Individual Thesis

dIAgnosis: A MULTI-MODEL LLM PLATFORM FOR CLINICAL DECISION SUPPORT

Panagiotis Menelaou

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2026

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

dIAgnosis: A Multi-Model LLM Platform for Clinical Decision Support

Panagiotis Menelaou

Supervisor

Prof. Elpida Keravnou-Papailiou

The Individual Thesis was submitted in partial fulfilment of the requirements for the Degree of Computer Science of the Department of Computer Science of the University of Cyprus

May 2026

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content.

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Professor Elpida Keravnou-Papailiou who guided, supported and advised me throughout the year. Her knowledge, experience and expertise had a significant impact on the outcome of this thesis, and I am truly grateful for her guidance.

I would also like to thank my family and friends for their continuous support, which kept me motivated and disciplined to do well on this journey.

Finally, I would like to thank all the medical students who participated in my survey, whose contributions played a crucial role in the assessment of this work.

Abstract

Large Language Models (LLMs) are increasingly demonstrating significant potential in healthcare, ranging from medical documentation to real-time diagnostic reasoning support. However, their clinical integration remains limited by serious liabilities, including hallucinations, probabilistic inconsistency, and data privacy constraints. In high-stakes environments where diagnostic errors can cause significant patient harm, relying on an unverified, single-model output introduces significant risk. This thesis presents dIAgnosis, a fully functioning web-based platform that coordinates an asynchronous Mixture-of-Agents (MoA) pipeline to deliver a structured, multi-model second opinion for clinical decision support. The system handles structured clinical inputs (patient history, vital signs, telemetry, and laboratory findings) and routes them in parallel to an ensemble of independent worker LLMs across disparate providers (OpenAI, Anthropic, Google Gemini, and Groq). A dedicated aggregator model (GPT-5.1) synthesizes these individual worker drafts into a unified, structured clinical report conforming to a strict JSON schema. The dIAgnosis platform features longitudinal patient monitoring via historical consultation tracking, bedside clinical calculators, an automated external evidence retrieval system grounded in the NCBI PubMed database, and an automated multi-stage panel debate workflow. Furthermore, an internal administrative subsystem evaluates report validity using biomedical text embeddings (PubMedBERT) to perform structural quality assurance, panel consensus analysis, and semantic divergence profiling. The architecture was evaluated on a clinician-informed rolling benchmark of ten distinct medical cases. The model panel achieved a 100% Top-3 hit rate and a 0% unsafe omission rate, indicating excellent diagnostic coverage, although it did not consistently surpass the single strongest frontier model in absolute Top-1 diagnostic accuracy. Preliminary usability testing via a medical student survey revealed a strong consensus that multi-model architecture, explicit alignment metrics, and minority opinion tracking substantially elevate clinician trust and visual safety calibration compared to traditional standalone systems.

Περίληψη

Τα Μεγάλα Γλωσσικά Μοντέλα παρουσιάζουν αυξανόμενες προοπτικές στον τομέα της υγείας, καλύπτοντας ένα ευρύ φάσμα εφαρμογών από την κλινική τεκμηρίωση έως την υποστήριξη κλινικών αποφάσεων. Ωστόσο, η ενσωμάτωσή τους στην κλινική πράξη παρεμποδίζεται σημαντικά από κρίσιμους περιορισμούς, όπως οι «παραισθήσεις» (hallucinations), η πιθανολογική ασυνέπεια και οι περιορισμοί στην προστασία των προσωπικών δεδομένων. Σε περιβάλλοντα υψηλού κινδύνου, όπου τα διαγνωστικά σφάλματα μπορούν να προκαλέσουν σοβαρή βλάβη στους ασθενείς, η αποκλειστική βασιμότητα στα αποτελέσματα ενός και μόνο μη επαληθευμένου μοντέλου εγκυμονεί σοβαρούς κινδύνους. Η παρούσα διατριβή παρουσιάζει το dIAgnosis, μια πλήρως λειτουργική διαδικτυακή πλατφόρμα η οποία συντονίζει μια ασύγχρονη αρχιτεκτονική Mixture-of-Agents με σκοπό την παροχή μιας δομημένης, πολυ-μοντελικής δεύτερης γνώμης για την υποστήριξη κλινικών αποφάσεων. Το σύστημα δέχεται δομημένα κλινικά δεδομένα και τα προωθεί παράλληλα σε μια ομάδα ανεξάρτητων μοντέλων-εργατών (worker models) από διαφορετικούς παρόχους (OpenAI, Anthropic, Google Gemini και Groq). Στη συνέχεια, ένα εξειδικευμένο μοντέλο-συνθέτης (aggregator model, GPT-5.1) αναλύει και συγχωνεύει τις επιμέρους διαγνώσεις σε μια ενιαία, δομημένη κλινική αναφορά που συμμορφώνεται με ένα αυστηρό σχήμα μορφοποίησης JSON. Η πλατφόρμα dIAgnosis υποστηρίζει διαχρονική παρακολούθηση ασθενών μέσω καταγραφής του ιστορικού των επισκέψεων, ενσωματωμένους κλινικούς υπολογιστές, ένα αυτοματοποιημένο σύστημα ανάκτησης εξωτερικών επιστημονικών τεκμηρίων βασισμένο στη βάση δεδομένων NCBI PubMed, καθώς και μια προαιρετική διαδικασία πολυ-μοντελικής αντιπαράθεσης και διαλόγου (panel debate). Επιπλέον, ένα εσωτερικό υποσύστημα διαχείρισης αξιολογεί την εγκυρότητα των αναφορών χρησιμοποιώντας βιοϊατρικές ενσωματώσεις κειμένου (PubMedBERT), εκτελώντας δομικό ποιοτικό έλεγχο, ανάλυση συναίνεσης και προσδιορισμό σημασιολογικών αποκλίσεων μεταξύ των μοντέλων. Η αρχιτεκτονική του συστήματος αξιολογήθηκε μέσω ενός δυναμικού συνόλου δοκιμών (benchmark) που περιλάμβανε 10 διακριτά ιατρικά περιστατικά βασισμένα σε πραγματικά κλινικά δεδομένα. Η ομάδα των μοντέλων πέτυχε ποσοστό εντοπισμού 100% στις τρεις επικρατέστερες διαγνώσεις (Top-3 hit rate) και 0% ποσοστό επικίνδυνων παραλείψεων, αποδεικνύοντας εξαιρετική διαγνωστική κάλυψη, αν και δεν ξεπέρασε σταθερά το ισχυρότερο μεμονωμένο μοντέλο σε απόλυτη ακρίβεια πρώτης

επιλογής (Top-1 accuracy). Προκαταρκτικές δοκιμές ευχρηστίας μέσω έρευνας σε φοιτητές ιατρικής έδειξαν ότι η πολυ-μοντελική αρχιτεκτονική, οι σαφείς δείκτες ευθυγράμμισης και η καταγραφή των απόψεων της μειοψηφίας ενισχύουν σημαντικά την εμπιστοσύνη των κλινικών γιατρών και τη βαθμονόμηση των παραμέτρων ασφαλείας σε σύγκριση με τα παραδοσιακά μεμονωμένα συστήματα.

Contents

Chapter 1 – Introduction..... 1

1.1 Overview of Large Language Models in Healthcare..... 1

1.2 The Multi Model Architecture..... 2

1.3 Objectives and Contributions..... 4

1.4 Thesis Structure..... 5

Chapter 2 - Related Work..... 7

2.1 Large Language Models in Clinical Practice..... 7

2.2 Multi-Model Approaches and Mixture of Agents..... 11

2.3 Evolution of Clinical Decision Support Systems..... 11

2.4 Existing Clinical Decision Support Systems..... 13

2.5 Positioning of this work..... 16

Chapter 3 - Requirements and System Specifications..... 19

3.1 Introduction..... 19

3.2 Requirements Analysis..... 19

3.2.1 Functional Requirements..... 20

3.2.2 Non-Functional Requirements..... 21

3.3 Specifications..... 24

3.3.1 Target Users..... 24

3.3.2 System Constraints and Assumptions..... 25

3.3.3 Use Case Diagram..... 26

Chapter 4 - Technologies and Tools..... 28

4.1 Introduction..... 28

4.2 Development Environment..... 28

4.2.1 Visual Studio Code..... 28

4.2.2 Python 29
4.2.3 Virtual Environment and Dependency Management 29
4.2.4 Git/Version Control 29
4.3 Frontend Technologies 30
4.3.1 Streamlit 30
4.3.2 Pandas 30
4.3.3 Plotly 30
4.3.4 FPDF2 31
4.4 Backend Technologies 31
4.4.1 FastAPI 31
4.4.2 Uvicorn 31
4.4.3 Pydantic 32
4.4.4 HTTPX 32
4.5 Database Technologies 32
4.5.1 SQLite 32
4.5.2 SQLAlchemy 32
4.5.3 aiosqlite 33
4.6 LLM APIs 33
4.6.1 OpenAI API 33
4.6.2 Groq API 33
4.6.3 Anthropic API 34
4.6.4 Gemini API 34
4.7 Embeddings and Evidence Retrieval 34
4.7.1 Sentence-BERT / SentenceTransformers 34
4.7.2 PyTorch 35
4.7.3 NCBI E-utilities / PubMed 35
4.8 Configuration and Security Tools 35

4.8.1 python-dotenv 35
4.8.2 Streamlit Secrets 36
4.8.3 FastAPI Middleware 36
Chapter 5 - System Design and Implementation 37
5.1 Introduction 37
5.2 System Architecture Overview 38
5.3 Backend Design and Implementation 40
5.4 Multi-Model Reasoning Pipeline 40
5.5 Aggregation and Report Generation 42
5.6 Longitudinal Follow-Up Design 43
5.7 Data Storage Design 43
5.8 Evaluation and Benchmarking Subsystem 44
5.9 Evidence Retrieval and Report Support 45
5.10 Frontend Design and User Workflows 46
5.11 Safety and Security Considerations 47
5.12 Code Structure 47
5.13 System Data Flow 48
5.14 Interface Screenshots 49
Chapter 6 - Summary 73
6.1 Conclusions 73
6.2 Limitations 75
6.3 Medical Student Survey 76
6.4 Future Work 79
6.5 Closing Remarks 80
Bibliography 82
Appendix A - Medical Student Survey A-1
A.1 Medical Student Survey Questions A-1

A.2 Medical Student Survey Answers A-11

Appendix B - User Manual / Testing Guide B-1

B.1 Requirements B-1

B.2 Configuration B-2

B.3 Installing Dependencies B-2

B.4 Starting the Backend B-2

B.5 Starting the Frontend B-3

B.6 Testing a New Consultation B-3

B.7 Testing Report Features B-4

B.8 Testing Patient History B-4

B.9 Testing Admin Analytics B-5

B.10 Testing Clinical Benchmark Evaluation B-6

B.11 Notes and Limitations B-6

Chapter 1 – Introduction

1.1 Overview of Large Language Models in Healthcare 1

1.2 The Multi Model Architecture 2

1.3 Objectives and Contributions 4

1.4 Thesis Structure 5

1.1 Overview of Large Language Models in Healthcare

A large language model (LLM) is a type of artificial intelligence (AI) trained on vast datasets using self-supervised learning, enabling it to interpret and generate human-like language [1]. The transformer architecture is the most widely adopted approach, where the model learns to predict the next word in a sequence, allowing it to produce coherent and contextually relevant responses [1]. LLMs are becoming a part of everyday life. Their capabilities over the recent years have been significantly expanding across all fields, with new models surpassing their predecessors as leading technology companies compete for the best outcome by pushing their boundaries to the limits [1]. Unlike traditional software that produces deterministic outputs, LLMs are probabilistic systems that generate responses based on learned statistical patterns rather than verified facts. This means that even the most advanced models can produce outputs that are confident in tone, yet factually incorrect, a phenomenon commonly referred to as hallucination [2]. This limitation becomes especially serious in fields where decisions directly affect human life, most notably healthcare, where an incorrect or misleading output is not merely an

inconvenience but a potential source of serious harm. Hallucinating is just one of several limitations that make the adoption of LLMs in high-stakes fields particularly challenging.

Nonetheless, healthcare is benefiting with LLMs, from clinical documentation and automation to clinical decision making in real-time, acting as a vital tool for doctors and medical experts [1]. Apart from reduced reliability, LLMs in the medical domain carry other serious risks: inconsistent outputs, data privacy concerns or even potential biases which make their uncritical usage a liability for most [1]. Addressing these limitations is therefore essential to move toward a safer and more structured incorporation of artificial intelligence (AI) into clinical decisions, one that clearly supports rather than replaces the critical judgement of experienced clinicians.

One of the main challenges facing LLMs in healthcare is the limited availability of training data [3]. Medical data is highly sensitive and subject to strict privacy regulations, which restricts how much of it can be used to train and improve these models. Without access to diverse and representative clinical data, LLMs struggle to perform reliably across different patient populations and medical contexts. A gradual and carefully managed introduction of AI into healthcare settings, starting with lower-risk applications and expanding over time as trust and safety are established, offers a more responsible path toward realizing the full potential of these models in medicine [4].

1.2 The Multi Model Architecture

A single-model architecture relies on one AI model to handle all aspects of a task independently, without input from other models. A multi-model architecture, by contrast, involves multiple models working together, either collaborating by sharing outputs, or operating in defined roles such as proposers and aggregators, to produce a more robust and reliable final result [5].

By nature, LLMs are not designed to be optimal in every aspect. Different models excel in different scenarios, and no single model currently outperforms all others across every task [5]. Much like humans, who each bring their own strengths and weaknesses, real impact and progress comes when individual capabilities are combined toward a common goal. The same principle applies to LLMs. By combining their independent outputs, each backed by its own distinct training data and architecture, the final result becomes significantly more reliable and complete than what any single model would produce alone [6]. This concept is known as multi-LLM collaboration. The most practical form for real-world applications is text-level collaboration, where each model's output becomes the input to another, allowing models to build on, verify, and refine each other's reasoning possibly reducing error and improving trust [5].

However, simply combining random models is not enough. Research has shown that using multiple copies of the same model, or models with very similar training, creates an echo chamber effect, where models reinforce each other's errors rather than correcting them, and debate converges on a single unchanging answer regardless of whether it is correct [7]. True diversity requires models that differ not only in their output but in their underlying training data, architecture, and provider.

That said, applying multi-model collaboration within the medical domain introduces additional challenges that go beyond simply combining outputs. Compatibility between models, the interpretability of their responses, and strict data privacy requirements, all impose significant constraints that general-purpose ensemble methods were not designed to handle [6]. Most existing approaches to LLM ensembles focus on open-ended tasks and do not account for the specific demands of clinical applications, such as structured outputs, domain-specific knowledge, and the need for explainable reasoning that clinicians can trust and act on.

Despite these challenges, the evidence points clearly in one direction. Ensemble methods consistently outperform individual models on medical benchmarks, suggesting that the

path toward more reliable clinical AI lies not in perfecting a single model only, but in combining multiple models in a structured and carefully designed way [6].

1.3 Objectives and Contributions

The main goal of this diploma project was to explore whether multiple LLMs working together could provide a more structured and trustworthy form of clinical decision support than a single model alone. Even though evidence exists that multi-model approaches outperform individual models, this idea has not been widely applied in the medical field, which is arguably where it is needed most. Critical reasoning, debate between different perspectives, and reaching a considered conclusion are exactly the qualities that clinical decision-making demands.

dIAgnosis is a fully working multi-model clinical decision support platform that gives medical professionals access to a structured second opinion on any clinical case. The system sends each case to multiple worker models from different providers, collects their independent assessments, and passes them to an aggregator model that produces a finalized clinical report. The platform also supports longitudinal patient monitoring through follow-up visits, maintaining a full consultation history for each patient.

The main contributions of this work are:

- A fully working multi-model clinical decision support platform
- A standardized worker prompt pipeline using models from multiple providers
- An aggregator model producing structured JSON clinical reports

- A longitudinal follow-up system with patient history tracking

- Optional debate round for second-pass review between model outputs

- Worker draft preservation allowing traceability of each model's individual reasoning

- A set of bedside clinical calculators and PubMed evidence lookup

- Internal validity metrics covering schema compliance and safety checks on every consultation

- PubMedBERT-based semantic consistency analysis for internal research use

- A clinical benchmark system with 10 clinician-informed cases comparing panel vs single model performance

- A rolling benchmark allowing cases to be run individually and results aggregated over time

1.4 Thesis Structure

This thesis is organised into six chapters:

- Chapter 1 introduces the problem, the motivation behind the work, and the contributions of the platform.

- Chapter 2 covers related work in the areas of large language models in medicine, ensemble approaches, hallucination and safety, and existing clinical decision support systems.

- Chapter 3 presents the requirements analysis and system specifications, including functional and non-functional requirements, target users, system assumptions, and a use case diagram.

- Chapter 4 presents the technologies and tools used to develop the platform, including the frontend, backend, database, LLM APIs, evidence retrieval, and configuration/security tools.

- Chapter 5 describes the system design and implementation, including the architecture, backend design, multi-model reasoning pipeline, aggregation process, database design, evaluation subsystem, frontend workflows, safety considerations, code structure, data flow, and interface screenshots.

- Chapter 6 concludes the thesis with a conclusion, limitations of the current system, a medical student survey and directions for future work.

Chapter 2 - Related Work

2.1 Large Language Models in Clinical Practice 7

2.2 Multi-Model Approaches and Mixture of Agents 11

2.3 Evolution of Clinical Decision Support Systems 11

2.4 Existing Clinical Decision Support Systems 13

2.5 Positioning of This Work 16

2.1 Large Language Models in Clinical Practice

Building on the general overview presented in Chapter 1, this section examines the specific ways LLMs have been applied in clinical practice and what the given research has revealed so far, along with their challenges and limitations.

Because of their ability to process and reason over large amounts of clinical information, LLMs can personalize responses based on individual patient data, which can prove useful for clinicians as a support tool [8]. Remote diagnosis is another area that can benefit greatly from LLMs, giving both patients and clinicians access to structured reasoning support regardless of location. Clinical report generation saves clinicians significant time and effort, allowing them to focus on more demanding tasks rather than routine documentation. Medical decisions require synthesis from various sources in order to reach a good diagnosis and prognosis, which is exactly what LLMs are capable of doing [8].

Systems like DDx PaLM-2 demonstrate this in practice, enabling interactive diagnosis assistance where clinicians and the model work together iteratively to arrive at a conclusion [8]. Similarly, NYUTron was trained on hospital data and showed strong performance in predicting in-patient mortality and readmission risk [8]. These systems show that when LLMs are fine-tuned on domain-specific medical data, their clinical usefulness increases significantly. General purpose models such as GPT-4 and GPT-3.5 have shown remarkable performance on medical question answering tasks, outperforming even expert clinicians on benchmarks like the USMLE medical licensing exam [8]. This is a clear indication that LLMs should not be underestimated in clinical settings. Nevertheless, when it comes to more complex open-ended tasks, general LLMs tend to underperform compared to models that have been fine-tuned specifically on medical datasets, which is expected given the specialized nature of clinical work [8].

In one study, ChatGPT achieved 60.3% accuracy in differential diagnosis based on patient history alone, rising to 76.9% when additional test results were provided [9]. While these numbers leave room for improvement, they represent a meaningful baseline for a general-purpose model operating in a specialised domain. Interestingly, LLMs have shown particular strength in rare and complex cases, where they have been shown to surpass average population consensus in reaching a correct diagnosis [9]. In patient triage, performance was comparable to that of resident physicians, suggesting real potential for LLMs in time-pressured clinical workflows [9].

A Harvard study published in Science found that AI identified the correct or near-correct diagnosis in 67% of emergency triage cases, outperforming human doctors who achieved 50-55% accuracy under the same conditions [10]. In treatment planning, the gap was even more pronounced. AI scored 89% compared to 34% for physicians using conventional resources like search engines [10]. These results suggest that LLMs are moving beyond passing medical exams and are beginning to demonstrate genuine clinical value in real-world settings [8]. This shift is already reflected in practice. Nearly one in five US physicians now use AI to assist with diagnosis [10]. The study tested only text-based patient data, leaving out critical signals like patient distress and physical appearance that

clinicians rely on. As of the EU, a WHO/Europe report found that 74% of EU countries are already using AI-assisted diagnostics and 63% use AI chatbots to support patient engagement [11]. All 27 EU member states recognize improved patient care as a primary driver of AI development, and nearly half have already created dedicated professional roles for AI and data science in health [11]. The focus is now shifting toward workforce training, ensuring clinicians can critically engage with these tools rather than simply adopting them uncritically [11].

Despite the promising results discussed above, several limitations and challenges prevent LLMs from being fully integrated into clinical practice. While research has shown that LLMs can perform impressively on diagnostic tasks, medicine is a domain where imperfect outputs carry serious consequences.

A study found that only 33% of GPT-4 generated discharge summaries were completely error-free, and the model only caught 80 out of 280 clinically relevant drug interactions that established tools would normally flag [12]. When tested on real patient cases, LLMs fell behind physicians by 16 to 25 percentage points, especially when clinical information was spread out over time [12]. Bias is a problem too. Over 90% of medical LLMs show measurable gender and racial bias, which could mean different patients receive different quality recommendations for no clinical reason [12]. Legal accountability is also unresolved. Nobody has figured out who is legally responsible when an LLM gets something wrong, if it is the clinician, the hospital, or the developers [12]. What is perhaps most concerning is that injecting a tiny amount of misinformation into training data, as little as 0.001% was enough to increase harmful outputs by 4.8%, and standard benchmarks failed to catch it [12].

Table 2.1.1 summarizes the main limitations identified across the literature [8]. These issues do not make LLMs useless, but they do make it clear that relying on a single model output without any checks is not a safe approach in medicine.

Limitation	Description
Hallucination	Models generate confident but factually incorrect outputs, which in a clinical setting can lead to misdiagnosis or inappropriate treatment recommendations.
Training Data Cutoff	LLMs are trained on data up to a fixed date and cannot account for recent medical developments, new treatments, or updated clinical guidelines.
Domain Data Limitations	Medical datasets are small, private, and difficult to access due to privacy regulations, limiting how well models can be trained on clinical data.
Black Box Problem	Clinicians cannot see how the model reached its conclusion, making it difficult to verify, trust, or challenge the output.
Behaviour Alignment	General purpose LLMs do not naturally respond with clinical precision and may produce answers that are too vague or conversational for medical use.
Privacy and Security	Patient data entered into LLM systems raises serious privacy concerns, and models may inadvertently leak sensitive information.
Regulatory and Accountability Gaps	No established framework exists for determining responsibility when an LLM contributes to a wrong clinical decision.

Table 2.1.1 — Main limitations of LLMs in clinical practice

Sources: [2], [3], [8]

2.2 Multi-Model Approaches and Mixture of Agents

Single model approaches have clear limitations as discussed in the previous section. Combining multiple models produces more consistent, structured, and less error-prone outputs than relying on any individual model alone [13]. One of the most relevant frameworks in this space is the Mixture-of-Agents (MoA) approach [13]. A multi-agent framework is a structured approach in which multiple AI agents, each with a defined role, collaborate within a shared environment to complete a task [13]. The framework coordinates how agents interact, share information, and contribute to a final output, providing a more organized and transparent form of AI reasoning [13]. The MoA framework works by dividing models into the roles of proposers and an aggregator. The proposers independently generate responses to the same input, while the aggregator synthesizes all proposer outputs into a single high-quality response. The aggregator does not simply pick the best answer but critically evaluates all inputs and produces a refined, comprehensive output. Wang et al. demonstrated that this approach outperforms GPT-4o on multiple benchmarks using only open-source models, while being twice as cost effective as GPT-4 Turbo [13].

In the medical domain, this kind of structured multi-model collaboration has been largely absent [6]. Most clinical AI systems rely on a single model output with no cross-checking or synthesis mechanism, which is arguably the domain where such checks are needed most [6]. dIAgnosis directly addresses this gap by implementing a Mixture of Agents architecture. Multiple worker models act as proposers, each independently assessing the same clinical case, and a dedicated aggregator model synthesizes their outputs into a single consolidated clinical report.

2.3 Evolution of Clinical Decision Support Systems

A clinical decision support system (CDSS) is a health information technology tool designed to analyze patient data and history in order to assist clinicians in making more informed medical decisions [14]. These systems aim to improve patient outcomes by

delivering relevant knowledge and recommendations at the right time during the care process [14].

