Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Agentic AI System for Hospital Management

Ashis Kumar Pati

ashispati@soa.ac.in

Institute of Technical Education and Research

Kaibalya Mohapatra

Institute of Technical Education and Research

Mukesh Kumar Padhi

Institute of Technical Education and Research

Shreya Tripathy

Institute of Technical Education and Research

Article

Keywords: Agentic AI System, Hospital Management, Reinforcement Learning, MultiAgent System, Memory System

Posted Date: August 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9856415/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

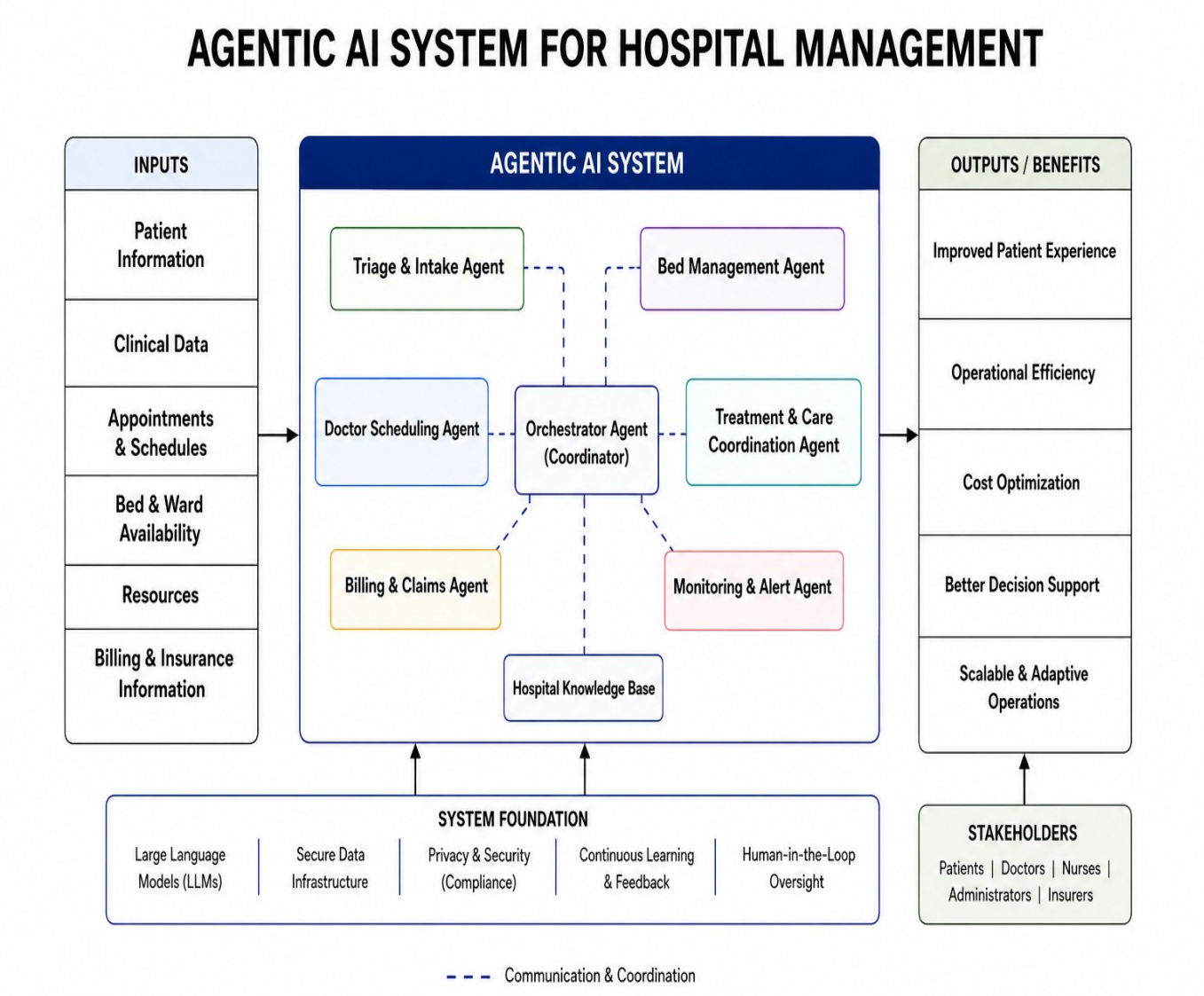
Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on November 5th, 2026. See the published version at https://doi.org/10.1038/s41598-026-78738-1.

Graphical Abstract

Agentic AI System for Hospital Management

Ashis Kumar Pati*, Kaibalya Mohapatra, Mukesh Kumar Padhi, Shreya Tripathy



Highlights

Agentic AI System for Hospital Management

Ashis Kumar Pati*, Kaibalya Mohapatra, Mukesh Kumar Padhi, Shreya Tripathy

- Hospital processes were made efficient through effective communication and coordination within various departments.
- Assisted in reducing waiting time for patients and decreased their work burden, enabling staff to attend to patients better.
- Increased the efficiency of the hospital's service delivery due to efficient operations.

Agentic AI System for Hospital Management

Ashis Kumar Pati*[a], Kaibalya Mohapatra[b], Mukesh Kumar Padhi[b] and Shreya Tripathy[b]

[a]Centre For Data Science, SOA University, Bhubaneswar, India ashispati@soa.ac.in, ashisinmath@gmail.com
[b]Centre For Data Science, SOA University, Bhubaneswar, India

ARTICLE INFO

Keywords:
Agentic AI System, Hospital Management, Reinforcement Learning, Multi-Agent System, Memory System

ABSTRACT

In modern-day hospital administration, there are many problems like long waiting time for patients, poor scheduling of physicians, lack of coordination between departments, among others. As such, this research suggests an Agentic AI System for Hospital Administration that utilizes a multi agent architecture and intelligence embedding to automate healthcare processes. This model entails a Patient Interaction Agent, a Care Coordination Agent, a Doctor Interaction Agent, and a Bed Allocation Agent, whose main roles are facilitating patients' entry and sorting out the patients' needs, allocation of doctors to attend to patients, coordination within the hospital, and hospital resource management, respectively. Through reinforcement learning, decision-making based on information processing, and synchronization, the framework becomes more adaptive and scalable. It is expected that due to intelligence distribution and intelligent decision-making, this model will minimize total waiting time, improve physicians' usage, and enhance scheduling of patients. Empirical analysis shows that hospital operations exhibit significant performance superiority of agentic models over the traditional models. Modern hospital administration faces major challenges such as prolonged patient waiting times, inefficient physician scheduling, poor interdepartmental coordination, and suboptimal resource utilization. Traditional hospital management systems often rely on static and centralized approaches that lack adaptability in dynamic healthcare environments. To address these limitations, this paper proposes a cooperative Multi-Agent Reinforcement Learning (MARL)-based intelligent hospital administration framework for workflow optimization and resource management. The proposed system consists of four collaborative agents: a Patient Interaction Agent for patient triage, a Doctor Scheduling Agent for dynamic physician allocation, a Care Coordination Agent for interdepartmental communication, and a Bed Allocation Agent for adaptive resource management. The agents utilize decentralized decision-making and reinforcement learning to optimize hospital operations in real time. The framework is evaluated using a hospital simulation environment and compared with conventional scheduling and management approaches. The results show that the effectiveness of the proposed system model is able to complete the tasks at a rate of 82.50% in a period of 5.12 seconds with a success rate in handling queries of 79.15%. Furthermore, the system attains a coordination efficiency of 87.50%, indicating effective collaboration among agents, while maintaining an agent conflict rate of 5.00%, ensuring stable and interference-free operations. The framework records a low communication overhead of 0.88, reflecting efficient inter-agent communication, and achieves a convergence rate of 96.4%, demonstrating learning stability and adaptability.This indicates that the model is efficient, scalable, and adaptable in managing intelligent hospitals.

