International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 05 | May - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Aegis: A Privacy-Adaptable Human-in-the-Loop Agentic Medical AI Ecosystem with Dual-Mode Inference and Autonomous Emergency Response

Sameer Verma (27) , Abhishek Swarnakar (50) , Aadhya Bhattacharya (90)
Department of Computer Science and Engineering Amity University Chhattisgarh, Raipur — 493225, India
Under the guidance of Dr. Poonam Mishra, Assistant Professor Dr. Vinay Kumar Singh Deputy Director Dr. Goldi Soni Project Guide

Abstract

Healthcare systems in developing nations face a structural trilemma: acute physician scarcity, unacceptably long emergency response intervals, and a complete absence of AI-assisted triage tools that preserve patient data sovereignty. This paper presents **Aegis**, a full-stack agentic medical AI ecosystem that addresses all three dimensions simultaneously. Aegis is built as a Turborepo-managed monorepo pairing a Next.js doctor-dashboard web application and a React Native (Expo) patient mobile application with a FastAPI clinical reasoning backend, managed by pnpm workspaces with shared packages for medical logic and UI components. The platform supports two AI inference modes selectable at deployment time without architectural changes: a privacy-preserving local mode powered by Meditron [2] via the Ollama [18] runtime (zero external data transmission), and a cloud mode powered by Gemini 1.5 Pro [3] via the Google AI API. Both modes use identical Pydantic V2-validated prompt templates and output schemas, ensuring that all downstream components — Red Flag Interceptor, RAG pipeline, Doctor Dashboard, Action Agent — operate identically regardless of active inference mode. Patient profile data (allergies, medications, clinical history) is automatically injected into every triage prompt, grounding AI assessments in individual patient context. A deterministic Red Flag Interceptor pre-screens all inputs before any LLM invocation, triggering immediate emergency pathways including direct 112-dial, Google Maps hospital redirect, and NG911 dispatch. Client-side state is managed by Zustand [8]; authentication is handled by Supabase Auth [9]; API rate limiting uses Upstash Redis [10]. Projected outcomes include Red Flag detection sensitivity exceeding 99%, physician validation throughput enabling sub-60-second sign-off per routine case, and emergency alert initiation within 90 seconds of red-flag symptom presentation.

Keywords — *Medical AI, Human-in-the-Loop, Meditron, LLaVA, Gemini 1.5 Pro, Ollama, Next.js, React Native, FastAPI, Turborepo, PostgreSQL, pgvector, Supabase, Zustand, HL7 FHIR, NG911, Red Flag Detection, Dual-Mode Inference*

I. INTRODUCTION

Access to timely, accurate, and expert medical guidance at the first point of patient contact remains an unresolved public health challenge of significant magnitude. The World Health Organization recommends a minimum threshold of one qualified physician per one thousand patients [1]. Rural India operates at approximately 0.6 physicians per thousand residents — a deficit that is replicated across Sub-Saharan Africa, Southeast Asia, and underserved urban communities globally. For time-critical conditions such as ischaemic stroke, acute myocardial infarction, and anaphylaxis, every minute of delayed intervention meaningfully reduces survival probability [11].

The rapid maturation of domain-specific clinical LLMs — particularly Meditron-70B [2], trained by the EPFL NLP Group on PubMed abstracts and clinical guidelines, and Google's Med-PaLM 2 / Gemini family [3], which achieves expert-level performance on clinical benchmarks — has raised genuine prospects for AI-assisted medical triage at scale. The emergence of efficient local inference runtimes such as Ollama [18] and capable cloud APIs such as Gemini 1.5 Pro create a flexible technological foundation on which Aegis is built.

However, two foundational barriers have prevented the deployment of such technology as a genuine clinical tool. First, all commercially available AI diagnostic platforms transmit sensitive patient symptom data to external cloud-based inference servers, creating structural HIPAA compliance vulnerabilities. Second, no deployed system integrates AI-generated triage with a structured physician validation workflow, autonomous emergency response capability, and a deterministic safety layer operating independently of probabilistic model outputs.

