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

AI-Driven Medical Misinformation Detection and Health Guidance System

B Dhanvith Naik[1], Joel Joseph Jecob[1], Maneesh S[1], Shrivatsa S[1], Abhishek Kumar K[2]

[1]Students, Department of Artificial Intelligence & Machine Learning, Vivekananda College of Engineering & Technology, Puttur, Karnataka, India

[2]Assistant Professor, Department of Artificial Intelligence & Machine Learning, Vivekananda College of Engineering & Technology, Puttur, Karnataka, India

Email: dhanvithnaik205@gmail.com,joeljosephjecob@gmail.com,maneeshsb4@gmail.com,vatsa142005@gmail.com

Abstract - Health advice online spreads faster than anyone can check it, and a person holding a lab report or a worrying symptom usually has to piece together help from several unrelated tools. This paper describes a web application that puts four of those jobs in one place: checking health claims, giving general guidance on symptoms, reading uploaded medical reports, and keeping a searchable personal record. Claims are labelled SUPPORTED, CONTRADICTED or UNCERTAIN by a TF-IDF and logistic regression classifier. Where more evidence is needed, the system retrieves passages with Sentence Transformers and FAISS, searches the web through Tavily, and can query the Google Fact Check API. A validator that runs independently of the language model then scores the answer and flags low confidence. Symptoms are matched against predefined rules, with Tesseract OCR reading text from images. PDF and image reports are parsed with PyPDF and Tesseract, and the measurements are stored in SQLite. The interface uses React and Vite on a FastAPI backend. Three worked examples demonstrate it. No aggregate accuracy figures are reported, because a formal benchmark has not yet been run. It gives general guidance only and does not replace a clinician.

Key Words: Health Misinformation, Claim Verification, Retrieval-Augmented Generation, FAISS, OCR, Personal Health Records.

1. INTRODUCTION

Ask a search engine whether you really need eight glasses of water a day and you will get a confident answer in under a second. Ask again in different words and the answer may change. Health information online behaves like this. There is a great deal of it, the quality is uneven, and almost none of it is sorted for the person who is reading [1], [2]. Generative AI has made the problem bigger, because convincing but false text is now cheap to produce [3].

Spotting a false claim is only half the job. A bare label such as “false” rarely changes what someone believes. People also need a short explanation they can follow and a source they can open [4], [5]. Language models can write that explanation fluently, but fluent is not the same as correct [6], and a medical model can be pushed into repeating wrong facts [7]. An assistant of this kind therefore has to lean on evidence that sits outside the model, and it has to say so plainly when the evidence is thin.

Claims are not the only health problem people bring to a screen. Someone receives a blood test as a PDF, or as a photograph of a printed sheet, and cannot tell which numbers matter. Someone else types a few symptoms into a search box and gets a list of frightening conditions. Last year’s results sit in a different folder from this year’s. In practice each of these needs its own tool, and none of the tools knows about the others.

This paper describes a web application that handles these jobs together. A user can type a health claim and receive a verdict with the evidence behind it. They can describe symptoms, in text or in a photo, and receive general guidance. They can upload a report in PDF or image form and have its test values pulled out and stored, and later they can search those records. The claim classifier uses TF-IDF features with logistic regression. Evidence comes from semantic retrieval with Sentence Transformers and FAISS, web search through Tavily and the Google Fact Check API. Groq and OpenRouter supply the language model that drafts answers, and an independent validator scores each result before it is shown. The frontend is written in React with Vite and the backend in Python with FastAPI.

The main contributions are as follows. First, we combine claim verification, symptom guidance, report reading and personal record keeping in one working application instead of four separate tools. Second, we pair a light classifier with retrieval and web evidence, and add a validation step that does not depend on the language model. Third, every answer carries its own evidence, a validation score and a disclaimer, and the system returns UNCERTAIN when it cannot find support. Finally, we describe the design, the testing approach and three worked examples, including the places where the system gets things wrong.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 explains the methodology and Section 4 the implementation. Section 5 covers testing, case studies and discussion, Section 6 looks at safety and transparency, and Section 7 concludes with limitations and future scope.