1. Introduction

Hospital management in contemporary times entails multiple stages of the process, which can be illustrated starting with the admission of patients to the process of diagnosing and treating the condition as well as issuing prescriptions, followed by billing patients at the end. However, each mentioned process will only be efficient enough if the transfer of data and coordination with various departments happens in time. Nevertheless, in traditional models, the management flow described above is typically interrupted because information is transferred manually, causing delays, low efficiency, and increased workload among health professionals [11].

Today's hospital information systems face many difficulties associated with the increase in patient numbers, limitations regarding hospital infrastructure, as well as high demands for high quality of services. Typically, current hospital information systems give preference to the digitization of records over providing smart collaboration between departments of hospitals, which causes many issues, such as long wait times for patients, poor allocation of resources, delays in prescription, and ineffective real-time decision making. Such problems indicate the need for developing a new hospital information system based on intelligent technologies that would enable better integration into workflow and coordination, see [18].

Moreover, the existing hospital information management faces some crucial problems related to inefficiency during patient scheduling and waiting periods. Additionally, there might be issues related to insufficient allocation of resources, which includes allocating beds, instruments, and personnel[12]. Another issue that could arise in hospitals concerns emergency cases where patients need to be prioritized in a proper way and promptly. Finally, administrative overload due to excessive paperwork and repeated entering of data into the system also poses certain challenges to healthcare professionals.

The concept of agentic AI represents a shift from the traditional rule-based framework to that of autonomous and intelligent decision-making using dedicated software agents[2]. In addition to making decisions based on perceptions of the environment and information processing, these

ORCID(s):

agents work together with other agents to carry out various functions. Examples of functions performed by agents in a hospital setting include communicating with patients, providing clinical support, managing pharmacological activities, and coordinating care among others. The capabilities of these agents to adapt to changes, prioritize actions, and optimize performance are enabled by technologies like Large Language Models.

1.1. Problem Statement

The functioning in modern-day hospitals involves multiple departments, such as OPD, emergency rooms, pharmacy, diagnostics, bed management, and other similar departments. Proper coordination between them is vital to ensure efficiency in operation. Nowadays, information systems in hospitals are fragmented because different departments use software that cannot interact with other departments' software. It results in isolated databases, delayed information transfer, and inefficient coordination in the process of providing medical assistance. Additionally, numerous existing hospital management systems are reactive and focus on rule-based processing and data gathering. There are no sophisticated processes that could help adapt to changing conditions of hospitals, such as a sudden change in the number of patients, prioritization of urgent cases, and changes in resources. As a result, hospitals face problems like ineffective scheduling of patients, improper utilization of resources, untimely diagnostics, and unnecessary administrative pressure on healthcare professionals. Therefore, one can conclude that the main drawback of modern hospital management systems lies in the fact that there is no coordinated system with intelligent and flexible features.

1.2. Proposed Solution

1. **Design of the Decentralized Multi-Agent System for Hospital Management :** This research suggests a decentralized approach where independent artificial intelligence agents work together to manage the entire hospital, from interacting with patients to coordinating physicians, managing medications, and allocating beds, with the goal of increasing overall operational effectiveness.
2. **Application of Artificial Intelligence and Large Language Models in** Decision Making :In this proposed framework, advanced artificial intelligence and Large Language Models are used to analyze patient data and clinical data to make intelligent decisions in real time.
3. **Coordination of Real-Time Operations and Resource Allocation :** To enable coordination and messaging between agents in real time, a communication scheme is developed, ensuring proper allocation of resources in emergency situations.
4. **Scalability and Flexibility of the System through Agent Collaboration :**This system allows for scalability based on patient load and emergencies through collaborative autonomous agents without causing any bottlenecks.
5. **Improvements in the Performance of Hospitals :**The proposed framework provides significant improvements in performance indicators compared to conventional hospital management systems.

1.3. Organisation of the Paper

Section 2 discusses the existing works related to Healthcare Management using AI. Section 3 explains the Proposed Agentic AI Framework that we have implemented in our project. Section 4 contains all the Mathematical Formulation used in our model. Section 5 explains the Experimental Setup and Implementation of each agent and the dataset used. Section 6 discusses the Result and Comparison of the agent and workflow. Section 7 looks at the Ethical and Security aspects in the deployment of the proposed AI-based multi-agents for healthcare. Section 8 addresses the Conclusion, Future Improvements, and Prospects of an AI-based multi-agent healthcare system for enhancing the effectiveness of healthcare scalability.

2. Related Work

2.1. AI in Healthcare Management

Abdul et al. **[1]** investigate various AI-based frameworks that could potentially improve the performance of healthcare systems during natural disasters. According to the results presented by the authors, the use of AI tools is highly beneficial for increasing the effectiveness of management activities due to the opportunities offered by machine learning, predictive analytics, and robotics. In general, the authors argue that AI technologies may significantly benefit disaster-related healthcare processes.

Alaskar et al. **[4]** review the essential contribution made by artificial intelligence in modern medicine as well as possible solutions provided to critical problems of the healthcare sector using computational approaches. Thanks to big healthcare data and advanced computing capabilities, AI provides numerous effective solutions to critical issues.

Amiri and Zahra **[5]** investigate the impact of information systems utilizing artificial intelligence on the field of healthcare management and present a taxonomy aimed at solving associated problems. Overall, the proposed taxonomy helps understand and develop further AI usage within the domain under consideration.

Junaid et al. **[13]** analyze current achievements in the field of healthcare management systems implemented on the basis of innovative technologies including the Internet of Things, artificial intelligence, blockchain technology, and other. The application of these technologies enables achieving personalized healthcare but poses several associated challenges linked to data complexity and standardization.

Sehrawat and Kumar **[17]** in suggested a sophisticated system designed to monitor elderly patients in order to facilitate their better treatment using wearable sensors, big data, and IoT solutions. The use of Hadoop MapReduce and a hybrid optimization-based deep learning approach (HDCO-DEL) allows managing and processing collected health data effectively.

2.2. Multi-Agent Systems in Healthcare

Kamal et al. **[3]** elaborate on ways of employing telemedicine, e-health, and MAS within the field of healthcare, especially concerning chronic pain management. The authors explain how the implementation of MAS helps achieve intelligent communication between multiple agents that ensures efficient healthcare services. Nonetheless, it is imperative to note the many benefits of telemedicine and e-health solutions in enhancing physician-patient communication and providing healthcare services.

Dhatterwal et al. **[9]** highlight problems in healthcare emerging due to growing populations and high prices. These researchers suggest the use of agent-based solutions as a way of dealing with these challenges through developing a general framework that utilizes swarm intelligence together with MAS for the purpose of collaboration, assistance in patient diagnosis, and handling patients' information.

Reffad et al. **[16]** consider the application of an Internet of Things-based approach that uses bio-inspired hybrid multi-agent technology in order to provide healthcare services. In particular, the authors make use of IGWO and PSO optimization approaches for increasing efficiency in distributing tasks, decreasing energy consumption, and improving performance in fog computing.

Pandey et al. **[15]** focus on possible applications of large language models and MAS in the process of automating prior authorization within healthcare systems. In particular, the researchers offer a framework according to which large-scale tasks are broken down into smaller subtasks and accomplished by agents without human involvement. According to empirical results, GPT-4 demonstrated 95.6 % accuracy.

Dhasarathan et al. **[8]** describe an e-healthcare model that combines MAS and deep learning. Using privacy metrics such as confidentiality, integrity, and availability, the researchers were able to control access to patients' data in real time. In other words, their solution allows regulating the access to patient's data by MAS agents.

2.3. Large Language Model Agents

Fan et al. **[10]** look at a multi-agent framework for simulating the interactions between doctors and patients. As such, their work seeks to explore ways to improve the performance of LLMs in different clinical roles. They introduce Multi-View Medical Evaluation (MVME), which enables researchers to evaluate the performance of LLMs in different clinical scenarios, such as symptom collection, diagnosis, and recommendations for examinations. The paper clearly illustrates that there is still much scope for improvement when it comes to utilizing LLMs in the healthcare sector.