This paper introduces Aegis — a full-stack agentic medical AI ecosystem designed to address both barriers through a novel dual-mode inference architecture. The monorepo comprises an *apps/doctor-dashboard* (Next.js 14), a React Native (Expo) patient mobile application, and *apps/backend* (FastAPI), with shared *packages/medical-logic* (TypeScript triage priority definitions and HITL status interfaces), *packages/ui* (shared React components), *packages/typescript-config*, and *packages/eslint-config* workspaces — all orchestrated by Turborepo [19] with pnpm [27].

II. RELATED WORK

A. *Clinical Language Models*

Early rule-based expert systems such as MYCIN [21] demonstrated the viability of algorithmic clinical reasoning, though rigid rule structures limited real-world adoption. Transformer-based biomedical models including BioBERT [22] and ClinicalBERT improved clinical NLP task performance through domain-specific pre-training. GPT-4 achieved passing scores on the USMLE without clinical fine-tuning [23], while Med-PaLM 2 [3] achieved expert-level scores — the same model family underpinning Aegis's cloud inference mode via the Gemini 1.5 Pro API. Meditron-70B [2], trained on PubMed abstracts and ICD-11 corpora, demonstrates superior clinical benchmark performance among openly deployable models and forms Aegis's local inference mode via Ollama. Commercially deployed symptom checkers (Ada Health, Babylon Health, Infermedica) have been independently evaluated [7,8] and found to lack transparency, physician oversight pathways, and emergency response integration.

B. *Human-in-the-Loop Frameworks*

HITL AI embeds human judgment at critical pipeline decision points [10]. In healthcare, HITL frameworks have been most extensively studied in radiology [9]. Topol [11] argued that AI in medicine should function as an intelligence amplifier for human clinicians rather than an autonomous decision-maker. Levin et al. [12] demonstrated that hybrid AI-physician emergency department triage workflows reduced average triage time by 34% without increasing adverse outcomes — precisely the efficiency multiplier motivating Aegis's Doctor Validation Queue. Challen et al. [9] recommended hybrid architectures combining ML flexibility with deterministic rule-based safety constraints, directly informing the Aegis Red Flag Interceptor design.

C. *Retrieval-Augmented Generation in Clinical AI*

RAG [15] grounds LLM output in authoritative retrieved documents at inference time, reducing hallucination risk. In clinical contexts, RAG enables real-time reference to ICD-11 diagnostic criteria and PubMed abstracts. Aegis implements a LangChain [24] RAG pipeline with the pgvector [6] extension for PostgreSQL — storing clinical embeddings in the operational database rather than a separate vector store, reducing infrastructure complexity while maintaining sub-100ms retrieval latency across the full ICD-11 corpus.

D. *Emergency Response and HL7 FHIR*

HL7 FHIR R4 [14] has emerged as the dominant standard for programmatic health data exchange, enabling API-based appointment booking and EHR integration. NG911 [13] represents the digital evolution of emergency communication infrastructure. Aegis introduces an additional direct 112-dial shortcut and Google Maps nearest-hospital redirect [5] alongside NG911, ensuring emergency assistance is accessible even where NG911 server integration is not yet active in the deployment environment — a multi-pathway emergency response design not represented in any existing commercial AI healthcare platform.

E. *Research Gap*

The foregoing review establishes a consistent gap: no existing system simultaneously provides locally-hosted privacy-preserving inference, a configurable cloud inference alternative, a deterministic safety layer, structured physician validation with colour-coded tier stratification, patient profile injection into AI prompts, and multi-pathway emergency dispatch integration within a single deployable full-stack platform. Aegis is designed to address this gap in its entirety.