2. RELATED WORK

2.1 Health misinformation online

Reviews of social media research agree that health misinformation is common, although the exact share depends on the platform, the topic and how it is measured [1]. Li et al. [2] follow it from creation through spread to correction and conclude that it keeps changing, which means any knowledge source behind a detector needs regular updating. Skafle et al. [8] studied COVID-19 vaccine posts and found the same themes returning again and again, so a correction that answers the specific theme is likely to work better than a plain “false”. Van der Linden [5] makes a related point from psychology: why people become receptive to a claim matters as much as the claim itself.

2.2 Automated detection

Feng et al. [9] survey machine learning, deep learning and knowledge-based detectors and list class imbalance,

inconsistent annotation and differences between datasets as recurring problems. Papanikou et al. [10] point out that many methods depend on social-network signals, such as how a post spreads. A stand-alone health tool has no such signals. That is one reason we work from the claim text and the evidence, and not from the user's network.

2.3 Evidence-based fact checking and retrieval

Sarrouti et al. [11] retrieve supporting evidence and compare it with a health claim, and show that choosing the right evidence is a task of its own. Kotonya and Toni [12] go a step further and generate an explanation alongside the verdict on a public-health dataset. Cui et al. [13] bring in structured medical knowledge, though keeping such a knowledge base current takes steady effort. Cao et al. [14] use several model reasoning stages to build evidence before verifying a claim, at the price of more computation, and Bukar et al. [15] combine retrieval from authoritative health sources with a transformer classifier.

Retrieval-augmented generation [16] is the idea behind most of these systems. Xiong et al. [17] benchmark it for medicine and show that retrieval quality strongly affects the final answer. Amugongo et al. [18] add a warning: retrieval does not remove hallucination when the retrieved material is poor.

2.4 Language models, chatbots and explanations

Wang and Zhang [19] review how large language models are used in medicine and list hallucination, privacy and reliability among the open problems. Han et al. [7] show that targeted wrong facts can change a medical model's behaviour while it still performs normally elsewhere, and Wei et al. [6] find that a convincing medical answer is not necessarily a correct one. Healthcare chatbots can make information easier to reach [20], and their replies can come across as empathetic [21], but empathy is not clinical expertise. Surveys of explainable AI [22], [23], [24] stress that users need explanations they can understand. Sadeghi et al. add that an explanation does not prove the prediction behind it is right, which is why we show evidence next to the explanation and not the explanation alone.

2.5 Symptom checkers

Wallace et al. [25] reviewed online symptom checkers and found that their diagnostic and triage accuracy varies and that they cannot be treated as a clinical assessment. We take that finding at face value. Our symptom module offers possible causes as general guidance, with a risk level and a disclaimer, and nothing more.

2.6 Research gap

The literature has strong work on each of these tasks by itself. What we could not easily find is a working application that verifies a claim against retrieved evidence, scores its own answer, gives careful symptom guidance, reads a user's medical reports and keeps them for later, all in one place. The weaknesses that keep appearing, namely poor retrieval, unreliable generated text and overconfident tools [6], [17], [25], shaped our design: evidence before answers, a validator that sits outside the language model, UNCERTAIN as an allowed result, and a clear statement that the tool supports the reader and does not diagnose.

3. PROPOSED METHODOLOGY

3.1 System architecture

The application follows a client-server design, shown in Fig. 1. The React and Vite frontend collects claims, questions, symptoms and report files and talks to the backend through REST calls. The FastAPI backend, run with Uvicorn and using Pydantic models to validate requests, decides which module should handle each request. Four modules sit behind it: health claim verification, symptom analysis, medical report processing and personal health records. A validation and safety stage checks results before they go back to the user, and SQLite holds the stored records. Tavily, Groq, OpenRouter and the Google Fact Check API connect to the backend as external services. API keys are read from environment variables and are not written into the code.

