one may overwrite the other's review state. Claim and release operations in the queue reduce this risk, but they do not replace tested concurrency behaviour. MEDBRIEF therefore lists a claimed item as assigned to another physician and relies on server-side state, not only on the interface.

Human review is also separate from machine output. A physician attestation records that the treating doctor reviewed the case details, verified the sources and resolved outstanding critical flags. AI summaries are labelled as drafts, and the physician can reject or regenerate a draft. These workflows improve accountability but do not replace clinical responsibility.

Regulatory and ethical expectations also apply. Medical data must be collected with patient consent, stored securely and accessed on a need-to-know basis. This paper does not assert compliance with any specific healthcare regulation. Compliance would require formal assessment of the hospital deployment, retention rules, key management and incident procedures.

Existing-system limitations are consequently framed as workflow risks: scattered documents hide relevant history, persuasive summaries conceal extraction uncertainty, and unstored files cannot be inspected later. MEDBRIEF addresses these risks by recording consent, source, confidence and review state rather than assuming that automated output is complete.

The evidence gap addressed by the project is therefore operational rather than a claim of a new language model. The proposed system connects kiosk intake, consent, document storage, extraction, grounded drafting, emergency alerts and physician sign-off in one workflow. Its distinction is the relationship between these records and the conditions required before information is presented as verified.

III. PROPOSED METHODOLOGY

MEDBRIEF processes each patient visit through identification, consent verification, document upload, extraction, draft summarization, alert evaluation and physician review. The workflow is designed so that every clinician-visible field can be related to its source document, confidence level and verification state. The validation path remains operational when a language model is unavailable, and a mock provider is distinguished from real AI output.

Patient identification: A new patient enters basic demographic information and receives a hospital registration number in the form MK-2026-XXXXXX. A returning patient enters the registration number or card identifier for a fast lookup. The system validates the submitted information, avoids creating unintended duplicate records and must retrieve actual backend records rather than display sample patients as real ones. The kiosk guides the patient through five steps: welcome, identity, language, consent and ready.

Consent management: Medical document processing begins only after consent has been recorded for the patient and workflow. Consent status is associated with the patient record and checked on the server, so that alternative API routes cannot bypass it. A test patient without consent appears in the queue with a waiting status, which exercises this block. Missing consent stops processing rather than

being assumed.

Upload validation: The upload API accepts PDF, PNG, JPEG and WEBP files with a maximum size of 15 MB. Type, size and request fields are validated before processing, and the interface reports accepted formats, progress and failures. A successful upload response is not treated as proof of permanent storage. The intended sequence is validation, consent check, durable file write, database record creation, extraction and review. A document should be marked stored only after the file write and the database operation both succeed.

Text extraction: An OCR pipeline converts scanned reports and photographed prescriptions into machine-readable text. The design preserves page and source information and reports low-confidence or unreadable regions. Failed extraction produces an explicit unreadable state, with the missing fields listed, instead of a fabricated summary. Manual review is offered when automated extraction is unreliable. The current standard OCR provider uses template behaviour, and integration of a real engine is a planned stage.

Field-level provenance: Each clinical field, such as chief complaint, current medication and known drug allergy, shows its source, a confidence score and a verification state. States visible in the portal include verified and unreadable. Deterministic field validation precedes AI summarization, and the physician retains authority to edit, confirm or reject every field. Extracted values are shown beside the original document so that misread handwriting, abbreviations, dates and laboratory values can be compared.

Grounded summarization: A summary service organizes extracted information into reason for consultation, relevant history, key findings, laboratory results, reported medications, warnings and missing information. A grounding validator checks that statements are supported by the source text. The summary must not treat absent information as negative; if a report does not mention a medication, the system does not state that the patient is not taking it. An OpenAI-compatible adapter connects a local language model [8], and a mock provider is used when the model is unavailable. The interface must identify which provider produced a draft.

Emergency red-flag detection: Clinically defined rules evaluate patient responses and extracted information. Alerts are grouped into immediate, urgent, staff-check and manual-help tiers, and each alert records its trigger, a pattern description, evidence references, a recommended triage action and a lifecycle timeline. Staff can acknowledge an alert or escalate it to the duty doctor. A kiosk help button raises a manual alert with the station location. Alerts show why they fired and are not diagnoses; the absence of an alert is not a guarantee of safety.

Queue priority: Queue priorities are based on application-defined rules and available information. Patients with red-flag alerts are surfaced for urgent attention, and the dashboard separates patients awaiting review, patients requiring urgent attention and confirmed records. Rules and weights must be documented and clinically reviewed, and the system should show why a patient has been prioritised.

Physician review: After secure login the doctor sees an intake queue with status labels such as ready, waiting and in consult, and may claim or release a review item. Opening a record shows patient information, documents, extracted fields, the draft summary, confidence and red-flag alerts. A critical unresolved item must be resolved before the physician attestation can be submitted. The attestation states that the physician reviewed the case details, verified sources, resolved outstanding critical flags and approved the record as accurate.

Failure handling: Failures in storage, OCR, language models or the database are intended to produce explicit states rather than silent success. The patient interface distinguishes registration completed, document uploaded, processing, submission received and processing failed, and must not report permanent storage or successful processing when an operation has failed. A degraded health state is visible to operators.

