IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

MediGuard: A Safety-First AI-Assisted Healthcare Companion Platform

Vidhi Ingle[1], Pratik Wankhede[2], Sayali Deshmukh[3], Sujal Jadhav[4], Prof. R. C. Awatade[5]

Student, Department of Artificial Intelligence and Data Science, Prof. Ram Meghe Institute of Technology & Research, Amravati, India[1,2,3,4]

Faculty, Prof. Ram Meghe Institute of Technology & Research, Amravati, India[5]

Abstract: Access to reliable health information, routine self-monitoring and appropriate specialist guidance remains uneven, and general-purpose chatbots can produce unsafe or overconfident medical statements. This paper presents CareBot AI, a modular web platform that combines a conversational health assistant with a deterministic safety layer, a vitals tracker, a medicine and hydration manager, a medical-report analyser based on optical character recognition (OCR), a rule-based specialist recommender, a hospital finder and a transfer-learning pipeline for X-ray classification. The system uses a Flask REST back end with JWT authentication and SQLAlchemy persistence, and a plain HTML/CSS/JavaScript front end. A central design principle is that emergency detection, measurement-threshold checks and disclaimers are rule-based and independent of the language model, and that every component that is not yet configured reports its status honestly instead of fabricating output. An automated suite of 55 tests, covering authentication, per-user data isolation, input validation, safety behaviour and graceful degradation, passes on the implemented back end. The paper describes the architecture, module design and safety mechanisms, and discusses current limitations and the validation required before any clinical use.

Keywords: Healthcare chatbot, AI health assistant, safety layer, medical report OCR, X-ray analysis, transfer learning, Flask REST API, health monitoring

I. INTRODUCTION

Digital health tools are increasingly used for symptom enquiry, self-monitoring and the interpretation of medical documents, yet many users still face fragmented applications, limited access to clinicians and difficulty understanding medical terminology. Conversational agents have shown promise for patient-facing tasks, but the evidence base is heterogeneous and their safety depends strongly on how they are designed [1]. Large language models (LLMs) encode a considerable amount of clinical knowledge [2], but they can also produce fluent and confident statements that are wrong, which is unacceptable when a user describes a possible emergency. International guidance on artificial intelligence for health therefore stresses transparency, accountability and human oversight [3].

CareBot AI is a healthcare companion that treats safety as an architectural property rather than a prompt-engineering afterthought. Instead of delegating everything to a language model, the platform separates the tasks that must be deterministic (emergency detection, vital-sign threshold checks, medicine logging, input validation and access control) from the tasks that benefit from generative language (explaining health information in plain language). All of these capabilities are delivered through one authenticated web application so that a user can monitor vitals, manage medicines and water intake, upload a medical report, ask questions and find a suitable specialist without switching tools. The main contributions of this work are as follows:

- A layered and modular architecture (REST API, service layer, relational data layer and machine-learning package) with one blueprint per feature.
- A rule-based safety layer that runs before and after the LLM call, so an emergency reply never depends on the model being reachable or well behaved.
- An honest-degradation policy: OCR, LLM, maps and X-ray modules return an explicit “unavailable” or “unsupported” status instead of generating invented results.
- Per-user data isolation, JWT-based authentication and a role-restricted administrator console.
- An automated test suite of 55 tests that passes on the implemented back end.

The rest of the paper is organised as follows. Section II reviews related work, Section III states the problem and objectives, Section IV presents the system architecture, Section V details the individual modules, Section VI discusses safety and

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510123

ethical design, Section VII reports testing results, Section VIII lists limitations and future scope, and Section IX concludes.

II. RELATED WORK

A. Conversational Agents and Language Models

A systematic review of conversational agents in healthcare found applications in self-management, education and support, but also considerable variation in evaluation quality and a need for careful safety design [1]. Transformer-based language models have since improved markedly: few-shot learners demonstrated strong general language ability [4], and later work showed that LLMs can encode clinical knowledge while still falling short of clinician-level reliability [2]. These results motivate using an LLM for plain-language explanation, but they also motivate guardrails that do not rely on the model itself.