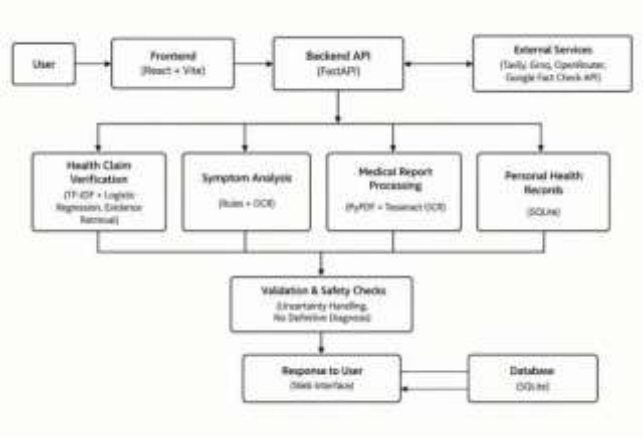


Fig -1: Overall system architecture

Fig. 2 shows how a request moves through the backend. A query is first checked against the existing dataset. If an answer is found it is returned. If not, the language model drafts a response and the system searches for sources. When no source is found the result is UNCERTAIN, and otherwise the draft receives a validation score and is returned. The same figure shows the two other branches: stored personal information can be passed to the language model as context, and an uploaded report goes through OCR into the database. Fig. 3 lists the use cases the interface supports.

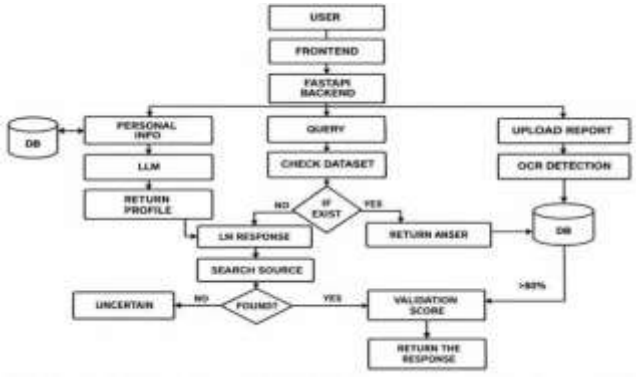


Fig -2: Flowchart of the system

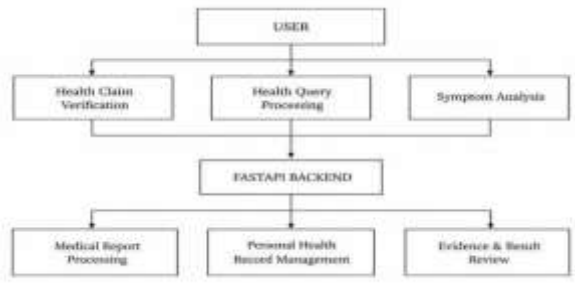


Fig -3: Use case diagram of the system

3.2 Health claim classification

A submitted claim is first cleaned and passed through a health-domain filter, so that the classifier is only applied to text that is actually about health. The text is then converted to TF-IDF features [26]. In the form used by scikit-learn, the weight of a term t in a claim d is

$$w(t, d) = \mathrm{tf}(t, d) \times [\ \ln((1 + N) / (1 + \mathrm{df}(t))) + 1\ ] \qquad (1)$$

where tf is how often the term appears in the claim, N is the number of training claims and $\mathrm{df}(t)$ is the number of claims that contain the term. A word found in almost every claim therefore counts for little, while a specific medical term counts for more. A multinomial logistic regression then gives the probability of each label:

$$P(y = k \mid x) = \exp(w_k \cdot x + b_k) / \Sigma_j \exp(w_j \cdot x + b_j) \qquad (2)$$

The most probable label (SUPPORTED, CONTRADICTED or UNCERTAIN) is reported, and the class probabilities give the confidence value shown beside it. This pair is quick and runs on an ordinary CPU, which suits an application meant for modest hardware. Its limits are discussed in Section 5.5.

3.3 Evidence retrieval and validation

The classifier sees only the claim text, so it cannot know what is true today. For that reason the system adds an evidence path. Sentence Transformers [27] turn the claim and the stored passages into vectors, and FAISS [28] returns the closest ones. When the stored material is not enough, Tavily searches the web, and the Google Fact Check API can be asked about claims that fact-checkers have already reviewed. Groq and OpenRouter are configured as language model providers, with different models available for different jobs. The model reads the retrieved evidence and drafts a short answer and a reason, which the interface shows under "Why this verdict?". Each piece of supporting evidence is listed with a stance, for example CONTRADICTS, and a relevance percentage, together with links to the sources.

The validator is the part we care about most. It runs apart from the language model, checks the evidence behind the draft and returns a validation score and a status such as LOW_CONFIDENCE. If the search finds nothing usable, the result is UNCERTAIN instead of a guess. Keeping the validator separate means that a fluent but unsupported draft cannot vouch for itself.