Wang et al. **[19]** study ColaCare, which is a multi-agent framework that uses both LLMs and expert models to improve the analysis of Electronic Health Records (EHRs). Their paper provides an insight into how DoctorAgents and MetaAgents cooperate in order to produce better clinical decisions. As a result, ColaCare achieves better performance in areas, such as mortality prediction and hospital readmission prediction.

Li et al. **[14]** introduce the concept of Agent Hospital, which is a simulated healthcare environment made up of autonomous patient, physician, and nurse agents enabled with the power of LLMs. By creating a healthcare environment in which physicians can practice autonomously by treating large numbers of simulated patients, the paper shows the benefits of using autonomous agent technologies in improving medical AI and education.

Cao et al. **[7]** talk about a three-tier framework to help clinicians adopt LLMs safely. Firstly, the authors classify it into three tiers, where Tier 1 refers to foundational competencies, such as prompt engineering, privacy considerations, and patient interaction. Secondly, Tier 2 emphasizes the significance of mastering LLM assessment skills, including bias identification and workflow integration. Lastly, Tier 3 includes advanced competencies, including ethical governance and model management.

Bao et al. **[6]** explore the Personalized Intelligent Outpatient Reception System (PIORS), an AI-powered system designed to reduce the workload of receptionist nurses in hospitals while improving patient service quality. The system combines an LLM-based virtual reception nurse with hospital information systems (HIS) to provide personalized and efficient assistance.

2.4. Limitations of Existing Systems

1. Although the existing literature [1], [4] and [8] considers artificial intelligence as an integral part of hospital administration, none of these studies incorporate Reinforcement Learning (RL).
2. Literature review [5], [10], and [19] applies the idea of using LLMs and MAS in separate contexts, but this study applies a full-fledged model based on a multi-agent system that involves agents of patient, doctor, and coordinator.
3. References [13] and [6] are focused on theoretical backgrounds or a survey analysis, while this paper is centered on a practical approach including agents' implementation, which the above-mentioned references fail to cover.
4. Literature review [3] and [15] focuses on AI-based decision-making, but they do not focus on continuous interaction with patients. In this paper, a patient interaction agent was used to facilitate real-time interaction.

3. Proposed Agentic AI Framework

The architecture of the multi-agent hospital management system is discussed in this segment, which consists of an intelligent distributed system capable of performing all hospital functions autonomously using artificial intelligence. Specialization plays a pivotal role in this architecture, where each individual function is assigned to a particular AI agent. A relational database is used to maintain the accuracy of all patient care procedures.

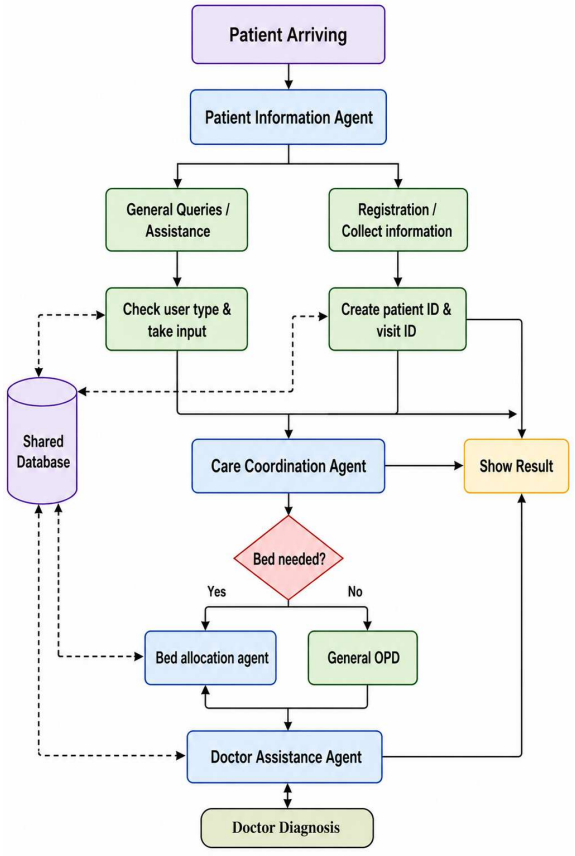


Figure 1: Model Architecture

3.1. System Architecture

The suggested model is composed of five theoretical levels, namely Data Layer, Agent Layer, Decision Layer, Communication Layer, and User Interface Layer. Each level consists of a set of different responsibilities that together constitute a pipeline covering every interaction of patients from their arrival until their discharge as shown in figure 1

The data processing in the system takes place in a linear sequential manner. As soon as the patient arrives at the clinic, the first step is the registration process, done using the Patient Intake Agent, which creates the patient visit record and collects the basic information about their vital signs. Once this stage is accomplished, the Triage and Scheduling Agent is used to determine the level of severity, the correct department where the patient needs to be admitted, and the physician to attend the patient. In case of an admission requirement, the Bed Allocation Agent will be used to select the most suitable clinical bed. Finally, the selected physician will use the Clinical Documentation Agent to diagnose, prescribe diagnostic tests, medications, and note down any observations during the visit. The System Architecture can be seen in Table 1

3.2. Agent Design

The architecture includes four different agents in addition to a vision model, which is problem-domain dependent. An agent is described by a set of features that include the functionality level, language model used, the toolbox, and the database tables.

Table 1
System Architecture

Layer	Description
Data Layer	All system entities rely upon a centralized relational database based on MySQL hosted on Aiven Cloud that acts as the one true source of truth. It keeps patient demographics, visits, vitals, triage, physicians, beds, medication, clinical documentation, test requests, procedures, billing, and reinforcement learning (RL) interaction history. All the agents interact with the centralized database only to read and write data.
Agent Layer	There are four unique AI agents at this layer: Patient Interaction Agent, Doctor Interaction Agent, Care Coordination Agent, and Bed Allocation Agent. All four agents have their individual large language model (LLM), tool kit, and logic implemented into them. They are called at specific points during the patient's journey.
Decision Layer	Intelligent decisions at this level is achieved through the combination of rule-based engines, large language models (LLMs), and a Deep Q-Network (DQN). The output of these decisions includes the severity score, department categorization, physician scheduling with consideration of preferences, and bed assignment with hard constraint implementation and risk analysis.
Communication Layer	Agent-to-agent communication does not take place directly, and rather takes place indirectly, using state propagation via the database, and structured function calling between agents. For example, the Patient Intake Agent triggers the Triage and Scheduling Agent by calling its handle() function using visit_id, thereby allowing the scheduling agent to access all the contextual information from the database.
User Interface Layer	Two terminal-based conversational interfaces are included in the project: hospital_interface() to interface with patients for admission, triage, and queries; as well as hospital_physician_interface() for use by physicians for entering clinical notes. Both interfaces take natural language input from users which is then parsed into tool calls by LLMs

3.2.1. Patient Interaction Agent

Patient Interaction Agent acts as the primary interface between the patients and the management system in hospitals, dealing with the admission of patients and queries or interactions related to patients. The agent utilizes the Llama-3.1-8B-Instant model through Groq API for efficient conversations with the users. The agent consists of two distinct components: Intake and Query/Interaction. Initially, the Intake component takes all the arriving patients. The tool is used with the force single tool-call intake_patient to ensure

that the model is doing just one real action and not inventing unsupported things. The intake_patient tool combines three actions in one operation: registering/retrieving information about the patient (creates a new entry if the patient is new and retrieves an entry if he visited before), creating a visit record, and registering the vital signs. New patients receive their unique identifier PAT_xxxx and store patient information in the 'patients' table. Returning patients can use the existing ID to prove their existence and retrieve information. In both cases, a new entry in the 'visits' table with OPD by default type and Pending status will be created along with recording of blood pressure, body temperature, oxygen saturation, and heart rate in the 'vitals' table. The prompt does not allow creating a new unique ID for a patient and assigning unknown values. Six different tools are available for use with the Query/Interaction component, performing all post-intake interaction automatically (tool_choice = auto): handle_appointment (handled by the Triage and Scheduling Agent), generate_full_bill, update_bill_after_diagnosis, view_bill, get_bed_details_by_visit, and discharge_patient. handle_appointment helps the user to schedule visits, while four remaining tools are related to generating, updating, viewing the full bill and discharging a patient from the ward. The consultation fees for a GP and specialist are set to 500 and 1,500 rupees respectively, while bed rates for different beds are 1,000 rupees for a general ward, 5,000 rupees for an ICU bed, and 8,000 rupees for an emergency bed. The patient interaction flowchart can be seen in Figure 2