Security methodology: Doctor and staff accounts authenticate before clinical endpoints become accessible, and role-based permissions are enforced on the server rather than by hiding buttons. Passwords use bcrypt hashing. Uploaded files must not be exposed through public URLs, and retrieval requires authorization. Errors must not reveal credentials, database details or patient information. Important actions, including document access, review activity and sign-off, are recorded in a clinical audit log. These safeguards are verified through security tests and are not assumed from the existence of code.

Persistence and recovery: PostgreSQL 16 accessed through Prisma is the intended primary database. The backend exposes health and status endpoints that report service state and database connectivity. An in-memory fallback allows demonstration when the database is unreachable, but data held in memory can disappear on restart. An older SQLite file also exists in the API data directory and may contain records [13]. It must be inspected and backed up before configuration changes, and no file or table may be deleted or reset until existing data has been checked.

IV. SYSTEM IMPLEMENTATION

The application uses a three-component architecture. The patient frontend, doctor portal and backend API run as separate workspaces and communicate through API endpoints. The backend owns validation, consent enforcement, extraction, summarization, authorization and persistence, while the frontends present intake, review and alerts. Source files, database records and processing services are intended to remain within the configured hospital environment. The system has not been deployed publicly, and the screens shown in this paper use synthetic demonstration data.

Frontend stack: The patient kiosk and the doctor portal are built with React [9] and TypeScript [12] and served during development by Vite [10] on ports 5175 and 5174 respectively. Reusable controls provide loading, disabled, empty and error states. A demonstration banner states that the data is synthetic and links to an explanation of which parts are real and which are mocked. A planned final stage applies a shared cream, blue and red design system with consistent colours, typography, buttons, cards and status indicators across both applications.

Patient kiosk views: The kiosk starts with a welcome page offering self intake and companion or staff-assisted intake. A progress bar shows welcome, identity, language, consent and ready stages. New patients register, returning patients look up their registration number, and both continue to intake questions answered by voice or touch. The header provides a language switch between English and Tamil, accessibility modes, a guide and an I Need Help button. A status line reports whether the clinical backend API is connected. A mobile portal card appears on the page, but the audit reports that a standalone remote patient portal has not been implemented.

Doctor portal views: A left navigation panel provides the OPD Intake Queue, Summary Review, Red-Flag Alerts, Admin and Analytics, Question Bank and Templates, Multilingual and Governance, Operational Notifications and Clinical Audit Log. The intake dashboard presents summary cards for patients in queue, pending doctor review, red-flag alerts and confirmed records. A provenance panel explains that deterministic field validation precedes AI summarization and shows each field with its verification state. The header shows backend health, uptime and the signed-in physician.

Backend stack: The Node.js [11] backend listens on port 5001 and exposes authentication, patient, consent, document, review, alert, audit and status endpoints. Prisma [6] provides database access and schema migrations for PostgreSQL 16 [7], and bcrypt hashes passwords [3]. A health endpoint reports the service name, version, environment and database state, and returns a degraded status when the database is disconnected. A separate status endpoint reports the connection target and whether in-memory fallback mode is active.

Environment configuration: Backend configuration is read from environment variables, including the database connection string. Credentials must not be committed to source control or exposed in frontend code, and errors must not reveal passwords or connection details. The illustrative development connection uses a local PostgreSQL instance, but it has not been verified as the working configuration and must not replace the actual environment value without checking.

Document handling: Uploads are accepted as PDF, PNG, JPEG or WEBP files up to 15 MB. The document service creates a storage reference and a metadata record containing the document identifier, patient identifier, file name, content type and reference. In the audited version the service does not write the uploaded file buffer to disk or object storage, so the original binary can be lost after the request completes. The doctor document-image route is generated by the application but is not registered in the backend. Stages two and three of the development plan address durable storage and secure retrieval.

Extraction and summarization: The OCR pipeline includes failure and low-confidence handling and reports unreadable documents with the file name, the missing fields and an Inspect Source control. The standard clinical OCR provider currently returns mock or template output. The summarization service includes grounding validation and an adapter for an OpenAI-compatible local language model [8], with a mock provider when the model is unavailable. Real OCR and language-model integration, and clear labelling of mock output, remain required before clinical use.

Review queue and attestation: The queue lists patients with status labels and intake times, and supports claiming and releasing items. A patient assigned to another physician is marked so that two doctors do not unintentionally overwrite each other. The summary review view shows critical items, a link to resolve them, the attestation checkbox with a version identifier, and Reject Draft and Regenerate controls. The sign-off action remains unavailable while a critical item is unresolved.

Red-flag alert view: The alert view lists raised alerts with filters for all, immediate, manual, urgent and staff-check tiers. A selected alert shows its tier, trigger, station, time, recommended triage action, pattern description, evidence references and lifecycle timeline, with Acknowledge Alert and Escalate to Duty Doctor actions. Manual help requests from kiosk stations appear in the same queue.

Audit and governance: Important actions are written to a clinical audit log, and the portal also provides governance views for multilingual content, question banks and operational notifications. Audit integrity is reported as operational but must be tested, including whether document access and sign-off events are recorded reliably. Test patients, such as a patient without consent and a concurrent-test patient, exercise consent blocking and queue states. No real patient data is shown in this paper.