3.4 Symptom analysis

The user types symptoms or uploads a picture that contains them. For images, Tesseract [29] reads the text, and from there the pipeline treats it like typed input. The symptoms are matched against predefined symptom-condition rules. The result lists a possible cause, a risk level, a short piece of prevention advice and the number of symptoms detected. Every result carries a note that it is not a medical diagnosis, and the advice is to see a doctor if symptoms are severe, getting worse or not improving. Because the rules are written down, they can be read and checked line by line, which matters for health advice.

3.5 Report processing and personal records

A report can be uploaded as PDF, PNG, JPG or WEBP, up to 10 MB. Text is taken from PDFs with PyPDF. Images are opened with Pillow and read with Tesseract. The backend then looks for test names, values, units and reference ranges, and gives each test a status of NORMAL, LOW, HIGH or UNKNOWN. The rows are stored in SQLite and the uploaded files are kept in a configured personal-reports folder. Users can search stored reports by term (for example sugar, Vitamin D or glucose), read summaries, compare values across reports, and ask questions that use the stored values as context. PDF extraction, OCR and storage all run locally.

3.6 Safety rules

A few rules apply to every module. The system never states a diagnosis; it says "possible" and gives guidance. When evidence is missing or conflicting, it says so. When symptoms look serious, it tells the user to see a professional. Inputs are validated before processing, and configuration values such as API keys are read from environment variables.

4. IMPLEMENTATION

4.1 Software

The backend is written in Python and the frontend in JavaScript. Table 1 lists the tools used in each layer. Node.js and npm are needed for the frontend, and Python with the required packages for the backend. Windows, Linux and macOS can all be used for development.

Table -1: Technology used in each layer

Layer	Tools
Interface	React, Vite
Backend	FastAPI, Uvicorn, Pydantic, python-dotenv
Claim classifier	scikit-learn (TF-IDF, logistic regression)
Retrieval	Sentence Transformers, FAISS, Transformers
External services	Tavily, Google Fact Check API, Groq, OpenRouter
Reports and OCR	PyPDF, Pillow, Tesseract OCR
Storage	SQLite
Testing	pytest

4.2 Hardware and network

The application does not need special hardware. The minimum we list is an Intel Core i3 processor at 1.9 GHz, a 64-bit operating system, 4 GB of RAM and 100 GB of disk space. Users reach it from a phone, tablet or computer with a modern browser. The machine that runs the backend needs the Tesseract executable on its PATH to read image reports. Internet access is needed only for the external services, while PDF extraction, OCR and SQLite storage work without it.

4.3 Optional training pipeline

The project also contains a training pipeline for Qwen2.5-0.5B-Instruct [30] with LoRA [31], meant for experiments with healthcare response generation. It is switched off by default and the main application does not depend on it. If it is enabled, GPU training is supported. The evidence validator is independent of this model, so changing or fine-tuning the language model does not change how answers are checked.

4.4 Claim data

The classifier is trained on a labelled collection of health claims. The size and sources of that collection have not been documented in detail yet, so this paper gives no dataset statistics. Building a larger and more varied claim set is the first item on our list of future work.

5. TESTING, CASE STUDIES AND DISCUSSION

5.1 Testing approach

Testing looked at each module on its own and then at the whole workflow, using valid, invalid and awkward inputs. Table 2 summarises the six kinds of test. Backend components are tested with pytest. Across all of them we paid special

attention to safety behaviour: possible conditions must not be shown as diagnoses, and weak or conflicting evidence must be reported as uncertain.

Table -2: Kinds of test and what each one checks

Test	What it checks
Unit	Single functions: API handlers, symptom matching, data processing, retrieval and validation
Integration	Frontend requests, FastAPI services, processing modules, external services and the database working together
Functional	Claim verification, symptom analysis, health queries, report processing and personal records
API	Request validation, response structure and error handling
Retrieval	Whether Sentence Transformers and FAISS return relevant passages, and what happens when none exist
Safety	No definitive diagnosis, no unsupported advice, and sensible handling of missing or conflicting evidence

This testing was about behaviour, not statistics. We have not run a labelled benchmark, so we report no accuracy, precision or recall for the classifier or the retrieval step. The three examples below are real screens from the application. They show what the system does, and they are not a measure of how often it is right.

