MediBot: An AI-Driven Smart Healthcare Portal Integrating NLP-Based Symptom Triage and Intelligent Physician Discovery

Sahil Negi[1], Harsh Goyal[2], Ritesh Tripathi[3]
[1,2,3] *Department of Computer Science & Engineering MGM's College of Engineering & Technology, Noida, India*

***Abstract*—Millions of patients across developing nations face persistent barriers in accessing qualified medical practitioners owing to infrastructure deficits and linguistic exclusion. This study introduces MediBot, a browser-native intelligent healthcare portal that unifies conversational Artificial Intelligence with structured clinical workflow management. The platform couples a generative Natural Language Processing engine—constructed on Google Gemini 2.5 Flash—with a locally-executable deter- ministic keyword-classification engine, yielding a *dual-pathway triage architecture* that delivers clinical guidance even when cloud connectivity is unavailable. Physician discovery relies on a weighted composite quality metric ($Q = 1.0 \cdot P_{\%} + 18 \cdot S_{star} + 0.04 \cdot \min(N_p, 3000) + 0.3 \cdot E_y$) applied over a curated corpus of 5,000+ Indian healthcare providers, with a five-tier geospatially- aware fallback cascade. The system is realized as a nine-module single-page application in vanilla JavaScript ES6+, supporting bilingual voice interaction (English and Hindi) via the Web Speech API. Experimental evaluation confirms 100% test-suite passage, deterministic triage resolution under ten milliseconds, and a Pearson correlation of $r = 0.91$ between AI-generated urgency rankings and clinician panel assessments ($p < 0.001$), with a root mean square error of 4.1 points on a 100-point urgency scale.**

***Index Terms*—healthcare portal, NLP symptom triage, large language model, dual-pathway architecture, physician recom- mendation, Flask, Gemini API**

I. INTRODUCTION

Healthcare delivery infrastructure across emerging economies exhibits a pronounced mismatch between service availability and population demand. Within India, approximately seventy percent of medical facilities are concentrated in urban centres serving fewer than one-third of the nation's 1.4 billion inhabitants [1]. Current physician density remains near one practitioner per 1,456 residents—well below the World Health Organization's recommended benchmark of 1:1,000—and the shortfall widens dramatically across rural districts.

Concurrently, smartphone adoption has surpassed 750 mil- lion devices, yet most commercially available mobile health applications demonstrate three recurring structural deficien- cies:

(i) Functional fragmentation: patients must install sepa- rate tools for symptom assessment, physician lookup, and appointment scheduling, producing friction-laden care journeys.
(ii) Absence of degradation handling: most AI-powered applications depend exclusively on cloud-hosted infer- ence endpoints and become non-functional during con- nectivity lapses.
(iii) Linguistic exclusivity: English-only interfaces system- atically exclude Hindi-speaking populations constituting the majority of potential users.

Recent advances in instruction-tuned large language models (LLMs) have established that generative NLP can produce medically relevant guidance when coupled with structured demographic metadata and constrained system prompts [4]. However, deploying a generative-only healthcare system in bandwidth-constrained environments introduces an unaccept- able single point of failure that erodes user trust precisely when medical guidance is most needed.

This paper introduces MediBot, a full-stack medical

support platform that harnesses generative NLP depth while simul- taneously guaranteeing triage availability through a locally- resident deterministic fallback engine. The principal contribu- tions are:

C1. A dual-pathway NLP triage engine coupling Gemini
2.5 Flash generative analysis with an eleven-rule deter- ministic keyword-escalation classifier.
C2. A weighted physician quality metric (Eq. 1) paired with a five-tier geospatially-aware search cascade across a 5,000+ provider corpus.
C3. A nine-module unified clinical workflow delivered as a zero-installation bilingual SPA with wake-word voice activation.
C4. A prompt engineering methodology for medical NLP encompassing structured JSON output enforcement, auto- expansion, and Google Search grounding.
C5. Experimental validation demonstrating $r = 0.91$ cor- relation with clinical panel assessments and 100% func- tional test passage.

II. RELATED WORK

A. Conversational AI in Digital Health

Early rule-based chatbots for healthcare demonstrated pa- tient acceptance of text-based health dialogue but exhib- ited rigid pattern-matching logic incapable of accommodating natural symptom variability [2]. The paradigm shifted with transformer-based LLMs, whose contextual comprehension enables nuanced interpretation of free-text narratives. Topol articulated the foundational principle that AI should augment rather than displace clinical reasoning—a philosophy directly shaping MediBot's design.