Quality tooling: The repository contains 25 automated test files with 301 passing tests, and production builds succeed for the patient frontend, doctor portal and backend. Git and GitHub provide version control, and Vercel is the deployment target mentioned in earlier work [14]. These results are regression evidence. They do not show that uploaded files survive a restart, that PostgreSQL is reachable or that real OCR reads medical reports.

Development plan: Work follows eight stages: database connectivity and durable persistence, permanent document storage, secure document retrieval and viewer, real OCR integration, grounded AI summaries, end-to-end patient-to-doctor integration, security and regression testing, and interface redesign with final deployment testing. The order is deliberate, because reliable storage must precede a polished interface.

Figures 1-4 summarize processing, data relationships, readability decisions and verification scope. The backend enforces consent and authorization; the frontends expose source, uncertainty and permitted actions.

V. ADVANTAGES

1. **Source Linked Review**

Traceable fields: Doctors can relate each extracted field to its source, confidence score and verification state rather than reading an unexplained paragraph. The original report remains the reference for comparison when handwriting, units or dates are uncertain.

Inspectable limitations: Unreadable fields, missing information and low-confidence extraction remain visible. A critical unresolved item blocks the attestation, so the physician must inspect the original before the record can be confirmed.

2. **Consent Aware Intake**

Consent before processing: Medical document processing is blocked when the required consent is absent, and consent is associated with the patient and workflow. Server-side checks reduce the chance that alternative routes bypass consent.

Reduced repetition: New and returning patients are identified through registration or lookup, which reduces repetitive manual entry and the risk of unintended duplicate records.

3. **Accessible Patient Experience**

Language and voice: English and Tamil interfaces, voice instructions, accessibility modes and a guide support patients with different literacy and technology experience. Touch and voice interaction avoid dependence on typing.

Assisted entry: Companion and staff-assisted intake allows family members or attendants to complete the process, while a persistent help button lets a patient request staff at any time.

4. **Safety Alerts and Physician Authority**

Explained alerts: Red-flag alerts display their tier, trigger, evidence and recommended triage action, which helps staff decide quickly why an alert was raised. Alerts are not presented as diagnoses.

Human final decision: AI summaries are draft aids, and the treating physician can edit, reject or regenerate them. The attestation records that the physician reviewed the case and resolved critical flags.

5. **Operational Transparency**

Visible system status: Both frontends display backend connectivity, and the health endpoint reports a degraded state when the database is disconnected. The audit therefore exposed storage limitations instead of hiding them behind a working demonstration.

Audit trail: Review activity, alert lifecycle events and sign-off are intended to be recorded in a clinical audit log, supporting accountability and later inspection.

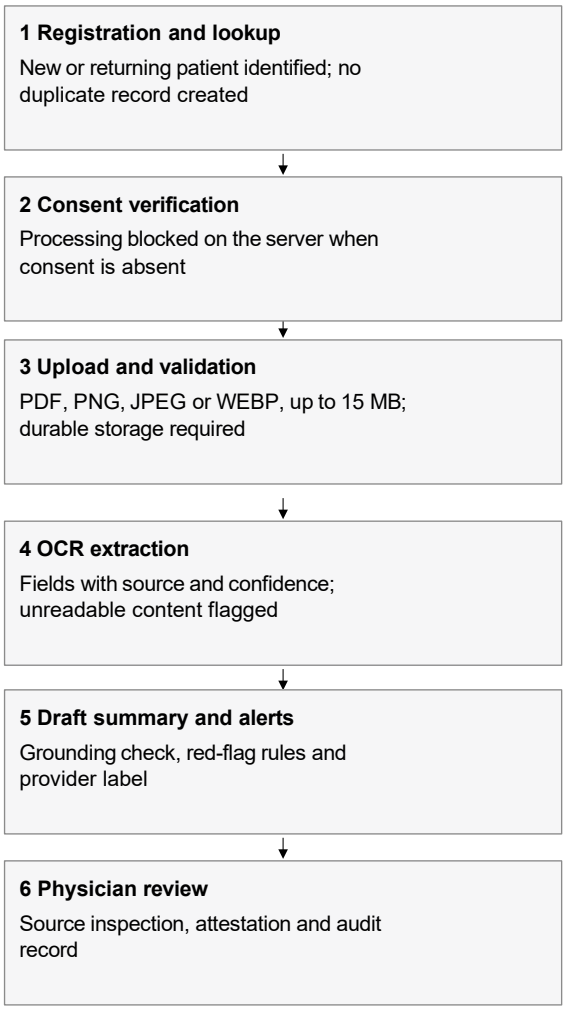
6. **Evaluation and Responsible Claims**

Measured verification: Automated tests, builds and an integration audit expose both passing results and open defects. Real OCR, durable storage and clinical evaluation remain pending.

Distinctive workflow: Consent, documents, extraction, summaries, alerts and physician decisions remain linked. Mock output, unreadable content and missing information cannot imply verified clinical data; clinical accuracy and universal usability are not established.