5.2 Worked example: a health claim

Fig. 4 shows the claim screen. We typed "do humans need to drink 8 glass of water daily to stay alive", exactly as it appears on screen, typo included. The system returned CONTRADICTED with a short answer: people do not have to drink exactly eight glasses to survive, because needs vary and food and other drinks count towards the total. Its "Why this verdict?" note cited Institute of Medicine figures of about 2.7 L a day for women and 3.7 L for men, from all fluids and food together. Fig. 5 and Table 3 show the four evidence items. All four were marked CONTRADICTS, with relevance between 85% and 95%, and four source links were given.

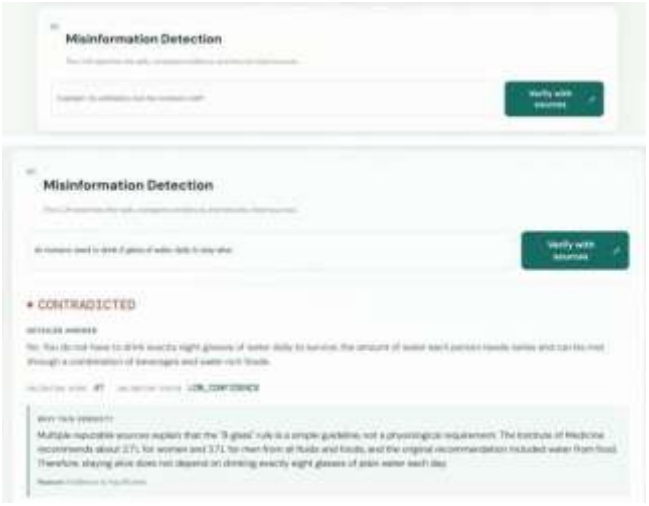


Fig -4: Claim entry screen and the result returned for the water claim



Fig -5: Supporting details and source links for the same claim

Table -3: Evidence items returned for the water claim (paraphrased)

#	Stance	Relevance	Point made
1	Contradicts	95%	Total intake of 9–13 cups comes from drinks and food; the 8-glass figure was never meant as plain water only
2	Contradicts	92%	Institute of Medicine totals of 2.7 L (women) and 3.7 L (men); the 8-glass rule is arbitrary
3	Contradicts	88%	Most people can ignore the rule if they drink when thirsty and eat a balanced diet
4	Contradicts	85%	An easy target, but needs vary: some people need less and some more

One detail deserves attention. The verdict was clear and all four items agreed, yet the validation score was only 47 and the status read LOW_CONFIDENCE, with the reason "Evidence is insufficient." This shows that the validator works apart from the model's own confidence, which is what we intended. Of the four links, two came from health systems, one was a video page and one was a pharmacy blog, so the mix of sources may have pulled the score down. We have not tested that explanation, and finding out what drives the score is a task for the next version.

5.3 Worked example: symptoms

Fig. 6 shows the symptom screen and one result. The input was headache, body pain and throat pain. The system detected three symptoms, named sinusitis as a possible cause, gave a low risk level, and advised rest, fluids and seeing a doctor if the symptoms were severe, worsening or not improving. No image was supplied, so the OCR status read NO IMAGE PROVIDED. The screen states clearly that the result is not a medical diagnosis. Three loosely related symptoms can fit many conditions, so the single cause named here is a starting point and not an answer.