3.2.2. Care Coordination Agent

This component will be referred to as the Clinical Triage and Scheduling Agent (CTSA). This component encompasses the most complicated architectural design within the entire system. It features a multi-step pipeline which consists of rule-based scoring, large language model reasoning, and physician scheduling considering patient preferences. The pipeline runs through ten steps. In step one, a rule-based engine scores the patient on a scale of 0 to 100 based on thresholds for vital sign measurements. In step two, a local Llama-3.2-3B model via Ollama validates this severity score and adds or subtracts up to 20 points based on the patient's age, chronic conditions, allergies, and medical history. These contextual factors cannot be considered by the fixed rules applied to vital sign measurements. In step three, another LLM invocation determines the department for the patient among five possible clinical departments including general medicine, cardiology, neurology, orthopedics, and ENT. In step four, a consultation check involving multiple departments checks if there should be consultations with any other departments depending on the patient's history; if yes, then consultation assignments are done without counting additional workload. Finally, the severity score determines five clinical zones including the red zone (severity $\geq$ 90), critical care (70 $\leq$ severity $\leq$ 89), step-down ward (50 $\leq$ severity $\leq$ 69), IPD (30 $\leq$ severity $\leq$ 49), and OPD (severity $<$ 30). If the severity level falls below 30, then it is considered an outpatient case except when a dedicated LLM

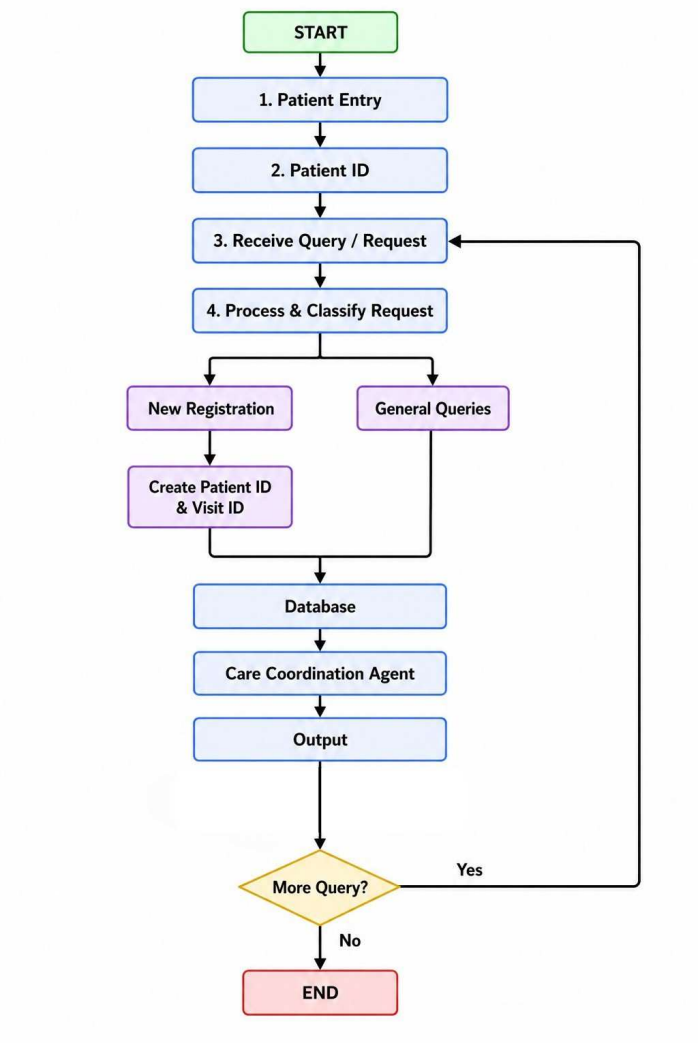


Figure 2: Patient Interaction Flowchart

invocation confirms the need for admission. The physician scheduling algorithm works on the principle of preference-based scheduling that is structured along four paths:

1. no preference—where the physician with the highest rank is chosen by rule without the assistance of the LLM
2. maintaining continuity from the previous doctor seen by the patient
3. selection of a preferred physician, resolved before the pipeline process through the specific interface
4. preference of a woman physician. The physicians are ranked based on a scoring system that includes +50 for departmental match, +30 for the matching of the specialization keywords, and +15 for cognitive load (sum of severity_score/100 for all active patients, if any). An additional bonus score of +5 can be provided considering the physician's experience. In case the physician selected by the patient's preference is not available, the LLM generates an empathic response for the patient depending on his/her severity.

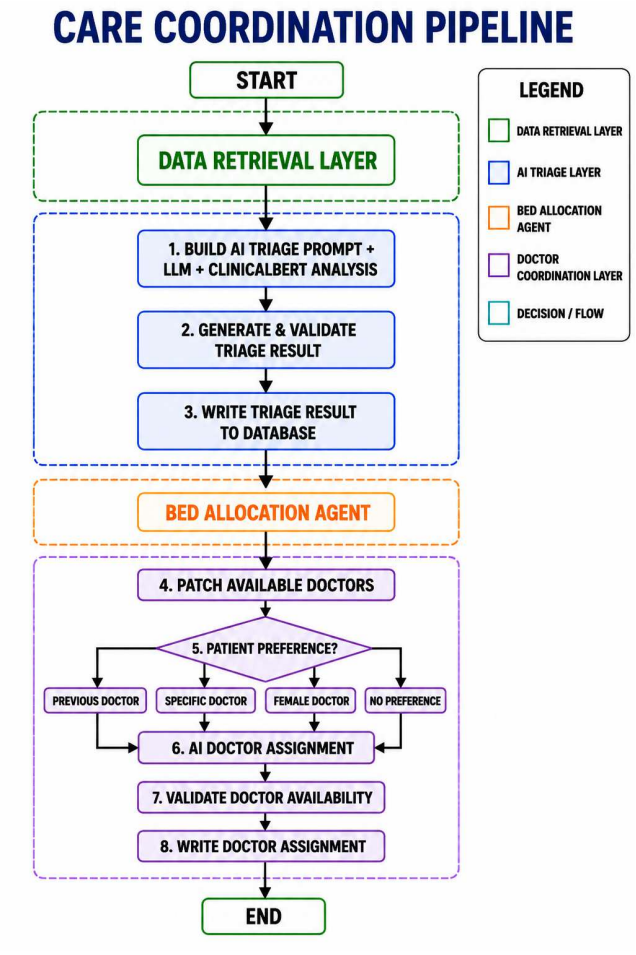


Figure 3: Care Coordination Flowchart

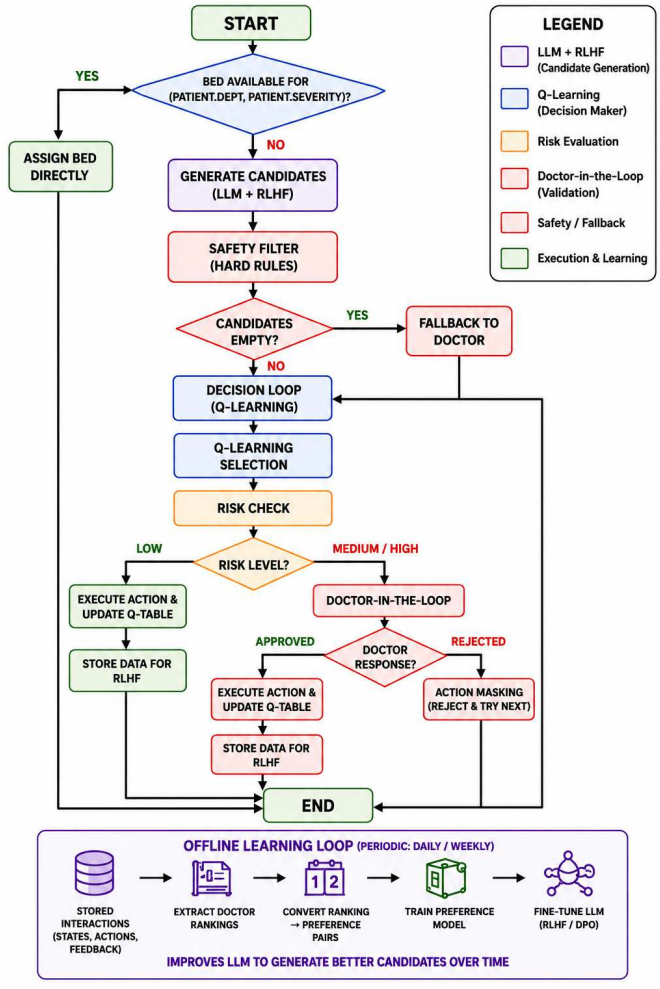


Figure 4: Bed Allocation Flowchart

The care coordination flowchart can be seen in Figure 3.