CDSSs have a longer history than many realize, evolving through four distinct phases over several decades [14]. In phase 1, the earliest systems dating back to 1959, used analogue methods. One example is sorting cards representing diagnoses against patient symptoms to suggest a differential. While innovative for their time, these standalone systems were inefficient, required manual data entry, and were limited to specific clinical domains. Phase 2 brought integration with early centralized clinical data repositories and nascent hospital information systems, beginning with the HELP system in Utah in 1967, the first to embed decision support directly into clinical workflows. This solved the problem of manual data entry and allowed systems to be proactive but created a new limitation: decision support content could not easily be shared or reused across institutions [14]. Phase 3 addressed this by introducing standards for sharing clinical knowledge such as the Arden Syntax, developed in 1989. These standards allowed decision support content to be encoded, stored and shared across institutions, overcoming the isolation problem of integrated systems. Still, phase 3 came with its own drawbacks. Too many competing standards existed, and any encoding standard inherently limited what could be expressed within it to what the original authors had anticipated. If a clinical need fell outside the scope of the standard, there was no way to represent it [14]. Phase 4 went further by separating decision support from clinical systems entirely through APIs and virtual medical record (VMR) interfaces. This modular approach allowed decision support to be consumed by any compatible system, regardless of the underlying platform [14].

Despite decades of development across all four phases, traditional CDSS remained constrained by rigid rule-based logic, narrow clinical scope, and limited ability to reason over unstructured clinical language. The emergence of LLMs represents a fundamental shift, replacing fixed rules with probabilistic reasoning over natural language and enabling a new generation of decision support tools capable of handling the complexity and variability of real clinical cases.

Building on the four architectural phases described above [14], the emergence of LLMs could be considered a fifth phase in the evolution of clinical decision support [9]. Unlike previous phases which required structured inputs and predefined rules, LLMs can reason over free text, handle ambiguity, and generalize across clinical domains in ways that earlier systems could not.

2.4 Existing Clinical Decision Support Systems

In current clinical settings, a wide range of CDSS tools exist. They vary from traditional EHR-integrated systems to newer AI-based platforms. In the US clinical decision support vendor market, Epic Systems Corporation leads at 27.1% market share across 1,886 hospital installations, followed by Oracle Cerner, now renamed Oracle Health, at 18.5% with 1,291 installations [15]. Both platforms have evolved significantly, embedding AI features such as predictive analytics, voice-driven documentation, and point-of-care decision support into their EHR systems [16], [17]. However, these systems remain primarily embedded workflow platforms. Although they increasingly include AI-enabled features, their main role is still to support documentation, prediction, order entry, and EHR-based workflow.

Beyond traditional EHR-integrated systems, a new generation of AI-powered standalone tools has emerged to address the limitations of rule-based decision support. These tools are widely used by clinicians to assist with differential diagnosis, clinical planning, and evidence retrieval. The most prominent among them are Isabel DDx, UpToDate Expert AI, Glass Health, and OpenEvidence, each offering distinct capabilities. A detailed comparison of these tools against dIAgnosis is presented in Table 2.5.1 in Section 2.5. A brief description of each tool follows:

- **Isabel DDx**

  Isabel DDx is one of the most established AI-powered differential diagnosis tools, with over 22 years of clinical validation and continuous refinement [18]. Clinicians

enter patient demographics, clinical features, lab results, and imaging findings, and the system generates a ranked list of diagnoses filtered by age, sex, and region. Results are color-coded by likelihood and include red-flag conditions that must not be missed, as well as medications that may have caused the presenting symptoms [18]. Despite its longevity and breadth, Isabel relies on a curated diagnostic knowledge base rather than LLM-based free-text reasoning.

- **UpToDate Expert AI**

UpToDate Expert AI is a conversational clinical assistant backed by evidence-based content curated from over 7,600 clinicians [19]. It allows clinicians to ask clinical questions and receive traceable answers with inline citations, follow-up question support, drug information, and voice input. Unlike Isabel DDx, it is not a differential diagnosis generator. It is primarily an evidence retrieval tool, and it remains grounded in a single curated knowledge base.

- **Glass Health**

Glass Health is a subscription-based AI platform designed for physicians to formulate differential diagnoses and clinical plans based on patient symptoms and history [20]. It provides evidence-based treatment options, assists with documentation through templates, and includes a community library where clinicians can share and access medical knowledge. Publicly available descriptions of Glass Health focus on single-assistant diagnosis, planning, and documentation support.

- **OpenEvidence**

OpenEvidence is an AI-powered evidence retrieval platform that draws from peer-reviewed sources to provide clinicians with evidence-based recommendations at the point of care [21]. According to company-reported figures, by mid-2025 it was used daily by over 40% of US physicians across more than 10,000 hospitals [21]. OpenEvidence also launched DeepConsult, an AI agent built specifically for physicians. This feature is positioned as an agentic research tool that can analyze and

cross-reference large numbers of peer-reviewed studies to generate structured research summaries [21].

However, all the deployed systems reviewed above are primarily single-system or single-assistant tools. In contrast, recent research has begun exploring multi-agent architectures, where multiple LLM agents contribute independent or role-specific reasoning before a final answer is produced [22]. These studies demonstrate the potential of multi-agent LLM frameworks in clinical diagnosis.

A study developing a Multi-Agent Conversation (MAC) framework inspired by real clinical multidisciplinary team discussions found that MAC significantly outperformed single models including GPT-3.5 and GPT-4 on rare disease diagnosis [23]. Optimal performance was achieved using four doctor agents and one supervisor agent, a structure that closely mirrors how clinical teams reason collaboratively in practice. MAC also outperformed other established methods including Chain of Thought and Self-Consistency approaches [23]. These findings suggest that multi-agent architectures represent a meaningful step forward in clinical AI, toward a more collaborative form of reasoning.

This is beginning to translate into real-world practice as well. BJC Healthcare is currently piloting LLM-enhanced multi-agent systems for end-of-life care planning, demonstrating that health systems are starting to move beyond single-model chatbot tools toward agentic clinical workflows [22].

Despite the variety of tools reviewed and the growing body of research into multi-agent approaches, existing multi-model frameworks have been evaluated primarily on benchmarks and remain research prototypes. One example is MDAgents, where multiple LLMs collaborate to assess a diagnosis as a group of agents, resembling real-world medical decision making (MDM) [24]. Deployed clinical tools therefore remain centered on single-assistant interaction, while the integration of multi-model consensus into a general-purpose clinician-facing platform remains largely unexplored.

2.5 Positioning of this work

This section compares dIAgnosis against the AI-powered clinical decision support tools reviewed in Section 2.4. While each tool offers valuable capabilities, they share several common limitations. Most rely on fixed, domain-specific knowledge bases or curated datasets rather than general-purpose LLM reasoning. Several focus primarily on evidence retrieval rather than active differential diagnosis support. Critically, none employ a multi-model consensus mechanism where independent models assess the same case, and an aggregator synthesizes their outputs. Table 2.5.1 compares dIAgnosis against selected AI-powered clinical decision support tools based on publicly available product documentation. The comparison focuses on functional presence rather than performance, regulatory status, or clinical validation.

✓ indicates that the feature is explicitly described in public documentation.

~ indicates partial or adjacent functionality.

✗ indicates that the feature could not be verified from public documentation.

Feature	**Isabel DDx**	**UpToDate Expert AI**	**Glass Health**	**OpenEvidence**	**dIAgnosis**
Ranked Differentials	✓	~	✓	✗	✓
Red-Flags and Warnings	✓	~	✓	~	✓
Evidence Lookup	✓	✓	✓	✓	✓
Drug Interaction Checking	~	~	~	~	~

Inline Citations	~	✓	✓	✓	✗
Voice Input	✗	✓	✓	~	✗
EHR/EMR Integration	✓	~	✓	~	✗
Multilingual Support	✓	~	✓	✗	✗
Structured Clinical Report	✗	~	✓	✓	✓
Patient Handout	✗	✗	✓	✓	✓
Multi-Model Consensus	✗	✗	✗	✗	✓
Patient Folder with History	✗	✗	~	~	✓

Table 2.5.1 – Comparison of CDSSs and dIAgnosis

Sources: official product documentation and publicly available feature descriptions for each tool [18], [19], [20], [21].

As shown in Table 2.5.1, the tools reviewed do not all serve the same purpose. Some are primarily differential diagnosis generators, while others function mainly as evidence retrieval assistants. Positioned among them, dIAgnosis covers a broad range of clinical features despite being a first version developed by a single person. While the concept of synergistic multi-modelling has legacy roots in second-generation deep expert systems, its modern instantiation using generative Large Language Models as autonomous agents represents a new frontier, with commercial platforms only beginning to experiment with agentic medical workflows as of 2026 [22]. It is important to note that dIAgnosis has not undergone professional clinical testing or regulatory validation and cannot be compared

to these tools on performance or safety grounds. The comparison here is limited to feature presence based on publicly available documentation.

Chapter 3 - Requirements and System Specifications

3.1 Introduction 19

3.2 Requirements Analysis 19

3.3 Specifications 24

3.1 Introduction

This chapter presents the requirements analysis and system specifications of the dIAgnosis platform. It defines the target users of the system, outlines the functional and non-functional requirements that guided its development, and presents a use case diagram that captures the main interactions between users and the system. Understanding these requirements is essential for contextualizing the design and implementation decisions described in the following chapters.

3.2 Requirements Analysis

This section outlines the functional and non-functional requirements that guided the development of dIAgnosis. Functional requirements define what the system must do,

while non-functional requirements define how the system must behave in terms of performance, reliability, security, and usability.

3.2.1 Functional Requirements

Functional requirements describe what the system must do. They define the main features, actions, and behaviors the application must support.

The system shall:

- Accept structured clinical case input including patient demographics, vital signs, symptoms, medical history, medications, allergies, and test results.
- Send each clinical case to multiple worker LLM models from different providers running in parallel.
- Collect and store each worker model's independent output as a separate draft.
- Pass all worker drafts to a dedicated aggregator model for synthesis.
- Generate a structured JSON clinical report containing ranked differentials, assessment, investigations, red flags, safety warnings, certainty level, and a patient handout.
- Display the final report across clinician-facing tabs including the clinical report, risk tools, evidence lookup, and optional debate review, while providing a separate analytics dashboard for internal evaluation.

- Support longitudinal patient monitoring through follow-up visits, maintaining a full consultation history for each patient.

- Support PubMed evidence lookup based on the leading differential diagnosis, suggested investigations, and safety signals.

- Provide bedside clinical calculators as decision support tools.

- Support importing and running clinical benchmark cases to compare panel vs single model performance.

- Store all consultations, worker drafts, and evaluation metrics in a persistent database.

- Allow clinicians to manage patient folders, rename visits, delete consultations, and create follow-ups from the sidebar.

- Perform internal structural quality checks on every generated report.

- Perform semantic panel consistency analysis using PubMedBERT embeddings to detect divergent worker outputs.

3.2.2 Non-Functional Requirements

The non-functional requirements describe the quality constraints and operational properties expected from the dIAgnosis prototype.

- **Performance**

  - The system shall run worker model calls concurrently so that the total consultation time is determined mainly by the slowest model response rather than the sum of all model response times.
  - The system shall provide visible progress feedback while the panel is running, including a loading state during model querying and report generation.

- **Reliability**

  - The system shall tolerate individual worker model failures. If one model fails, times out, or returns invalid JSON, the failure shall be stored and displayed without preventing successful worker outputs from being reviewed.
  - The system shall normalize and validate generated report fields before display so that missing or malformed sections do not crash the frontend.

- **Security**

  - The system shall keep all LLM provider API keys outside the frontend. API credentials shall be loaded from environment variables or a local environment configuration file and used only by backend/server-side code.
  - The frontend shall not display API keys, raw environment configuration, or sensitive provider credentials.

- **Usability**

  - The interface shall guide the user through the consultation workflow using clearly separated sections for patient identification, vitals,

history, symptoms, presentation, report review, evidence, debate, and analytics.

- The final report shall be presented in a structured tabbed layout so that urgent concerns, summary information, diagnoses, investigations, and safety warnings can be located without reading raw model output.

- **Consistency**

  - The aggregator output shall follow a fixed JSON schema containing the same required report fields for every consultation.
  - The frontend shall render reports using the normalized schema so that reports generated from different model combinations maintain a consistent layout and terminology.

- **Maintainability**

  - The implementation shall separate major responsibilities into distinct modules, including frontend interface, backend API, database models, CRUD operations, LLM orchestration, report utilities, evidence retrieval, evaluation, and configuration.
  - Model selection and provider configuration shall be isolated from the core workflow so that model changes do not require rewriting the consultation or report-generation logic.

- **Scalability**

  - The system shall allow the model panel size and model providers to be changed through configuration.
  - Adding a new worker model shall require updating the model configuration and provider-call logic only, without changing the database schema or frontend report layout.

- **Safety**

- Every generated report shall include a visible clinical safety disclaimer.
- The system shall present outputs as decision support only and shall include red flags, warnings, and escalation information where relevant, rather than presenting the generated diagnosis as definitive.

- **Portability**

  - The system shall run as a Python-based prototype using installable dependencies and local configuration.
  - The prototype shall support local SQLite persistence so it can be run without requiring an external database server.

- **Transparency**

  - The system shall preserve worker drafts separately from the aggregated report so that the final consensus can be traced back to individual model outputs.
  - The system shall expose internal support and agreement metrics so that users can inspect whether the final report was strongly or weakly supported by the worker panel.

3.3 Specifications

This section presents the system specifications of dIAgnosis, covering the target users, system constraints, assumptions, and a use case diagram.

3.3.1 Target Users

The primary target users of dIAgnosis are medical professionals, including doctors, clinicians, and physicians, who may require structured decision support during clinical consultations. The platform is intended to function as a second opinion and reasoning-support tool rather than as a replacement for clinical judgement. A secondary user group consists of medical students and trainees, who may use the platform in supervised educational contexts to submit clinical cases, review multi-model outputs, and compare the generated reasoning with their own clinical reasoning. Researchers may also use the platform to study multi-model clinical reasoning behavior and to run clinical benchmark evaluations comparing panel-based outputs with single-model performance.

3.3.2 System Constraints and Assumptions

The following constraints and assumptions apply to the current version of dIAgnosis:

- The system always requires a stable internet connection, as all LLM inference is performed through external provider APIs rather than locally hosted models. Valid API keys must be configured for the required provider services. In the current implementation, OpenAI is required for final aggregation, while additional provider keys enable the multi-model worker panel.

- The system is dependent on third-party LLM providers. Any changes to provider pricing, model availability, rate limits, or API structure may affect system behavior without modification to the platform itself. Model outputs are probabilistic and non-deterministic, meaning that identical inputs may produce different outputs across runs.

- dIAgnosis has not undergone clinical validation or regulatory review of any kind. It is a research prototype and should not be used for actual clinical decision making without appropriate professional oversight. The system is explicitly designed as a decision support tool and does not claim diagnostic authority.

- The platform has not been tested for concurrent multi-user deployment. It is designed for single-user or demonstration use and may experience performance degradation under simultaneous access from multiple users.

- The system assumes that all clinical input is provided in English. Non-English inputs are not supported and may produce unreliable outputs.

- The quality of the system output is directly dependent on the quality of the clinical information entered by the user. Incomplete, inaccurate, or ambiguous input will reduce the reliability of the report generated.

- The system assumes that the clinician using the platform has sufficient medical knowledge to critically evaluate the generated output and will not act on it without applying their own professional judgement.

3.3.3 Use Case Diagram

Figure 3.3.1 presents a main use case of dIAgnosis from the clinician's perspective. The clinician can complete a structured clinical intake, view the generated clinical report, ask follow-up questions, inspect supporting evidence, use clinical calculators, manage patient records, and create follow-up visits for returning patients.

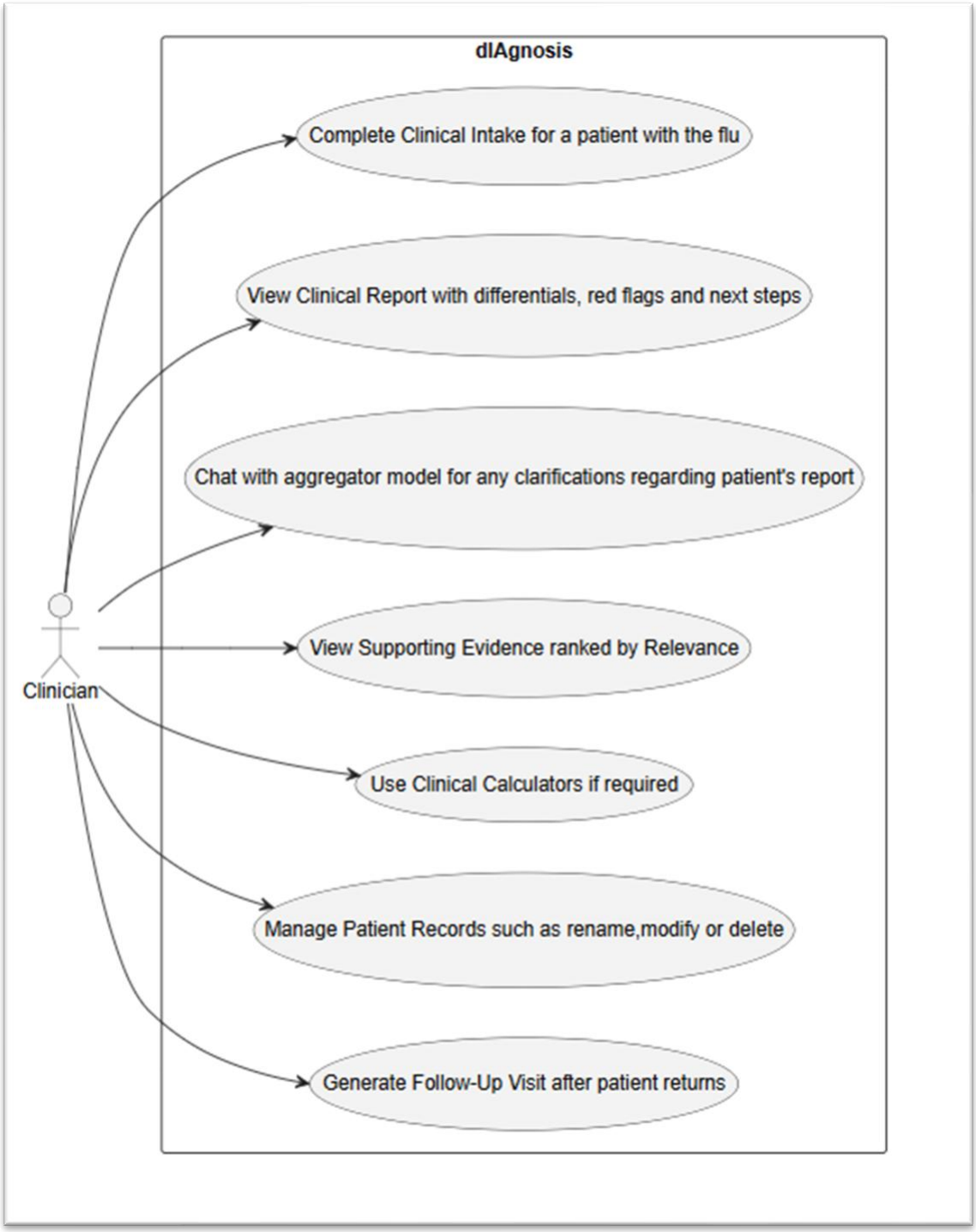


Figure 3.3.1 - Use Case Diagram for dIAgnosis

Chapter 4 - Technologies and Tools

4.1 Introduction 28

4.2 Development Environment 28

4.3 Frontend Technologies 30

4.4 Backend Technologies 31

4.5 Database Technologies 32

4.6 LLM APIs 33

4.7 Embeddings and Evidence Retrieval 34

4.8 Configuration and Security Tools 35

4.1 Introduction

This chapter presents all the technologies and tools used in the development of dIAgnosis. For each technology, a brief description is provided along with its specific role within the platform. The chapter is organised by category, covering the development environment, frontend and backend frameworks, database technologies, LLM APIs, embedding and evidence retrieval libraries, ending with configuration and security tools.

4.2 Development Environment

4.2.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful open-source code editor developed by Microsoft. It supports a wide range of programming languages and offers features such as syntax highlighting, intelligent code completion, debugging, and an integrated terminal. Visual Studio Code was used as the primary development environment for dIAgnosis due to its extensive Python support, its Git integration, and its large ecosystem of extensions that streamlined the development process [25].

4.2.2 Python

Python is a high-level, object-oriented programming language known for its simplicity, flexibility, and clean syntax. It offers a vast ecosystem of libraries and frameworks covering everything from web development to machine learning and runs across all major operating systems. Python was chosen as the primary language for dIAgnosis due to its extensive support for AI and data science libraries, its modular structure, and its compatibility with all the frameworks and APIs used in the platform [26].

4.2.3 Virtual Environment and Dependency Management

A virtual environment encapsulates all the libraries and dependencies required for a project, isolating them from the global Python installation. This ensures that the exact versions of each library used in dIAgnosis are preserved and that the project can be recreated consistently on any machine. The virtual environment contains no project code and can be deleted and rebuilt at any time from the requirements file [27].

4.2.4 Git/Version Control

Git is the most widely used version control system in the world. It tracks changes to files over time, allowing developers to monitor the history of a project, revert to previous states, and collaborate effectively. All source code was tracked using Git, allowing

changes to be monitored and maintained in a structured development history. The project repository is hosted on GitHub at https://github.com/PanagiotisM003/dIAgnosis, providing a remote backup and a clean record of the codebase for submission and review [28].

4.3 Frontend Technologies

4.3.1 Streamlit

Streamlit is an open-source Python framework designed for building interactive data applications with minimal effort. It is widely used by data scientists and AI/ML engineers to rapidly develop and deploy web-based interfaces without requiring frontend development expertise. Streamlit was chosen for dIAgnosis due to its clean and responsive user interface, its extensive built-in components for data visualisation and layout, and the ease with which applications can be deployed and shared [29].

4.3.2 Pandas

Pandas is an open-source Python library that provides flexible and efficient data structures for data manipulation and analysis [30]. It is widely used in data science and engineering for handling tabular data. In dIAgnosis, Pandas is used primarily in the admin dashboard and benchmark analytics to organise, filter, and display consultation data and benchmark results in structured table format.

4.3.3 Plotly

Plotly is an open-source Python library for creating interactive, publication-quality charts and visualisations. In dIAgnosis, Plotly is used in the admin dashboard to render the internal QA quality profile charts and the panel consistency trend charts, providing

clinicians and researchers with a visual overview of system performance across consultations [31].

4.3.4 FPDF2

FPDF2 is a lightweight Python library for generating PDF documents programmatically. It is the successor of the original PyFPDF library and is designed for simple and fast PDF creation without external dependencies. In dIAgnosis, FPDF2 is used to generate downloadable PDF versions of the final clinical report, allowing clinicians to save and share structured consultation outputs [32].

4.4 Backend Technologies

4.4.1 FastAPI

FastAPI is a modern, high-performance Python web framework for building APIs. It offers performance comparable to NodeJS and Go, while remaining easy to use and fast to develop with. FastAPI was chosen for dIAgnosis as the backend framework due to its speed, its automatic generation of interactive API documentation, and its native support for asynchronous operations which is essential for managing parallel LLM API calls across multiple worker models [33].

4.4.2 Uvicorn

Uvicorn is a lightweight Asynchronous Server Gateway Interface (ASGI) web server for Python, designed to serve asynchronous web applications. In dIAgnosis, Uvicorn is used to run the FastAPI backend, enabling the server to handle asynchronous requests efficiently, particularly important when managing concurrent LLM API calls during multi-model consultations [34].

4.4.3 Pydantic

Pydantic is the most widely used data validation library for Python, known for its speed and reliability. It uses Python type hints to define and enforce data schemas, making it easy to validate incoming requests and outgoing responses. In dIAgnosis, Pydantic is used to define the request and selected response models for the FastAPI backend, ensuring that clinical case inputs and generated report outputs conform to the expected structure before being processed or stored [35].

4.4.4 HTTPX

HTTPX is a fully featured HTTP client for Python that supports both synchronous and asynchronous APIs, as well as HTTP/1.1 and HTTP/2. In dIAgnosis, HTTPX is used for HTTP communication with external services, including asynchronous LLM provider requests in the backend and PubMed E-utilities requests in the evidence lookup module [36].

4.5 Database Technologies

4.5.1 SQLite

SQLite is a C-language library that implements a fast, self-contained, and highly reliable SQL database engine. Unlike client-server databases, SQLite stores the entire database in a single file on disk, requiring no separate server process or configuration. It was chosen for dIAgnosis due to its simplicity, portability, and suitability for a single-user research prototype where a lightweight local database is sufficient [37].