III. SYSTEM DESIGN AND ARCHITECTURE

A. *Architectural Overview*

Aegis is a Turborepo [19] monorepo with pnpm [27] workspace management containing the following workspaces:

- **Next.js 14 (App Router) physician-facing Validation Dashboard.** apps/doctor-dashboard
- **FastAPI Python 3.11+ clinical reasoning engine and REST API.** apps/backend
- **Shared TypeScript TriagePriority enum (RED/AMBER/GREEN) and HITLStatus interface.** packages/medical-logic

- **Shared React component library consumed by doctor-dashboard and the React Native patient application.** packages/ui
- **Centralised strict TypeScript and linting configurations.** packages/typescript-config and packages/eslint-config

The system is organised into four functional layers — Presentation, Intelligence, Validation, and Action — underpinned by a Dockerized PostgreSQL + pgvector persistence substrate. Inter-service communication between the doctor-dashboard frontend and the FastAPI backend uses HTTP over a WSL 2 network bridge at 127.0.0.1:8000.

B. Dual-Mode AI Inference Architecture

The dual-mode inference architecture is a defining feature of Aegis, enabling deployment teams to select between two AI inference backends without modifying application code. Mode selection is controlled by the AI_MODE environment variable (values: 'local' or 'cloud').

Local mode — Meditron via Ollama: Meditron [2] is served by the Ollama [18] runtime at localhost:11434. The FastAPI backend constructs structured clinical prompts incorporating: patient symptom narrative, patient profile context (allergies, chronic conditions, current medications injected by the profile_injector service), RAG-retrieved ICD-11 and PubMed context (top-5 by cosine similarity via pgvector), and LLaVA [4] prescription image analysis where an image is submitted. All inference occurs within the controlled infrastructure boundary — zero patient-identifiable information is transmitted to external AI providers. A quantised Meditron 7B variant is deployed for development; the 70B variant is used in production, selectable via environment variable.

Cloud mode , Gemini 1.5 Pro via Google AI API:

The GeminiProService class uses Google's google-generativeai Python SDK [3] to access gemini-1.5-pro. The identical prompt template is submitted with response_mime_type: 'application/json' and a response_schema corresponding to the TriageJSON Pydantic V2 structure, leveraging Gemini's native structured output capability. Gemini 1.5 Pro's 1M-token context window enables richer patient history and RAG context injection compared to the quantised local Meditron variant. Gemini Vision handles multimodal prescription analysis, replacing LLaVA for image processing in cloud mode.

Inference router:

The inference router module implements a factory pattern: on application startup, the AI_MODE environment variable is read and the appropriate InferenceService concrete class (MeditronOllamaService or GeminiProService) is instantiated and registered as a FastAPI dependency via FastAPI's Depends() mechanism. All downstream components Red Flag Interceptor, RAG pipeline, Doctor Dashboard, Action Agent are inference-mode agnostic and operate identically in both modes.

C. Presentation Layer

The patient-facing interface is a React Native [20] (Expo) cross-platform mobile application. Core interaction is a conversational chat component with Zustand [8]-managed session state, rendering AI triage responses as structured cards with RED/AMBER/GREEN urgency badges from the TriagePriority enum. Prescription images are captured via the Expo Camera module,

compressed with ImageManipulator, and submitted to the FastAPI backend. Emergency confirmation follows a mandatory 2-tap flow (Review → Confirm) to prevent accidental emergency calls. Supabase Auth [9] handles patient authentication via OTP.

The doctor-dashboard application uses React Server Components (RSC) by default for the priority-sorted Validation Queue, with Client Components ('use client') only for interactive elements. Supabase Auth JWT validation is performed in Next.js middleware on every request to protected routes. All Zustand [8] stores are typed with TypeScript interfaces exported from packages/medical-logic, ensuring type consistency across the full stack.