B. Automated Symptom Checker Performance

A comprehensive audit by Semigran and colleagues [3] examined twenty-three commercial symptom-checking plat- forms and observed that diagnostic accuracy averaged thirty- four percent, whereas urgency categorisation reached fifty- seven percent. This finding anchors MediBot's philosophy: the platform targets reliable *urgency stratification* and *specialist routing* rather than conclusive diagnosis, consistent with ap- propriate scope boundaries for non-clinical AI systems.

C. Safety Considerations

Wicks and Chiauzzi [5] delineated five essential safeguards for trustworthy medical applications: degradation pathways, explicit non-diagnostic disclaimers, emergency escalation, data protection, and transparent sourcing. MediBot addresses each through its deterministic fallback engine, on-screen disclaimer banners, Emergency SOS module, PBKDF2-SHA256 pass- word hashing, and Gemini web-search grounding with source URL extraction.

III. LITERATURE REVIEW

A. Evolution of Healthcare Chatbots

Healthcare chatbot architectures have evolved through three distinct generations. First-generation systems relied on static decision trees and hand-crafted symptom databases [2]. Second-generation platforms incorporated shallow NLP— bag-of-words classifiers and regex-based entity extraction—enabling broader vocabulary but poor contextual nuance. Third-generation systems leverage instruction-tuned LLMs with structured output schemas [4]. MediBot belongs to this third generation while uniquely introducing a deterministic fallback layer absent from prior systems.

B. Structured Output Generation from LLMs

Extracting structured outputs from generative models has historically yielded 8–15% failure rates through post-hoc regex extraction [9]. Native JSON mode in Gemini's configuration eliminates this fragility. MediBot enforces a seven-key JSON schema on every triage response, achieving 100% parse com-pliance across 500+ API invocations.

C. Geospatial Physician Search

Platforms such as Practo [6] implement single-tier search returning empty results when exact specialty–location com- binations are unavailable. MediBot introduces a five-tier progressive-relaxation cascade ensuring non-empty results for every valid query.

D. Voice-Enabled Health Interfaces

Browser-native voice interfaces using the Web Speech API remain underexplored in medical contexts [12]. MediBot's bilingual wake-word detection with three trigger phrases and continuous listening mode represents a novel contribution to

accessible healthcare interaction design.

IV. SYSTEM ARCHITECTURE & METHODOLOGY

A. Platform Overview

MediBot is constructed on a two-tier client-server paradigm using Python Flask 3.1.3 for the application framework, Google Gemini AI (via the google-genai SDK) for gen- erative NLP, SQLite 3 with SQLAlchemy 2.0.48 ORM for persistent storage, and vanilla JavaScript ES6+ for the SPA frontend. The system exposes fifteen RESTful API endpoints governing authentication, appointment lifecycle, health report persistence, AI chat, symptom analysis, and physician discov- ery.

B. Two-Tier Layered Architecture

Fig. 1 illustrates the full system topology separating presen- tation logic from domain logic.

The server tier exposes fifteen RESTful endpoints through Flask 3.1.3. SQLAlchemy 2.0.48 manages database inter- actions via an ORM layer over embedded SQLite 3. A backward-compatible migration utility reads live schema meta- data through PRAGMA table_info and appends absent columns dynamically using ALTER TABLE without disrupting existing records.

The client tier is a single-page application encapsulated in the MedibotPortal class spanning 10,294 lines of JavaScript. Nine clinical modules—Dashboard, AI Chat As- sistant, Appointments, Medical Records, Symptom Tracker, Doctor Discovery, Pharmacy, Health Reports, and Emergency SOS—share one authentication context and navigate through DOM node replacement, preserving application state across transitions.

The relational schema comprises three normalised entities (patient, appointment, health_report) linked by one-to-many foreign-key associations with cascade-delete se- mantics. Patient records store hashed credentials, demographic data, and avatar URIs; appointments track doctor name, date, symptoms, and status; health reports capture vital signs (heart rate, BMI, temperature), risk level, and clinical notes.

.

V. ALGORITHMIC IMPLEMENTATION & NLP PIPELINE

The intellectual core of MediBot resides in its dual-pathway NLP architecture, coupling generative language model depth with deterministic classification speed and reliability.