4.5.2 SQLAlchemy

SQLAlchemy is a Python SQL toolkit and Object Relational Mapper that allows developers to interact with databases using Python objects rather than writing raw SQL queries. It provides a high-level abstraction over the database layer while still allowing full control over the underlying SQL when needed. In dIAgnosis, SQLAlchemy is used to define and manage the database models for consultations and worker drafts, handling all database operations in a structured and maintainable way [38].

4.5.3 aiosqlite

aiosqlite is a Python library that provides an asynchronous interface to SQLite databases. It allows SQLAlchemy to perform database operations without blocking the event loop, which is essential in an async FastAPI application. In dIAgnosis, aiosqlite serves as the async driver that enables non-blocking database reads and writes during multi-model consultations [39].

4.6 LLM APIs

4.6.1 OpenAI API

The OpenAI API provides programmatic access to OpenAI's language models, allowing developers to integrate advanced AI capabilities directly into their applications. In dIAgnosis, the OpenAI API is used both within the worker panel and for final report aggregation. The OpenAI model can act as an independent worker alongside models from Groq, Anthropic, and Google Gemini, while the aggregator model synthesizes all worker outputs into the final structured clinical report. This design separates the independent reasoning stage from the synthesis stage, even when the same provider is used for both roles [40].

4.6.2 Groq API

Groq is an AI inference provider that offers API access to a range of large language models with a focus on extremely fast inference speeds. Unlike some larger providers, Groq specialises in low-latency model serving, making it particularly suitable for applications that require rapid responses. In dIAgnosis, the Groq API is used to access additional worker models from different providers, contributing to the genuine model diversity that the multi-model consensus architecture depends on [41].

4.6.3 Anthropic API

The Anthropic API provides programmatic access to Claude, Anthropic's family of large language models. In dIAgnosis, the Anthropic API is used as an additional worker model provider, further diversifying the panel of models contributing to the multi-model consensus pipeline [42].

4.6.4 Gemini API

The Google Gemini API gives access to Google's Gemini models, which are among the most capable general-purpose language models currently available. In dIAgnosis, Gemini is included as one of the supported worker model providers, adding another distinct perspective to the multi-model panel and strengthening the diversity of reasoning across the consultation pipeline [43].

4.7 Embeddings and Evidence Retrieval

4.7.1 Sentence-BERT / SentenceTransformers

SentenceTransformers is a Python library for computing text embeddings using pre-trained transformer models. It converts text into dense vector representations that capture semantic meaning, enabling similarity comparisons between pieces of text. In dIAgnosis, SentenceTransformers is used with the PubMedBERT model to compute embeddings of

worker model outputs and measure semantic similarity between them, supporting the semantic panel consistency analysis in the admin dashboard [44].

4.7.2 PyTorch

PyTorch is an open-source deep learning framework that provides tensor computation and automatic differentiation capabilities. In dIAgnosis, PyTorch is not used directly in the application logic but serves as a required dependency for the SentenceTransformers library, which relies on it to load and run the PubMedBERT embedding model [45].

4.7.3 NCBI E-utilities / PubMed

The NCBI E-utilities are a set of server-side programs that provide programmatic access to the Entrez database system at the National Center for Biotechnology Information (NCBI), including PubMed which contains over 35 million biomedical literature records. They accept structured URL-based queries and return results in standard formats such as XML and JSON. In dIAgnosis, the E-utilities API is used by the evidence lookup module to retrieve and rank relevant PubMed articles based on the leading differential diagnosis generated in the clinical report [46].

4.8 Configuration and Security Tools

4.8.1 python-dotenv

python-dotenv is a Python library that reads key-value pairs from a .env file and loads them as environment variables at runtime. In dIAgnosis, python-dotenv is used to manage all sensitive configuration values including API keys for OpenAI, Groq, Anthropic, and Google Gemini, ensuring they are kept outside the source code and never exposed in version control [47].

4.8.2 Streamlit Secrets

Streamlit Secrets is a built-in secrets management feature provided by the Streamlit framework. It allows sensitive configuration values to be stored in a dedicated secrets file that is kept separate from the application code. In dIAgnosis, Streamlit Secrets is used on the frontend side to store and access the backend API URL and the frontend-to-backend API key without hardcoding them in the source code [48].

4.8.3 FastAPI Middleware

dIAgnosis implements a custom HTTP middleware in the FastAPI backend that inspects every incoming request before it reaches any route handler. If a valid API key is not present in the request headers, the middleware rejects the request immediately, preventing unauthorised access to the backend endpoints. This restricts backend access to clients that possess the configured API key [49].

Chapter 5 - System Design and Implementation

5.1 Introduction 37

5.2 System Architecture Overview 38

5.3 Backend Design and Implementation 40

5.4 Multi-Model Reasoning Pipeline 40

5.5 Aggregation and Report Generation 42

5.6 Longitudinal Follow-Up Design 43

5.7 Data Storage Design 43

5.8 Evaluation and Benchmarking Subsystem 44

5.9 Evidence Retrieval and Report Support 45

5.10 Frontend Design and User Workflows 46

5.11 Safety and Security Considerations 47

5.12 Code Structure 47

5.13 System Data Flow 48

5.14 Interface Screenshots 49

5.1 Introduction

This chapter describes the design and implementation of dIAgnosis, focusing on how the system was structured and how its main features were built. While the previous chapter

introduced the technologies used, this chapter explains how those technologies were combined to form a working clinical decision-support prototype. The system follows a modular architecture, separating the user interface, backend logic, database layer, external model providers, evidence retrieval, and evaluation tools. This separation made the platform easier to develop, test, maintain, and extend.

The chapter also presents the reasoning workflow used by dIAgnosis. A clinical case is entered through the Streamlit frontend, processed by the FastAPI backend, sent to several large language model providers, and then combined into a structured consensus report. Additional functions, such as longitudinal follow-up, PubMed evidence lookup, internal quality assessment, benchmark evaluation, debate rounds, and PDF export, are also described. Together, these components form the full implementation of the prototype.

5.2 System Architecture Overview

dIAgnosis was designed as a web-based system with a clear separation between the frontend, backend, database, and external services. The frontend is implemented using Streamlit and provides the main interface used by the clinician or researcher. It is responsible for collecting clinical information, displaying generated reports, showing visual analytics, and offering access to evidence lookup, debate, follow-up, and administrative tools.

The backend is implemented using FastAPI and acts as the main processing layer of the system. It receives structured requests from the frontend, validates the incoming data, manages the multi-model reasoning pipeline, stores results, and returns structured responses. The backend communicates with external LLM APIs through asynchronous HTTP requests, allowing several model calls to be handled efficiently.

The database layer uses SQLite with SQLAlchemy and aiosqlite. It stores consultations, worker model outputs, final reports, evaluation results, clinical benchmark cases, and

benchmark runs. External services include OpenAI, Groq, Anthropic, Google Gemini, and NCBI E-utilities for PubMed evidence retrieval. Configuration and security are handled through environment variables, Streamlit secrets, and an API-key middleware

Figure 5.2.1 shows the general system architecture of dIAgnosis. The clinician interacts with the Streamlit frontend, which sends authenticated HTTP requests to the FastAPI backend. The backend manages the multi-model reasoning process, communicates with external LLM providers, stores consultation data in the SQLite database layer, and supports evaluation and benchmarking. PubMed evidence retrieval and configuration/security components are also shown as supporting parts of the system.

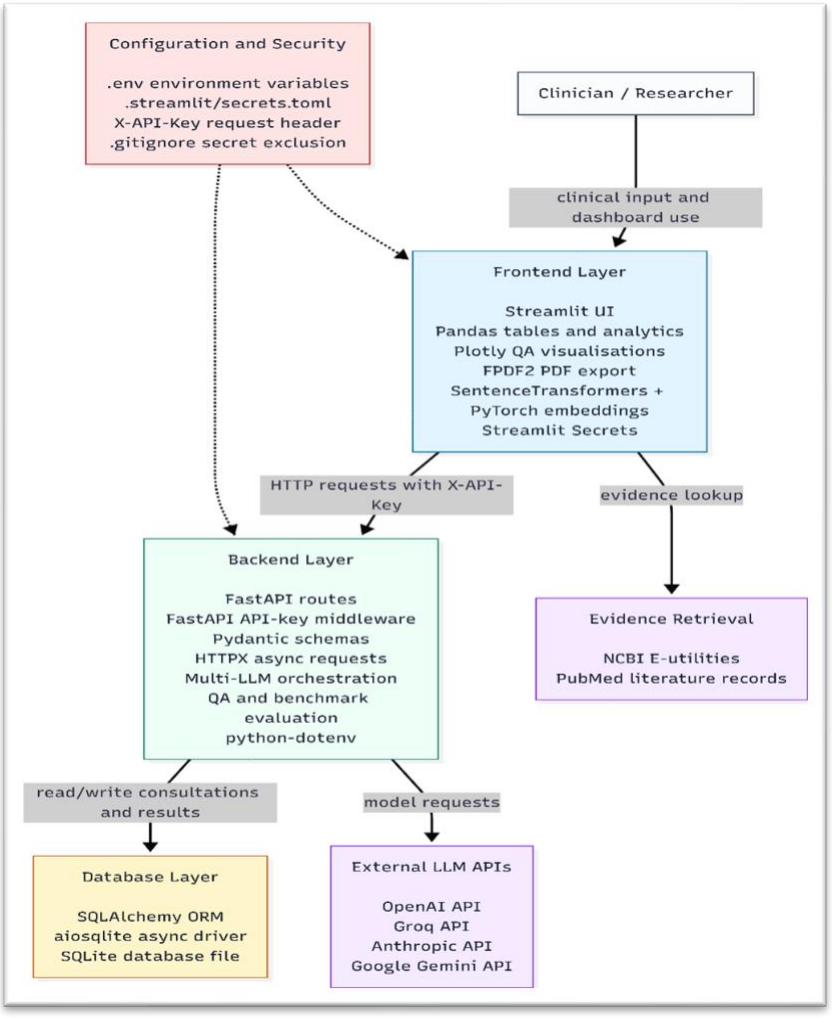


Figure 5.2.1 – General System Architecture of dIAgnosis

5.3 Backend Design and Implementation

The backend is the central coordination layer of dIAgnosis. It exposes the API endpoints used by the frontend and manages the main system operations. FastAPI was used because it supports asynchronous request handling, structured request validation, and automatic API documentation. This was particularly useful because the system depends on multiple external API calls, especially during the multi-model reasoning stage.

The backend includes endpoints for generating a consultation report, retrieving consultation history, deleting or renaming records, running debate rounds, follow-up chat with single-model, importing clinical benchmark cases, running benchmark evaluations, and retrieving aggregate benchmark results. These endpoints are defined in the backend application and use Pydantic models to ensure that incoming requests follow the expected structure.

A custom API-key middleware is also implemented in the backend. When the API_KEY environment variable is configured, incoming requests must include the correct X-API-Key header. This provides a simple access-control mechanism between the frontend and backend. If no API key is configured, the system can still run openly during local development.

5.4 Multi-Model Reasoning Pipeline

The main reasoning process of dIAgnosis is based on a multi-model worker pipeline. Instead of relying on a single model response, the backend sends the same structured clinical prompt to several configured LLMs. Each model acts as an independent worker and produces its own assessment, ranked differential diagnoses, suggested investigations, red flags, warnings, certainty level, and patient-facing explanation.

The worker models are selected from the MODELS list in backend/models_config.py. Each model configuration contains an internal identifier, provider name, provider-specific model name, display name, and enabled status. During a consultation, the backend first filters this list to include only models marked as enabled. If a maximum number of models is provided, the system then selects the first enabled models up to that limit. In the current consultation workflow, the frontend sends max_models = 5, so the five enabled configured models are used.

The worker calls are executed in parallel using Python asynchronous execution. In the backend, query_all() creates one asynchronous task for each selected model and runs them together using asyncio.gather(). Therefore, the total waiting time is mainly affected by the slowest model response rather than the sum of all model response times. This is why the interface can convene a five-model panel without waiting for each provider sequentially.

After the worker responses return, the backend normalizes the outputs. If a model returns valid JSON, its response is marked as successful and stored as a worker draft. If a model fails, times out, or returns invalid JSON, the failure is recorded with that model's status instead of stopping the whole pipeline. The system only stops report generation if all worker models fail.

The successful worker drafts are then passed to a dedicated aggregator model. In the current implementation, the aggregator uses GPT-5.1 through the OpenAI API. The aggregator receives the structured patient context together with the worker outputs and produces the final JSON consensus report. The worker panel consists of five enabled models: GPT-5.1 from OpenAI, Llama-3.3-70B Versatile from Groq, Qwen-3-32B from Groq, Claude Sonnet 4.5 from Anthropic, and Gemini 3 Pro Preview from Google. These models were selected to provide provider diversity across the panel, rather than relying on multiple models from a single provider. The aggregator model is also configured separately in the backend and is used after the worker stage to synthesize the independent drafts into the final report. The following table shows their role in the implementation:

Role	Provider	Model
Worker	OpenAI	GPT-5.1
Worker	Groq	Llama-3.3-70B Versatile
Worker	Groq	Qwen-3-32B
Worker	Anthropic	Claude Sonnet 4.5
Worker	Google Gemini	Gemini 3 Pro Preview
Aggregator	OpenAI	GPT-5.1

Table 5.4.1 - Table of LLMs used in dIAgnosis

5.5 Aggregation and Report Generation

After the worker models return their individual outputs, dIAgnosis generates a final consensus report using an aggregator model. The purpose of the aggregator is not simply to copy one worker response, but to compare the worker outputs and produce a structured synthesis. It considers the level of agreement between models, repeated diagnoses, investigation suggestions, safety warnings, red flags, and certainty levels.

The final report follows a consistent structure. It includes a clinical assessment, ranked differential diagnoses, suggested investigations, red flags, warnings, and certainty statement. This structured format makes the output easier to display in the frontend, store in the database, evaluate using internal metrics, and export as a PDF.

The aggregator prompt includes safeguards to discourage unsupported conclusions. It is instructed to remain consistent with the worker panel and to explain its reasoning when objective clinical signals point toward important diagnoses. This design helps reduce the

risk of a final report introducing a conclusion that is not supported by the broader model panel.

5.6 Longitudinal Follow-Up Design

dIAgnosis includes longitudinal follow-up functionality so that consultations are not treated as isolated events. When a patient has previous stored visits, the system can retrieve relevant historical information and include it as context in the next consultation. This allows the model panel to consider previous symptoms, background conditions, investigations, and earlier assessments.

The frontend groups consultations by patient and provides shortcuts for follow-up visits. When a follow-up consultation is created, previous history can be included in the prompt sent to the worker models and aggregator. The interface also displays follow-up context so that the user can see how the current visit relates to earlier records.

This feature is important because clinical reasoning often depends on change over time. A symptom may be more significant if it is new, worsening, persistent, or different from a previous presentation. By including longitudinal context, the system is better aligned with real clinical workflows.

5.7 Data Storage Design

The system uses SQLite as its database engine because it is lightweight, portable, and suitable for a research prototype. SQLAlchemy is used to define and interact with the database models, while aiosqlite provides asynchronous database access. This allows the backend to perform database operations without blocking the event loop.

The database stores consultation records, patient details, query parameters, worker model drafts, final reports, evaluation outputs, clinical benchmark cases, benchmark runs, and benchmark results. This storage design supports both normal clinical consultation workflows and research-focused evaluation workflows.

Keeping worker outputs and final reports separately is an important design choice. It allows the system to compare individual model responses with the aggregated report, calculate internal QA metrics, and inspect whether the final report reflects the broader panel. It also supports later analysis of model agreement and disagreement.

5.8 Evaluation and Benchmarking Subsystem

The evaluation subsystem was implemented to support internal analysis of report quality, worker agreement, and model-panel behavior. It is not intended to determine whether a diagnosis is clinically correct. Instead, it evaluates whether the final report is structurally complete and whether its content is supported by the individual worker drafts.

For each generated report, the system calculates structural QA metrics. These include:

- **Report support**

  Measures how strongly the final report overlaps with the worker outputs.

- **Worker agreement**

  Measures how consistently the panel converged on similar diagnoses.

- **Format completeness**

  Checks whether the required report fields are present.

- **Unsupported-content risk**

Highlights parts of the final report that are not strongly reflected in the worker drafts.

- **Critical minority signals**

Identifies high-risk diagnoses or warnings raised by a smaller subset of workers.

The system also includes a clinical benchmark workflow. Benchmark cases can be imported from template data or uploaded as JSON, stored in the database, selected by the user, and run through the same multi-model pipeline used for normal consultations. The generated outputs are then compared against reference case data. The benchmark module reports metrics such as panel match score, Top-1 accuracy, Top-3 hit rate, unsafe omission rate, and panel-versus-best-single win rate. It also provides case-level comparisons showing the predicted diagnosis, reference diagnosis, exact match, family-level match, Top-3 hit, and critical diagnosis coverage.

In addition, the system includes semantic panel consistency analysis. This uses PubMedBERT embeddings through SentenceTransformers to compare the diagnostic focus of worker model outputs. Instead of only comparing exact diagnosis labels, the system compares the meaning of the worker responses in embedding space. This allows it to identify semantic divergence, outlier worker outputs, and critical minority signals. These results do not prove that a model is right or wrong, but they help highlight cases where the panel disagreed in a way that may require manual review.

5.9 Evidence Retrieval and Report Support

The evidence retrieval module allows the user to search for biomedical literature related to the generated consensus report. This function is available through the Evidence tab after a report has been generated. The system uses NCBI E-utilities to query PubMed and retrieve publication metadata programmatically.

The PubMed query is built from the structured report rather than from a separate user-entered search term. The module first identifies the primary diagnostic entity from the leading item in the report's ranked differential diagnosis list. It can also use concise investigation or safety-signal terms from the report when these are suitable for search. The query is then sent to PubMed using the E-utilities search endpoint, and the returned PubMed IDs are used to retrieve article summaries.

Retrieved articles are ranked before display. The ranking considers factors such as article type, relevance to the report concepts, and publication recency. For example, guidelines, systematic reviews, meta-analyses, reviews, and clinical trials are weighted more strongly than lower-level article types. The frontend displays the article title, journal, publication date, evidence type, relevance score, and a short explanation of why the source was selected.

This evidence feature is designed as a support tool rather than a validation mechanism. Retrieved articles may help the clinician review the report, but they do not prove that the generated diagnosis is correct. The evidence tab therefore connects the model output to biomedical literature for inspection, while leaving final interpretation to the clinician.

5.10 Frontend Design and User Workflows

The frontend was designed in Streamlit to provide a structured clinical workspace. The main workflow begins with clinical intake, where the user enters patient details, presenting complaint, history, medications, allergies, vital signs, and other relevant context. The interface then sends this information to the backend to generate a multi-model consensus report.

After a report is generated, the frontend displays the result across several tabs. These include the main clinical report, risk tools, evidence lookup, and optional debate

functionality. The user can review the ranked differential diagnoses, investigations, safety warnings, red flags, patient handout, and model-panel outputs. The system also supports follow-up questions through a chat feature linked to the generated report.

The frontend also includes an admin dashboard. This dashboard displays internal QA metrics, panel consistency information, benchmark results, and trend visualizations. Plotly is used for interactive charts, Pandas is used for tabular data, and FPDF2 is used to export the final report as a downloadable PDF

5.11 Safety and Security Considerations

Because dIAgnosis is a clinical decision-support prototype, safety considerations were included throughout the design. The system presents itself as a support and evaluation tool rather than an autonomous diagnostic authority. Its outputs are intended to be reviewed by a clinician before influencing patient care.

The generated reports include red flags, warnings, certainty statements, and patient-facing explanations. The system also includes internal QA checks that help identify structural problems, weak panel support, or possible unsupported content. These features do not eliminate risk, but they create additional review points around the generated output.

Security-related measures include storing provider API keys outside the source code, using .env files for backend secrets, using Streamlit secrets for frontend configuration, excluding secret files through .gitignore, and applying API-key middleware between the frontend and backend. These measures help reduce the risk of exposing sensitive configuration values or allowing unintended backend access.

5.12 Code Structure

The application is organized into separate backend, frontend, data, and documentation areas. The backend folder contains the FastAPI application, database configuration, ORM models, CRUD operations, LLM client logic, evaluation logic, and model configuration. This keeps the server-side responsibilities grouped in one part of the application.

The frontend folder contains the Streamlit application and supporting modules. These modules handle API endpoint configuration, reusable UI components, evidence retrieval, embedding access, QA visualisations, report parsing, PDF generation, session state, and styling. This separation keeps the main app file more manageable and makes individual features easier to maintain.

The data folder contains local data assets such as clinical benchmark templates and database files. This structure makes the project easier to understand because each folder has a clear role.

5.13 System Data Flow

The system data flow begins when the user enters clinical information through the Streamlit interface. The frontend packages this information into a structured request and sends it to the FastAPI backend using an HTTP request. If API-key protection is enabled, the request must include the correct X-API-Key header.

The backend validates the request and constructs a clinical prompt for the worker models. The prompt is sent asynchronously to the configured LLM providers. Each successful worker response is collected and stored. The aggregator then receives the worker outputs and produces the final structured consensus report.

After the final report is generated, the backend evaluates the output, stores the consultation and model drafts in the database, and returns the result to the frontend. The

frontend displays the report, allows the user to inspect evidence, run a debate round, ask follow-up questions, export a PDF, or review the case later through the history and dashboard views.

5.14 Interface Screenshots

The workflow starts with a structured intake form, where the user creates a new consultation for Patient 1 and enters the patient details, vitals, history, medications, allergies, risk factors, symptoms, examination notes, and lab/ECG findings. This shows that the system first converts the clinical encounter into structured input fields before any model is called.