Figure 1 MEDBRIEF **processing**

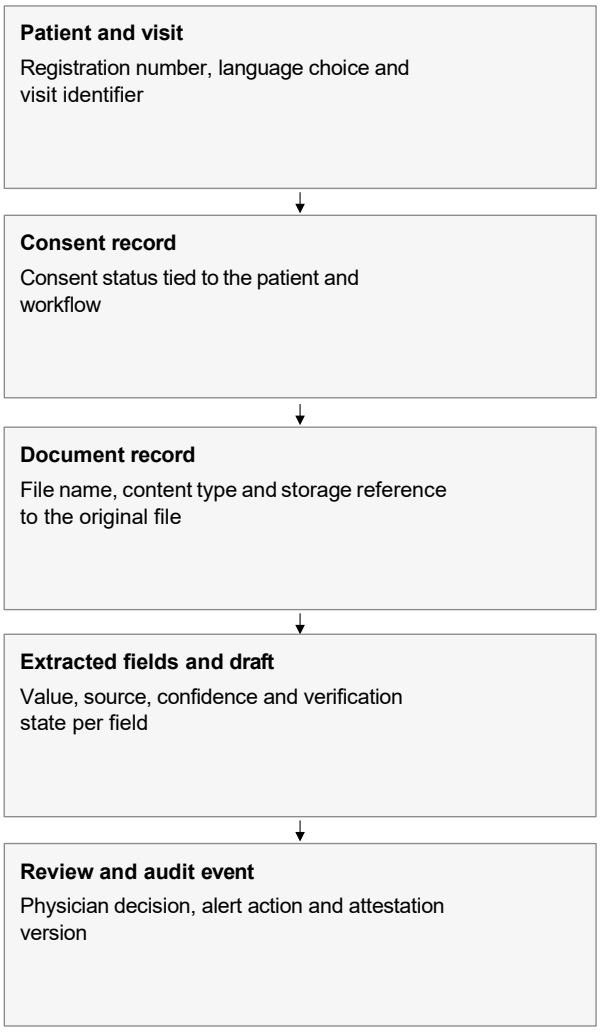
Consented intake to a reviewed record



Mock OCR and language-model providers are labelled as such; they do not replace the real services.

Figure 2 Review data relationships

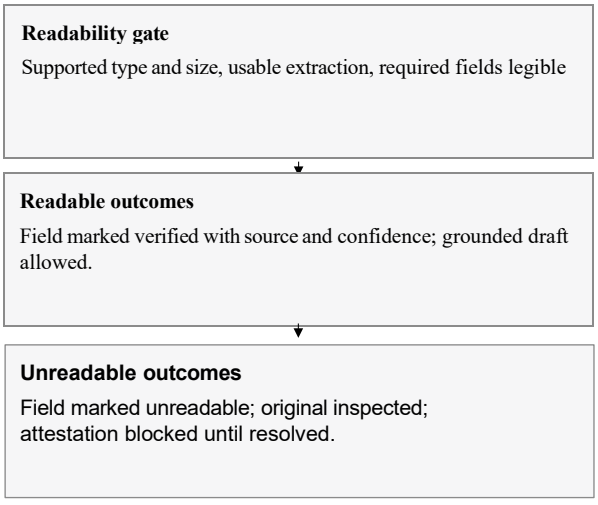
Durable records with explicit ownership



Physician decisions remain separate from AI drafts. A new draft does not inherit an earlier approval.

Figure 3 Readability decision

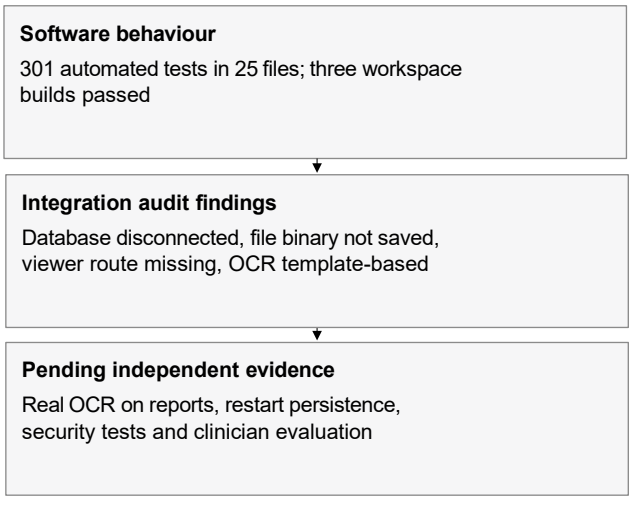
Unreadable content is never summarized silently



Resolution means the physician inspected the source, not that OCR output was correct.

Figure 4 Verification boundaries

Measured local results and their scope



Tests and audits do not establish clinical accuracy or deployment readiness.

VI. RESULTS AND ANALYSIS

The integration audit of the current repository reported 301 passing automated tests across 25 test files and successful production builds for all three workspaces: the patient frontend, the doctor portal and the backend API. The audit also reported that authentication bypasses were removed, that seeded development accounts use bcrypt hashing, and that registration, patient lookup, consent checks, the upload API, the review queue with claim and release operations, summary grounding, red-flag detection and audit logging are implemented. These tests establish specific software behaviours; they are not a penetration test, a clinical validation or evidence of real-world accuracy.

The audit found four defects that the unit tests did not reveal. First, the backend health endpoint returned a degraded status with the database disconnected, and the status endpoint reported that the database server was unreachable at localhost:5432 and that the application was running in in-memory fallback mode. Records held only in memory may disappear when the backend restarts. Second, an SQLite database file in the API data directory may contain existing records, so the authoritative database must be identified and backed up before any change. Third, the document service generates a storage reference but does not write the uploaded file buffer, so the original binary may be lost, and the unregistered document-image route prevents the portal from opening the original report. Fourth, the standard OCR provider uses mock or template behaviour, so extraction on arbitrary documents has not been demonstrated.