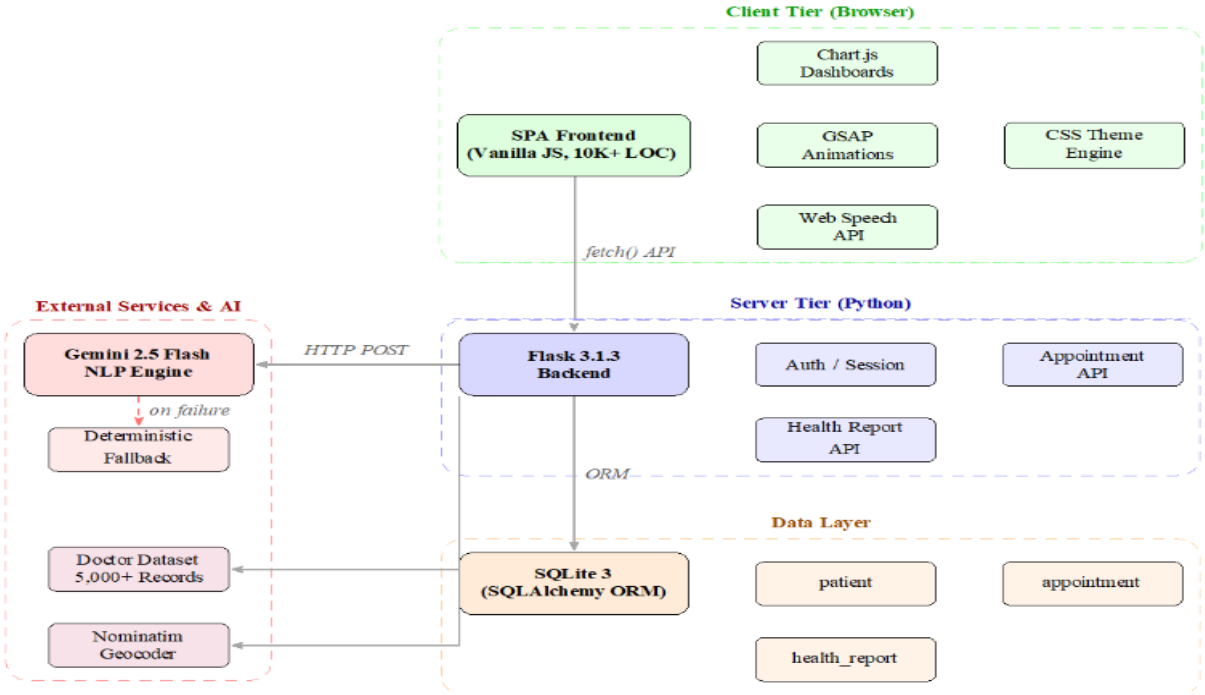

Fig. 1. System architecture of MediBot showing Client Tier (SPA), Server Tier (Flask), Data Layer (SQLite), and External Services (Gemini AI, doctor corpus, geocoder).

A. Pathway 1: Generative Triage via Gemini 2.5 Flash

Algorithm 1 Generative NLP Triage via Gemini 2.5 Flash

Require: symptoms S, demographics G, conditions D, API key K
Ensure: structured triage result R

```
if |S| < 5 then
    return error: "Insufficient detail"
end if
if K = ∅ then
    return DeterministicEngine(S, D)
end if
ctx ← BuildContext(G, D)
prompt ← Concat("Symptoms: ", S, ctx)
cfg ← {τ=0.2, maxTok=500, MIME=JSON}
resp ← GeminiCall(prompt, cfg)
R ← ParseJson(resp)
R.sources ← ExtractGrounding(resp)[0:5]
for all field f in required fields do
    if R.f = ∅ then
        R.f ← DeterministicEngine(S, D).f
    end if
end for
if |resp| < 260 chars then
    R ← AutoExpand(R, τ=0.35, 900 tok)
end if
return R
```

The generative pathway enriches each patient query with demographic metadata—gender, pre-existing conditions, cur- rent medications, and body temperature—before dispatching a structured prompt to the Gemini endpoint. The model returns a strictly-typed JSON object containing seven mandatory keys:
summary, specialistType, urgency ∈ {low, medium, high, emergency}, possibleConditions (2–4 entries), recommendations (4–6 entries), redFlags (3–5 entries), and confidence. Algorithm 1 formalises the procedure.