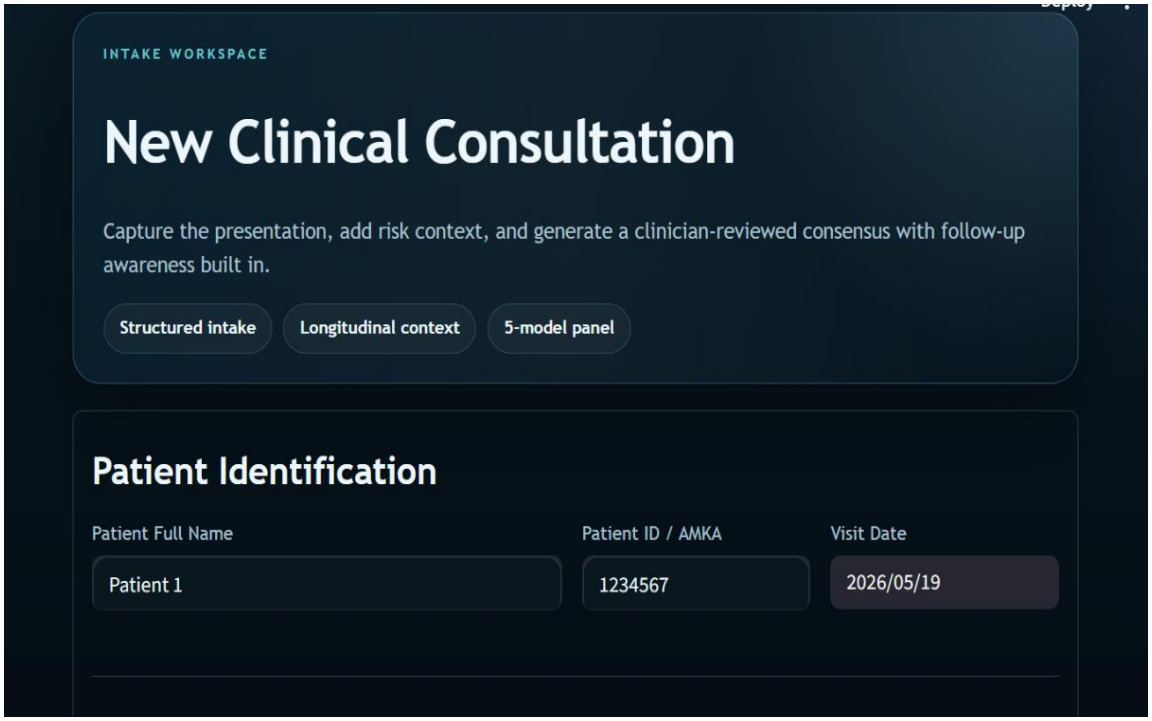


Figure 5.14.1 – New Consultation Screen (1)

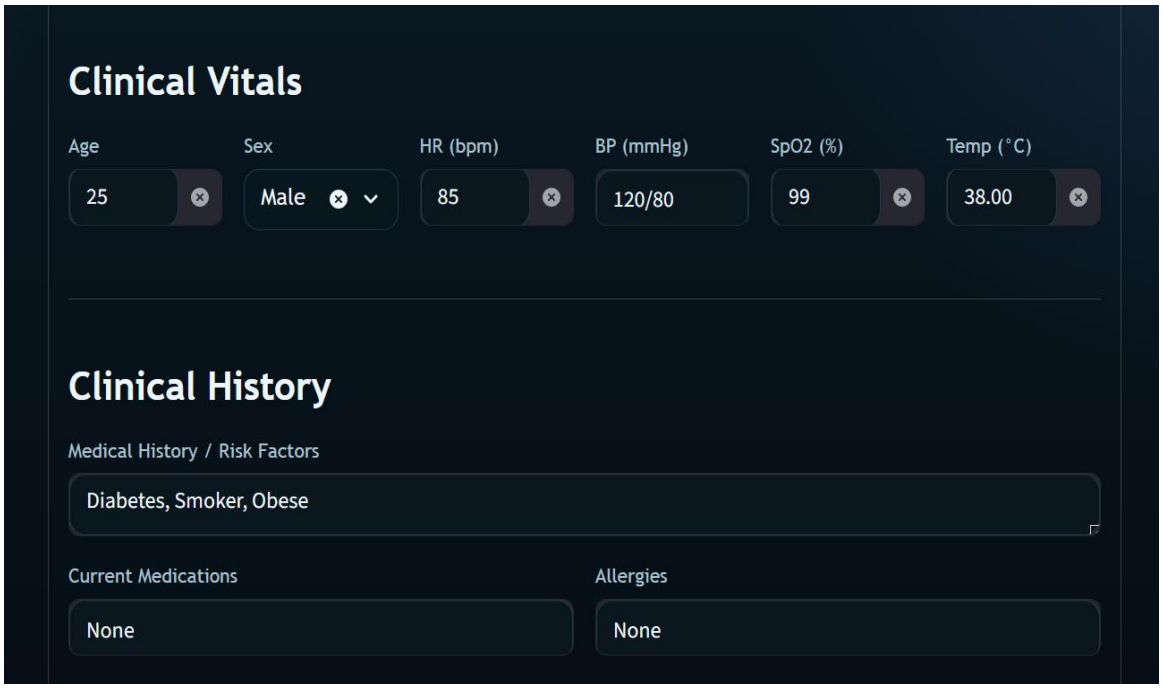


Figure 5.14.2 – New Consultation Screen (2)

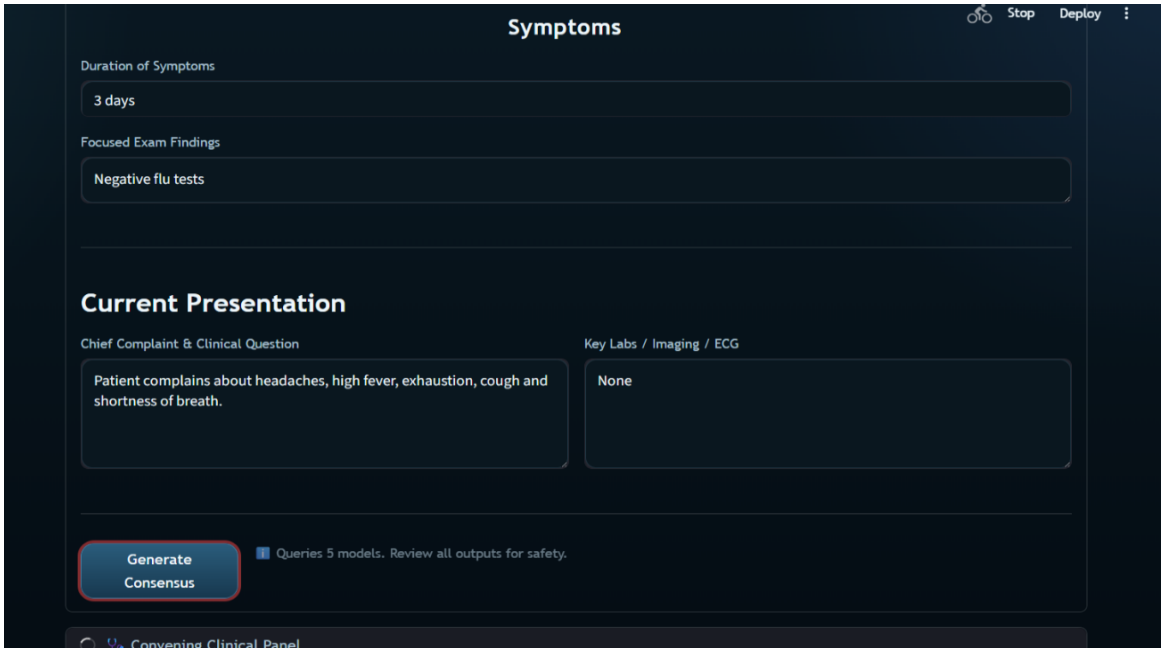


Figure 5.14.3 – New Consultation Screen (3)

After the form is completed, the user selects Generate Consensus. This triggers the backend workflow that sends the structured case to the configured model panel. The loading state shows that dIAgnosis is convening the clinical panel and sending the case to multiple models.

The sidebar shows how consultations are stored and organized. The entered case is saved under Patient 1, with the visit date attached to the consultation. From this panel, the user can reopen a previous visit, create a follow-up visit, or manage stored records.

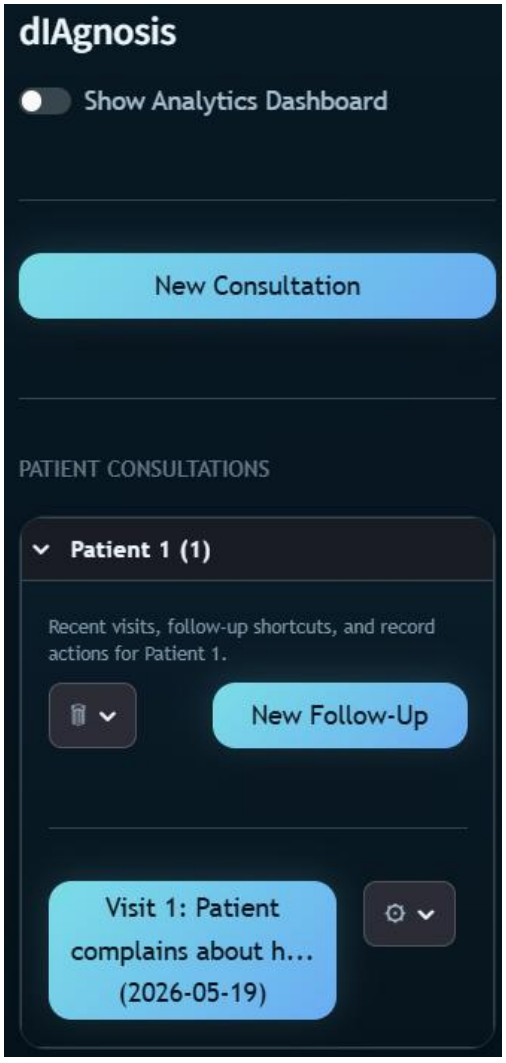


Figure 5.14.4 – Sidebar

Once the panel finishes, the system opens the Clinical Consensus Report. The header identifies Patient 1, the visit date, panel agreement, safety disclaimer, vital telemetry, consensus status, leading diagnosis, certainty level, escalation state, and next best test. In this case, the leading diagnosis is community-acquired pneumonia, with medium certainty and urgent escalation suggested.

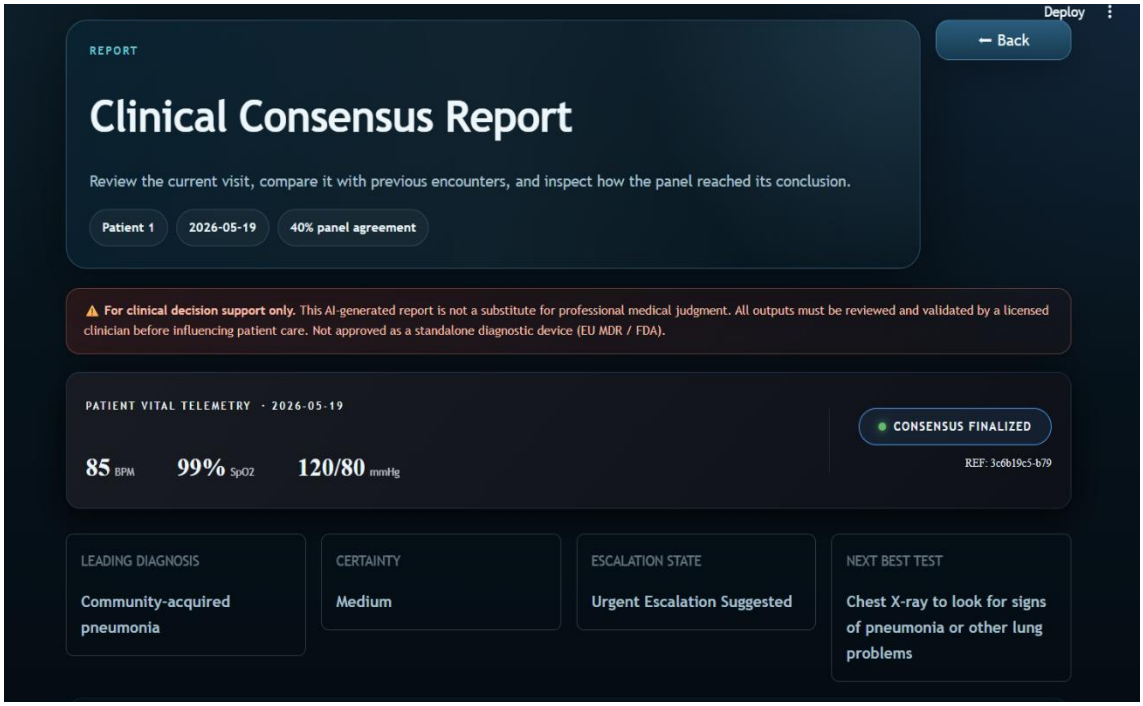


Figure 5.14.5 – Final Consensus Report Header

The next report section, Immediate Concern, shows how the system surfaces the highest-priority safety state before the detailed report. It presents the escalation recommendation, explains why the case needs attention, identifies the current visit, and shows whether previous visits are available for comparison. This demonstrates that the report is not only a diagnosis list but also organizes the output by urgency and context.

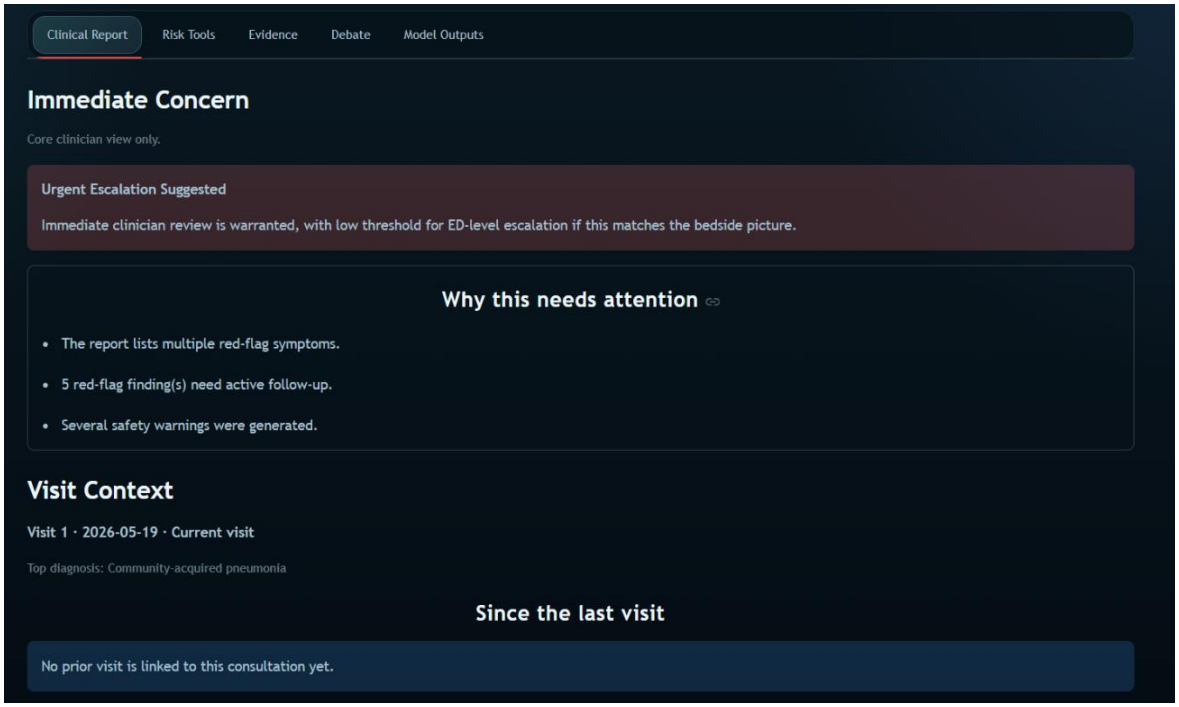


Figure 5.14.6 – Clinical Report Tab (1)

The Summary section gives the condensed narrative version of the report. It turns the structured input and model outputs into a readable snapshot of the consultation. Beside it, the certainty panel shows the aggregator's confidence level. The certainty panel shows the aggregator's confidence as medium, while the support metrics summarize worker agreement and alignment.

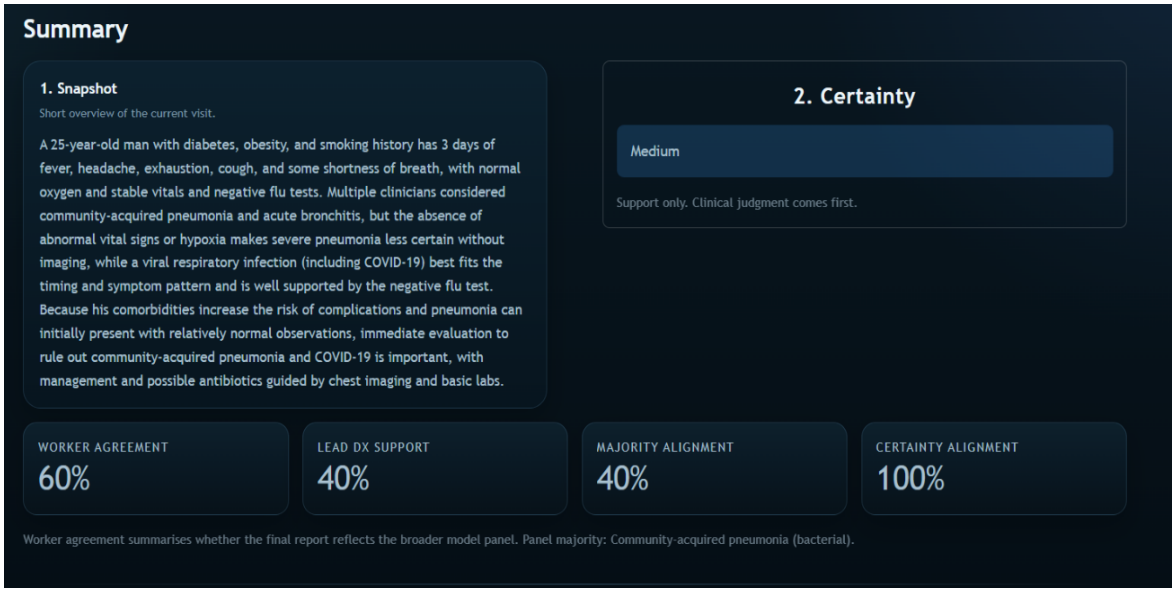


Figure 5.14.7 – Clinical Report Tab (2)

The Likely Diagnoses section presents the ranked differential diagnosis. Community-acquired pneumonia is ranked first, followed by acute bronchitis and suspected COVID-19. Each diagnosis is displayed with an ICD-10 code and agreement percentage, showing how strongly the panel supported each possibility.

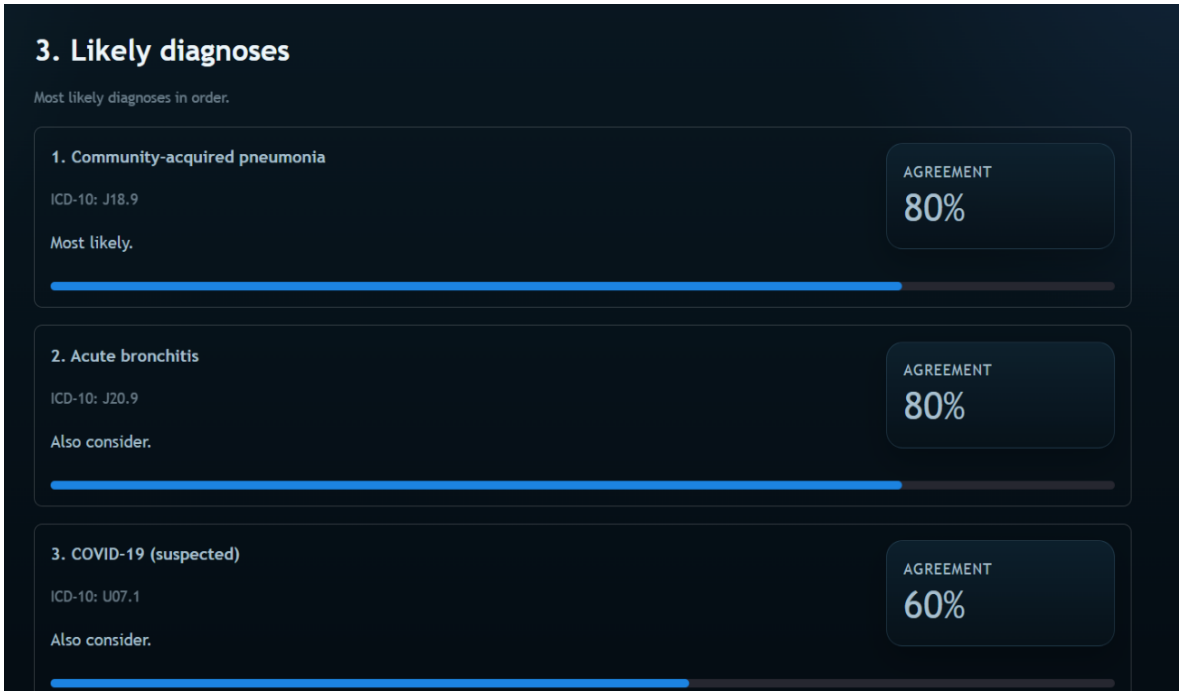


Figure 5.14.8 – Clinical Report Tab (3)

The Assessment and Recommended Tests sections provide the expanded clinical interpretation. The report explains why pneumonia remains important despite stable vital signs and recommends chest X-ray, COVID-19 testing, blood count, inflammatory markers, metabolic testing, and repeat vital monitoring.

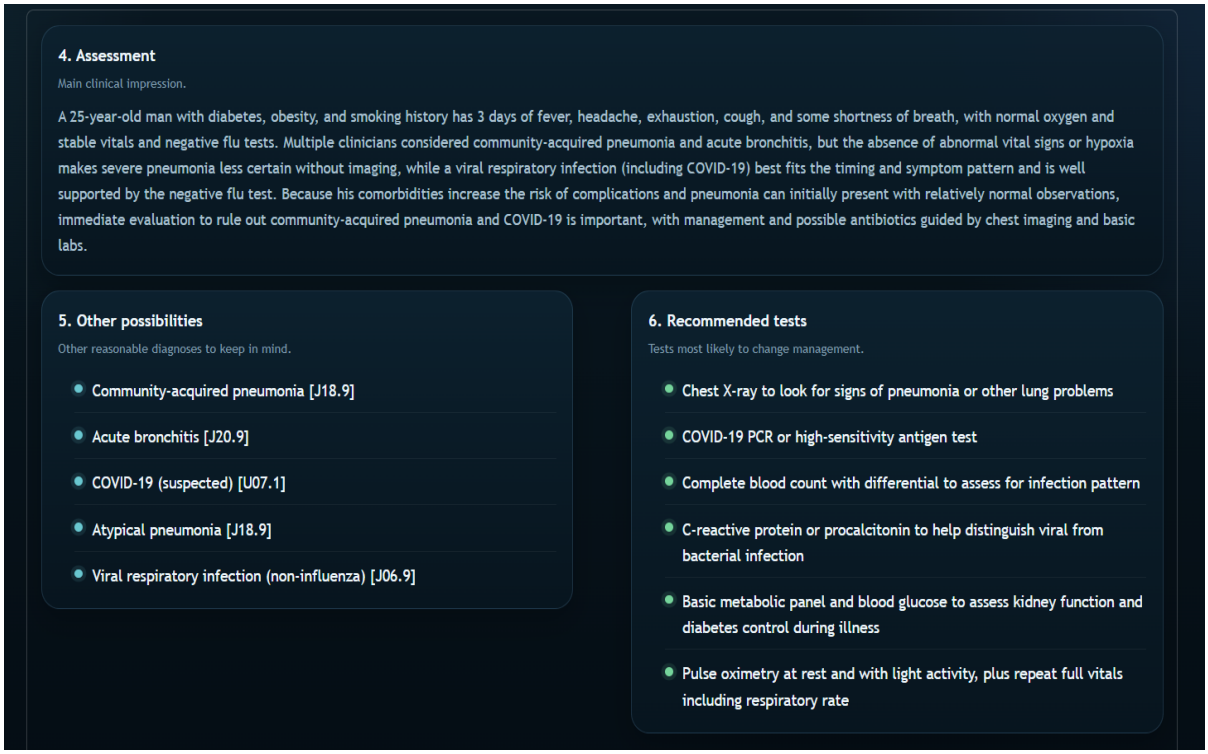


Figure 5.14.9 – Clinical Report Tab (4)

Red flags and Safety checks sections show the safety section of the report. It highlights warning signs and safety checks that should be reviewed before reassurance, discharge, or follow-up planning, along with options to download a pdf of the finalized report or a clinician note.

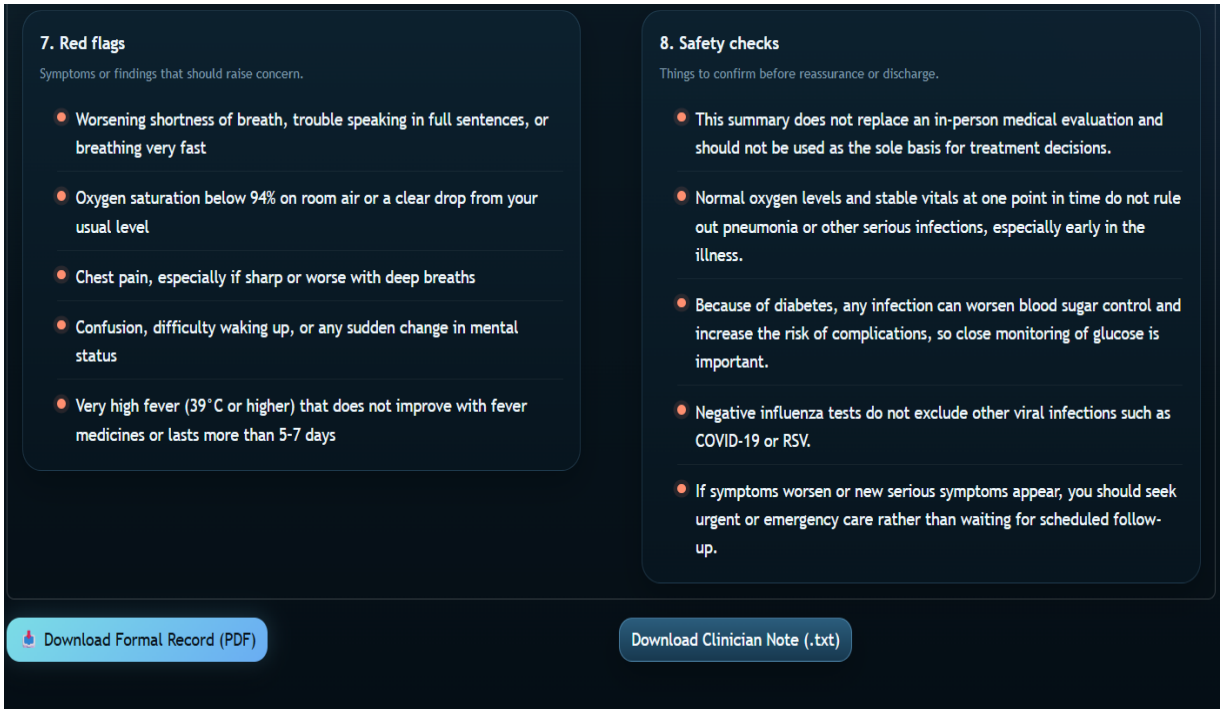


Figure 5.14.10 – Clinical Report Tab (5)

The follow-up chat screen shows how the finalized report can be queried after generation. This screen demonstrates that the chat assistant answers using the finalized report context rather than acting as a separate general-purpose chatbot allowing clinicians to analyze the report with the help of the aggregator.