B. Deep Learning for Medical Imaging

Deep convolutional networks have achieved dermatologist-level skin-lesion classification [5] and have been applied to retinal and paediatric chest images using transfer learning [6]. CheXNet reported radiologist-level pneumonia detection on chest X-rays [7]. Transfer learning, in which a network pre-trained on a large source domain is adapted to a smaller target task, is surveyed in [8], and lightweight backbones such as MobileNetV2 make this practical on modest hardware [9]. These studies rely on large curated datasets and rigorous held-out evaluation. CareBot AI therefore keeps its X-ray module gated until a permitted dataset is supplied and does not report accuracy figures that have not been measured.

C. OCR for Clinical Documents

Open-source OCR engines such as Tesseract [10] can convert scanned reports to text, but laboratory reports vary widely in layout and print quality. CareBot AI therefore treats OCR output as an aid for reading rather than as ground truth: the original text is preserved, and any structured field extracted from it is flagged as low confidence.

III. PROBLEM STATEMENT AND OBJECTIVES

Existing consumer health applications typically address a single need, and general chatbots do not guarantee safe behaviour when a user describes a serious symptom. The problem addressed here is to design an integrated, privacy-aware healthcare companion whose safety-critical behaviour is predictable, testable and independent of any single AI component. The objectives are:

- to provide secure registration, login and per-user storage of all health records;
- to track blood pressure, heart rate, medicines and water intake with cautious, non-diagnostic feedback;
- to provide an AI health assistant that never presents itself as a doctor and escalates emergencies immediately;
- to extract text from uploaded medical reports and to support CNN-based X-ray classification when a validated dataset is available;
- to suggest a relevant specialist and nearby facilities, and to give administrators visibility of system status.

IV. SYSTEM ARCHITECTURE

A. Layered Overview

CareBot AI follows a layered client–server design that uses the REST architectural style [11]. As shown in Fig. 1, the presentation layer is a static HTML/CSS/JavaScript client that exchanges JSON messages with the API layer over HTTP. Protected requests carry a JWT bearer token. The API layer delegates to a service layer that contains the business logic, which in turn uses a relational data layer, file storage for uploads, and an optional machine-learning package. External services (an LLM provider, the Tesseract engine and a maps API) are optional and are accessed only through the service layer.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