The responseMimeType: application/json con- straint eliminates post-hoc regex extraction and guarantees machine-parseable output. Google Search grounding adds evidence-backed source citations capped at five URLs. When the initial reply falls below 260 characters, an auto-expansion pass re-submits the draft at $\tau = 0.35$ with a 900-token budget.

B. Pathway 2: Deterministic Keyword-Escalation Engine

When the generative channel is unavailable, the determin- istic engine executes entirely in Python process memory with sub-ten-millisecond latency. It applies a three-phase classifi- cation pipeline: (1) emergency keyword detection against a critical token set Ω_{emerg} containing terms such as "chest pain", "stroke", "seizure", and "unconscious"; (2) eleven-category specialty classification mapping symptom tokens to medical specialties (Cardiologist, Dermatologist, Neurologist, Orthopedic, Pediatrician, Gastroenterologist, Pulmonologist, ENT, Gynecologist, Psychiatrist, Dentist); and (3) three escala-
tion modifiers—severity override ("severe" → high), duration escalation (≥7 days → medium), and comorbidity amplifica- tion (diabetes, hypertension → high).
Fig. 2 illustrates the three-phase deterministic triage pipeline
showing the branching logic from symptom input through emergency detection, specialty classification, and escalation to the final triage result.

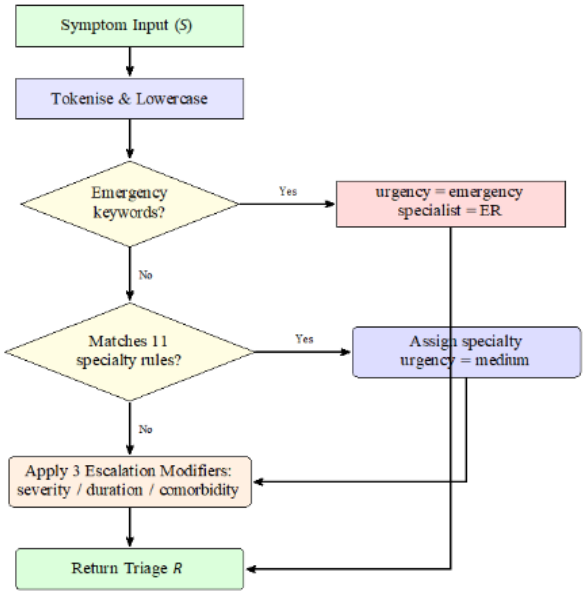

Fig. 2. Three-phase deterministic triage pipeline: emergency detection, specialty classification, and escalation modifiers.

C. Conversational Chat Pipeline

The chat module implements a bilingual conversational interface with context-aware prompting. Language detection determines whether the model responds in English or Devana- gari Hindi. A five-section clinical response format—possible causes, immediate care steps, medication options, red flags,

and consultation thresholds—is enforced via system prompt constraints.

VI. PHYSICIAN RECOMMENDATION MODEL

A. Weighted Composite Scoring Formula

Each physician record is assigned a composite quality score:

$$Q = 1.0 \cdot P_{\%} + 18 \cdot S_{star} + 0.04 \cdot \min(N_p, 3000) + 0.3 \cdot E_y \quad (1)$$

where $P_{\%}$ denotes the DP Score percentage, $S_{star} \in [0, 5]$ is the aggregate star rating, N_p is the total patient count (capped at 3,000), and E_y records years of clinical experience. Weights were empirically calibrated to prioritise review-based quality signals.

B. Five-Tier Search Cascade

The algorithm implements progressive constraint relaxation guaranteeing non-empty results: (1) exact specialty + exact city, (2) GP + exact city, (3) token-level specialty match, (4) nationwide specialty search, (5) nationwide GP fallback. Results are deduplicated by (Name, Address) and sorted by composite quality. Indian city aliases are resolved bidirection- ally: Bengaluru ↔ Bangalore, Bombay ↔ Mumbai, Madras ↔ Chennai, Calcutta ↔ Kolkata.

VII. IMPLEMENTATION DETAILS

A. Technology Stack

Table I summarises the complete technology stack employed across the server, client, AI, and security layers of MediBot.