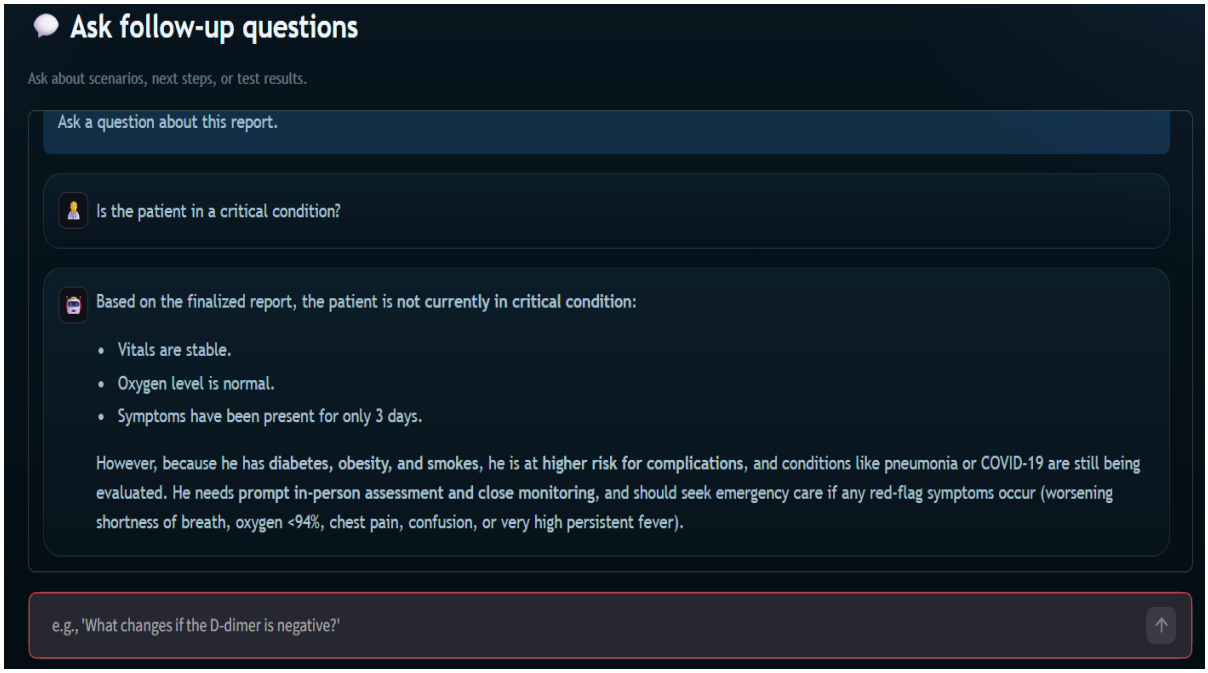


Figure 5.14.11 – Clinical Report Tab (6)

The Risk Tools tab shows additional calculators integrated into the same workflow. The user can switch between HEART, Wells, PERC, qSOFA, and GCS. The displayed HEART calculator produces a score from manually selected fields. This shows that dIAgnosis supports structured bedside scoring alongside the AI report, but the calculator is a separate tool controlled by its own input fields.

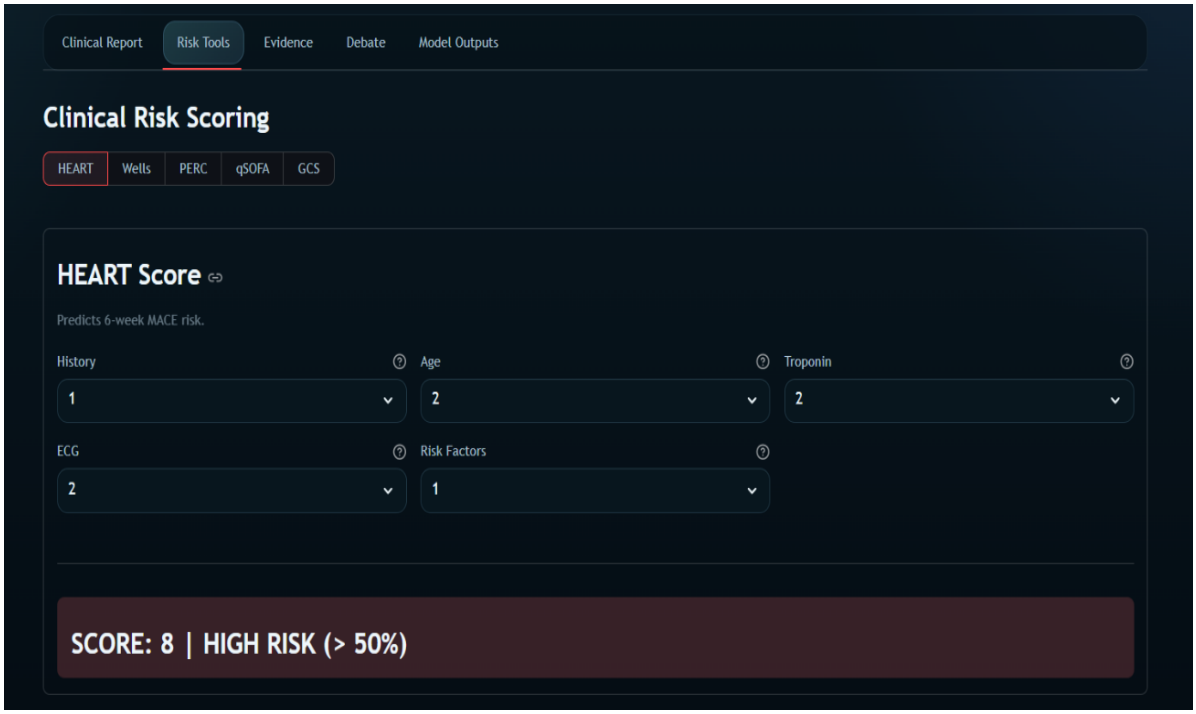


Figure 5.14.12 – Risk Tools Tab

The Evidence tab shows the evidence-grounding workflow. After the report is generated, the system identifies the primary diagnostic entity, builds a structured PubMed query, retrieves relevant sources, ranks them, and explains why each source was selected. This screen demonstrates how dIAgnosis connects the generated consensus to external biomedical literature for review.

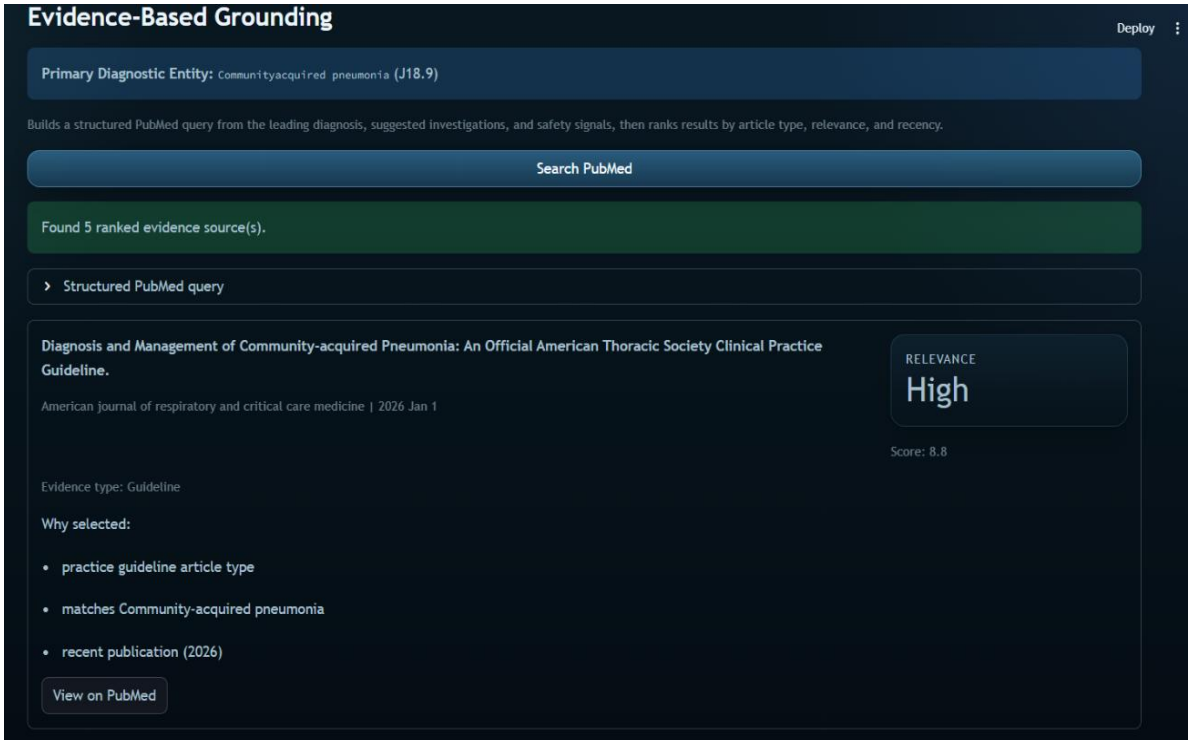


Figure 5.14.13 – Evidence Tab

The Debate tab shows an optional second-pass workflow. The app first preserves the original worker drafts, creates a neutral peer summary, and then allows the models to revise their positions after seeing the panel-level summary. The debate summary displays agreement, format, and risk-signal changes. This demonstrates that debate is an optional review layer, not the default report-generation step. The debate workflow shows that the original worker outputs were reviewed again. This screen displays the neutral peer-review summary that is sent back to the worker models before they revise their positions.

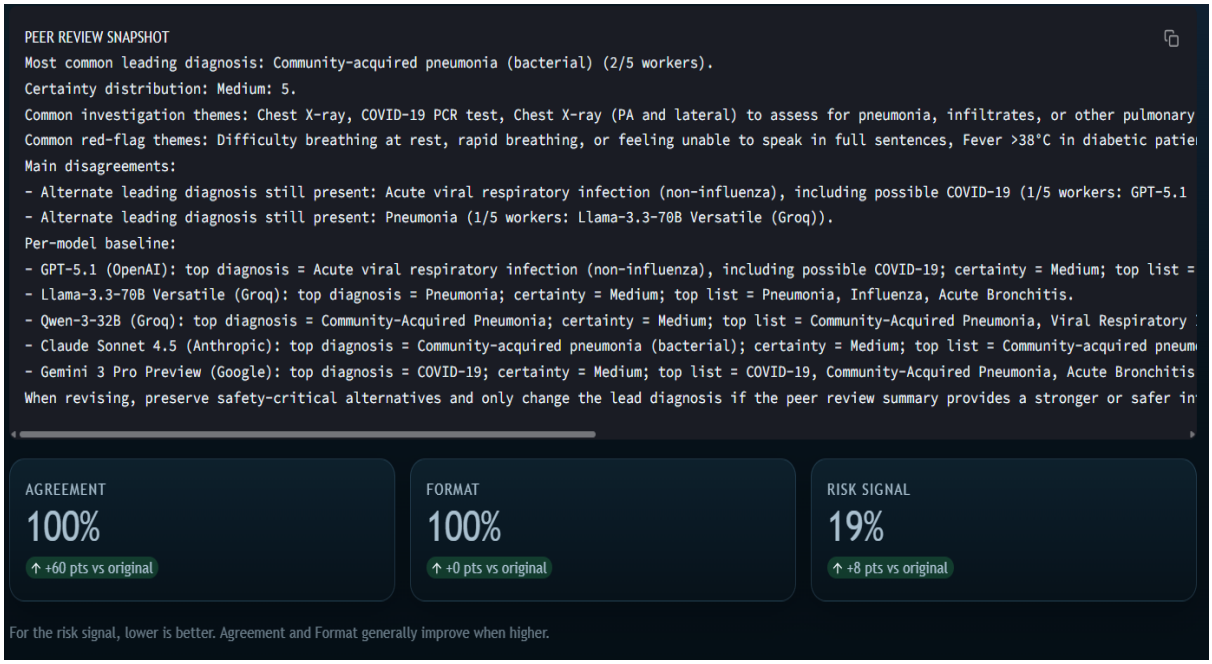


Figure 5.14.14 - Debate Tab (1)

After the debate round, the system displays revised worker positions and a separate debated consensus report. The original report remains unchanged, while the debated report is shown as an additional comparison output.

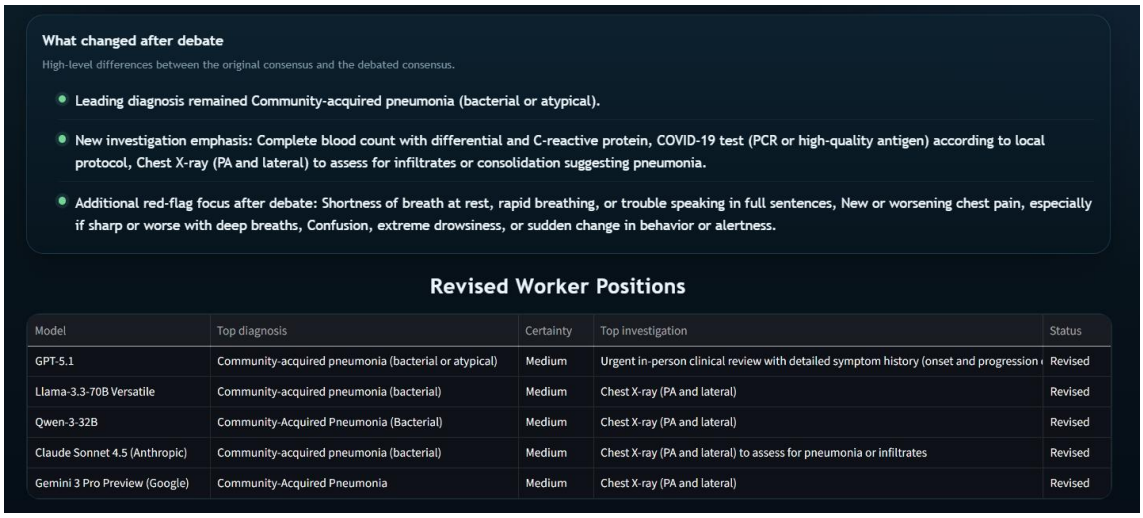


Figure 5.14.15 - Debate Tab (2)

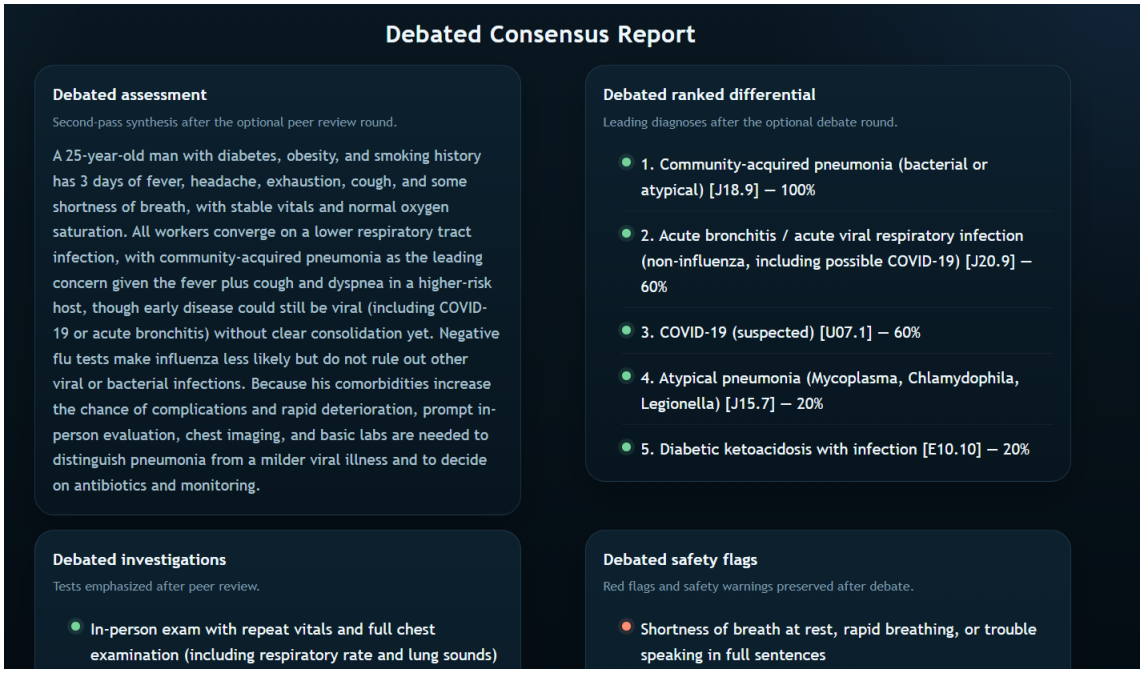


Figure 5.14.16 - Debate Tab (3)

The Model Outputs tab shows the original individual model responses before aggregation. Each model is listed with completion status and can be expanded. This screen demonstrates transparency: the user can inspect the raw worker-level outputs that contributed to the consensus. Each model can be expanded to inspect its top diagnosis, certainty, differential list, investigations, red flags, warnings, and raw structured response.

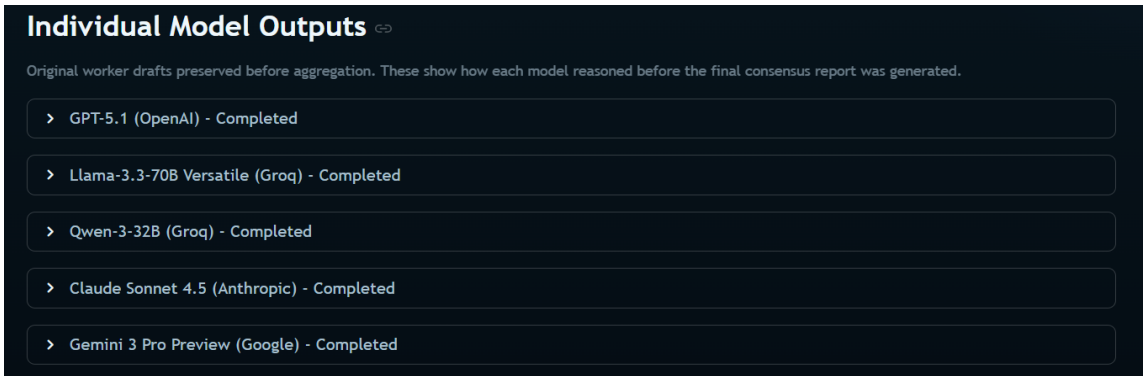


Figure 5.14.17 - Model Outputs Tab (1)

The expanded model output shows one model's full structured response, including provider, top diagnosis, certainty, differential count, assessment, ranked differentials, investigations, red flags, warnings, and raw reasoning fields. This demonstrates traceability from the final consensus back to the individual model-level contribution.

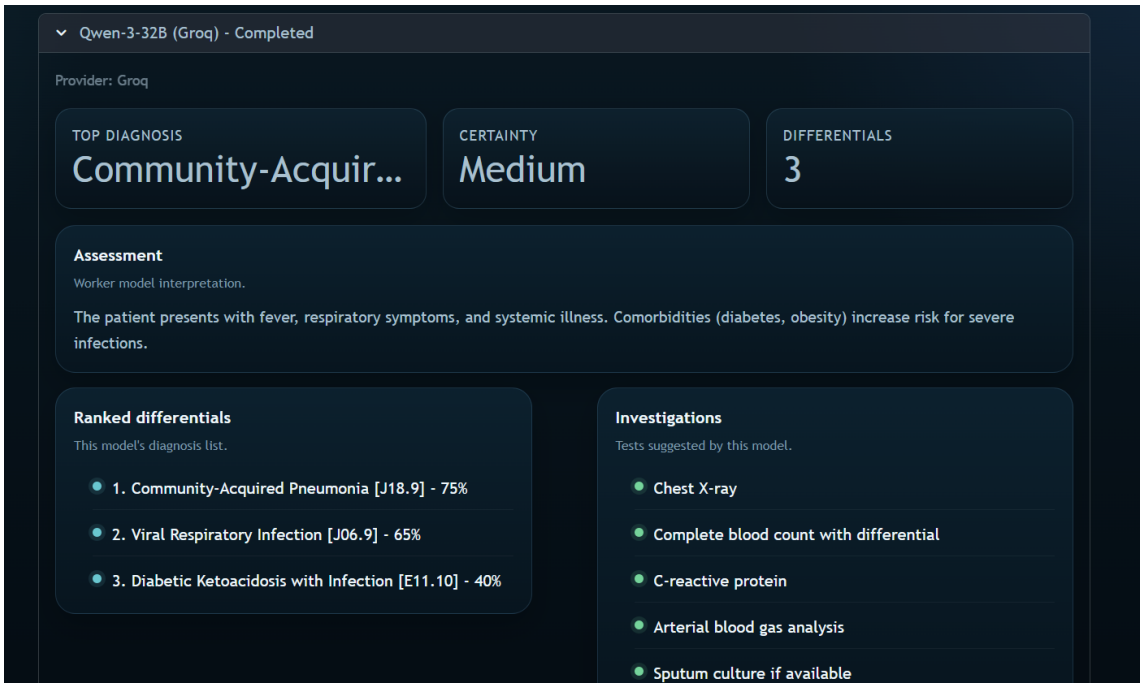


Figure 5.14.18 - Model Outputs Tab (2)

The System Analytics and Oversight screen shows the internal quality-assurance layer of dIAgnosis. This page is not part of the patient-facing clinical report, and it is not intended to prove whether the diagnosis is medically correct. Instead, it checks whether the generated reports are structurally complete, whether the final consensus is supported by the worker models, and whether any parts of the report may need manual review. At the top of the screen, the dashboard summarizes all stored consultations currently being reviewed. In this example, the system has reviewed 2 consultations. The overall report support is 64.3%, meaning that, on average, the final reports are moderately supported by the outputs produced by the worker models. Worker agreement is 56.7%, showing that the model panel did not fully converge on the same conclusions across the stored cases. Format completeness is 100%, meaning the final reports contain the required structured fields. Unsupported-content risk is 19.0%, which indicates that some final report content may not be strongly reflected in the worker drafts.

Figure 5.14.19 - System Analytics (1)

The Quality Profile section breaks these values into more specific support signals. Format completeness is 100%, which means the report structure is complete and follows the expected schema. Lead diagnosis support is 40%, meaning only part of the worker panel independently placed the final leading diagnosis as their own top diagnosis. Top-2 diagnosis support is 60%, meaning more workers included the final leading diagnosis somewhere within their two most likely diagnoses, even if it was not always ranked first. Majority alignment is 40%, meaning the final report only partially matched the most common worker position. Certainty agreement is 90%, showing that the final certainty level is mostly consistent with the certainty levels produced by the workers. Investigation support is 71.8%, meaning the recommended tests in the final report overlap fairly well with investigations suggested by the models. Safety-item support is 30.5%, meaning many red flags or warnings in the final report were not strongly repeated across the individual worker outputs.

Area	Average	Direction	Meaning
Format completeness	100.0%	Higher is better	The final report contains the required fields and follows the expected report structure.
Lead diagnosis support	40.0%	Higher is better	Worker models independently supported the final leading diagnosis as their own first choice.
Top-2 diagnosis support	60.0%	Higher is better	Worker models included the final leading diagnosis within their two most likely diagnoses.
Majority alignment	40.0%	Higher is better	The final report agrees with the panel majority rather than an isolated worker output.
Certainty agreement	90.0%	Higher is better	The final certainty level is consistent with the worker panel's certainty distribution.
Investigation support	71.8%	Higher is better	Suggested investigations overlap with recommendations made by the worker models.
Safety-item support	30.5%	Higher is better	Red flags and warnings in the final report are supported by the worker outputs.