The summary review screen shows how the safeguards behave. For a photographed prescription named apollo_rx_2026.png, the portal reports that the document could not be reliably digitized, lists the missing fields as medication name, dosage and frequency, and allergies and reactions, shows the extracted value as handwritten and unreadable, and offers Inspect Source. One critical item remains unresolved, so the physician attestation stays unavailable. This shows that unavailable information is not turned into a normal summary. Because the OCR provider is template-based, the result demonstrates the workflow state rather than OCR accuracy.

Overall, the results support the structure of the workflow and the presence of consent, provenance and review controls. They do not yet establish persistence across restarts, real OCR accuracy, document retrieval for doctors, grounded summaries on real reports, security under attack or clinical safety. These items correspond to development stages one to seven and require integration tests with real files rather than additional unit tests alone.

VII. CONCLUSION

MEDBRIEF implements an AI-assisted healthcare document management and clinical support prototype that connects kiosk registration, consent, document upload, extraction, draft summarization, emergency alerts and physician review. Its principal objective is not simply to generate a summary, but to make the basis and limits of each clinical field inspectable. Readable fields display source, confidence and verification state, and unreadable inputs retain explicit uncertainty and direct the physician to the original document.

The reported tests and builds demonstrate concrete implementation behaviour, but the integration audit shows that passing tests do not prove reliable operation. The database was disconnected, uploaded files were not persisted, the document viewer route was incomplete and OCR remained template-based. MEDBRIEF therefore remains an assistive prototype with synthetic data, and clinical responsibility remains with the treating physician.

The integrated Registration to Consent to Document to Summary to Physician Review workflow is the project's strongest practical feature. Its completion depends on resolving persistence, storage, retrieval and OCR before further interface work. Further evaluation should extend this workflow to consented real documents and carefully measured hospital pilots.

VIII. FUTURE WORK

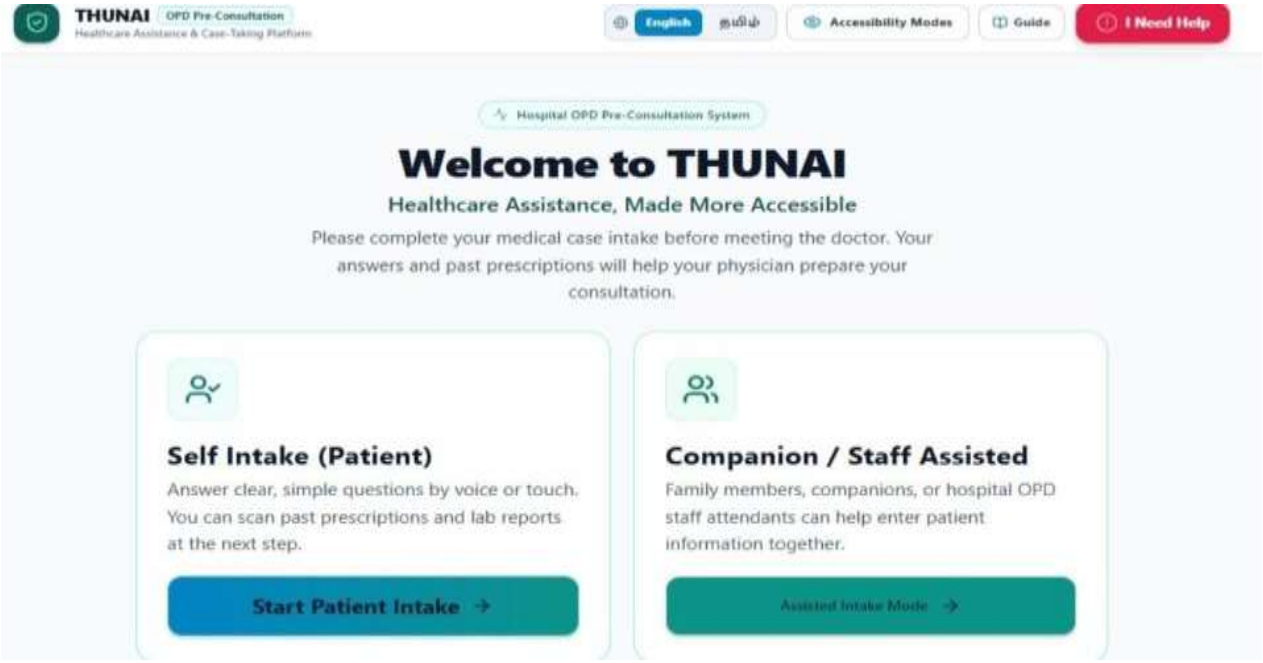
Restoring durable persistence is the highest-priority next activity. The team should identify the active database configuration, investigate the unavailable PostgreSQL connection, inspect and back up the SQLite file without deleting it, verify Prisma migrations and confirm that records survive a backend restart. The application should not fall back silently to in-memory storage for operations that require persistence.