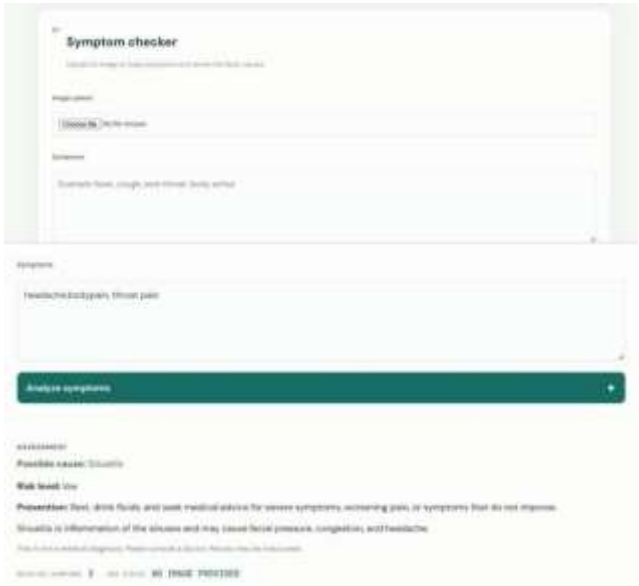


Fig -6: Symptom checker screen and the assessment returned

5.4 Worked example: a medical report

Fig. 7 shows the records screen, where a report is uploaded and stored reports are searched, and the test details extracted from one blood report. Each row has a test name, a value, a reference range and a status. The screen is useful, and it also shows the weak spot of this module. Rows such as Phone and Mobile appear as tests. Some names are garbled (Pov, Mev and ROW stand for PCV, MCV and RDW). Some values and ranges are plainly misread: the RBC count has a range of 14–169, the MCV value is 7200, and the platelet count is 11,550,000. These are the kind of errors that OCR followed by line-by-line parsing produces. Users should always compare the extracted table with the original report.

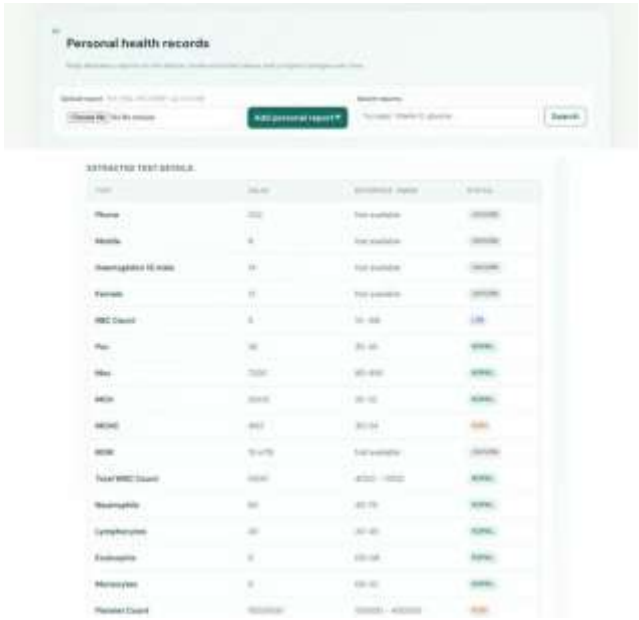


Fig -7: Personal health records screen and the test details extracted from a report

5.5 Discussion and limitations

Putting four tasks in one application has a practical benefit. A user does not have to move between tools, and the claim checker, the symptom module and the report reader all follow the same rules about uncertainty and disclaimers. The limits are just as real, and we would rather list them than hide them.

First, there is no quantitative evaluation yet, and the claim data is not described in detail. Second, a TF-IDF classifier sees words and not meaning. A claim that uses the same words as a true one but means the opposite can be misread, and a topic that is missing from the training data will be handled poorly. This is a main reason the system leans on evidence and a validator. Third, answer quality depends on what the search returns, and as the water example shows, source quality varies. Fourth, the symptom rules cover only the conditions written into them. Fifth, report parsing is fragile when the scan or the layout is poor, as Fig. 7 shows. Finally, some features send text to outside services, so claim and query text leaves the machine, and stored report values would too if they were passed as context to a hosted model. Users should be told this. The system supports the reader and does not replace a doctor.

6. SAFETY AND TRANSPARENCY

Because the application deals with health, we treated transparency as a feature and not as an afterthought. A claim result shows the label, a detailed answer, the reason for the verdict, the evidence with its stance and relevance, links to the sources, and the validation score and status. A user who doubts the verdict can open the sources and judge for themselves. The explanation text is written by a language model, so on its own it is not proof [23]. That is why the evidence and the validation score sit right beside it.

Uncertainty is shown, not hidden. A claim with no usable evidence returns UNCERTAIN, a weak evidence set returns a low-confidence status, and symptom results always include a disclaimer and advice to see a professional when symptoms are serious. Reports and extracted values stay on the machine that runs the application, apart from any text sent to the external services discussed in Section 5.5. The validator is independent of the language model, so a change of model provider does not change how answers are judged.

7. CONCLUSION AND FUTURE SCOPE

This paper presented a web application that brings health claim verification, symptom guidance, medical report reading and personal record keeping into one place. A TF-IDF and logistic regression classifier gives the first label, Sentence Transformers, FAISS, Tavily and the Google Fact Check API supply evidence, and a validator that is independent of the language model scores the result. OCR and PDF parsing turn reports into searchable records. The system can say UNCERTAIN, shows its sources, and never presents a possible condition as a diagnosis. It is meant to support readers and not to replace clinical advice.