3.2.3. *Bed Allocation Agent*

In summary, the proposed Bed Allocation Agent utilizes the paradigm of two-agent reinforcement learning. First, we have a locally-deployed Qwen2.5-7B LLM (via Ollama), acting as a clinical ranker and returning a shortlisted of up to 5 beds in JSON format, where each suggestion is accompanied by the clinical rationale in one sentence. The fallback strategy to be used should a LLM crash occurs is based on the set of heuristics rules.

As for the second agent, this will be a DQN whose input is an 11-dimensional normalized state vector, which includes parameters such as severity level, department, type of bed required, emergency flag, bed availability at various levels, percentage of occupied beds in the department/hospital, patient age, and LLM override. Before being scored by the DQN, all candidates need to be filtered through a set of hard constraints, namely patients whose severity is 90 % + will be assigned an Emergency bed, while those with severity ≥ 70 are forbidden from being allocated a General bed altogether. Then a three-level risk classifier is used to classify the selected candidate from DQN as either LOW, MEDIUM, or HIGH risk. The HIGH-risk allocation can only be completed after doctor-in-the-loop approval and entry in the database. The training of the RLHF algorithm is supplemented by collecting its outputs (doctor feedback, alternate ranking, etc.) and rewards in respective tracking tables (bed_allocation_log, doctor_bed_feedback). Offline DPO training for LLM candidate generation is scheduled to happen in Phase 2. Bed allocation flowchart is described in Figure 4.

3.2.4. *Doctor Interaction Agent*

Clinical Documentation Agent is meant for use by doctors in hospitals as a means of documenting patient consultation outcomes. Unlike patient-facing agents using cloud-hosted large language models, the agent works based on the locally deployed Qwen2.5-7B using Ollama, meaning it ensures that no sensitive clinical data leaves the hospital's private network. The agent uses a ReAct-style agentic loop, whereby it keeps on calling on different tools as long as the language model thinks that there are still relevant actions to be performed. The agent has nine possible tools to invoke;

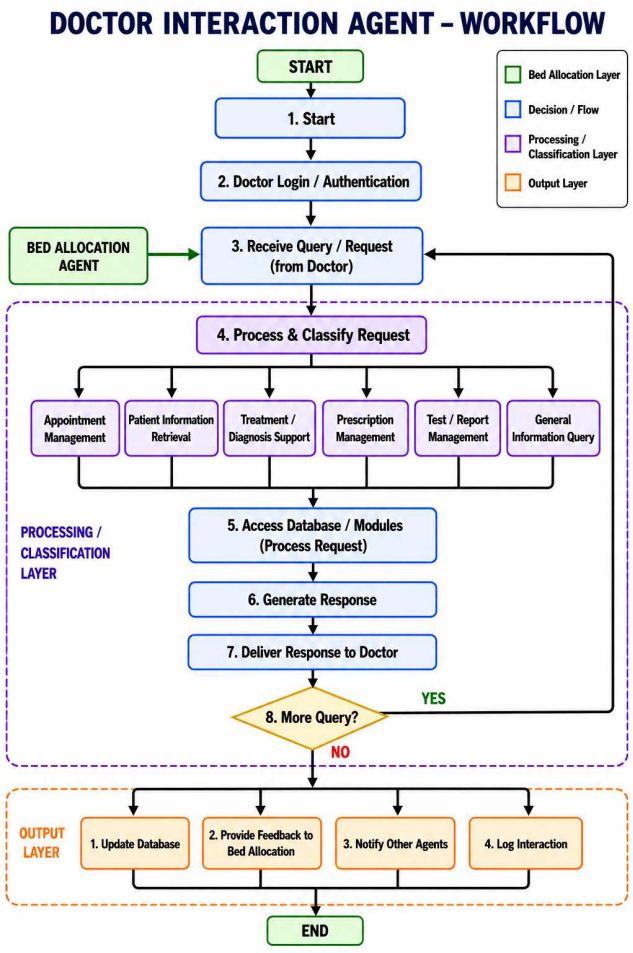


Figure 5: Doctor Interaction Flowchart

they include add_clinical_note, add_diagnosis, order_test, prescribe_medicine, schedule_procedure, `mark_admission_required`, mark_followup, update_visit_severity, and `record_vitals`. All these tools correspond to particular database tables' write operations. The system prompt has been written in such a way that it enforces strict grounding; the agent cannot come up with any value not explicitly given in the input, whether it is visit id, physician id, name of drugs, etc. In addition, the agent cannot repeat similar tool calls since it will lead to duplicate database write operations. Example of an input may be: "Patient diagnosed with viral fever. Order CBC. Prescribe Paracetamol 500 mg twice daily for 5 days." The detailed flowchart of doctor interaction agent is shown in Figure 5.

3.2.5. Medical Model

The Medical Model serves as an additional path in the process of training, providing patients with the ability to upload their radiology images (X-rays, MRIs, and CT images) and getting a description or summary produced using artificial intelligence. The Medical Model uses the vision-language model Qwen2-VL-7B-Instruct, which has seven billion parameters and uses the 4-bit NF4 quantization method implemented with BitsAndBytes in order to save VRAM resources.

The Adaptive Low-Rank Adaptation (AdaLoRA) technique was used to fine-tune the model based on the rank budget assigned to the attention and feed forward projection layers (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj). The fine-tuning process utilized the SLAKE dataset (SLAKE-VQA-English), which consists of medical images, related questions, and answers concerning radiology. Finally, the following configuration was used to train the model: init_r = 16, target_r = 12, learning rate LR = 2e-5, 15 epochs, and balanced closed questions increase ('yes/no'). The last feature allows avoiding class imbalance while predicting whether the answer should be a 'yes/no' question. Once receiving a query for image processing from the patient, the Medical Model will start providing disease-related information.

3.3. Memory system

The system implements memory at three distinct levels: session memory, persistent structured memory, and decision audit memory. The Memory Architecture is discussed above in Table 2

Moreover, the embeddings of the symptoms that correspond to each patient visit are generated using the Bio_ClinicalBERT (emilyalsentzer/Bio_ClinicalBERT) model and consist of 768 dimensions. The purpose is to provide a robust semantic understanding of the patient's chief complaint or their symptoms. Currently, they are documented in conjunction with triage decisions within the decision audit layer, while their purpose is to serve as input features in embedding-based triage validations.