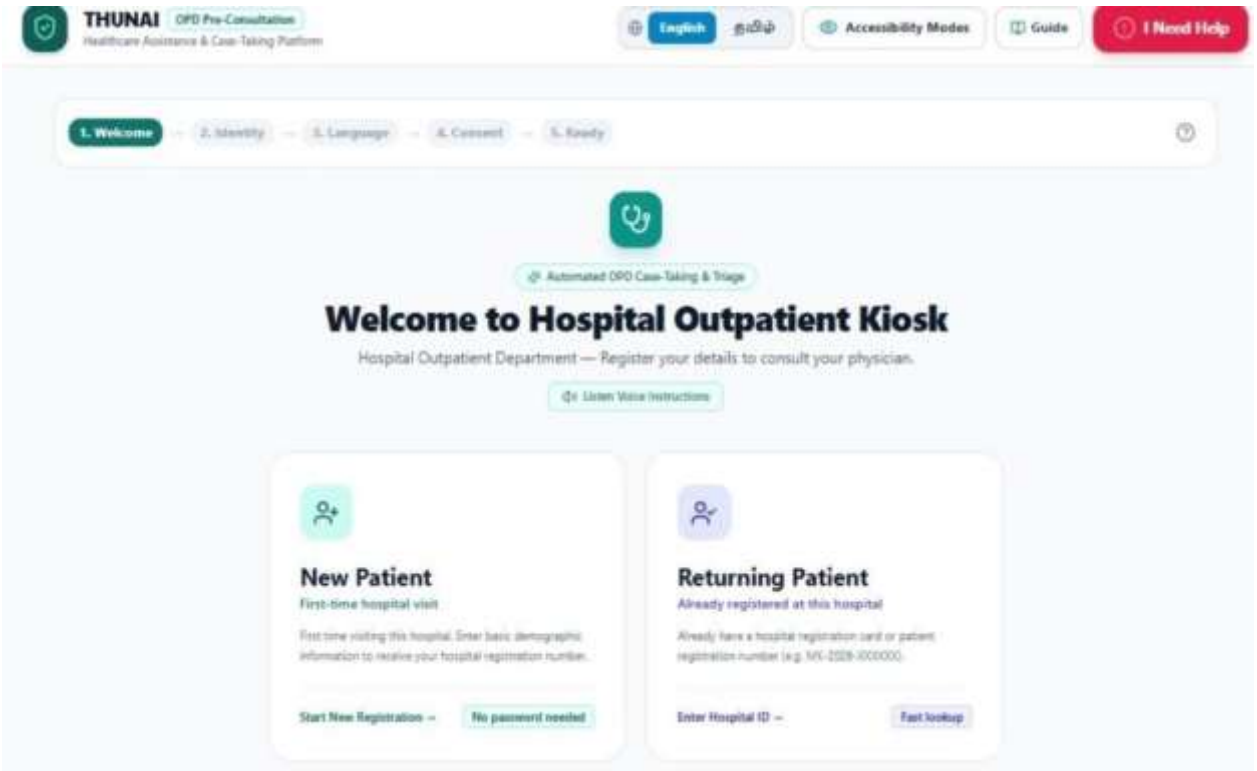
Permanent document storage and secure retrieval follow. Uploaded binaries should be written to durable storage, linked to database records, checked for integrity and retrieved through authorized endpoints rather than public URLs. The missing document-image route should be registered and connected to the doctor portal, with loading, error and unsupported-file states.

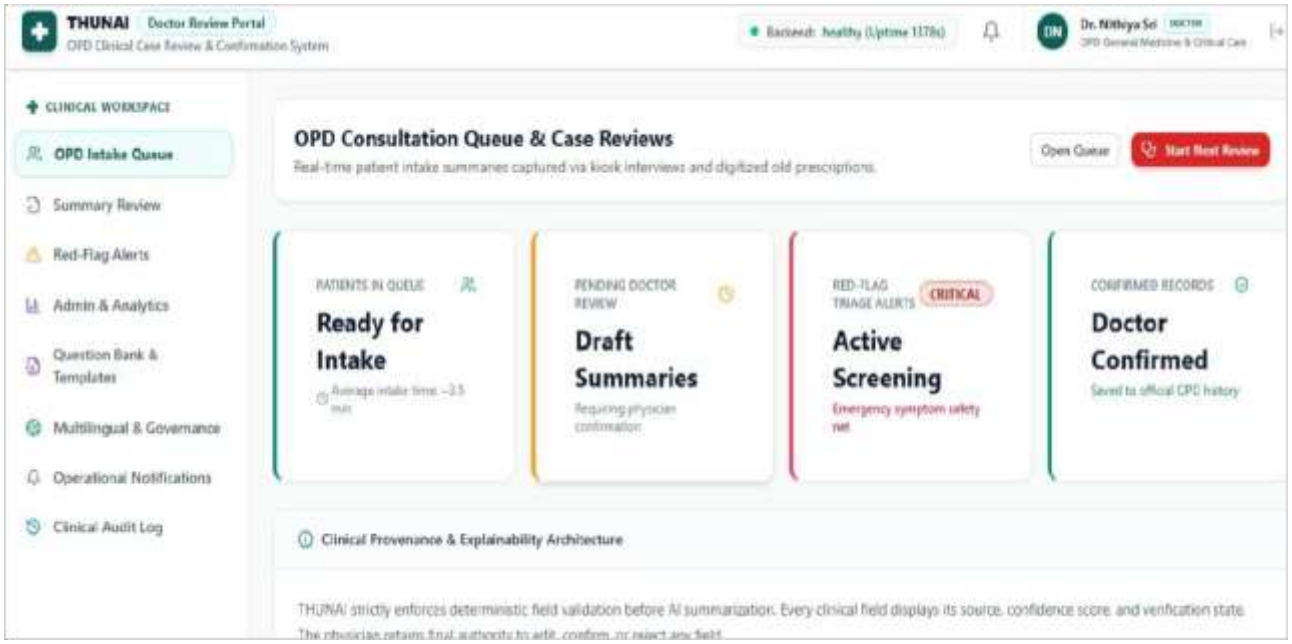
A real OCR engine should then process PDFs and images, preserve page information and report low-confidence content, with a manual-review fallback tested on varied medical reports. The summary pipeline should connect the intended language model, validate clinical statements against source text and distinguish mock output from real AI output. Red-flag rules require review by clinicians before