Much remains to be done. The first task is a proper evaluation, with a larger and more varied claim set and measured accuracy for the classifier, the retrieval step and the validator. We also want to understand what drives the validation score, widen the set of trusted evidence sources, and improve retrieval indexing for complex questions. Report parsing needs the most care: more report formats, better handling of poor scans, and checks that catch values like the ones in Fig. 7. Personal records could gain trend views over longer periods. The LoRA training pipeline can be explored further while keeping the independent validator, and additional validation rules can be added for conflicting evidence.

DECLARATIONS

Ethical approval and consent: The study did not involve recruiting participants or any intervention with people. Reports uploaded to the application are stored locally on the machine

that runs it. Institutional ethical requirements and data-use permissions should be confirmed by the authors' institution before any clinical deployment or external publication.

Data and code availability: The project source code and trained model files are kept as project materials and are not publicly deposited at present. They may be made available by the corresponding author on reasonable request, subject to institutional permission.

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Conflict of interest: The authors declare that they have no conflict of interest related to this study.

ACKNOWLEDGEMENT

The authors thank the Principal, Dr. Mahesh Prasanna K., and the Head of the Department of Artificial Intelligence & Machine Learning, Dr. Radhika Shetty D S, for their support and encouragement. We are also grateful to the management, and to the teaching and non-teaching staff of the department, for providing the facilities and help needed to carry out this work.