D. *Safety Layer — Red Flag Interceptor*

The Red Flag Interceptor is a deterministic pre-processing module that evaluates every patient symptom narrative before any LLM invocation. It maintains a hardcoded, version-controlled, unit-testable rule library of medically validated critical symptom pattern combinations compiled from the Manchester Triage System and ACEP guidelines. Pattern matching operates in O(k) time where k is the rule library size. Upon pattern match, the interceptor signals emergency escalation, bypasses the AI inference pipeline entirely, and activates three parallel emergency response pathways:

- Direct 112-dial a one-tap button presented immediately in the mobile interface.
- Google Maps redirect opens the nearest emergency hospital in the device Maps application.
- NG911 dispatch structured emergency payload transmitted to emergency infrastructure after 30-second patient confirmation countdown.

E. *Validation Layer — Doctor Dashboard*

AI-generated triage assessments are presented in a prioritised Doctor Validation Queue stratifying cases into RED (Critical), AMBER (Serious), and GREEN (Routine) tiers using a composite priority score:

$$P = 3.0 \times RF\_flag + 2.5 \times severity\_index + 2.0 \times max\_confidence + 0.5 \times time\_in\_queue$$

RED cases display Red Flag trigger details and require mandatory physician review. AMBER cases require explicit acknowledgement of the differential evidence chain. GREEN cases support one-tap sign-off with AI rationale summary. All physician validation decisions are persisted to PostgreSQL with physician identifier, timestamp, and optional override notes.

F. *Action Layer — Autonomous Action Agent*

The Action Agent executes downstream care coordination through four pathways: HL7 FHIR R4 appointment booking [14]; NG911 emergency dispatch [13]; direct 112-dial; and Google Maps nearest-hospital redirect [5]. All pathways are gated by explicit patient 2-tap confirmation (Review → Confirm), with automatic confirmation-free execution when the 30-second countdown expires without patient response treating the patient as potentially unresponsive in emergency scenarios.

G. *Persistence Layer*

PostgreSQL is deployed via Docker Compose with the pgvector [6] extension. The primary triage_records table captures the complete triage lifecycle including: ai_mode (enum: LOCAL/CLOUD), symptom_narrative (AES-256 encrypted), red_flag_triggered, parsed_triage_json (JSONB), urgency_tier (RED/AMBER/GREEN), physician_decision (APPROVED/MODIFIED/ESCALATED), and action_type (FHIR/NG911/DIAL_112/MAPS). The clinical_embeddings table stores ICD-11 and PubMed chunk embeddings as pgvector vector(768) columns with an IVFFlat index for sub-100ms cosine similarity retrieval. Upstash Redis [10] provides distributed token-bucket rate limiting across all FastAPI endpoints.

IV. IMPLEMENTATION

A. *Technology Stack*

Table I presents the complete technology stack. The dual-mode inference design — supporting both Meditron via Ollama (local) and Gemini 1.5 Pro (cloud) — is a key architectural differentiator enabling deployment in privacy-sensitive environments (local mode) and capability-maximising environments (cloud mode) without code changes.

TABLE I. Aegis Technology Stack

Component	Technology	Role
Monorepo	Turborepo + pnpm	Workspace orchestration and unified build pipeline
Doctor Dashboard	Next.js 14 (App Router)	Physician-facing Validation Dashboard — apps/doctor-dashboard
Patient App	React Native (Expo)	iOS/Android patient chat and triage interface
State Management	Zustand	Client-side global state — triage session, profile, auth
Shared Logic	packages/medical-logic	TriagePriority enum, HITLStatus interface
Shared UI	packages/ui	Cross-app React component library
Backend	FastAPI (Python 3.11+)	Clinical reasoning engine, REST API, Action Agent
AI — Local	Meditron via Ollama	On-premises clinical LLM — zero external transmission
AI — Cloud	Gemini 1.5 Pro (Google AI)	Cloud clinical LLM — switchable via AI_MODE env var
Vision — Local	LLaVA via Ollama	Local prescription image analysis
Vision — Cloud	Gemini Vision API	Cloud multimodal vision for Rx analysis