Figure 5.14.20 - System Analytics (2)

The Support and Risk Over Time chart shows how report support and unsupported-content risk change across stored consultations. The blue line represents report support, where higher values are better. The red dotted line represents unsupported-content risk, where lower values are better. In this example, report support remains in the moderate range, while unsupported-content risk increases from the first stored case to the second. This allows the user to identify cases that may require closer inspection later, especially if support drops or risk rises.

Figure 5.14.21 - System Analytics (3)

The Cases Needing Review table lists the consultations that the QA system believes deserve manual inspection. Each row includes the patient, visit date, number of worker models, report support, worker agreement, format completeness, risk signal, and critical minority score. Patient 2 has lower support and a higher risk signal than Patient 1, so it appears as a stronger review candidate. Patient 1 has better support and lower unsupported-content risk, but it is still listed because worker agreement and lead diagnosis support are not perfect.

Figure 5.14.22 – System Analytics (4)

The Semantic Panel Consistency section adds another internal review layer. Instead of only comparing exact diagnosis names, it uses biomedical embeddings to compare the meaning of the diagnostic outputs. This matters because two models may use different words for clinically related concepts. For example, “community-acquired pneumonia” and “pneumonia” are semantically close, while “acute viral respiratory infection” is related but meaningfully different. The screen reports 2 cases analyzed, mean panel divergence of 75.7%, 1 semantic outlier, and 2 critical minority signals. These values show that the stored consultations include meaningful differences in how worker models framed the diagnosis.

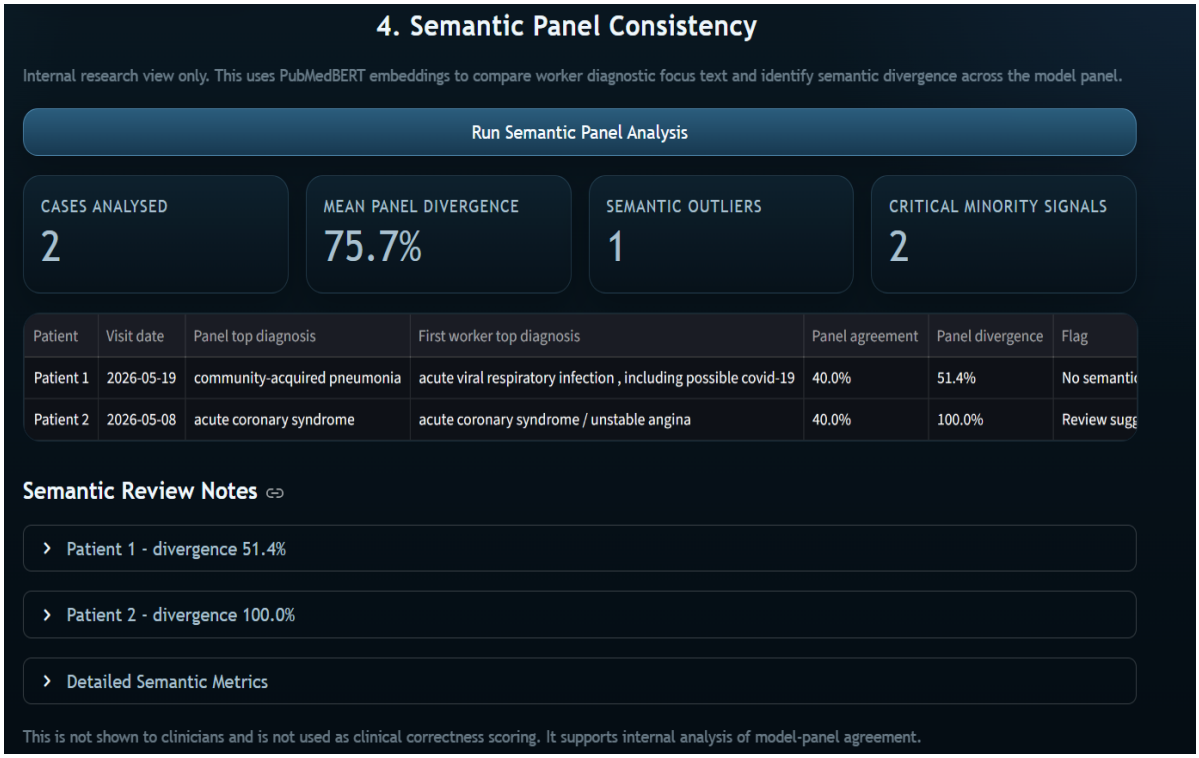


Figure 5.14.23 - System Analytics (5)

The Clinical Evaluation section shows a separate benchmark workflow used to test the system against predefined reference cases. Unlike the live consultation workflow, this section is not about managing an individual patient visit. Its purpose is to evaluate how well the model panel performs on cases where a reference diagnosis is already known.

The first screen shows the benchmark import area. The user can import template cases or upload a benchmark JSON file. This makes the evaluation workflow reusable. The system can be tested with the included cases, or with a new external benchmark set. The user can also choose how many benchmark models to run, which allows the same evaluation process to be repeated with different panel sizes.

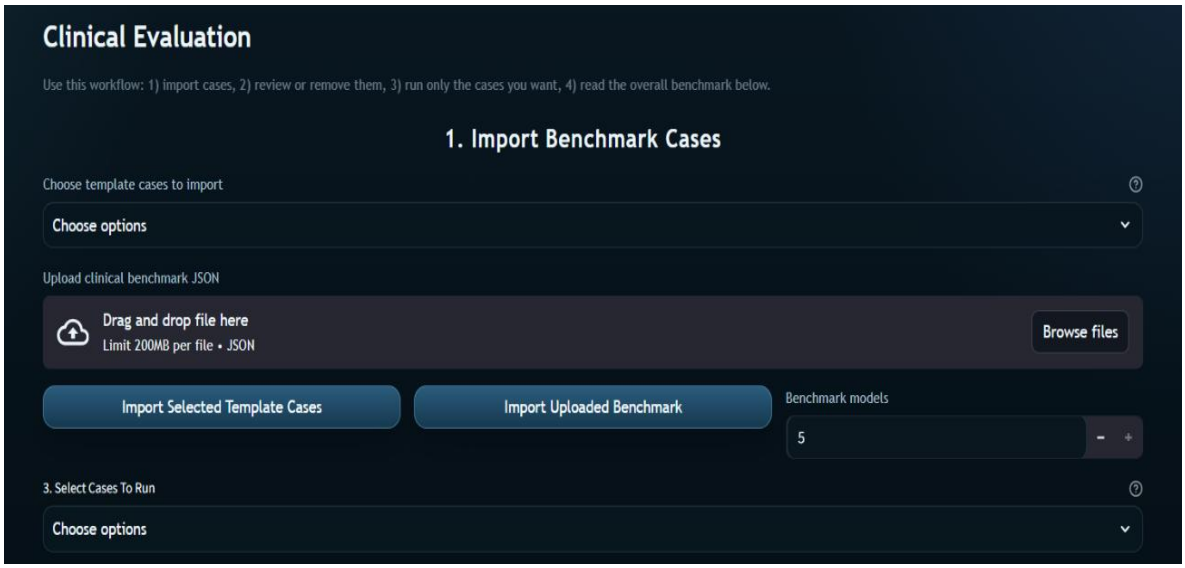


Figure 5.14.24 - System Analytics (6)

The Review Imported Cases table shows the benchmark cases currently loaded into the system. In this example, 10 cases have been imported. Each row lists the primary differential, the published case title, and the source type. This step is important because it lets the user inspect the benchmark set before execution and confirm that the imported cases are the intended reference cases.

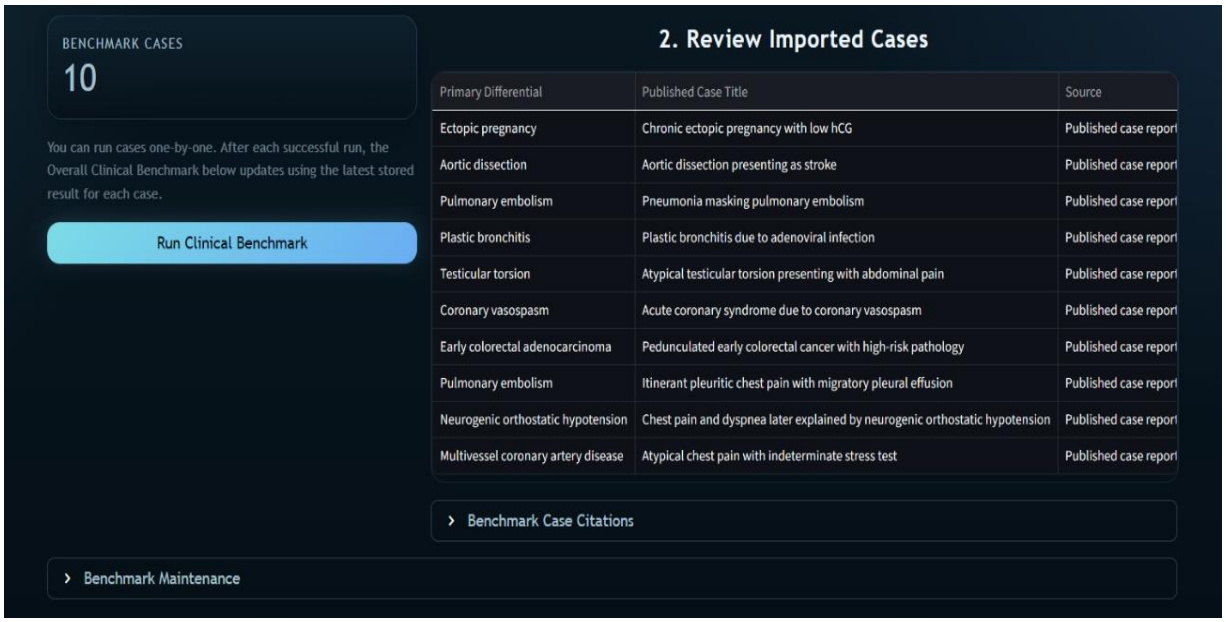


Figure 5.14.25 - System Analytics (7)

The Overall Clinical Benchmark screen summarizes performance across the full benchmark set. In this example, 10 cases are covered. The panel match score is 73.0%, which represents the overall diagnostic match between the panel output and the reference diagnoses, including partial or family-level matches. The panel Top-1 accuracy is 70.0%, meaning the panel's first diagnosis matched the reference diagnosis in 7 out of 10 cases. The panel Top-3 hit rate is 100.0%, meaning the correct reference diagnosis appeared somewhere in the panel's top three differential diagnoses for every benchmark case. The unsafe omission rate is 0.0%, meaning no critical reference diagnosis was completely missed. The panel-versus-best-single win rate is 10.0%, meaning the full panel outperformed the best individual model in only 1 out of 10 cases.

Figure 5.14.26 - System Analytics (8)

The Panel Case Results table gives the case-level breakdown behind the aggregate metrics. It compares the panel's predicted top diagnosis with the reference top diagnosis for each benchmark case. The Top-1 Match column shows whether the first prediction was an exact match. The Top-1 Family Match column gives partial credit when the predicted diagnosis belongs to the same diagnostic family as the reference. The Top-3 Hit column shows whether the correct diagnosis appeared anywhere in the top three. This is useful because a model may not rank the correct diagnosis first but may still include it in the differential.

Panel Case Results

Case	Predicted Top Dx	Reference Top Dx	Top-1 Match	Top-1 Family Match	Top-3 Hit
Pulmonary embolism	Pulmonary embolism	Pulmonary embolism	1	1	1
Neurogenic orthostatic hypotension	Esophageal structural or motility disorder (e.g., stricture, achalasia, malignancy)	Neurogenic orthostatic hypotension	0	0	1
Multivessel coronary artery disease	Coronary artery disease	Multivessel coronary artery disease	1	0.9	1
Testicular torsion	Testicular torsion	Testicular torsion	1	1	1
Coronary vasospasm	Coronary artery vasospasm / Prinzmetal angina with myocardial infarction	Coronary vasospasm	0	0.682	1
Early colorectal adenocarcinoma	High-risk T1 (pT1) colorectal adenocarcinoma in a malignant polyp	Early colorectal adenocarcinoma	1	0.9	1
Aortic dissection	Acute Stanford type A aortic dissection with stroke (cerebral malperfusion or emb	Acute aortic dissection	1	0.9	1
Pulmonary embolism	Pulmonary embolism with or without pulmonary infarction	Pulmonary embolism	1	0.9	1
Plastic bronchitis	Adenoviral pneumonia	Plastic bronchitis	0	0	1
Ectopic pregnancy	Ovarian ectopic pregnancy (primary ovarian implantation)	Chronic ectopic pregnancy	1	0.9	1

Figure 5.14.27 - System Analytics (9)

The Per-Model Summary compares the individual worker models across the same 10 benchmark cases. It reports each model's mean clinical match score, Top-1 accuracy, and unsafe omission rate. In this example, GPT-5.1 has the highest mean clinical match score and Top-1 accuracy, while several models have a small unsafe omission rate. This table helps determine whether the panel is adding value compared with individual models, or whether one model is consistently stronger.

Per-Model Summary

model	cases	mean_clinical_match_score	top1_accuracy	unsafe_omission_rate
GPT-5.1 (OpenAI)	10	0.7436	0.8	0
Llama-3.3-70B Versatile (Groq)	10	0.6479	0.7	0.1
Claude Sonnet 4.5 (Anthropic)	10	0.6278	0.6	0.1
Gemini 3 Pro Preview (Google)	10	0.6149	0.6	0
Qwen-3-32B (Groq)	10	0.5905	0.6	0.1

Figure 5.14.28 – System Analytics (10)

The Case-Level Benchmark Detail table compares the panel score against the best single-model score for each case. It lists the case ID, case title, panel score, best individual model, best individual score, and whether the panel beat the best single model. This is important because aggregate accuracy alone does not show where aggregation helps. In the screenshot, the panel beats the best single model in only one case, while in most cases the best individual model performs as well as or better than the panel.

Figure 5.14.29 - System Analytics (11)

Overall, the Clinical Evaluation workflow demonstrates that dIAgnosis includes an external evaluation mode separate from normal patient use. It allows benchmark cases to be imported, reviewed, executed, and scored against known references. The results show both aggregate performance and detailed case-level behavior, making it possible to evaluate whether the multi-model panel improves diagnostic coverage, reduces unsafe omissions, and adds value over single-model outputs.

Chapter 6 - Summary

6.1 Conclusions 73

6.2 Limitations 75

6.3 Medical Student Survey 76

6.4 Future Work 79

6.5 Closing Remarks 80

6.1 Conclusions

The central question driving this project was whether coordinating multiple large language models could produce a more reliable and trustworthy form of clinical decision support than relying on a single model alone. The short answer, based on what dIAgnosis demonstrated, is yes, but not in the way that initially seems most obvious.

The most striking benchmark result is also the one most worth examining carefully. The full five-model panel outperformed the single best individual model in only one out of ten benchmark cases, a 10% win rate. On the surface, that might seem to undermine the whole premise of the project. But the framing matters here. GPT-5.1, the top-performing individual model, happened to be both a worker and the aggregator in this implementation, which means the panel was partly competing against itself. More importantly, Top-1 accuracy is not the only relevant metric in a clinical context, and arguably not the most important one. The panel achieved a 100% Top-3 hit rate and a 0% unsafe omission rate across all ten cases. No critical reference diagnosis was completely

missed by the panel. That is a meaningful safety property that a single model, however strong, does not guarantee by design.

What multi-model collaboration offers is not primarily higher accuracy at the top position, it is broader coverage, visibility of disagreement, and the preservation of minority opinions. When the panel includes a model that flags a high-risk alternative diagnosis that the majority did not rank first, that signal is not lost. It is visible in the worker drafts, surfaced in the internal QA metrics, and available for the clinician to act on. A single model response cannot offer that. The value of dIAgnosis lies less in whether the top diagnosis is always correct and more in whether the overall output creates a safer, more inspectable reasoning environment for the clinician using it.

The internal evaluation metrics reinforced this picture. Worker agreement across the stored consultations averaged 56.7%, which means the panel did not simply converge on a single shared answer. That level of divergence is not a failure, it is exactly the kind of diagnostic plurality that a multi-model system is supposed to surface. The aggregator's job is then to synthesise those perspectives into a structured output, not to pick the most popular one. Whether the aggregation step is doing this well in every case is harder to verify with the current benchmark, and that is a legitimate limitation.

Beyond the diagnostic performance results, the project demonstrates that a fully working multi-model clinical decision support platform can be built, evaluated, and refined by a single developer within the scope of an undergraduate thesis. dIAgnosis integrates a five-model asynchronous reasoning pipeline, a structured aggregation stage, longitudinal patient tracking, PubMed evidence retrieval, internal QA metrics, semantic divergence analysis using PubMedBERT, bedside calculators, debate rounds, and a clinical benchmark system. The fact that all of these components work together in a coherent clinical workflow is, in itself, a meaningful contribution.

The role dIAgnosis is designed for is as a second opinion tool, not an autonomous diagnostic system. Its outputs are structured to be reviewed, not acted on blindly. The red

flags, certainty statements, worker draft visibility, and minority opinion tracking are all deliberate design choices oriented toward supporting clinician judgement rather than replacing it. That framing should not change even as the system improves.

6.2 Limitations

Every limitation worth discussing here is one that emerged from actually building and testing the system, not simply a pre-declared constraint.

The benchmark set of ten cases is the most significant evaluation constraint. Ten cases is enough to illustrate how the system behaves, but nowhere near enough to draw reliable conclusions about diagnostic performance across different specialties, patient populations, or levels of case complexity. The cases were clinician-informed, but they were not drawn from a validated benchmark dataset, and they skewed toward relatively well-defined presentations. How the system handles rare conditions, ambiguous histories, or cases with deliberately incomplete information remains untested.

The internal QA metrics are useful for flagging consultations that may deserve a second look, but they are not clinical correctness measures. A high report support score does not mean the diagnosis is right. A low worker agreement score does not mean the consensus is wrong. These metrics are tools for directing attention, not for validating medical conclusions, and they should be presented to users in a way that makes that distinction clear.

The PubMed evidence retrieval module connects the generated report to relevant biomedical literature, but it does not verify the diagnosis. A retrieved guideline or systematic review is supporting context for a clinician to engage with, not confirmation that the model's reasoning was sound. The current ranking algorithm weights article type and recency but does not assess relevance at the sentence or claim level, which limits how useful the retrieved sources actually are in practice.

Finally, the system has not been tested with licensed clinicians in real clinical workflows. The feedback available comes from a medical student survey, which is a useful early signal but not a substitute for evaluation by experienced practitioners. How a consultant physician, a junior doctor under time pressure, or a specialist in a narrow field would use and interpret the output is still an open question.

6.3 Medical Student Survey

A small survey was conducted with medical students and early-career medical trainees to gather feedback on the perceived usefulness, trust, and safety concerns surrounding dIAgnosis. The survey received 12 responses. The participants included students from different stages of medical training: 5 respondents were in their 1st or 2nd year, 1 was in 3rd year, 1 was in 4th year, 4 were in their 5th or 6th clinical years, and 1 was in internship or residency.

The responses showed that the participants were already familiar with general-purpose AI tools. Nine out of twelve respondents reported using tools such as ChatGPT, Claude, or Gemini daily, while two used them a few times per week. This suggests that the target educational audience is already exposed to AI-assisted reasoning, making trust, safety, and correct integration into clinical learning especially important.

Overall, respondents were comfortable adopting new digital tools, with an average score of 4.08 out of 5. However, trust in AI for real clinical decisions was more cautious, with an average score of 3.42 out of 5. This difference is important: students may be open to using AI tools, but they do not automatically trust them in high-stakes clinical contexts.

The screenshots and interface were generally rated as clear and easy to read, with an average score of 4.33 out of 5. The leading diagnosis in the example case was rated as moderately plausible, with an average score of 3.58 out of 5. When asked what they would do based on the report, half of the respondents stated that they could not judge from the

report alone, while four would order the suggested investigations and two would escalate to a senior immediately. This shows that respondents did not treat the system output as a final answer, but as something requiring further clinical verification.

The multi-model design was viewed positively. Compared with using a single AI model, 4 respondents said the multi-model approach would make them trust the answer significantly more, and 5 said it would make them trust it slightly more. Only 3 felt about the same. The agreement score was also considered useful: 8 out of 12 respondents said they would become more cautious when model agreement was lower.

Safety-related features were also valued. Eleven out of twelve respondents said that a made-up-information risk score would make them more careful with the report. The minority warning feature, where one model flags a serious alternative diagnosis even if the majority supports another diagnosis, was also seen as useful: 3 respondents called it the most important feature, 4 called it important, and 5 said it was useful but not essential.

The main concerns were consistent with the risks discussed earlier in the thesis. The most common worry was AI confidently giving wrong information that sounds correct, selected by 10 out of 12 respondents. Other major concerns included doctors becoming worse at reasoning because of AI reliance, selected by 8 respondents, and junior doctors or students treating AI suggestions as correct without questioning them, selected by 7 respondents.

Respondents also identified conditions that would increase their trust in dIAgnosis. The most common requirement was checking the system against published clinical guidelines, selected by 9 out of 12 respondents. Eight respondents wanted published research showing diagnostic performance, six wanted official audit or certification, and six wanted institutional backing from a hospital or department. Free-text responses also emphasized trusted sources, guideline references, clearer reasoning, and real-world statistics.

The average likelihood of recommending dIAgnosis to a medical peer was 6.33 out of 10. This suggests moderate interest, but not unconditional acceptance. The survey therefore supports the positioning of dIAgnosis as an educational and decision-support prototype rather than a tool ready for unsupervised clinical use.

Overall, the survey suggests that medical students see value in the multi-model structure, visible agreement scores, minority warnings, follow-up context, and report transparency. At the same time, their responses show clear caution around hallucination, overreliance, accountability, and lack of formal validation. These findings reinforce the need for clinician oversight, guideline grounding, stronger evidence integration, and larger-scale evaluation before such a system could be considered for real clinical deployment.

In the following table, the key findings from the survey are presented.

Survey Item	**Result**
Number of respondents	12
Daily AI tool use	9/12
Average comfort with digital tools	4.08/5
Average trust in AI for real clinical decisions	3.42/5
Average interface clarity	4.33/5
Average recommendation score	6.33/10
Multi-model approach increased trust	9/12
Hallucination was a concern	10/12
Guideline validation required for trust	9/12

Table 6.3.1 – Medical Student Survey Key Findings

6.4 Future Work

Several directions could improve dIAgnosis beyond the current prototype.

- The system should be evaluated with licensed clinicians using a larger and more diverse set of clinical cases. The current benchmark and survey provide early feedback, but they are not enough to establish clinical reliability. Future studies should include real clinicians, more specialties, rare conditions, emergency cases, and cases with incomplete or ambiguous information.

- The benchmark workflow could be expanded. A larger benchmark set would make it possible to compare panel performance more reliably against individual models. Future versions could include more detailed reference standards, severity labels, gold-standard investigations, treatment plans, and specialty-specific case groups.

- The evidence retrieval system could be improved. The current PubMed lookup retrieves supporting literature related to the leading diagnosis, but future work could provide stronger guideline grounding, inline citations inside the generated report, and automatic linking between each recommendation and its supporting source. Integration with recognised clinical guidelines such as NICE, ESC, IDSA, or local hospital protocols would increase trust.

- The system could improve how it handles uncertainty. Future versions could provide clearer calibration of certainty scores, explain why models disagree, and distinguish between diagnostic uncertainty, missing data, and safety-driven caution. This would help clinicians understand not only what the system suggests, but how confident the panel is and why.

- The multi-model pipeline could be made more adaptive. Instead of always using the same fixed panel, the system could select models based on case type, specialty, urgency, or previous model performance. It could also use specialist agents, such as cardiology, infectious disease, emergency medicine, or radiology-focused agents, before aggregation.

- The platform could be integrated with electronic health record systems using standards such as FHIR. This would reduce manual data entry and allow the system to use richer longitudinal patient data. However, this would also require stronger privacy, security, and governance controls.

- Future work should include stronger deployment and safety controls. This includes multi-user support, audit logs, role-based access, privacy-preserving data handling, cost and latency monitoring, and formal clinical safety evaluation. These steps would be necessary before any real-world clinical deployment could be considered.

- The use of fine-tuned medical language models. The current implementation relies on general-purpose LLMs accessed through external APIs. Future versions could incorporate models fine-tuned on medical datasets, clinical guidelines, diagnostic reasoning tasks, or structured case reports. Such models could improve the quality of differential diagnosis generation, reduce clinically irrelevant outputs, and produce reasoning that is better aligned with medical terminology and practice. Fine-tuned medical models could therefore significantly improve both the accuracy and consistency of the platform, especially when combined with the existing multi-model consensus and aggregation architecture.