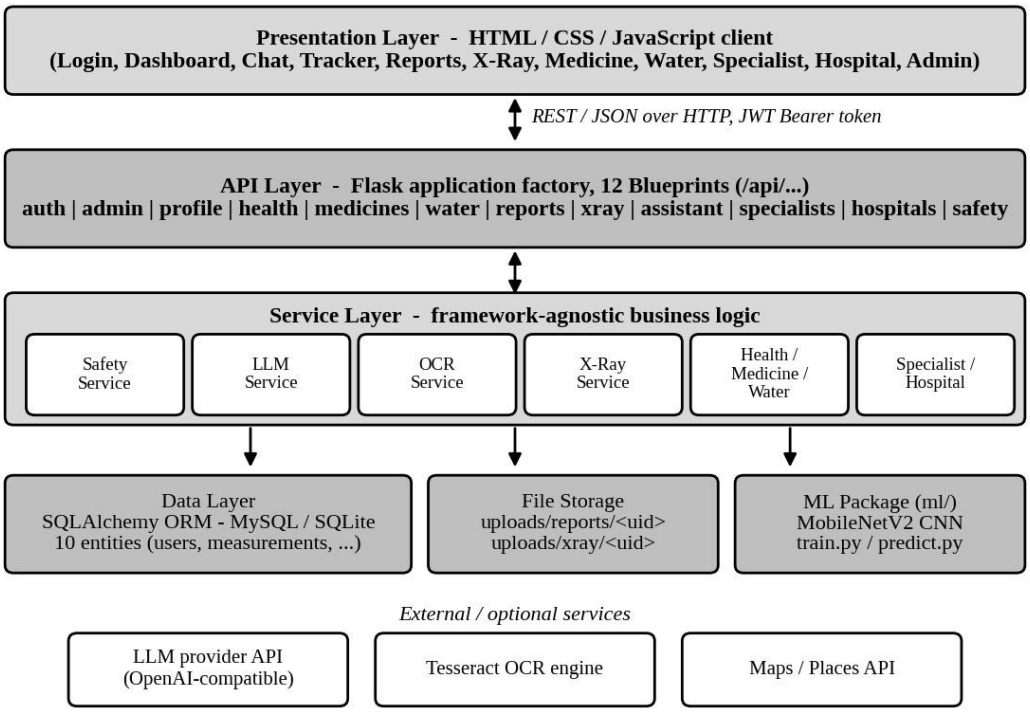


Fig. 1 Layered architecture of CareBot AI

B. Back-End Organisation

The back end is built with Flask [12] using the application-factory pattern. The factory creates the application, enables cross-origin requests for the separate front end, initialises the database and JWT extensions, and registers twelve blueprints under the /api prefix: auth, admin, profile, health, medicines, water, reports, xray, assistant, specialists, hospitals and safety. Route modules handle only the HTTP concerns of parsing, validation and response shaping. Business logic lives in framework-agnostic service modules, and helper modules provide validators, password hashing and a uniform response envelope of the form {success, message, data}.

Centralised error handlers convert 404, 405, 413 and unhandled exceptions into JSON error responses without exposing stack traces, and the application logs request, authentication and error events without logging secrets. On first start the application creates all tables and seeds three demonstration facilities and one administrator account. The database defaults to a local SQLite file for development and can be switched to MySQL through a single connection-string setting.

C. Data Model

The data layer consists of ten entities mapped with SQLAlchemy (Table I). Every user-owned record carries an indexed foreign key to the owning user, which the routes use to restrict every query to the authenticated user.

TABLE I ENTITIES OF THE DATA MODEL

Entity	Purpose	Key attributes
User	Account and role	name, email (unique), password hash, role (patient/admin)
HealthProfile	Basic profile	age, gender, known conditions, preferences
Measurement	Vital-sign reading	systolic, diastolic, heart rate, value/unit, source, timestamp
Medicine	Medicine schedule	name, dosage, schedule time, start and end date
MedicineLog	Dose history	medicine id, scheduled time, status (taken/missed)

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

WaterLog	Hydration entry	amount in ml, timestamp
Report	Uploaded report	file names, type, extracted text, structured data, OCR status
Entity	**Purpose**	**Key attributes**
XrayResult	X-ray analysis record	region, predicted class, confidence, model version, supported flag
Conversation	Assistant history	user message, assistant response, timestamp
Facility	Hospital / clinic	name, specialty, address, phone, source

D. Technology Stack

Table II summarises the technologies used. The stack follows common open-source practice so that the prototype can be reproduced on an ordinary workstation.

TABLE II TECHNOLOGY STACK

Layer	**Technology**
Front end	HTML5, CSS3 and JavaScript (no framework, no build step)
Web framework	Flask 3.0 with Flask-CORS
Authentication	Flask-JWT-Extended (JWT access tokens), Werkzeug password hashing
Persistence	Flask-SQLAlchemy; SQLite (development) or MySQL via PyMySQL
OCR (optional)	Tesseract through pytesseract, Pillow and pdf2image
Deep learning (optional)	TensorFlow/Keras with a MobileNetV2 backbone
LLM (optional)	Any OpenAI-compatible chat-completion endpoint, set through environment variables
Testing	pytest with an in-memory SQLite configuration

V. MODULE DESIGN

A. Authentication and Access Control

Users register with a name, e-mail address and password. Passwords are stored only as salted hashes produced by the Werkzeug security helpers. On login the server issues a JSON Web Token [13] with a twelve-hour lifetime, and the client sends it in an Authorization header. Logging out places the token identifier in a server-side blocklist so that the token is rejected afterwards. Two roles exist, patient and admin. Every data route reads the user identity from the token and filters queries by it, and every admin route additionally verifies the admin role and returns HTTP 403 otherwise.

B. Health Tracker and Safety Thresholds

Users record systolic and diastolic blood pressure and heart rate. Before storage, readings are checked against broad plausibility ranges (systolic 40–300 mmHg, diastolic 20–200 mmHg, heart rate 20–250 bpm) so that impossible values never reach the database; this is a data-quality check and not a medical judgement. Each reading is labelled with its source (user entered or device). After saving, the safety service compares the values with cautious thresholds (Table III). When a value is outside these limits, the response includes a safety note that states plainly that it is not a diagnosis and suggests contacting a healthcare professional.

TABLE III CAUTIOUS THRESHOLDS USED BY THE SAFETY SERVICE

Parameter	**Lower limit**	**Upper limit**
Systolic blood pressure (mmHg)	≤ 90	≥ 180
Diastolic blood pressure (mmHg)	not checked	≥ 120
Heart rate (bpm)	≤ 40	≥ 130

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

The trend endpoint reports the mean systolic pressure and heart rate together with a direction of increasing, decreasing or stable. The direction is obtained by comparing the mean of the later half of the recorded values with the mean of the earlier half. The method is intentionally simple and explainable and is not clinical trend analysis.