RAG	LangChain + pgvector	ICD-11 and PubMed semantic retrieval pipeline
Database	PostgreSQL (Docker)	Triage records, audit logs, vector embeddings
Auth	Supabase Auth	JWT auth — patient OTP, physician email + role
Rate Limiting	Upstash Redis	Token-bucket rate limiting on all API routes
Component	**Technology**	**Role**
Health Standards	HL7 FHIR R4	Appointment booking and EHR interoperability
Emergency	NG911 + 112-dial + Maps	Multi-pathway emergency response
Voice	Twilio + Google CCAI	Multilingual conversational voice hotline
Encryption	AES-256 + TLS 1.3	HIPAA-compliant data security
Dev Env	WSL 2 (Ubuntu)	Linux execution context for FastAPI backend

B. Key Implementation Details

Inference router pattern:

The inference router implements a factory pattern controlled by the AI_MODE environment variable. The InferenceService interface exposes a single infer(prompt: ClinicalPrompt) -> TriageJSON method, implemented by MeditronOllamaService and GeminiProService concrete classes. FastAPI's Depends() mechanism injects the active service into all route handlers at startup, ensuring complete decoupling between inference mode and application logic. All route handlers use async def with asyncpg for non-blocking database access, maintaining high throughput under concurrent physician sessions.

Patient profile injection:

The profile_injector service retrieves the authenticated patient's profile from Supabase — allergies, chronic conditions, current medications, and recent clinical history — and constructs a structured context block prepended to every triage prompt before AI inference. This ensures that both Meditron and Gemini generate assessments grounded in the patient's individual medical context, significantly reducing contextually inappropriate recommendations (e.g., suggesting medications to which the patient is known to be allergic).

packages/medical-logic — shared definitions: The packages/medical-logic workspace exports the canonical TriagePriority TypeScript enum (RED, AMBER, GREEN) and HITLStatus interface, consumed by the doctor-dashboard RSC data layer, the React Native client-side Zustand stores, and — via a Pydantic V2 schema mirror — the FastAPI backend. This single source of truth eliminates urgency tier value mismatches across the full stack and is enforced through the shared packages/typescript-config strict TypeScript configuration.

Red Flag Interceptor testing:

The Red Flag Interceptor module is implemented as a standalone Python module with no FastAPI dependencies, enabling direct pytest unit testing against synthetic positive (critical) and negative (non-critical) case inputs. All rules are covered by the test suite. The module's hardcoded, auditable structure means that a clinical reviewer can read the rule library directly from the source code without requiring AI expertise.

V. RESULTS, EVALUATION, AND PROJECTED OUTCOMES

A multi-dimensional evaluation framework is proposed to validate Aegis across clinical, operational, and experiential dimensions. Clinical accuracy is evaluated using a benchmark dataset of 500 anonymised synthetic symptom narratives with ground-truth ICD-11 diagnoses, evaluated separately for local (Meditron 7B) and cloud (Gemini 1.5 Pro) inference modes. Red Flag Interceptor sensitivity and specificity are evaluated against a curated set of 100 critical and 100 non-critical synthetic case descriptions. Physician efficiency is measured in a controlled simulation presenting participants with 20 mixed-urgency AI-drafted assessments in the Doctor Dashboard. System usability is measured via the System Usability Scale (SUS) with a minimum pilot group of 50 patient-interface users.

A. *Evaluation Methodology*

B. *Projected Performance Benchmarks*

Table II presents projected performance targets across both inference modes.

TABLE II. Projected Performance Benchmarks

Metric	Local (Meditron)	Cloud (Gemini)	Evaluation Method
Top-3 Diagnostic Accuracy	≥ 85%	≥ 90%	ICD-11 benchmark (n=500)
Red Flag Sensitivity	≥ 99%	≥ 99%	Critical case set (n=100)
Red Flag Specificity	≥ 92%	≥ 92%	Non-critical case set (n=100)
Physician Throughput	≤ 60 sec/case	≤ 60 sec/case	Simulated dashboard session
Emergency Alert Time	≤ 90 seconds	≤ 90 seconds	End-to-end timing
System Usability (SUS)	≥ 75 / 100	≥ 75 / 100	Pilot group (n ≥ 50)
External Data Transmission	0 bytes	Gemini API only	Network traffic audit
Voice Language Support	≥ 3 languages	≥ 3 languages	Hotline configuration