6.5 Closing Remarks

Medicine has always dealt with uncertainty by seeking more than one opinion. A difficult case gets referred, a second consultant is called, a multidisciplinary team sits down together. The underlying logic is not that any single clinician is unreliable, but rather that consequential decisions deserve more than one perspective, and that disagreement between experts is itself useful information. The reasoning behind dIAgnosis is the same. A conclusion reached by multiple independent models, with the disagreements kept visible and the minority opinions preserved, is structurally more trustworthy than one that is not.

That principle is straightforward. Making it work in practice is less so. The pipeline must be fast enough to be usable, the aggregation must be genuinely synthetic rather than just picking the most confident worker, the output must be structured enough for a clinician to navigate under pressure, and the uncertainty must be communicated in a way that informs rather than overwhelms. Every design decision in dIAgnosis was shaped by those constraints, and most of the limitations identified in this chapter trace back to places where the balance has not yet been fully found.

What this project establishes is that the architecture is feasible, the implementation is coherent, and the early signals from medical students suggest the core design decisions are pointing in the right direction. Agreement scores, minority warnings, worker draft transparency, and longitudinal context are not features added for completeness, they are the parts of the system that carry the most clinical meaning, and the survey responses suggest that even early-stage users recognise that.

There is real work still ahead. Validation with licensed clinicians, larger benchmarks, stronger evidence grounding, and proper deployment infrastructure are all necessary before a system like this could be considered for any real clinical setting. But the direction is the right one. The future of clinical AI is probably not a single model that gets good enough to trust on its own, it is systems designed from the ground up to make their reasoning visible, their uncertainty explicit, and the clinician the final authority. dIAgnosis is one step toward that.

Bibliography

[1] U. Mumtaz, A. Ahmed, and S. Mumtaz, “LLMs-Healthcare : Current Applications and Challenges of Large Language Models in various Medical Specialties,” Feb. 26, 2024, *arXiv*: arXiv:2311.12882. doi: 10.48550/arXiv.2311.12882.

[2] G. Perković, A. Drobnjak, and I. Botički, “Hallucinations in LLMs: Understanding and Addressing Challenges,” in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, May 2024, pp. 2084–2088. doi: 10.1109/MIPRO60963.2024.10569238.

[3] C. U. Ogdu, S. Gurbuz, M. Karakose, and E. Hanoglu, “Medical Implications of LLM Based Clinical Decision Support Systems in Healthcare,” in *2025 29th International Conference on Information Technology (IT)*, Feb. 2025, pp. 1–4. doi: 10.1109/IT64745.2025.10930252.

[4] S. Gottlieb and L. Silvis, “How to Safely Integrate Large Language Models Into Health Care,” *JAMA Health Forum*, vol. 4, no. 9, p. e233909, Sep. 2023, doi: 10.1001/jamahealthforum.2023.3909.

[5] S. Feng *et al.*, “When One LLM Drools, Multi-LLM Collaboration Rules,” Feb. 06, 2025, *arXiv*: arXiv:2502.04506. doi: 10.48550/arXiv.2502.04506.

[6] H. Yang, M. Li, H. Zhou, Y. Xiao, Q. Fang, and R. Zhang, “One LLM is not Enough: Harnessing the Power of Ensemble Learning for Medical Question Answering,” *medRxiv*, p. 2023.12.21.23300380, Dec. 2023, doi: 10.1101/2023.12.21.23300380.

[7] A. Estornell and Y. Liu, “Multi-LLM Debate: Framework, Principals, and Interventions”.

[8] F. Liu *et al.*, “Application of large language models in medicine,” *Nat Rev Bioeng*, vol. 3, no. 6, pp. 445–464, Jun. 2025, doi: 10.1038/s44222-025-00279-5.

[9] E. Yu *et al.*, “Large Language Models in Medicine: Applications, Challenges, and Future Directions,” *International Journal of Medical Sciences*, vol. 22, no. 11, pp. 2792–2801, May 2025, doi: 10.7150/ijms.111780.

[10] R. Booth and R. B. U. technology editor, “AI outperforms doctors in Harvard trial of emergency triage diagnoses,” *The Guardian*, Apr. 30, 2026. Accessed: May 02, 2026.[Online]. Available: https://www.theguardian.com/technology/2026/apr/30/ai-outperforms-doctors-in-harvard-trial-of-emergency-triage-diagnoses

[11] U. Nations, “New WHO/Europe report provides first-ever snapshot of AI in health care across European Union Member States,” United Nations Western Europe. Accessed: May 02, 2026. [Online]. Available: https://unric.org/en/new-who-europe-report-provides-first-ever-snapshot-of-ai-in-health-care-across-european-union-member-states/

[12] Y. Artsi *et al.*, “Challenges of Implementing LLMs in Clinical Practice: Perspectives,” *J Clin Med*, vol. 14, no. 17, p. 6169, Sep. 2025, doi: 10.3390/jcm14176169.

[13] J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou, “Mixture-of-Agents Enhances Large Language Model Capabilities,” Jun. 07, 2024, *arXiv*: arXiv:2406.04692. doi: 10.48550/arXiv.2406.04692.

[14] A. Wright and D. F. Sittig, “A four-phase model of the evolution of clinical decision support architectures,” *International Journal of Medical Informatics*, vol. 77, no. 10, pp. 641–649, Oct. 2008, doi: 10.1016/j.ijmedinf.2008.01.004.

[15] “Clinical Decision Support System Vendors by Market Share.” Accessed: May 03, 2026. [Online]. Available: https://www.definitivehc.com/resources/healthcare-insights/top-clinical-decision-support-system-vendors

[16] “Epic EHR Software: Pricing, Features, Demo & Comparison.” Accessed: May 03, 2026. [Online]. Available: https://www.ehrinpractice.com/epic-ehr-software-profile-119.html

[17] “Oracle Health. Advancing how health happens.” Accessed: May 03, 2026. [Online]. Available: https://www.oracle.com/health/clinical-suite/electronic-health-record/

[18] “Isabel DDx Companion | Isabel Healthcare.” Accessed: May 03, 2026. [Online]. Available: https://www.isabelhealthcare.com

[19] “AI in UpToDate: New Generative Solutions for Medical Professionals.” Accessed: May 03, 2026. [Online]. Available: https://www.wolterskluwer.com/en/solutions/uptodate/ai-clinical-decision-support

[20] “Glass health – Center for Digital Innovation and AI.” Accessed: May 03, 2026. [Online]. Available: https://cinia.blogs.usj.edu.lb/en/ai-educational-tool/glass-health/

[21] S. Team, “How OpenEvidence AI is transforming clinical decision-making,” Sermo. Accessed: May 03, 2026. [Online]. Available: https://www.sermo.com/resources/openevidence-ai/

[22] “How multi-AI agents can improve clinical decision support | Healthcare IT News.” Accessed: May 03, 2026. [Online]. Available: https://www.healthcareitnews.com/news/how-multi-ai-agents-can-improve-clinical-decision-support

[23] X. Chen *et al.*, “Enhancing diagnostic capability with multi-agents conversational large language models,” *NPJ Digit Med*, vol. 8, p. 159, Mar. 2025, doi: 10.1038/s41746-025-01550-0.

[24] Y. Kim *et al.*, “MDAgents: An Adaptive Collaboration of LLMs for Medical Decision-Making,” Oct. 30, 2024, *arXiv*: arXiv:2404.15155. doi: 10.48550/arXiv.2404.15155.

[25] “Documentation for Visual Studio Code.” Accessed: May 09, 2026. [Online]. Available: https://code.visualstudio.com/docs

[26] “General Python FAQ,” Python documentation. Accessed: May 07, 2026. [Online]. Available: https://docs.python.org/3/faq/general.html

[27] “venv — Creation of virtual environments,” Python documentation. Accessed: May 07, 2026. [Online]. Available: https://docs.python.org/3/library/venv.html

[28] K. Kerr, “What is Git? Our beginner’s guide to version control,” The GitHub Blog. Accessed: May 07, 2026. [Online]. Available: https://github.blog/developer-skills/programming-languages-and-frameworks/what-is-git-our-beginners-guide-to-version-control/

[29] “FAQ - Streamlit Docs.” Accessed: May 07, 2026. [Online]. Available: https://docs.streamlit.io/

[30] “pandas documentation — pandas 3.0.2 documentation.” Accessed: May 07, 2026. [Online]. Available: https://pandas.pydata.org/docs/

[31] “Plotly.” Accessed: May 07, 2026. [Online]. Available: https://plotly.com/python/

[32] v2.8.7, “fpdf2.” Accessed: May 07, 2026. [Online]. Available: https://py-pdf.github.io/fpdf2/index.html

[33] “FastAPI.” Accessed: May 07, 2026. [Online]. Available: https://fastapi.tiangolo.com/

[34] “Uvicorn.” Accessed: May 07, 2026. [Online]. Available: https://uvicorn.dev/

[35] “Welcome to Pydantic,” Pydantic Docs. Accessed: May 07, 2026. [Online]. Available: https://pydantic.dev/docs/validation/latest/get-started/

[36] “HTTPX.” Accessed: May 07, 2026. [Online]. Available: https://www.python-httpx.org/

[37] “SQLite Home Page.” Accessed: May 07, 2026. [Online]. Available: https://sqlite.org/

[38] “SQLAlchemy.” Accessed: May 07, 2026. [Online]. Available: https://www.sqlalchemy.org

[39] *aiosqlite: asyncio bridge to the standard sqlite3 module*. Accessed: May 07, 2026.

[40] "OpenAI API Platform Documentation." Accessed: May 07, 2026. [Online]. Available: https://developers.openai.com/api/docs

[41] "Groq is fast, low cost inference.," Groq. Accessed: May 07, 2026. [Online]. Available: https://groq.com/

[42] "API Overview," Claude API Docs. Accessed: May 07, 2026. [Online]. Available: https://platform.claude.com/docs/en/api/overview

[43] "Gemini API reference | Google AI for Developers." Accessed: May 07, 2026. [Online]. Available: https://ai.google.dev/api

[44] "SentenceTransformers Documentation — Sentence Transformers documentation." Accessed: May 07, 2026. [Online]. Available: https://sbert.net/

[45] "PyTorch documentation — PyTorch 2.11 documentation." Accessed: May 07, 2026. [Online]. Available: https://docs.pytorch.org/docs/stable/index.html

[46] *Entrez® Programming Utilities Help*. National Center for Biotechnology Information (US), 2010. Accessed: May 07, 2026.

[47] *python-dotenv: Read key-value pairs from a .env file and set them as environment variables*. Python. Accessed: May 07, 2026.

[48] "Secrets management - Streamlit Docs." Accessed: May 07, 2026. [Online]. Available: https://docs.streamlit.io/

[49] “Middleware - FastAPI.” Accessed: May 07, 2026. [Online]. Available: https://fastapi.tiangolo.com/tutorial/middleware/

Appendix A - Medical Student Survey

A.1 Medical Student Survey Questions

Section A — About You
Three quick questions about your background.

Q1. What stage of medical training are you currently in? *

- ◯ 1st or 2nd year (pre-clinical)
- ◯ 3rd year (early clinical)
- ◯ 4th year
- ◯ 5th or 6th year (final clinical years)
- ◯ Internship / residency (PGY-1 or later)
- ◯ Postgraduate / specialty training

Q2. How often do you use general-purpose AI tools (ChatGPT, Claude, Gemini, etc.) when studying or thinking through clinical cases? *

- ◯ Daily
- ◯ A few times a week
- ◯ Occasionally (a few times a month)
- ◯ Rarely
- ◯ Never

Q3. How comfortable are you generally with adopting new digital tools in your studies or clinical work? *

	1	2	3	4	5	
Not comfortable at all	◯	◯	◯	◯	◯	Very comfortable

Section B — Your Current Trust in AI for Medicine

Q4. How much do you currently trust AI answers (e.g., from ChatGPT) when a real clinical decision is at stake? *

	1	2	3	4	5	
Do not trust them at all	○	○	○	○	○	Trust them as much as a competent colleague

Q5. Which of these worries about using AI in medicine feel real to you? *

Tick all that apply.

- [ ] The AI confidently giving wrong information that sounds correct
- [ ] It being unclear who is responsible if the AI causes a mistake
- [ ] The AI working less accurately for certain groups of patients (e.g., women, ethnic minorities, rare diseases)
- [ ] Doctors becoming worse at reasoning on their own because they rely on AI
- [ ] Junior doctors and students treating AI suggestions as the right answer without questioning
- [ ] Patient information being sent to outside companies running the AI
- [ ] The lack of official rules or certification for these tools
- [ ] AI being available only in well-funded hospitals
- [ ] I do not have major worries about this

Q6. If an AI tool contributes to a diagnostic error that harms a patient, who do you think should be primarily accountable? *

- ○ The doctor who used the tool
- ○ The hospital that allowed it to be used
- ○ The company that built the AI
- ○ The regulator that approved the AI
- ○ Responsibility shared, agreed in advance
- ○ Not sure

Section C — Reading a dIAgnosis Report

Read the case and the dIAgnosis output below before answering the next 5 questions.
Test Case

https://jmedicalcasereports.biomedcentral.com/articles/10.1186/s13256-021-02666-z

Patient / History

- 54-year-old male
- History of tobacco use
- History of GERD
- BMI 29
- Family history: father with CAD/MI and CABG in his 50s

Presenting Symptoms

- Sudden substernal chest pain
- Radiation to left arm and jaw
- Pain intensity 7/10
- Episode lasted about 10 minutes
- Pain resolved spontaneously before treatment
- 3-week history of intermittent exertional chest pain
- Exertional pain relieved by rest
- No dyspnea
- No diaphoresis
- No nausea or vomiting
- No fever, cough, abdominal, urinary, or bowel symptoms

Vitals / Exam

- Blood pressure: 139/85 mmHg
- Heart rate: 81 bpm
- Hemodynamically stable
- Physical exam reported as unremarkable

Initial Tests

- ECG: no ischemic changes
- Serial troponins x3: negative, < 0.010 ng/mL
- CBC: normal
- CMP: normal
- TSH: normal

Cardiovascular Risk Data

- Total cholesterol: 235 mg/dL
- Triglycerides: 408 mg/dL
- HDL: 26 mg/dL
- LDL: not calculable because triglycerides were high

Further Cardiac Testing

- Echocardiogram: normal LV systolic function
- No wall motion abnormalities
- Exercise treadmill ECG stress test:
  - 95% of age-predicted maximal heart rate achieved
  - 10 METs
  - interpreted as intermediate probability of ischemia / indeterminate

Doctor Assessment / Clinical Reasoning

- Chest pain remained concerning for coronary ischemia despite negative troponins
- Symptom pattern was clinically important:
  - exertional
  - relieved by rest
  - radiation to arm/jaw
  - worsening pattern over several weeks
- Stress test was not reassuring enough to stop workup
- Cardiology felt further anatomical evaluation was needed
- CCTA was chosen because the treadmill ECG result was indeterminate

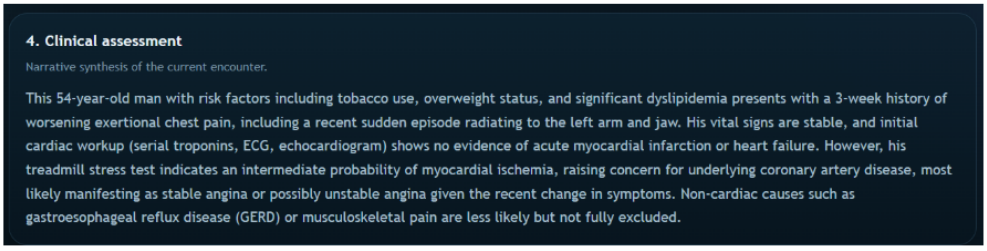
Definitive Findings

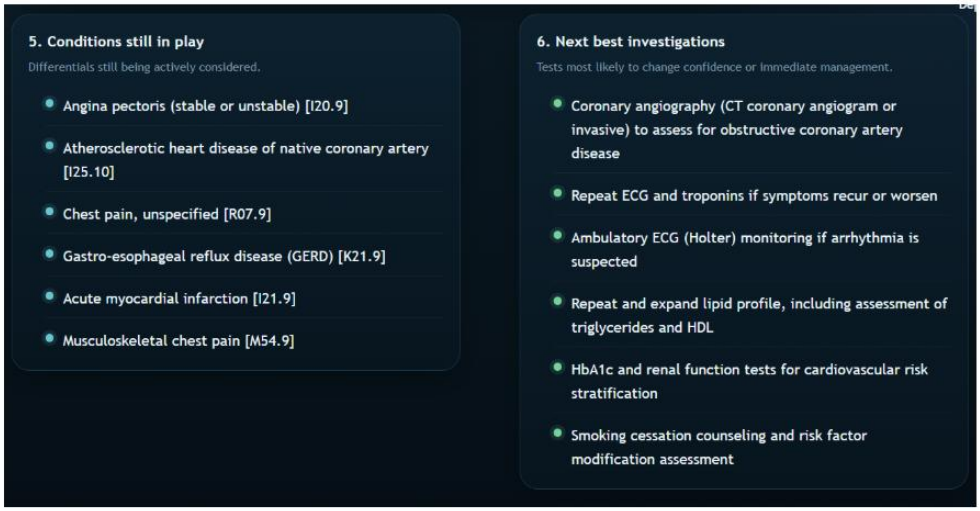
- Coronary CT angiography showed significant LAD disease
- Cardiac catheterization confirmed severe multivessel CAD

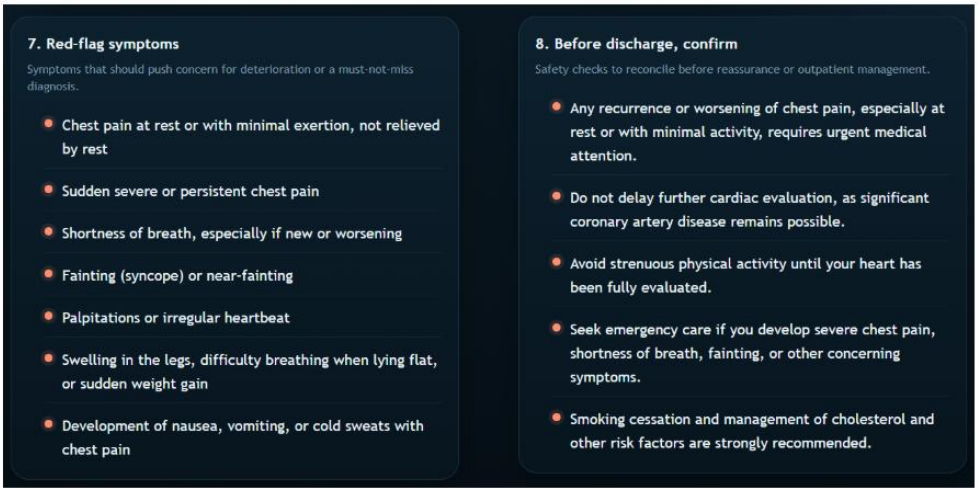
Final Outcome

- Diagnosed with multivessel coronary artery disease
- Required CABG

Screenshots from dIAgnosis Assesment







Clarification:
The AI tool correctly identifies angina and CAD as top diagnoses, but it doesn't reach the final conclusion of the case report "*multivessel CAD requiring CABG* " because that required invasive angiography data that wasn't yet available at the point the AI was consulted. This is expected and appropriate, the AI is reasoning from pre-angiography data, just as the clinicians were at that stage.

Q7. How clear and easy it is to read the screenshots? *

	1	2	3	4	5	
Confusing	○	○	○	○	○	Very clear

Q8. How clinically plausible does the leading diagnosis seem given the case? *

	1	2	3	4	5	
Not plausible	○	○	○	○	○	Highly plausible

Q9. If you were the doctor in charge of this patient, what would you do next based on this report? *

- ○ Order the investigations the tool suggests as the first step
- ○ Order different investigations than the ones the tool suggests
- ○ Escalate to a senior immediately, regardless of the report
- ○ Ignore the report and do my own work-up from scratch
- ○ I cannot judge from this report alone

Q10. How much would you actually rely on this output to inform a real decision? *

	1	2	3	4	5	
Would ignore it	○	○	○	○	○	Would weight it as heavily as a colleague's opinion

Q11. What is missing from the report above that you would want to see before relying on it?

E.g., the AI's reasoning, references to guidelines, contraindications, why other diagnoses were rejected, etc.

Your answer

Section D — How dIAgnosis Features Would Change Your Behavior

Each of the next 5 questions describes one feature of dIAgnosis in plain language and asks what you would do. Answer based on your honest reaction — there are no expected answers.

Q12. dIAgnosis asks five different AIs the same medical question instead of just one, and shows where they agreed or disagreed. Compared to using a single AI like ChatGPT alone, would this make you: *

- ◯ Trust the answer significantly more
- ◯ Trust it slightly more
- ◯ Feel about the same
- ◯ Trust it less (more places it could go wrong, not fewer)
- ◯ I would not trust either approach

Q13. Imagine the report says "5 of 5 AIs agreed" on a diagnosis in one case, and "2 of 5 agreed" in another case. Would seeing this number change how cautious you are about the diagnosis? *

- ◯ Yes, strongly — I would be much more cautious with the lower number
- ◯ Yes, somewhat
- ◯ Not really
- ◯ No — I would verify either result the same way
- ◯ No — I would not believe the number itself

Q14. The tool also shows a "made-up-information risk", a number from 0–100% showing how likely it estimates the AI may have invented part of its answer. If you saw this number on a real report, would it: *

- ◯ Make me much more careful with the report
- ◯ Make me somewhat more careful
- ◯ Not really change anything
- ◯ I would not believe the number itself

Q15. Imagine 4 of the 5 AIs say "pneumonia" but 1 raises "pulmonary embolism" , and the system flags this as a warning so you don't miss the rare-but-serious option. How valuable is this kind of warning? *

- ◯ The most important feature of the tool
- ◯ Important
- ◯ Useful but not essential
- ◯ Probably causes more anxiety than it adds insight
- ◯ Not useful

Q16. On a follow-up visit, the tool sends the AI both today's case and the previous visit, plus how many days have passed, and asks the AI to focus on what has changed. How would this affect your view of the tool? *

- ◯ Strongly more useful than a tool without this
- ◯ Somewhat more useful
- ◯ About the same — I prefer assessing each visit on its own
- ◯ I would not trust an AI to compare two visits reliably

Section E — Use Intention & Conditions for Trust

Q17. If dIAgnosis were freely available tomorrow, how often would you actually use it? *

- ◯ Daily during clinical rotations
- ◯ A few times a week
- ◯ Only for unfamiliar or complex cases
- ◯ As a study or learning aid, not on real patients
- ◯ I would not use it

Q18. What would have to be true for you to trust dIAgnosis on a real patient? *
Tick all that apply.

- ☐ It is checked against published clinical guidelines (NICE, ESC, IDSA, etc.)
- ☐ It is officially audited or certified by a regulator
- ☐ There is published research showing how often it gets diagnoses right
- ☐ My hospital or department uses it and stands behind it
- ☐ It plugs into the medical record system I already use
- ☐ A senior doctor I trust recommends it
- ☐ The AI cites sources for every claim it makes
- ☐ A doctor must always sign off before any action is taken on its output
- ☐ I cannot imagine trusting it on real patients under any conditions

Q19. Imagine you used dIAgnosis on a case and disagreed with its leading diagnosis. How likely would you be to override the tool and follow your own clinical judgment? *

	1	2	3	4	5	
Defer to the tool	◯	◯	◯	◯	◯	Always follow my own judgment

Section F — Recommendation & Criticism

Q20. On a scale of 0 to 10, how likely are you to recommend dIAgnosis to a peer studying medicine? *

	0	1	2	3	4	5	6	7	8	9	10	
Not at all likely	○	○	○	○	○	○	○	○	○	○	○	Extremely likely

Q21. What is the single biggest worry preventing you from using a tool like this on real patients?

Your answer

Q22. What is one feature that, if added, would make you significantly more likely to use this?

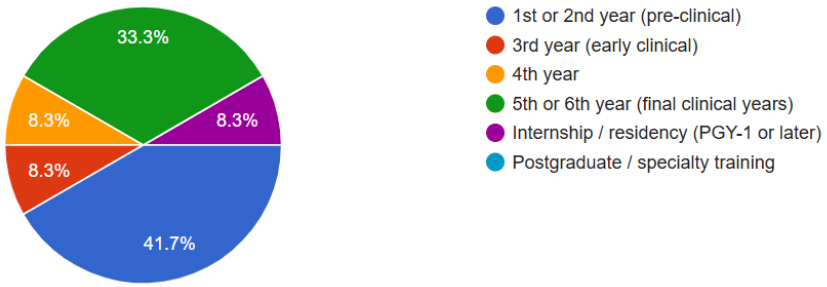
Your answer

A.2 Medical Student Survey Answers

Q1. What stage of medical training are you currently in?