3.4. Technologies Used :

The considered architecture was implemented in Python 3.10+ and includes managing the workflow of multiple agents, the reinforcement learning pipeline, and database interaction. The Groq API was applied to perform inference of the Llama-3.1-8B-Instant model in real-time for patient interaction tasks. Ollama is responsible for executing the Qwen2.5-7B and Llama-3.2-3B locally deployed models due to the confidentiality requirements of sensitive clinical information. PyTorch was chosen to implement the reinforcement learning model based on DQN for bed allocation, replay buffer management, and AdaLoRA training. PEFT with AdaLoRA fine-tunes the Qwen2-VL-7B model to reduce the number of trainable parameters without affecting performance. Qwen2-VL-7B-Instruct acts as the vision-language model to address Medical VQA tasks by processing textual and radiological image data using 4-bit quantization for computational efficiency. Bio_ClinicalBERT is responsible for generating 768-dimensional embeddings from patient complaints and symptoms. Persistent storage of patient information, workflow management, and reinforcement learning data is ensured through a MySQL database deployed on Aiven Cloud and configured for high availability. To simulate hospital operations and perform medical

Table 2
Memory Architecture

Memory Type	Mechanism	Scope and Purpose
Session Memory	Python dictionary stored in memory during processing that uses visit_id or patient_id as keys.	This construct is stored in memory for the duration of one user session and allows for the history lookup functionality (using show_history() method) and also ensures consistency in context between several queries made during one interaction.
Persistent Structured Memory	The MySQL-backed relational database hosted on the Aiven Cloud.	This storage layer provides a long patient history. The patient history allows functionalities like resolving doctor preference from earlier sessions and severity adjustment due to knowledge about chronic diseases.
Decision Audit Memory	JSONL (flat-file) (rl_log.jsonl) along with RL tables in MySQL.	All such decisions made by agents are recorded in terms of full state-action pairs: triage scores, bed assignment, physician allocation, and feedback from physicians. Such database can be used for training and fine-tuning of the LLM component of the Bed Allocation Agent via RLHF/DPO.

VQA tasks, the SLAKE-VQA-English dataset and synthetically generated hospital records are utilized. In addition, the VQA-RAD and SLAKE datasets from Hugging Face are used to fine-tune the Qwen2-VL-7B-Instruct model for VQA tasks, with closed-question samples from the SLAKE dataset being oversampled during training.

The different technologies used and their roles are described in Table 3

4. Mathematical Formulation

This section presents the formal mathematical foundations of the proposed multi-agent hospital management system. Every formula and constant presented here has been verified directly against the implemented code.

4.1. Entity Set Definitions

The hospital system operates over three primary entity sets. The Patient Visit Set is as: P = p1, p2, . . . , pn where each patient visit pi is characterized by a tuple of demographic, clinical,and contextual attributes including age, gender, chief complaint, symptoms,vitals, chronic conditions, allergies, medical history, and visit type. The vitals sub-tuple consists of blood pressure (systolic), SpO2 , temperature, blood sugar,heart rate, and respiration rate.

The Doctor Set is defined as: D = d1, d2, . . . , dm where each doctor dj is described by their department, specialization, years of experience, current patient load, maximum capacity, availability status, and gender.

The Bed Set is defined as: B = b1, b2, . . . , bk where each bed bl is described by its type, department, occupancy status, and priority score, with:

bed type $\in$ {General, ICU, Emergency}

4.2. Multi-Objective Optimization

The system simultaneously optimizes four operational objectives that together capture the core tension in hospital resource management: maximizing care quality and resource utilization while minimizing patient waiting time and workload imbalance.

Objective 1 — Minimize Average Patient Waiting Time : Waiting time for each patient is the difference between when treatment begins and when the patient arrived. The system minimizes the average across all patients:

$$\min W = \frac{1}{n}\sum_{i=1}^{n}\left[T_{\text{treatment}}(p_i) - T_{\text{arrival}}(p_i)\right] \tag{1}$$

where $T_{\text{treatment}}(p_i)$ is the time the assigned doctor begins treating patient pi , and $T_{\text{arrival}}(p_i)$ is the arrival timestamp recorded in the visits table. This is reduced by fast triage, rapid doctor assignment, and immediate bed allocation.

Objective 2 — Maximize Bed Utilization
The system maximizes the fraction of beds in active use:

$$\max U_{\text{bed}} = \frac{\left|\left\{b_l \in B : \text{occupancy_status}(b) = \text{occupied}\right\}\right|}{|B|} \tag{2}$$

where | B | denotes the total number of bed. This is subject to hard safety constraints enforced by the system: no General bed may be assigned when the patient severity score is 70 or above, and only an Emergency bed may be assigned when the severity score is 90 or above.

Objective 3 — Minimize Doctor Workload Variance
Rather than counting patients equally, the system uses a weighted cognitive load metric for each doctor. This is computed directly by the get weighted load() function in the code as the sum of severity scores (normalized to [0, 1]) of all active patients assigned to that doctor:

$$L_j = \sum_{p_i \in A(d_j)} \frac{\sigma(p_i)}{100} \tag{3}$$

Table3
Technologies and their Roles

Component	Technology	Role in System
Programming Language	Python 3.10+	Everything associated with agent logic, database interface, reinforcement learning (RL) training, and fine-tuning of models.
Cloud LLM API	Groq API	Runs Llama-3.1-8B-Instant to support the Patient Intake and Query Agents, with sub-second latency for inference, making it ideal for real-time interaction with patients.
Local LLM Inference	Ollama	Runs Qwen2.5-7B locally for the Clinical Documentation Agent and Bed Allocation Agent, and Llama-3.2-3B for the CTSA. It is done to ensure that sensitive clinical data stays on premises
Deep Learning Framework	PyTorch	Implements DQN to allocate beds, using the BedDQN neural network, the ReplayBuffer class, and the AdaLoRA fine-tuning training loop for the Medical VQA model.
Parameter-Efficient Fine-Tuning	PEFT (AdaLoRA)	Qwen2-VL-7B fine-tuned on the SLAKE medical VQA dataset. The AdaLoRA dynamically prunes low-importance singular value components from attention layers to reduce the number of trainable parameters but keep capacity unchanged.
Vision-Language Model	Qwen2-VL-7B-Instruct	Used as the base model for Medical VQA tasks, which process input image and text data as inputs. It uses quantize 4-bit representation via BitsAndBytes for efficiency.
Clinical NLP Model	Bio_ClinicalBERT	Computes embeddings of the size of 768 for patient symptoms and chief complaints for use in the CTSA for enhanced logging in terms of embeddings.
Relational Database	MySQL on Aiven Cloud	Provides a persistent central storage for more than 15 tables with patient records, workflow data, and RL data tracking. Aiven provides cloud hosting for the service, which guarantees high availability.
Hospital Dataset	SLAKE-VQA-English; Simulated Hospital Records	SLAKE provides curations of Q/A pairs with radiology images for VQA fine-tuning. Operational hospital data including patient records, visits, vitals, doctor assignment, and beds state are generated synthetically and stored in MySQL.
Model Fine-Tuning Datasets	VQA-RAD, SLAKE	Datasets used for training are VQA-RAD (flaviagiammarino/vqa-rad) and SLAKE (mdwiratathya/SLAKE-vqa-english). HuggingFace Datasets provides them. The final model trains only on SLAKE with closed-question oversampling.

where A(dj) is the set of active non-discharged patients currently assigned to doctor dj , and $\sigma(p_i)$ is the final triage severity score of patient pi . For example, a doctor managing four ICU patients with severity $\approx$ 85 carries a load of Lj = 3.4, whereas a doctor managing four OPD patients with severity $\approx$ 20 carries only Lj = 0.8. The system then minimizes the variance of this load across all doctors:

$$\min \mathrm{Var}(L) = \frac{1}{m}\sum_{j=1}^{m}\left(L_j - \bar{L}\right)^2 \tag{4}$$

where $\bar{L}$ is the mean weighted load across all m doctors.

Objective 4 — Maximize Service Quality

Service quality captures three dimensions: whether the patient was assigned a doctor from the correct department, whether the patient's stated doctor preference was fulfilled, and whether the allocated bed type matched the clinically required type:

$$Q = \sum_{i=1}^{n}\Big[\gamma_1 \cdot \mathbb{1}\left[\mathrm{dept_match}(p_i, d_j^*)\right] + \gamma_2 \cdot \mathbb{1}\left[pref_satisfied(p_i)\right] + \gamma_3 \cdot \mathbb{1}\left[bed_type_match(p_i, b_l^*)\right]\Big] \tag{5}$$

where $\mathbb{1}[\cdot]$ is the indicator function, d_j^* describes the assigned doctor, b_l^* represents the assigned beds and $\gamma_1, \gamma_2, \gamma_3 \geq 0$ are non-negative weighting coefficients.