C. Medicine and Water Management

The medicine manager supports creating, listing, updating and deleting medicines with a name, dosage, schedule time and optional start and end dates. Each dose can be logged as taken or missed, and deleting a medicine also removes its logs. The water tracker records intake in millilitres and returns a summary containing today's total, progress against a configurable daily target (2000 ml by default), and the seven-day total and average. Both modules use plain deterministic logic; no LLM is involved.

D. AI Health Assistant

The assistant pipeline is shown in Fig. 2. A chat request is first authenticated and validated. The safety service then scans the message for emergency expressions such as chest pain, difficulty breathing, severe bleeding, unconsciousness, seizure, stroke, overdose, severe allergic reaction and expressions of self-harm. If a match is found, a fixed escalation message that urges the user to contact local emergency services is returned, the exchange is stored, and the LLM is never called.

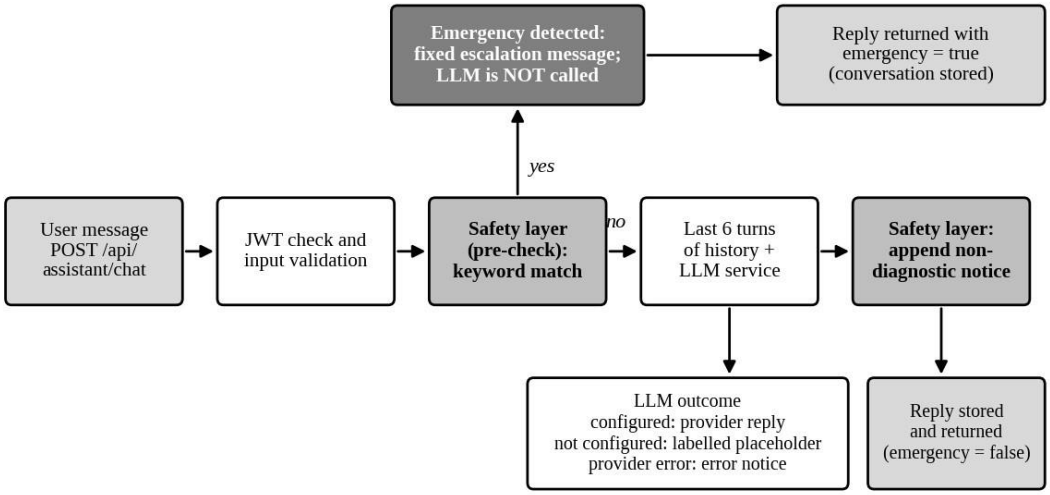


Fig. 2 Safety-gated request flow of the AI health assistant

Otherwise the service builds a request containing a system instruction and the six most recent stored turns of the user's own conversation. The system instruction requires the model to avoid definitive diagnoses, avoid prescribing treatments or dosages, use plain language, advise professional consultation when judgement is needed, and escalate emergencies. The request is sent to a configurable OpenAI-compatible endpoint with a low temperature (0.4) and a 20-second timeout. A standard non-diagnostic notice is appended to every model reply. When no provider is configured, the assistant returns a clearly labelled placeholder rather than a fabricated answer, and a provider failure is reported as an error notice instead of crashing the request.

E. Specialist Recommendation and Hospital Finder

The specialist recommender maps symptom keywords to one of seven specialties (cardiology, dermatology, neurology, orthopaedics, paediatrics, psychiatry and general medicine) using an ordered rule table, and defaults to a general physician when nothing matches. The explanation always states that the output is a general suggestion rather than a clinical determination. The hospital finder filters a facility table by specialty. Seeded records are marked with the source "demo" so they cannot be mistaken for live data, and the service is structured so that a maps or places API can replace the table lookup without changing the route.

F. Medical Report Analyzer

Report upload accepts PDF, PNG and JPEG files up to a configurable size (10 MB by default). File names are sanitised, stored under unique names in a per-user directory, and checked against the allowed extensions. Text is extracted with Tesseract [10]; PDF pages are first converted to images. A simple pattern matcher then identifies lines of the form "label:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

number unit" and stores them as structured fields, each marked as low confidence, while lines that do not match are left out instead of being guessed. The record keeps an explicit OCR status of success, unavailable, failed or pending. If the OCR packages or the Tesseract binary are not installed, the upload still succeeds and the status is reported as unavailable.