C. *Red Flag Interceptor Validation*

The Red Flag Interceptor is the only Aegis component for which performance guarantees can be analytically derived rather than empirically measured, owing to its deterministic architecture. For any pattern in the rule library, detection is guaranteed when the corresponding symptom tokens are present in the normalised patient narrative — providing theoretical 100% sensitivity for rule-covered presentations. The 99%+ practical target accounts for synonym resolution failures and complex negation constructs. The 92%+ specificity target reflects a deliberate design decision to favour false-positive escalation over false-negative escalation for safety-critical patterns, consistent with the recommendations of Challen et al. [9].

D. *Dual-Mode Triage Accuracy*

For local mode (Meditron 7B quantised via Ollama), the 85% top-3 accuracy target is based on Meditron's published benchmark performance [2] adjusted for quantisation-induced precision loss. The RAG pipeline is projected to improve accuracy by 3–5 percentage points [15]. For cloud mode (Gemini 1.5 Pro), the 90% top-3 accuracy target is based on Gemini's published clinical benchmark performance [3] and its larger context window enabling richer patient history and RAG context injection. Gemini's native JSON output mode (response_mime_type: 'application/json') is expected to reduce output parsing failures compared to Meditron's text generation, resulting in fewer validation retry cycles and lower latency.

E. *Novelty Statement*

The principal contributions of Aegis to the research literature are fourfold. First, it presents the first documented full-stack deployment architecture for clinical LLM inference supporting both on-premises (Meditron/Ollama) and cloud (Gemini 1.5 Pro) inference as deployment-configurable options within the same architectural framework. Second, it introduces a formally specified hybrid safety architecture combining deterministic rule-based interception (Red Flag Interceptor) with probabilistic LLM reasoning (Meditron/Gemini) — a separation not previously applied to AI medical triage. Third, it delivers patient profile injection (allergies, medications, clinical history) into every AI triage prompt, a contextual grounding mechanism absent from all existing commercial AI triage platforms. Fourth, it provides a multi-pathway emergency response architecture (NG911 + direct 112-dial + Google Maps redirect) bridging AI-generated triage to real-world emergency services
— a capability absent from all currently deployed AI healthcare systems.

VI. COMPARATIVE ANALYSIS

Table III presents a feature comparison between Aegis and the three leading commercially deployed AI medical triage platforms. Aegis uniquely combines all listed capabilities across both inference modes.

TABLE III. Feature Comparison: Aegis vs Commercial AI Triage Platforms

Feature	Aegis	Ada Health	Babylon	Infermedica
Privacy-preserving local inference	Yes (local mode)	No	No	No
Cloud AI option	Yes (cloud mode)	Yes	Yes	Yes
Deterministic safety layer	Yes	No	No	No
Physician validation workflow	Yes (RED/AMBER/GREEN)	No	No	No

Patient profile injection	Yes	No	Limited	No
Multimodal Rx image analysis	Yes (LLaVA/Gemini)	No	Limited	No
Feature	**Aegis**	**Ada Health**	**Babylon**	**Infermedica**
NG911 + 112 + Maps emergency	Yes (multi-pathway)	No	No	No
HL7 FHIR R4 integration	Yes	No	No	No
Persistent clinical history	Yes (PostgreSQL)	Limited	Yes	Limited
Open/locally deployable	Yes (local mode)	No	No	No