Copy chart

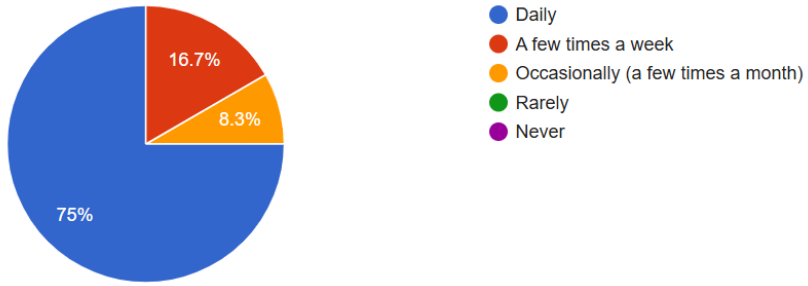
12 responses



Q2. How often do you use general-purpose AI tools (ChatGPT, Claude, Gemini, etc.) when studying or thinking through clinical cases?

Copy chart

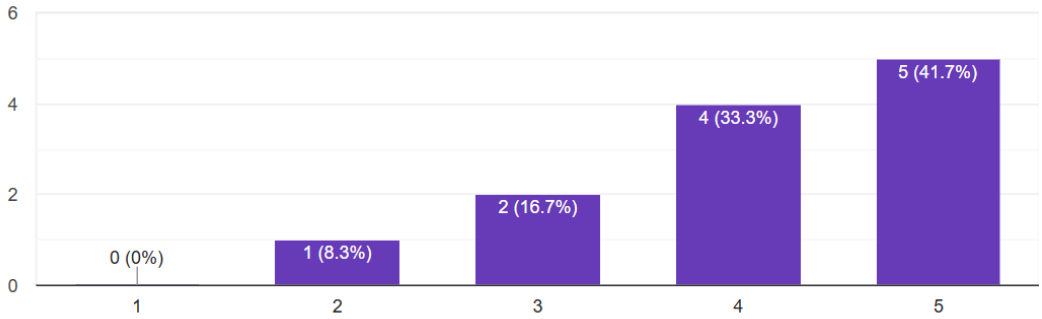
12 responses



Q3. How comfortable are you generally with adopting new digital tools in your studies or clinical work?

Copy chart

12 responses

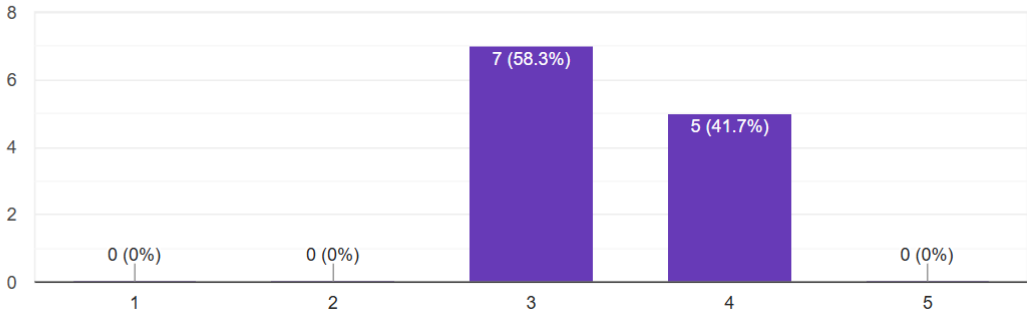


Section B — Your Current Trust in AI for Medicine

Q4. How much do you currently trust AI answers (e.g., from ChatGPT) when a real clinical decision is at stake?

Copy chart

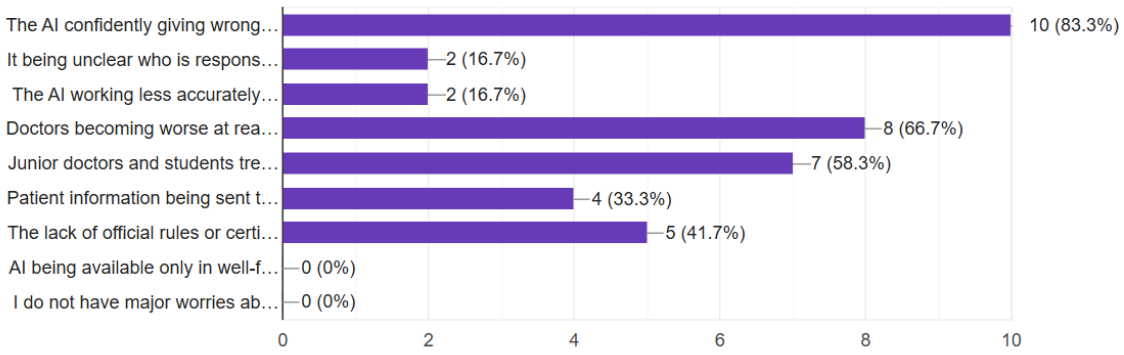
12 responses



Q5. Which of these worries about using AI in medicine feel real to you?

Copy chart

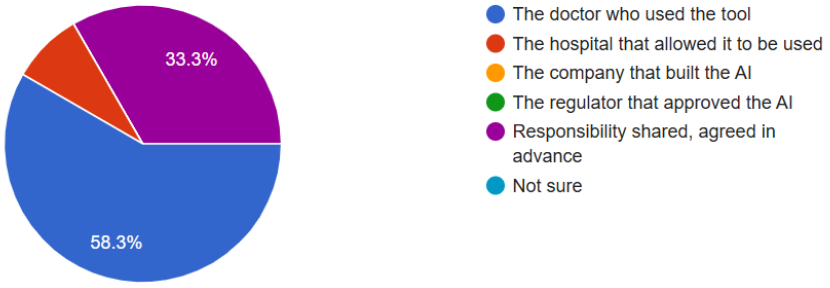
12 responses



Q6. If an AI tool contributes to a diagnostic error that harms a patient, who do you think should be primarily accountable?

Copy chart

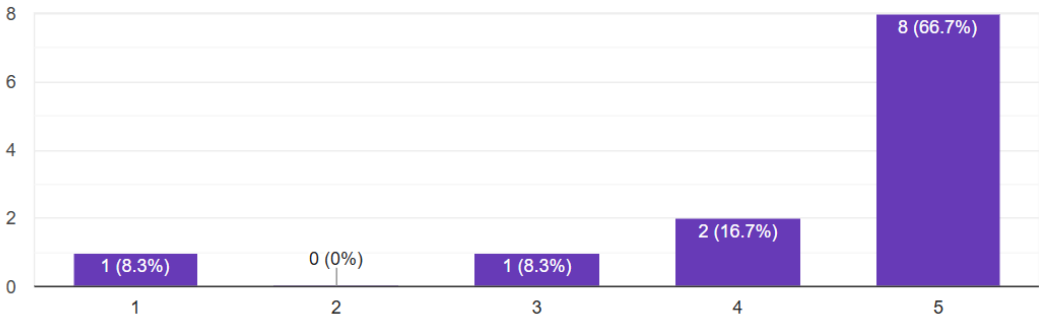
12 responses



Q7. How clear and easy it is to read the screenshots?

Copy chart

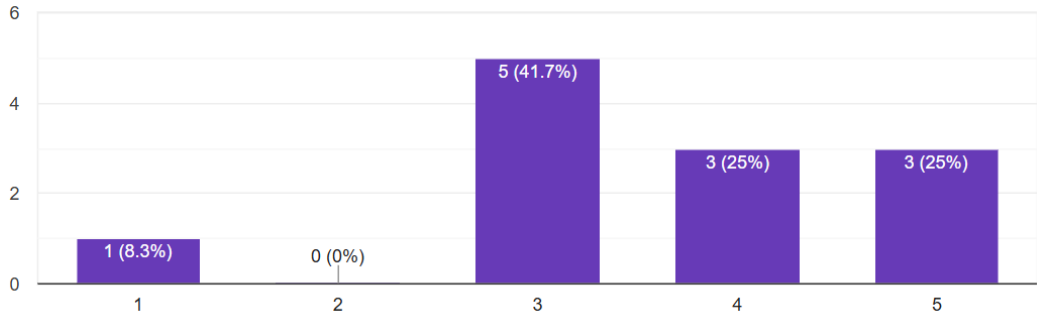
12 responses



Q8. How clinically plausible does the leading diagnosis seem given the case?

Copy chart

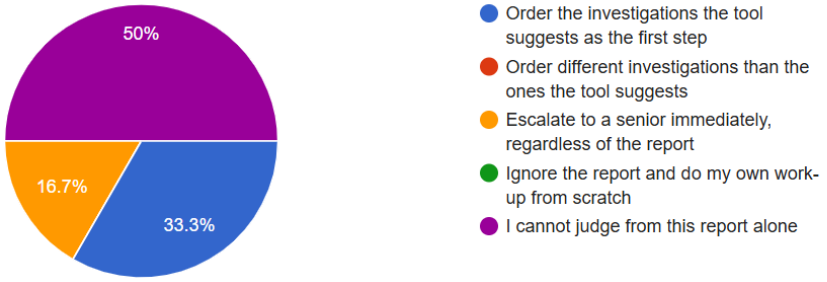
12 responses



Q9. If you were the doctor in charge of this patient, what would you do next based on this report?

Copy chart

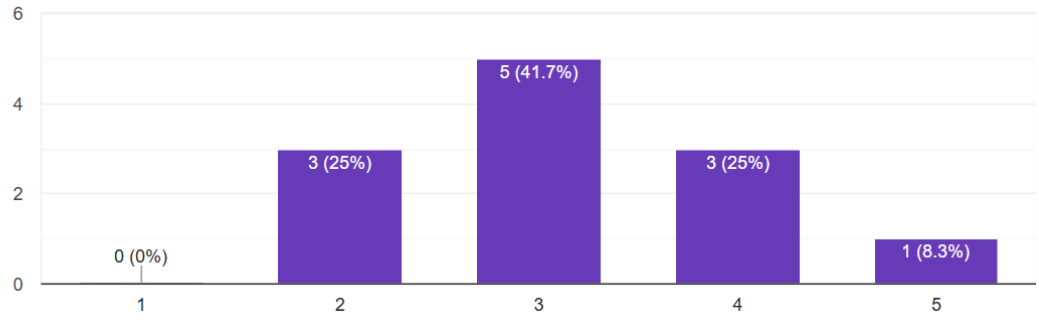
12 responses



Q10. How much would you actually rely on this output to inform a real decision?

Copy chart

12 responses



Q11. What is missing from the report above that you would want to see before relying on it?

5 responses

Summarise responses

all of the suggestions above

Why other diagnoses were rejected

Ai reasoning, references to guidelines and why it rejected other diagnoses

guidelines

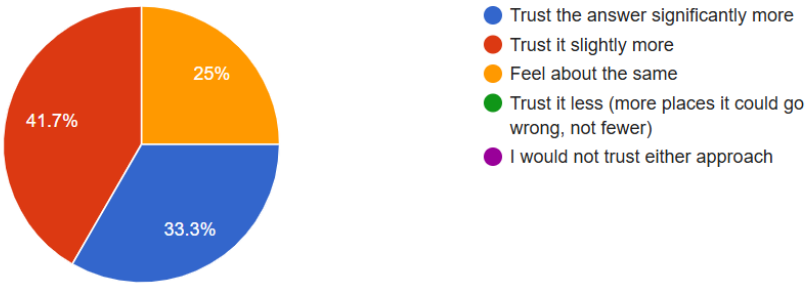
Different possibilities of diagnosis

Section D — How dIAgnosis Features Would Change Your Behavior

Q12. dIAgnosis asks five different AIs the same medical question instead of just one, and shows where they agreed or disagreed. Compared to using a single AI like ChatGPT alone, would this make you:

Copy chart

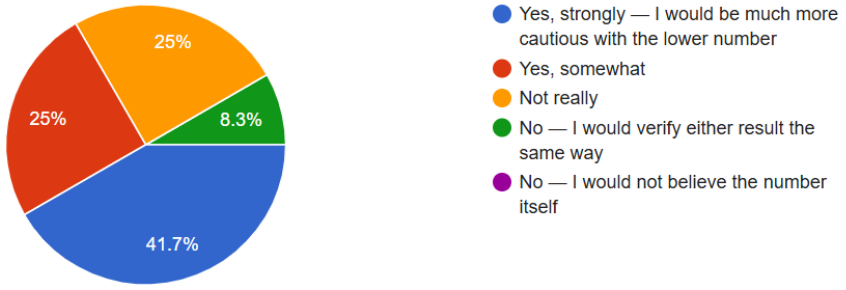
12 responses



Q13. Imagine the report says "5 of 5 AIs agreed" on a diagnosis in one case, and "2 of 5 agreed" in another case. Would seeing this number change how cautious you are about the diagnosis?

Copy chart

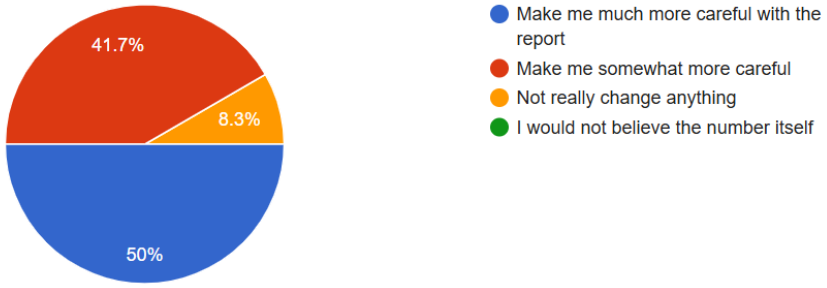
12 responses



Q14. The tool also shows a "made-up-information risk", a number from 0–100% showing how likely it estimates the AI may have invented part of its answer. If you saw this number on a real report, would it:

Copy chart

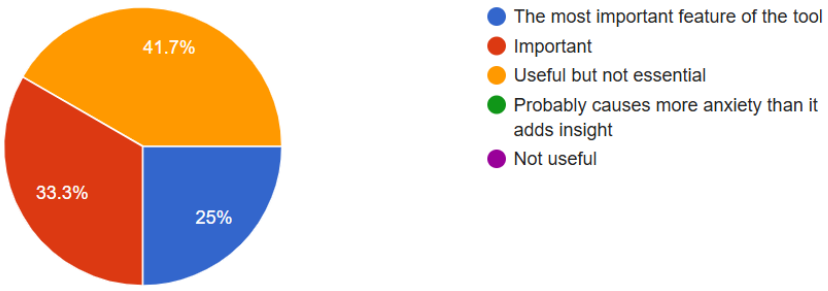
12 responses



Q15. Imagine 4 of the 5 AIs say "pneumonia" but 1 raises "pulmonary embolism" , and the system flags this as a warning so you don't miss the rare-but-serious option. How valuable is this kind of warning?

Copy chart

12 responses



Q16. On a follow-up visit, the tool sends the AI both today's case and the previous visit, plus how many days have passed, and asks the AI to focus on what has changed. How would this affect your view of the tool?

Copy chart

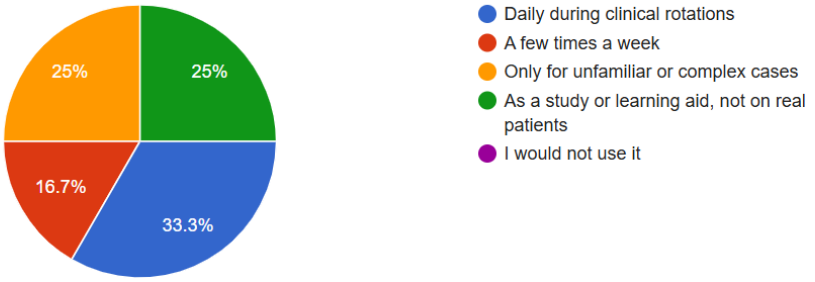
12 responses

Section E — Use Intention & Conditions for Trust

Q17. If dIAgnosis were freely available tomorrow, how often would you actually use it?

Copy chart

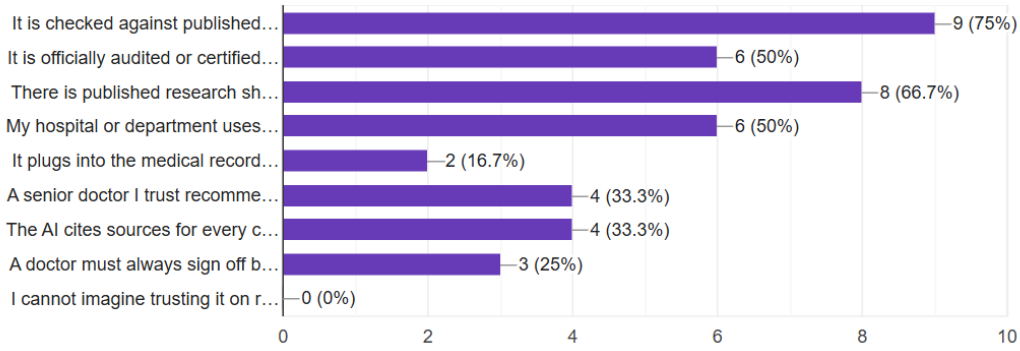
12 responses



Q18. What would have to be true for you to trust dIAgnosis on a real patient?

Copy chart

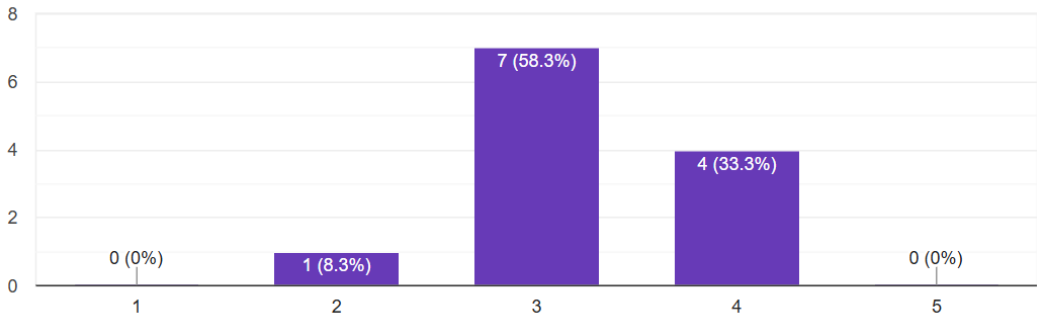
12 responses



Q19. Imagine you used dIAgnosis on a case and disagreed with its leading diagnosis. How likely would you be to override the tool and follow your own clinical judgment?

Copy chart

12 responses

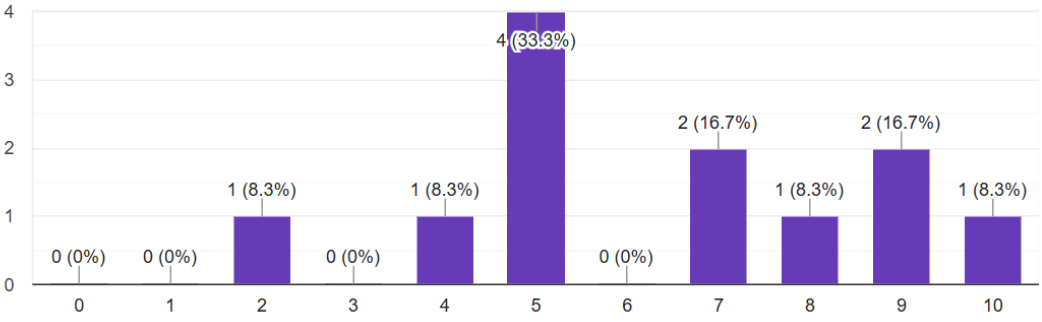


Section F — Recommendation & Criticism

Q20. On a scale of 0 to 10, how likely are you to recommend dIAgnosis to a peer studying medicine?

Copy chart

12 responses



Q21. What is the single biggest worry preventing you from using a tool like this on real patients?

7 responses

Summarise responses

Sources it's using and the reasoning

the wrong answers

Ai making up information

Responsibility and possibility of poor judgment

Wrong answers, cause we talking about putting someone's life at risk

losing my critical thinking

Missing critical aspects of a case that the Ai doesn't focus on

Q22. What is one feature that, if added, would make you significantly more likely to use this?

5 responses

Summarise responses

Trusted sources and international guidelines

real life statistics

Photo analysis

Real life Statistics

How other cases like the one I have have been handled

Appendix B - User Manual / Testing Guide

This appendix explains how to run and test the dIAgnosis prototype locally. The system is intended for demonstration and research purposes only. It has not been clinically validated and must not be used for real patient care.

B.1 Requirements

To run the prototype, the following are required:

- Python installed on the machine
- The dIAgnosis project folder or GitHub repository
- A stable internet connection
- Python dependencies installed from requirements.txt
- Valid API keys for the configured LLM providers
- A local .env file containing the required configuration values

The system uses external LLM APIs, so it will not generate reports unless valid API keys are configured.

B.2 Configuration

Before running the system, create a .env file in the project root. The file should contain the backend API key, database path, and LLM provider keys.

Example configuration:

API_KEY=your_backend_api_key

OPENAI_API_KEY=your_openai_key

GROQ_API_KEY=your_groq_key

ANTHROPIC_API_KEY=your_anthropic_key

GEMINI_API_KEY=your_gemini_key

DATABASE_PATH=./data/app.db

The actual API keys should not be committed to version control or shared publicly.

B.3 Installing Dependencies

Open a terminal inside the project folder and run:

```
pip install -r requirements.txt
```

If using a virtual environment, activate it before installing the dependencies, for example:

```
.\venv\Scripts\Activate
```

B.4 Starting the Backend

Run the FastAPI backend from the project root:

```
uvicorn backend.main:app --reload
```

The backend should start locally at:

http://127.0.0.1:8000

B.5 Starting the Frontend

Open a second terminal in the project folder and run:

```
streamlit run frontend/app.py
```

The frontend should open in the browser at:

http://localhost:8501

B.6 Testing a New Consultation

To test the main workflow:

1. Open the Streamlit frontend.

2. Enter patient identification details.

3. Enter vital signs, symptoms, history, medications, allergies, and available test results.

4. Select Generate Consensus.

5. Wait for the system to query the model panel and generate the report.

Expected result: the Clinical Consensus Report should appear with a leading diagnosis, certainty level, ranked differentials, investigations, red flags, safety checks, and worker agreement information.

B.7 Testing Report Features

After a report is generated, test the main tabs:

1. Open the **Clinical Report** tab and review the summary, assessment, diagnoses, investigations, red flags, and safety checks.

2. Open the **Risk Tools** tab and test calculators such as HEART, Wells, PERC, qSOFA, and GCS.

3. Open the **Evidence** tab and run a PubMed search based on the leading diagnosis.

4. Open the **Debate** tab and start an optional debate round.

5. Open the **Model Outputs** tab and expand individual worker responses.

Expected result: the user should be able to inspect the final report, supporting tools, evidence retrieval, debate output, and original worker drafts.

B.8 Testing Patient History

To test longitudinal functionality:

1. Generate a consultation for a patient.

2. Return to the sidebar.

3. Create a follow-up visit for the same patient.

4. Enter new symptoms or updated findings.

5. Generate a new consensus report.

Expected result: the system should preserve the patient's previous consultation and allow the new visit to be linked to the patient's history.

B.9 Testing Admin Analytics

To test the internal analytics dashboard:

1. Generate at least one consultation.

2. Open the admin/system analytics section.

3. Review report support, worker agreement, format completeness, unsupported-content risk, and cases needing review.

4. Run semantic panel consistency analysis if available.

Expected result: the dashboard should display structural QA and panel consistency metrics for stored consultations.

B.10 Testing Clinical Benchmark Evaluation

To test the benchmark workflow:

1. Open the clinical evaluation section.

2. Import the provided benchmark template cases.

3. Select cases to run.

4. Choose the number of benchmark models.

5. Run the clinical benchmark.

6. Review the overall benchmark summary and case-level results.

Expected result: the system should report benchmark metrics such as panel match score, Top-1 accuracy, Top-3 hit rate, unsafe omission rate, and panel-versus-best-single win rate.

B.11 Notes and Limitations

The prototype depends on external LLM providers, so response time and output may vary. API failures, rate limits, or missing keys may prevent some models from responding. The system is designed for local testing and demonstration, not public clinical deployment.