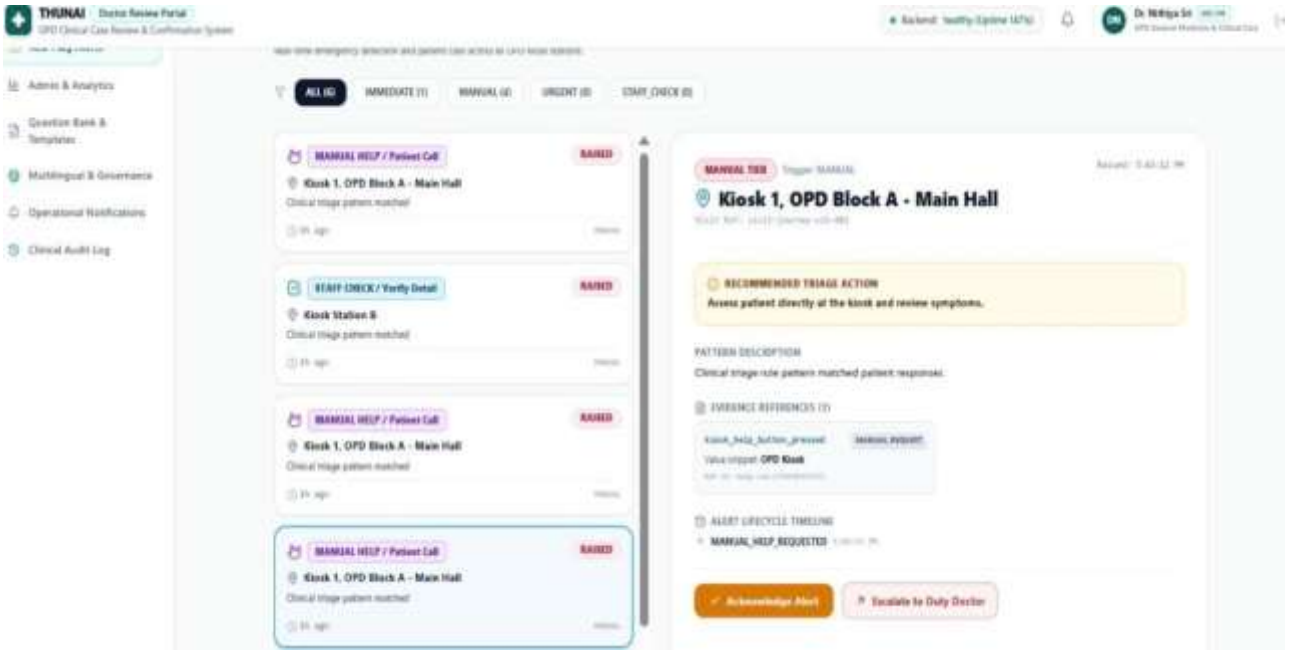
use, and emergency warnings must never imply that the absence of a warning means a patient is safe.