Combined Scalarized Objective : The four objectives are combined into a single minimization problem using a weighted scalarization :

$$\min Z = \alpha \cdot \overline{W} + \beta \cdot (1 - U_{\text{bed}}) + \gamma \cdot \text{Var}(L) - \delta \cdot Q \quad (6)$$

where $\alpha, \beta, \gamma, \delta \geq 0$ are coefficients that balance the relative importance of each objective.

4.3. Triage Severity Score — Hybrid Rule-LLM Formulation :

The triage severity score is computed through three sequential stages. All threshold values and constants are taken directly from calculate severity from vitals() and validate triage() in the Care Coordination Agent code.

Stage 1 — Rule-Based Score

The scoring begins with a fixed base value of 10. A visit-type bonus is added first, defined as:

$$\Delta_{\text{visit}} = \begin{cases} 20, & \text{if visit_type} = \text{Emergency} \\ 10, & \text{if visit_type} = \text{IPD} \\ 0, & \text{if visit_type} = \text{OPD} \end{cases} \quad (7)$$

Vital-sign increments are then added based on threshold conditions, and the score is capped at 100:

$$\sigma_{\text{rule}}(p_i) = \min\Big(100,\ 10 + \Delta_{\text{visit}} + \sum_k w_k \cdot \mathbb{1}\left[vital_k \in \textit{critical range}\right]\Big) \quad (8)$$

where w_k represents the increment associated with condition k, $\mathbb{1}$ denotes the indicator function and the score is capped ar 100.

The exact increment values w_k, verified from the code, are given in Table 4. Within each vital sign group, only the first matching condition applies (implemented as `elif` branches). If $\sigma_{\text{rule}} \geq 60$ after all increments are applied, the emergency flag is automatically set to `True`.

Stage 2 — LLM Validation and Clamping

The local LLM (llama3.2:3b via Ollama) receives the rule-based score along with the patient's age, chronic conditions, allergies, and medical history, and proposes a contextually adjusted score $\sigma_{\text{LLM}} \in [0, 100]$. The final score is computed using the following exact clamping formula, taken directly from lines 1049–1050 of the code:

$$\sigma_{\text{clamped}} = \max\left(0,\ \min(100,\ \sigma_{\text{LLM}})\right) \quad (9)$$

Table4
Rule-Based Severity Score Vital Sign Increments

Vital Sign Condition	Increment w_k	Emergency Flagged
$SpO_2 < 80\%$	+35	yes
$SpO_2 < 90\%$	+25	yes
$SpO_2 < 94\%$	+10	no
Systolic BP > 200 mmHg	+30	yes
Systolic BP > 180 mmHg	+20	yes
Systolic BP > 160 mmHg	+15	no
Systolic BP < 90 mmHg	+25	yes
Temperature > 105°F	+25	yes
Temperature > 103°F	+15	no
Temperature > 101°F	+5	no
Blood Sugar > 400 mg/dL	+20	yes
Blood Sugar > 300 mg/dL	+10	no
Heart Rate > 140 bpm	+20	yes
Heart Rate > 120 bpm	+10	yes
Heart Rate > 100 bpm	+5	no

$$\sigma_{\text{final}} = \max\Big(0,\ \sigma_{\text{rule}} - 20,\ \min(100,\ \sigma_{\text{rule}} + 20),\ \sigma_{\text{clamped}}\Big) \quad (10)$$

The net LLM adjustment is:

$$\Delta = \sigma_{\text{final}} - \sigma_{\text{rule}}, \qquad \Delta \in [-20, +20] \quad (11)$$

This formula guarantees that no matter what the LLM outputs, the final score cannot deviate from the rule-based score by more than ±20 points, preserving clinical safety.

Stage 3 — Hard Override Rules

After LLM validation, a second deterministic pass (validate triage()) enforces absolute minimum severity floors based on critical vital signs. These overrides cannot be bypassed by any LLM output, and follow the general form:

$$\sigma_{\text{final}} \leftarrow \max(\sigma_{\text{final}}, \theta_k) \quad (12)$$

if vital condition k is met The specific floor values θ_k verified from the code are given in Table 5.

Severity Zone Mapping :

The final severity score maps to a clinical zone and required bed type:

$$zone(p_i) = \begin{cases} \text{Red Zone} \rightarrow \text{Emergency bed}, & \text{if } \sigma_{\text{final}} \geq 90 \\ \text{Critical level} \rightarrow \text{ICU bed}, & \text{if } 70 \leq \sigma_{\text{final}} < 90 \\ \text{Step Down Ward} \rightarrow \text{General bed}, & \text{if } \sigma_{\text{final}} < 70 \end{cases} \quad (13)$$

OPD patients receive a doctor assignment only. An exception applies when the LLM determines that patient vulnerability factors such as advanced age or multiple comorbidities warrant admission, in which case a bed override flag $f_{\text{override}} = 1$ is set.

Table 5
Hard Override Minimum Severity Floors (validate triage)

Condition	Floor θ_k	Emergency Flagged
$SpO_2 < 80\%$	90	yes
$SpO_2 < 90\%$	75	yes
Systolic BP > 200 mmHg	90	yes
Systolic BP > 180 mmHg	70	yes
Temperature > 105-F	80	yes
Temperature > 104-F	65	no
Blood Sugar > 400 mg/dL	65	yes
Heart Rate > 140 bpm	80	yes
Heart Rate > 120 bpm	60	yes
$\sigma_{\text{final}} \geq 60$ (any cause)	NIL	yes

4.4. Doctor Assignment Scoring Function

For each patient p_i requiring department dept(p_i), every available doctor d_j is scored using a composite function implemented in *score_doctor()*. The function rewards department match, specialization relevance, low cognitive load, and clinical experience:

$$\begin{aligned}\text{score}(d_j, p_i) = 50 \cdot \mathbb{1}\left[\text{dept}(d_j) = \text{dept}(p_i)\right] \\ + 30 \cdot \mathbb{1}\left[\text{spec_match}(d_j, p_i)\right] \\ + \left[\left(1 - \min\left(\frac{L_j}{C_j^{\max}}, 1\right)\right) \times 15\right] \\ + \min\left(\frac{exp_years(d_j)}{5}, 5\right)\end{aligned} \tag{14}$$

$dept(d_j)$ represents the department of doctor d_j, $dept(p_i)$ denotes the department assigned by the triage process, $\mathbb{1}$ represents the indicator function, $spec_match(d_j, p_i)$) equals 1 if the doctor's specialization matches the patient's clinical requirement and 0 otherwise, L_j denotes weighted cognitive load of doctor, $C_j^{\max}$ denotes s the maximum patient capacity of doctor d_j, $exp_years(d_j)$ describes years of clinical experience.

The four components contribute as follows. A score of +50 is awarded for an exact department match between the doctor and the triage-assigned department. A score of +30 is awarded if the doctor's specialization string contains any keyword from the known specialization list for that department (`DEPT_TO_SPEC` mapping in the code). Between 0 and +15 is awarded as a load-based bonus, where L_j is the weighted cognitive load defined in Section 4.2 and $C_j^{\max}$ is the doctor's maximum patient capacity (defaulting to 25). Between 0 and +5 is awarded as an experience bonus of one point per five years of experience, capped at 5. The optimal doctor is then selected as:

$$d_j^* = \arg \max_{d_j \in D_{\text{avail}}} \text{score}(d_j, p_i) \tag{15}$$

where $D_{\text{avail}} \subseteq D$ is the filtered set of doctors satisfying availability status = Available and current load $< C_j^{\max}$. When a patient states a preference (previous doctor, specific named doctor, or female doctor), this scoring is applied within the preference-filtered pool rather than the full set.

4.5. Reinforcement Learning Formulation — Bed Allocation Agent

Bed allocation is modeled as a Markov Decision Process (MDP). All hyper-parameter values are taken directly from the constants defined in the Bed Allocation Agent code.