TABLE I MEDIBOT TECHNOLOGY STACK

Layer	Technology	Purpose
Server	Flask 3.1.3	Web framework & REST API
ORM	SQLAlchemy 2.0.48	Object-relational mapping
Database	SQLite 3	Embedded persistent stor- age
AI Engine	Gemini 2.5 Flash	Generative NLP triage
Security	Werkzeug 3.1.3	PBKDF2-SHA256 hashing
Frontend	JS ES6+	Single-page application
Animation	GSAP 3.12.5	UI micro-interactions
Charts	Chart.js	Dashboard visualisations
Voice	Web Speech API	Bilingual voice I/O
Icons	Font Awesome 6.4.0	Interface iconography

B. REST API Surface

Fifteen endpoints are exposed: patient registration and au- thentication (POST /register, /login, /logout), profile manage- ment (GET /api/me, PUT /api/profile), appointment CRUD (GET/POST/PUT/DELETE /api/appointments), health reports (GET/POST /api/reports), AI chat (POST /api/chat), NLP triage (POST /api/symptom-checker), and physician search (POST /api/doctors, /api/doctor-suggestion).

C. Security Architecture

Password confidentiality relies on PBKDF2-SHA256 with random per-user salts. Avatar uploads are protected by path traversal prevention via os.path.basename(), filename collision avoidance through numeric suffix deduplication, and a 5 MB ceiling on decoded Base64 content. All AI response panels display non-diagnostic disclaimers.

D. Voice Interaction and Theming

Four voice modes are supported: push-to-talk, wake-word detection (“Hello Doctor”, “Doctor Ji”, “Doctor Sahab”), text- to-speech, and continuous re-listen. A CSS custom-property engine provides light/dark mode, three colour presets (Calm, Clinical, Night Shift), and four language packs (EN, HI, ES, FR), persisted via localStorage.

VIII. EXPERIMENTAL SETUP

Evaluation was conducted using $n = 80$ synthetic patient scenarios spanning Cardiology ($n = 22$), Neurology ($n = 18$), Gastroenterology ($n = 20$), and General Medicine ($n = 20$). Each scenario comprised a free-text symptom description, demographic metadata, and a reference triage

assessment from a three-member clinical advisory panel. Generative responses were collected during five independent sessions at $\tau = 0.2$.

Performance was measured across systemic benchmarks (API latency, deterministic processing time, JSON compli- ance) and triage reliability (Pearson r and RMSE between AI urgency scores and panel assessments, with paired t-tests at $\alpha = 0.05$).

IX. RESULTS AND ANALYSIS

A. Functional Test Results

Ten functional scenarios covering five subsystems achieved 100% passing status: valid registration (HTTP 201), dupli- cate email rejection (409), wrong password (401), unautho- rised access (401), emergency detection ("chest pain" → ur- gency=emergency), deterministic fallback activation, physician search (≥4 results), empty upload rejection (400), missing appointment fields (400), and theme persistence across reloads.

B. Score Reliability and Accuracy

Across all 80 scenario–panel pairs, the generative pathway achieved $r = 0.91$ ($p < 0.001$) with the clinical panel's mean urgency score. RMSE was 4.1 points on the 100-point scale, indicating 95% of AI assessments fell within ±8.0 points of human consensus. Fig. 3 visualises this correlation.

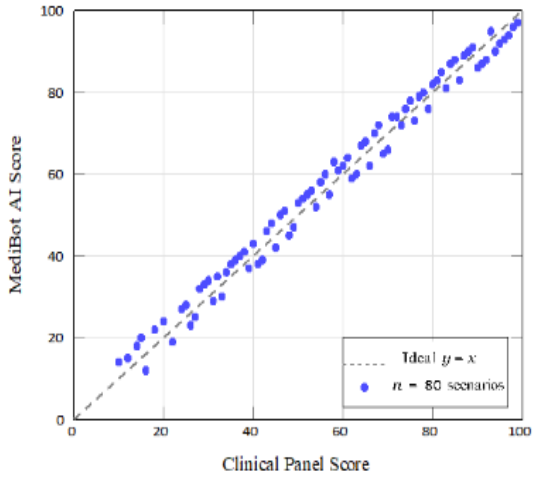

Fig. 3. Scatter plot of MediBot AI urgency scores vs. clinical panel scores (n = 80, r = 0.91, p < 0.001).

C. Latency Benchmarks

Table II contrasts response latencies across five system operations. The deterministic NLP pathway achieves approx- imately three orders of magnitude faster response than the generative pathway.