REFERENCES

1. V. Suarez-Lledo and J. Alvarez-Galvez, "Prevalence of health misinformation on social media: Systematic review," J. Med. Internet Res., vol. 23, no. 1, Art. no. e17187, 2021, doi: 10.2196/17187.
2. Y. J. Li, J. J. Marga, C. M. K. Cheung, X. L. Shen, and M. K. O. Lee, "Health misinformation on social media: A systematic literature review and future research directions," AIS Trans. Hum.-Comput. Interact., vol. 14, no. 2, pp. 116–149, 2022, doi: 10.17705/1thci.00164.
3. H. R. Saeidnia, S. Jahani, N. Ghiasi, and H. Keshavarz, "Generative AI and health misinformation: Production, propagation, and mitigation—A systematic review," BMC Public Health, vol. 26, no. 1, p. 693, 2026, doi: 10.1186/s12889-025-26148-9.
4. Y. Francis, F. J. Lantry, M. Echeverry, and M. Siddiq, "Effective countermeasures to health misinformation and disinformation for global health engagement practitioners: A rapid review," Mil. Med., vol. 190, no. Suppl_2, pp. 478–482, 2025, doi: 10.1093/milmed/usaf237.
5. S. van der Linden, "Misinformation: Susceptibility, spread, and interventions to immunize the public," Nat. Med., vol. 28, pp. 460–467, 2022, doi: 10.1038/s41591-022-01713-6.
6. Q. Wei, Z. Yao, Y. Cui, B. Wei, Z. Jin, and X. Xu, "Evaluation of ChatGPT-generated medical responses: A systematic review and meta-analysis," J. Biomed. Inform., vol. 151, Art. no. 104620, 2024, doi: 10.1016/j.jbi.2024.104620.
7. T. Han et al., "Medical large language models are susceptible to targeted misinformation attacks," npj Digit. Med., vol. 7, Art. no. 288, 2024, doi: 10.1038/s41746-024-01282-7.
8. I. Skafle, A. Nordahl-Hansen, D. S. Quintana, R. Wynn, and E. Gabarron, "Misinformation about COVID-19 vaccines on social media: Rapid review," J. Med. Internet Res., vol. 24, no. 8, Art. no. e37367, 2022, doi: 10.2196/37367.
9. X. Feng, J. Luo, Y. Yang, D. El Baz, and L. Shi, "Health misinformation detection: Approaches, challenges and opportunities," INQUIRY, vol. 62, 2025, doi: 10.1177/00469580251384784.
10. V. Papanikou, P. Papadakos, T. Karamanidou, T. G. Stavropoulos, E. Pitoura, and P. Tsaparas, "Health misinformation in social networks: A survey of information technology approaches," Future Internet, vol. 17, no. 3, Art. no. 129, 2025, doi: 10.3390/fi17030129.
11. M. Sarrouti, A. Ben Abacha, Y. Mrabet, and D. Demner-Fushman, "Evidence-based fact-checking of health-related claims," in Findings Assoc. Comput. Linguist.: EMNLP 2021, 2021, pp. 3499–3512, doi: 10.18653/v1/2021.findings-emnlp.297.
12. N. Kotonya and F. Toni, "Explainable automated fact-checking for public health claims," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2020, pp. 7740–7754, doi: 10.18653/v1/2020.emnlp-main.623.
13. L. Cui, H. Seo, M. Tabar, F. Ma, S. Wang, and D. Lee, "DETERRENT: Knowledge guided graph attention network for detecting healthcare misinformation," in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2020, pp. 492–502, doi: 10.1145/3394486.3403092.
14. X. Cao, J. Li, T. Ying, and Z. Liu, "From collaborative evidence generation to trustworthy detection: A foundation model-based framework for medical misinformation verification," Inf. Process. Manag., vol. 64, no. 1, Art. no. 105028, 2027, doi: 10.1016/j.ipm.2026.105028.
15. I. M. Bukar, B. M. Abduljalil, B. S. Maina, and A. Hassan, "An evidence-grounded retrieval-augmented transformer framework for health misinformation verification," arXiv:2608.02310, 2026, doi: 10.48550/arXiv.2608.02310.
16. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.
17. G. Xiong, Q. Jin, Z. Lu, and A. Zhang, "Benchmarking retrieval-augmented generation for medicine," in Findings Assoc. Comput. Linguist.: ACL 2024, 2024, pp. 6233–6251.
18. L. M. Amugongo, P. Mascheroni, S. Brooks, S. Doering, and J. Seidel, "Retrieval augmented generation for large language models in healthcare: A systematic review," PLOS Digit. Health, 2025, doi: 10.1371/journal.pdig.0000877.
19. D. Wang and S. Zhang, "Large language models in medical and healthcare fields: Applications, advances, and challenges," Artif. Intell. Rev., vol. 57, Art. no. 299, 2024, doi: 10.1007/s10462-024-10921-0.
20. K. Mohamed Jasim, A. Malathi, S. Bhardwaj, and E. C.-X. Aw, "A systematic review of AI-based chatbot usages in healthcare services," J. Health Organ. Manag., vol. 39, no. 6, pp. 877–899, 2025, doi: 10.1108/JHOM-12-2023-0376.
21. A. Howcroft, A. Bennett-Weston, A. Khan, J. Griffiths, S. Gay, and J. Howick, "AI chatbots versus human healthcare professionals: A systematic review and meta-analysis of empathy in patient care," Br. Med. Bull., vol. 156, no. 1, pp. 1–13, 2025, doi: 10.1093/bmb/ldaf017.
22. I. D. Mienye, G. Obaido, N. Jere, et al., "A survey of explainable artificial intelligence in healthcare: Concepts, applications, and challenges," Inform. Med. Unlocked, vol. 51, Art. no. 101587, 2024, doi: 10.1016/j.imu.2024.101587.
23. Z. Sadeghi, R. Alizadehsani, M. A. Cifci, S. Kausar, R. Rehman, P. Mahanta, et al., "A review of explainable artificial intelligence in healthcare," Comput. Electr. Eng., vol. 118, Art. no. 109370, 2024, doi: 10.1016/j.compeleceng.2024.109370.
24. H. W. Loh, C. P. Ooi, S. Seoni, P. D. Barua, F. Molinari, and U. R. Acharya, "Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022)," Comput. Methods Programs Biomed., vol. 226, Art. no. 107161, 2022, doi: 10.1016/j.cmpb.2022.107161.
25. W. Wallace et al., "The diagnostic and triage accuracy of digital and online symptom checker tools: A systematic review," npj Digit. Med., vol. 5, Art. no. 118, 2022, doi: 10.1038/s41746-022-00667-w.
26. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manag., vol. 24, no. 5, pp. 513–523, 1988.

27. N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in Proc. Conf. Empirical Methods Natural Lang. Process. and Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), 2019, pp. 3982–3992.
28. J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2021.
29. R. Smith, “An overview of the Tesseract OCR engine,” in Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR), 2007, pp. 629–633.
30. A. Yang et al. (Qwen Team), “Qwen2.5 technical report,” arXiv:2412.15115, 2024.
31. E. J. Hu et al., “LoRA: Low-rank adaptation of large language models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2022.