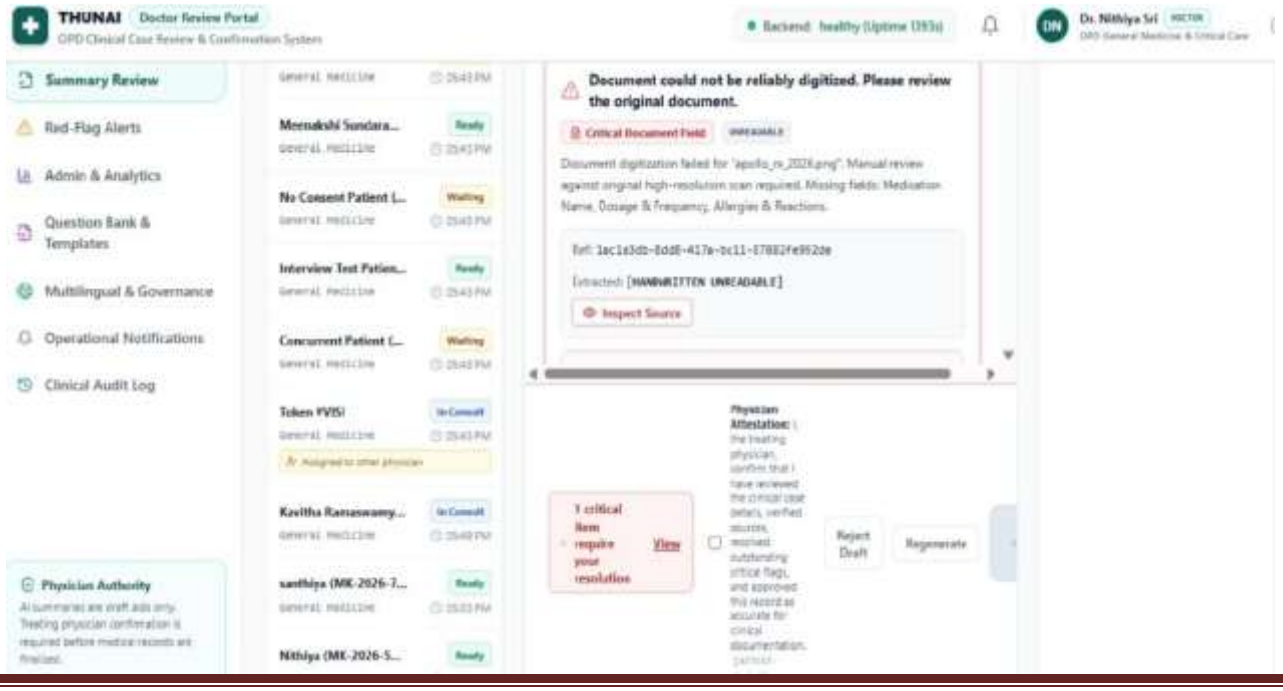
An end-to-end test with a real report and server-side security testing must precede deployment. Both frontends will then adopt a shared design system based on cream, blue and red tones, with accessible forms, clear status indicators and responsive layouts, followed by deployment checks of environment variables and database configuration. Any public service would require independently reviewed authorization, secure hosting, monitoring and incident procedures. Future improvements should preserve the conservative principle that unavailable evidence is not evidence of a verified record.






THUNAI
Doctor Review Portal
OPD Clinical Case Review & Confirmation System
CLINICAL WORKSPACE
OPD Intake Queue
Summary Review
Red-Flag Alerts
Admin & Analytics
Question Bank & Templates
Multilingual & Governance
Operational Notifications
Clinical Audit Log
OPD Consultation Queue & Case Reviews
Real-time patient intake summaries captured via kiosk interviews and digitized old prescriptions.
Open Queue
Start Next Review
PATIENTS IN QUEUE
Ready for Intake
PENDING DOCTOR REVIEW
Draft Summaries
Requiring physician confirmation
RED-FLAG TRIAGE ALERTS
CRITICAL
Active Screening
Emergency symptom safety net
CONFIRMED RECORDS
Doctor Confirmed
Saved to official OPD history
Clinical Provenance & Explainability Architecture
THUNAI strictly enforces deterministic field validation before AI summarization. Every clinical field displays its source, confidence score, and verification state.


Kiosk 1, OPD Block A - Main Hall
RECOMMENDED TRIAGE ACTION
Assess patient directly at the kiosk and review symptoms.


THUNAI
Doctor Review Portal
Summary Review
Red-Flag Alerts
Admin & Analytics
Question Bank & Templates
Multilingual & Governance
Operational Notifications
Clinical Audit Log
Document could not be reliably digitized. Please review the original document.
Inspect Source
Physician Attestation:
Reject Draft
Regenerate
Physician Authority

REFERENCES

[1] P. Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 2020, pp. 9459-9474. https://arxiv.org/abs/2005.11401

[2] Z. Ji et al. Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), 2023, Article 248. https://doi.org/10.1145/3571730

[3] N. Provos and D. Mazieres. A Future-Adaptable Password Scheme. USENIX Annual Technical Conference, 1999, pp. 81-91.

[4] Tesseract OCR contributors. Tesseract User Manual. Official documentation. https://tesseract-ocr.github.io/tessdoc/

[5] Microsoft. Document layout analysis with Document Intelligence. Official documentation. https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/prebuilt/layout

[6] Prisma. Prisma ORM Documentation. Official database toolkit documentation. https://www.prisma.io/docs

[7] PostgreSQL Global Development Group. PostgreSQL 16 Documentation. Official database documentation. https://www.postgresql.org/docs/16/

[8] OpenAI. API Reference. Official documentation for the chat completions interface used by OpenAI-compatible adapters. https://platform.openai.com/docs/api-reference

[9] Meta Open Source. React Documentation. Official web library documentation. https://react.dev

[10] Vite contributors. Vite Guide. Official build tool documentation. https://vite.dev/guide/

[11] OpenJS Foundation. Node.js Documentation. Official runtime documentation. https://nodejs.org/docs/latest/api/

[12] Microsoft. TypeScript Documentation. Official language documentation. https://www.typescriptlang.org/docs/

[13] SQLite contributors. SQLite Documentation. Official database documentation. https://www.sqlite.org/docs.html

[14] Vercel. Vercel Documentation. Official deployment platform documentation. https://vercel.com/docs

[15] MEDBRIEF project implementation records. Integration Audit, Health and Status Endpoint Output, Test and Build Results and Development Stage Plan. Local engineering artefacts, October 2026. Online references [4]-[14] accessed 27 October 2026.