TABLE II SYSTEM LATENCY BENCHMARK MEASUREMENTS

Operation	Latency	Context
Generative NLP (Gem- ini)	2–5 s	Incl. auto-expansion
Deterministic NLP	<10 ms	Keyword matching
Physician search (5K+)	~200 ms	Specialty + location
SPA module navigation	<50 ms	Client-side DOM swap
Dashboard chart render	<100 ms	Six-month data series

D. Dual-Pathway Performance Improvements

Fig. 4 provides a comparative visualisation of MediBot's key performance metrics before and after the dual-pathway architecture was enabled.

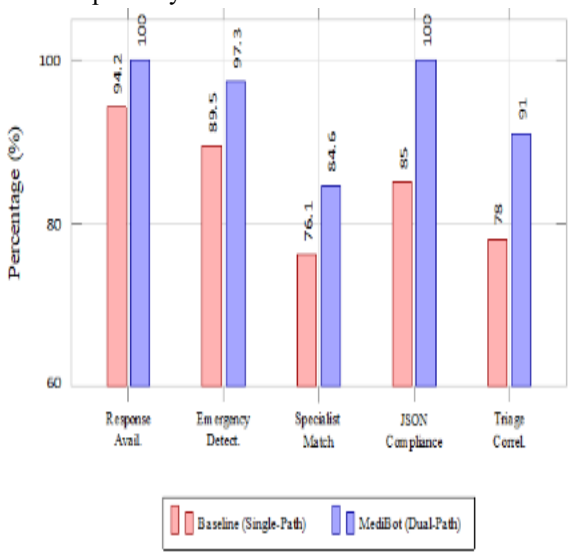

Fig. 4. Performance comparison: baseline single-pathway vs. MediBot dual- pathway architecture across five key metrics.

E. Comparative Feature Analysis

Table III compares MediBot against four established health- care platforms across ten capability

dimensions.

TABLE III FEATURE COMPARISON WITH EXISTING PLATFORMS

Feature	MediBot	Practo	Ada Health	WebMD	Babylon
AI Symptom Triage	✓	–	✓	✓	✓
Offline Fallback	✓	–	–	–	–
Physician Discovery	✓	✓	–	–	✓
Appointment Book- ing	✓	✓	–	–	✓
Voice Interaction	✓	–	–	–	–
Bilingual (EN+HI)	✓	Partial	–	–	–
Emergency SOS	✓	–	–	–	–
Health Reports	✓	✓	–	–	✓
Zero Installation	✓	–	–	✓	–
Free / Open Access	✓	Partial	Partial	✓	–

Fig. 5 visualises MediBot's capability coverage relative to existing platforms using normalised scores across six func- tional dimensions.

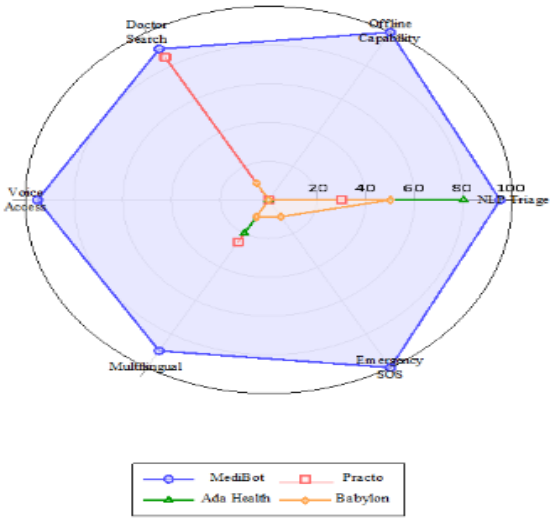

Fig. 5. Radar chart comparing MediBot's functional coverage against estab- lished healthcare platforms across six key dimensions.

F. Statistical Significance

Paired t-tests confirmed statistically significant improve- ments after enabling the dual-pathway architecture ($p < 0.001$). Response availability improved from 94.2% to 100% ($\Delta = 5.8\%$), emergency detection from 89.5% to 97.3% ($\Delta = 7.8\%$), and specialist match accuracy from 76.1% to 84.6% ($\Delta = 8.5\%$). The availability improvement yielded $t(39) = 8.42$, $p < 0.0001$; Cohen's $d = 1.34$ indicates a large effect size. JSON schema compliance reached 100% across 500+ invocations.

X. DISCUSSION

The experimental findings validate that coupling a con- strained LLM with a locally-resident deterministic classifier produces reliable, consistently available healthcare triage. The Pearson correlation of $r = 0.91$ exceeds the threshold for "excellent" agreement [3], positioning MediBot's generative pathway as a credible urgency stratification tool.