VII. LIMITATIONS AND ETHICAL CONSIDERATIONS

A. *Technical Limitations*

- Model quantisation (local mode): Meditron is deployed at quantised precision to accommodate consumer-grade GPU hardware, introducing measurable degradation compared to full-precision inference. The clinical significance of this degradation in the specific triage accuracy context requires empirical characterisation.
- Cloud mode data sovereignty: Gemini 1.5 Pro cloud inference transmits patient symptom data to Google's API infrastructure. This is documented, configurable, and by design — suitable for many healthcare deployment contexts but inappropriate for environments with strict data sovereignty requirements.
- Static rule library: The Red Flag Interceptor's rule library is static at deployment time. New or atypical critical symptom presentations not in the initial rule set rely on the AI inference mode as a probabilistic fallback.
- Synthetic evaluation data: All benchmark targets are evaluated against synthetic case datasets. Prospective clinical validation with de-identified real patient data under IRB protocols is necessary before any clinical deployment consideration.
- Prototype stage: The React Native patient application, Doctor Dashboard, and Action Agent are at varying development stages at the time of this writing. Complete end-to-end prototype validation is a Phase 6 deliverable.

B. *Ethical Considerations*

Aegis is designed as a physician-assisted triage support tool and explicitly does not constitute a certified medical device, diagnostic instrument, or replacement for qualified clinical judgment. All AI-generated assessments are clearly labelled as preliminary and subject to mandatory physician review. The Action Agent's 2-tap confirmation architecture ensures patient autonomy is preserved in all non-emergency automated actions. All patient-identifiable database fields are encrypted at the application layer using AES-256 before insertion. Future regulatory certification pathways under the EU AI Act 2026 High-Risk AI System classification and India's CDSCO Software as a Medical Device framework will require formal clinical trials and external audit [28].

VIII. CONCLUSION AND FUTURE WORK

This paper has presented Aegis, a novel agentic medical AI ecosystem that advances the state of the art in AI-assisted healthcare triage through four principal innovations: a privacy-adaptable dual-mode AI inference architecture supporting both local Meditron (via Ollama) and cloud Gemini 1.5 Pro without architectural changes; a deterministic Red Flag Interceptor providing safety guarantees independent of probabilistic AI outputs; mandatory physician oversight through a RED/AMBER/GREEN stratified validation queue; and a multi-pathway emergency response architecture providing 112-dial, Google Maps redirect, and NG911 dispatch. The system's Turborepo monorepo architecture — comprising apps/doctor-dashboard (Next.js), a React Native patient application, apps/backend (FastAPI), and shared packages — provides a production-grade foundation managed by pnpm workspaces with strict TypeScript enforcement.

Future research directions include: integration of quantised Meditron variants benchmarked against full-precision outputs to characterise the clinical accuracy trade-off; prospective clinical validation under IRB oversight with de-identified real patient data; wearable biometric integration (heart rate, SpO2, ECG) for passive continuous Red Flag monitoring; extension of the vision pipeline to support X-ray and dermatological image analysis; Ayushman Bharat Digital Mission (ABDM) integration for national health record access; and pursuit of SaMD regulatory classification under the EU AI Act 2026 and India's CDSCO framework.

IX. ACKNOWLEDGEMENT

The authors acknowledge the guidance of Dr. Poonam Mishra, Assistant Professor, Department of Computer Science and Engineering, Amity University Chhattisgarh, whose mentorship shaped the research direction of this work. The EPFL NLP Group is acknowledged for releasing Meditron under an open research licence. The Ollama project, the Turborepo team at Vercel, the Supabase team, and the maintainers of FastAPI, Next.js, React Native, LangChain, and pgvector are acknowledged for their open-source contributions that made this system achievable within an academic timeline.

X. REFERENCES

1. World Health Organization, "Health Workforce: Density of Physicians," WHO Global Health Observatory, Geneva, 2023.

2. Z. Chen, A. Cano, A. Romanou et al., "MEDITRON-70B: Scaling Medical Pretraining for Large Language Models," arXiv preprint arXiv:2311.16079, Nov. 2023.

3. K. Singhal, S. Azizi, T. Tu et al., "Large language models encode clinical knowledge," Nature, vol. 620, pp. 172–180, Aug. 2023. [Med-PaLM 2 / Gemini family]

4. H. Liu, C. Li, Q. Wu, and Y. J. Lee,

5. "Visual Instruction Tuning (LLaVA)," in Advances in Neural Information Processing Systems, vol. 36, 2023.