G. X-Ray Analyzer

The X-ray module (Fig. 3) is implemented as a separate machine-learning package so that it can be trained and evaluated independently of the web application. Images are resized to 224×224 pixels and scaled to the range [0, 1]. The classifier uses a MobileNetV2 backbone [9] pre-trained on ImageNet and frozen by default, followed by global average pooling, dropout (0.3), a 128-unit ReLU layer and a softmax output with one unit per class. It is compiled with the Adam optimiser and categorical cross-entropy using TensorFlow [14]. A training script reads an ImageFolder-style dataset (train, optional validation and test folders), and a dataset-inspection helper reports the actual classes and image counts before training.

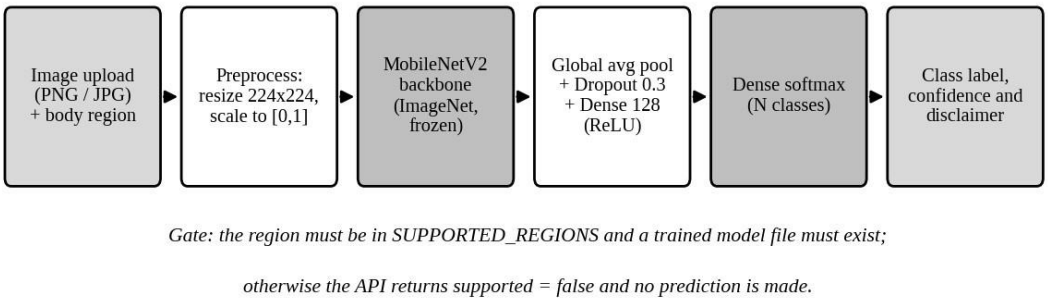


Fig. 3 Transfer-learning pipeline of the X-ray analyzer

Inference is gated: a region is accepted only if it has been listed as supported and a trained model file exists for it. In every other case the API returns supported = false with an explanatory message and makes no prediction. Each analysis is stored together with the region, predicted class, confidence, model version and supported flag, and every successful result carries a statement that it is an AI-assisted prediction and not a diagnosis. In the current prototype no dataset has been configured, so all regions are reported as unsupported; no accuracy figure is claimed.

H. Administrator Console

The administrator console, which requires an account with the admin role, shows project-wide record totals, a list of member accounts and profile summaries, and a module-connectivity view. The connectivity view distinguishes registered API routes from feature readiness, so that a module with an unconfigured LLM, unavailable OCR, an unconfigured X-ray model or demonstration-only hospital data is clearly identified. A demo-reset operation removes non-admin accounts and their records and is therefore intended for demonstration use only.

VI. SAFETY, PRIVACY AND ETHICAL DESIGN

A. Non-Diagnostic Positioning

Consistent with international guidance on AI for health [3], CareBot AI never presents itself as a clinician. Every generated reply carries a non-diagnostic notice, threshold notes explicitly state that they are not diagnoses, specialist suggestions are labelled as general guidance, and X-ray output is described as AI-assisted and subject to review by a qualified physician.

B. Honest Degradation

Each optional component reports its true state. Without an LLM key the assistant says so; without Tesseract the report status is unavailable; without a dataset the X-ray endpoint returns an unsupported response; and demonstration hospital data is labelled as such. This policy prevents the system from silently inventing clinical content and makes the configuration state visible to administrators.

C. Privacy and Data Isolation

Health data are sensitive. All user-owned queries are filtered by the authenticated identity, uploaded files are stored in per-user directories, secrets are read from environment variables rather than source code, passwords are hashed, and error responses do not leak internal details. The test suite verifies that one user cannot read another user's medicines, measurements or reports.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510123

VII. TESTING AND RESULTS

A. Test Setup

The back end was verified with an automated pytest suite that uses an in-memory SQLite configuration, so the tests never touch the development database. Tests call the API through Flask's test client and cover both successful behaviour and error paths.

B. Results

Table IV lists the number of tests per area. All 55 tests passed with no failures in the final run.