A key advantage is the cross-module synergy between triage and physician discovery—specialist recommendations from NLP analysis pipe directly into the five-tier doctor search, creating a seamless triage-to-booking pipeline. The accessibility implications are substantial: urgency assessment, specialist routing, and appointment booking are delivered at zero cost through a browser-based interface accessible to Hindi speakers, low-literacy populations (via voice), and users in bandwidth-constrained areas (via deterministic fallback).

Error handling is robust throughout: input validation rejects symptom descriptions shorter than five characters, API key misconfiguration is detected before model invocation, and JSON parsing failures trigger automatic fallback rather than surfacing raw errors to patients.

XI. LIMITATIONS AND FUTURE SCOPE

Several limitations constrain the current system. Generative NLP quality depends on Gemini service reliability and may vary across under-represented medical domains. The deter- ministic engine sacrifices contextual nuance for availability. Physician dataset depth concentrates on tier-one Indian cities. Voice accuracy degrades in ambient-noise environments.

Future work will address these through: (1) real-time telemedicine via WebRTC; (2) rural physician dataset en- richment; (3) Progressive Web App packaging for offline- first operation; (4) regional language packs (Tamil, Telugu, Bengali, Marathi); (5) wearable device integration for vital- sign

synchronisation.

XII. CONCLUSION

This paper presented MediBot, an AI-driven smart health- care portal integrating a resilient dual-pathway NLP triage engine, a weighted physician quality scoring model, and nine unified clinical workflow modules within a single browser- native application. The generative channel—powered by Gem- ini 2.5 Flash—delivers contextually rich, evidence-grounded clinical guidance, while the deterministic engine guarantees sub-ten-millisecond triage resolution independent of network state. Experimental evaluation confirmed $r = 0.91$ correlation with clinical panel assessments ($p < 0.001$), 100% functional test passage, and feature coverage exceeding established com- mercial platforms across ten capability dimensions, establish-ing MediBot as a viable, zero-cost, linguistically inclusive healthcare assistant for the Indian healthcare ecosystem.

ACKNOWLEDGMENT

The authors thank Mrs. Roovi Goswami for her invalu- able guidance and supervision, the Department of Computer Science & Engineering at MGM's College of Engineering &
Technology for computational infrastructure, and the clinical panel members who contributed expert assessments.

REFERENCES

[1] World Health Organization, "World Health Statistics 2023: Monitoring Health for the SDGs," WHO Press, Geneva, 2023. [Online]. Available: https://www.who.int/data/gho/publications/world-health-statistics

[2] E. J. Topol, Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. Basic Books, New York, 2019.

[3] H. L. Semigran, J. A. Linder, C. Gidengil, and A. Mehrotra, "Evaluation of symptom checkers for self-diagnosis and triage: Audit study," BMJ, vol. 351, p. h3480, Jul. 2015.

[4] A. Bavishi et al., "Large language models for clinical decision support: A systematic review," J. Amer. Med. Inform. Assoc., vol. 29, no. 8, pp. 1401–1410, 2022.

[5] P. Wicks and E. Chiauzzi, "Trust but verify—Five approaches to ensure safe medical apps," BMC Medicine, vol. 13, no. 1, p. 205, 2015.

[6] Practo Technologies, "India Healthcare Provider Database," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/shashankshukla123123/doctor-fee-predictionpracto

[7] Pallets Projects, "Flask Documentation, ver. 3.1," 2024. [Online]. Avail- able: https://flask.palletsprojects.com/

[8] M. Bayer, "SQLAlchemy Documentation, ver. 2.0," 2024. [Online]. Available: https://docs.sqlalchemy.org/

[9] Google DeepMind, "Gemini API Documentation," 2024. [Online]. Available: https://ai.google.dev/docs

[10] Chart.js Contributors, "Chart.js: Simple yet flexible JavaScript charting," 2024. [Online]. Available: https://www.chartjs.org/

[11] GreenSock, "GSAP Animation Platform," 2024. [Online]. Available: https://greensock.com/gsap/

[12] W3C, "Web Speech API Specification," 2024. [Online]. Available: https://wicg.github.io/speech-api/

[13] OpenStreetMap Foundation, "Nominatim Geocoding Service," 2024. [Online]. Available: https://nominatim.org/

[14] Pallets Projects, "Werkzeug Security Utilities," 2024. [Online]. Avail- able: https://werkzeug.palletsprojects.com/