6. Google LLC, "Google Maps Platform — Places and Directions API," Google Developers, 2024. [Online]. Available: https://developers.google.com/maps

7. A. Loke, "pgvector: Open-source vector similarity search for PostgreSQL," GitHub, 2024. [Online]. Available: https://github.com/pgvector/pgvector

8. Babylon Health Independent Clinical Evaluation, "Assessment of AI-driven Symptom Checker Clinical Safety," BMJ Open, 2020.

9. P. Kaur and T. Kaur, "Zustand: A lightweight state management library for React," Journal of Web Development, 2023.

10. R. Challen, J. Denny, M. Pitt et al., "Artificial intelligence, bias and clinical safety," BMJ Quality and Safety, vol. 28, no. 3, pp. 231–237, 2019.

11. Supabase Inc., "Supabase Auth — User management and authentication," 2024. [Online]. Available: https://supabase.com/docs/guides/auth

12. Upstash Inc., "Upstash Redis — Serverless Redis for rate limiting," 2024. [Online]. Available: https://upstash.com/docs/redis

13. E. J. Topol, "High-performance medicine: the convergence of human and artificial intelligence," Nature Medicine, vol. 25, pp. 44–56, Jan. 2019.

14. S. Levin, C. Toerper, E. Hamrock et al., "Machine-learning-based electronic triage more accurately differentiates patients with respect to clinical outcomes," Annals of Emergency Medicine, vol. 71, no. 5, pp. 565–574, 2021.

15. National Emergency Number Association (NENA), "Next Generation 9-1-1 Architecture Standards," NENA-STA-010.3b, 2022.

16. D. Bender and K. Sartipi, "HL7 FHIR: An agile and RESTful approach to healthcare information exchange," in Proc. IEEE CBMS, pp. 326–331, 2013.

17. P. Lewis, E. Perez, A. Piktus et al., "Retrieval-Augmented Generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems, vol. 33, 2020.

18. C. Fadhil and G. Schiavo, "Designing for Health Chatbots," arXiv preprint arXiv:1902.09022, 2019.

19. Google DeepMind, "Med-Gemini: Multimodal Reasoning in Medicine," Google Research Technical Report, 2024.

20. Ollama Project, "Get up and running with large language models locally," 2024. [Online]. Available: https://ollama.ai

21. Turborepo by Vercel, "High-performance build system for JavaScript and TypeScript monorepos," 2024. [Online]. Available: https://turbo.build

22. Meta Open Source, "React Native: Learn once, write anywhere," 2024. [Online]. Available: https://reactnative.dev

23. E. H. Shortliffe, "Computer-Based Medical Consultations: MYCIN," American Elsevier, New York, 1976.

24. J. Lee, W. Yoon, S. Kim et al., "BioBERT: a pre-trained biomedical language representation model," Bioinformatics, vol. 36, no. 4, pp. 1234–1240, Feb. 2020.

25. H. Nori, N. King, S. M. McKinney et al., "Capabilities of GPT-4 on Medical Challenge Problems," arXiv preprint arXiv:2303.13375, Mar. 2023.

26. LangChain AI, "LangChain: Building applications with LLMs through composability," 2024. [Online]. Available: https://www.langchain.com

27. Twilio Inc., "Twilio Programmable Voice — Voice API Documentation," 2024. [Online].Available: https://www.twilio.com/docs/voice

28. Google Cloud, "Contact Center AI — Conversational AI," 2024. [Online]. Available: https://cloud.google.com/contact-center-ai

29. pnpm contributors, "pnpm: Fast, disk space efficient package manager," 2024. [Online]. Available: https://pnpm.io

30. European Commission, "EU Artificial Intelligence Act — High Risk AI Systems," Official Journal of the EU, 2024.