TABLE IV AUTOMATED TEST SUITE SUMMARY

Area	Tests	Behaviour verified
Authentication	8	Registration, duplicate e-mail, missing fields, login, wrong password, protected-route rejection, current user, logout token revocation
Profile	3	Empty profile, create and update, authentication required
Health tracker	7	Add reading, safety note on out-of-range value, rejection of implausible values, empty trends, trends with data
Medicines	6	Create, missing name, update, dose logging, invalid status, delete
Water	3	Log, invalid amount, daily and weekly summary
Reports	4	No file, invalid extension, upload without OCR installed, empty list
X-ray	5	Unconfigured status, unsupported region, missing region, no file, invalid extension
Assistant	4	Honest placeholder without LLM, emergency detection, missing message, history recording
Specialist / hospital	5	Recommendation, default specialty, missing symptom, facility search and filter
Security isolation	3	User cannot access another user's medicines, measurements or reports
Admin	7	Role enforcement (403 for patients), summary, member profiles, module connectivity
Total	**55**	**55 passed, 0 failed**

C. Discussion

The results show that the safety-critical behaviour is deterministic and testable: the emergency path returns a fixed response without invoking the model, out-of-range readings always produce a cautious note, unconfigured components degrade to explicit statuses, and cross-user access is rejected. The tests confirm functional correctness of the back end only. No clinical accuracy study, usability study or load test has been performed, and none is claimed.

VIII. LIMITATIONS AND FUTURE SCOPE

The current prototype has the following limitations, which also define the future work:

- **Front-end integration.** The static client currently falls back to local sample data when the API is unavailable; binding every page to the authenticated API is ongoing.
- **X-ray model.** No dataset has been supplied and no model has been trained. Future work includes training on a permitted public dataset, evaluating on held-out data with sensitivity and specificity, and adding visual explanations such as saliency maps.
- **Lexical safety rules.** Emergency and specialist detection rely on keyword matching and can miss paraphrases or other languages. Semantic and multilingual classifiers should be evaluated alongside the rules, not instead of them.
- **Production hardening.** The token blocklist is held in memory, the password rule is minimal, cross-origin access is open and the default secrets are for development. A deployment would require a persistent blocklist, a stronger password policy, restricted origins, HTTPS and managed secrets.
- **Data sources.** Hospital data are seeded demonstration records, and report field extraction uses simple patterns. Integration with a maps API and layout-aware extraction are planned.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510123

- **Validation.** The platform has not been clinically validated. Clinician review and a usability study with real users are necessary before any health-related deployment.

IX. CONCLUSION

This paper presented CareBot AI, a modular healthcare companion that integrates a conversational assistant, health and medicine tracking, report OCR, specialist guidance and an X-ray analysis pipeline behind a secure REST API. Its distinguishing feature is a safety-first design in which emergency detection, threshold checks, disclaimers, access control and honest status reporting are deterministic and independent of the language model. A suite of 55 automated tests confirms the intended behaviour of the back end. The main remaining work is to connect the complete client to the API, train and rigorously evaluate the X-ray model on a permitted dataset, and carry out clinical and usability validation, after which the platform could be assessed for real-world use.

ACKNOWLEDGMENT

The authors thank Prof. R. C. Awatade and Prof. Ram Meghe Institute of Technology & Research, Amravati, for their guidance and support during this work.

REFERENCES

[1] L. Laranjo et al., "Conversational agents in healthcare: a systematic review," *J. Am. Med. Inform. Assoc.*, vol. 25, no. 9, pp. 1248–1258, 2018.
[2] K. Singhal et al., "Large language models encode clinical knowledge," *Nature*, vol. 620, pp. 172–180, 2023.
[3] World Health Organization, *Ethics and Governance of Artificial Intelligence for Health: WHO Guidance*. Geneva, Switzerland: WHO, 2021.
[4] T. B. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
[5] A. Esteva et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115–118, 2017.
[6] D. S. Kermany et al., "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.
[7] P. Rajpurkar et al., "CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning," arXiv:1711.05225, 2017.
[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.
[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.
[10] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, vol. 2, 2007, pp. 629–633.
[11] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.
[12] Pallets Projects, "Flask documentation," https://flask.palletsprojects.com (accessed Oct. 29, 2026).
[13] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," *RFC 7519*, Internet Engineering Task Force, 2015.
[14] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

 This work is licensed under a Creative Commons Attribution 4.0 International License