State Space : At each allocation step, the environment state st is an 11- dimensional normalized vector constructed by build state vector():

$$\begin{aligned}s_t = \Bigg[& \frac{\sigma}{100}, \frac{d_{\text{enc}}}{5}, \frac{b_{\text{enc}}}{2}, e_{\text{flag}}, \\ & \frac{n_{\text{exact}}}{2}, \frac{n_{\text{type}}}{10}, \frac{n_{\text{total}}}{50}, \\ & \rho_{\text{dept}}, \rho_{\text{overall}}, \frac{age}{100}, f_{\text{override}} \Bigg]\end{aligned} \tag{16}$$

where σ is the final triage severity score, e_{flag} is the emergency flag, $d_{\text{enc}} \in \{0, 1, 2, 3, 4, 5\}$ encodes the patient's department; $b_{\text{enc}} \in \{0, 1, 2\}$ encodes the required bed type (General, ICU, Emergency); n_{exact} is the count of beds matching both the required type and the patient's department; n_{type} is the count matching the required type in any department; n_{total} is the count of all available beds; ρ_{dept} and ρ_{overall} are fractional occupancy rates for the patient's department and the hospital overall respectively; and $f_{\text{override}} \in \{0, 1\}$ is the LLM bed override flag.

Action Space :

The agent selects from a discrete set of actions corresponding to indices into the LLM-generated candidate list:

$$A = \{0, 1, 2, 3, 4\}, \quad |A| = K = 5 \quad (\texttt{TOP_K} = 5)$$

Candidates that fail the hard constraint validator are masked to $-\infty$ before Q-value scoring, ensuring that they can never be selected regardless of the learned policy.

Policy — Epsilon-Greedy with Exponential Decay :

The agent follows an epsilon-greedy policy. The exploration rate decays exponentially with the number of allocation steps, following the exact formula from the code (lines 849–851)

$$\pi(a \mid s_t) = \begin{cases} \arg\max_a \ Q(s_t, a; \theta), & \text{with probability } 1 - \varepsilon_t \\ \text{Uniform}(A_{\text{valid}}), & \text{with probability } \varepsilon_t \end{cases} \tag{17}$$

$$\varepsilon_t = \varepsilon_{\min} + \left(\varepsilon_{\max} - \varepsilon_{\min}\right) \cdot \exp\left(\frac{-step_{\text{count}}}{N_{\text{decay}}}\right) \tag{18}$$

$$\varepsilon_{\max} = 1.0, \qquad \varepsilon_{\min} = 0.05, \qquad N_{\text{decay}} = 1000$$

Q-Network Architecture (BedDQN) :
The action-value function $Q(s, a; \theta)$ is approximated by a feed-forward neural network with the following architecture:

$$\begin{aligned} Q(s, a; \theta) : \mathbb{R}^{11} \rightarrow \mathit{Linear}(11 \rightarrow 64) \rightarrow \mathit{ReLU} \\ \mathit{Linear}(64 \rightarrow 32) \rightarrow \mathit{ReLU}\ \mathit{Linear}(32 \rightarrow 5) \rightarrow \mathbb{R}^5 \end{aligned} \tag{19}$$

A separate target network with parameters θ^- is maintained for training stability. It is updated via soft updates after every training step.

$$\theta^- \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta^-, \qquad \tau = 0.01 \tag{20}$$

Reward Function :
The composite reward signal is computed by compute reward() and consists of four additive components:

$$R(s_t, a_t) = R_{\text{type}} + R_{\text{dept}} + R_{\text{safety}} + R_{\text{feedback}} \tag{21}$$

The exact reward values are discussed in Table 6.

Table 6
Reward Function Components (exact values from implementation)

Component	Value	Condition
R_{type}	+10.0	Bed type matches the severity-required type
R_{dept}	+5.0	Bed department matches patient's department
R_{safety}	-50.0	$\sigma(p_i) \geq 90$ and bed type is General
R_{safety}	-30.0	$\sigma(p_i) \geq 70$ and bed type is General
R_{unsafe}	-20.0	Risk level is HIGH and no doctor review was done
R_{approved}	+15.0	Doctor-in-the-loop approves the allocation
R_{rejected}	-20.0	Doctor-in-the-loop rejects the allocation

In abstract form, this reward structure maps to the general formulation:

$$R = -\alpha W - \beta O + \gamma S \tag{22}$$

where W represents the latent cost of a sub-optimal bed assignment that may require future patient transfer, O represents the resource overload or clinical safety mismatch penalty, and S represents the service quality reward from bed type correctness, department match, and doctor approval. The coefficients $\alpha, \beta, \gamma > 0$ balance these three terms.

DQN Training — Bellman Loss with Experience Replay :
Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer $\mathcal{D}$ of capacity 10,000. Once the buffer contains at least 32 transitions, a mini-batch of 32 samples is drawn uniformly at random, and the following squared Bellman error loss is minimized:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\left(r + \gamma \cdot \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right] \tag{23}$$

with the following exact hyper-parameter values from the code:

$$\gamma = 0.95, \ \eta_{\text{Adam}} = 1 \times 10^{-3}, \ |batch| = 32, \ |\mathcal{D}| = 10{,}000$$

Table 7
Hospital Workflow Database Schema Summary

Table	Key Attributes	Purpose
patients	patient_id, age, gender, chronic_conditions, allergies, medical_history	Patient demographics and medical background
visits	visit_id, visit_type, severity_score, department, emergency_flag, visit_status	Per-visit clinical metadata and current status
vitals	bp, oxygen, temperature, heart_rate, sugar, respiration_rate	Time-stamped physiological measurements
triage_results	severity_score, department_suggested, emergency_flag, secondary_departments	Triage pipeline outputs per visit
doctor_assignments	doctor_id, assignment_type (Primary/Consult), priority_level, room_no	Doctor-patient allocation records
doctors	specialization, department, current_load, max_capacity, experience_years, gender	Doctor profiles and real-time availability
bed_resources	bed_type, department, occupancy_status, priority_score	Real-time bed inventory and allocation state
diagnosis	diagnosis_text, disease_category, severity, followup_required	Doctor-recorded diagnostic outcomes
prescriptions	drug_name, dosage, frequency, duration	Medication orders per visit
billing	consultation_fee, bed_charges, medicine_cost, total_amount, payment_status	Generated bills and payment tracking
bed_allocation_log	state_json, ncandidates_json, selected_bed_id, reward, doctor_feedback	RL training data for bed allocation agent

5. Experimental Setup and Implementation

5.1. Dataset used

The architecture employs two types of data sets: artificial hospital workflow data and the publicly available medical Visual Question Answering data set. The hospital workflow database can be seen in Table 7

5.1.1. Hospital Workflow Dataset (Synthetic)

A robust database system was created using artificially created data to represent a hospital setting. The database is maintained on MySQL using Aiven Cloud and includes fifteen related tables that cover the life cycle of patients. The tables, along with the approximate number of entries, are listed below in Table 5.1

The synthetic data was created so that all five severity zones could be included – Red Zone, Critical Care, Step Down Ward, Inpatient Department (IPD), and Outpatient Department (OPD). This made sure that there was enough variation not only for the RL training but also for testing the pipeline. Types of visits that were covered in the data included Emergency, IPD, and OPD. Patients' records consisted of various ages, comorbidities, and different combinations of vitals to make sure that all branches of the triage rule engine were exercised. The availability of real-world data from hospitals is hindered by strict rules about patient privacy, confidentiality issues, and limited access to health care data. Therefore, the need to create a synthesized dataset that would mimic realistic workflow patterns of hospitals but ensure diversity of data, scalability, and safety.

5.1.2. Medical VQA Dataset (SLAKE)

We have taken the medical dataset from Dataset Link

For the Medical VQA model optimization, the SLAKE dataset (SLAKE-VQA-English, available at HuggingFace via mdwiratathya/SLAKE-vqa-English) was used. SLAKE is a bilingual medical VQA dataset consisting of medical

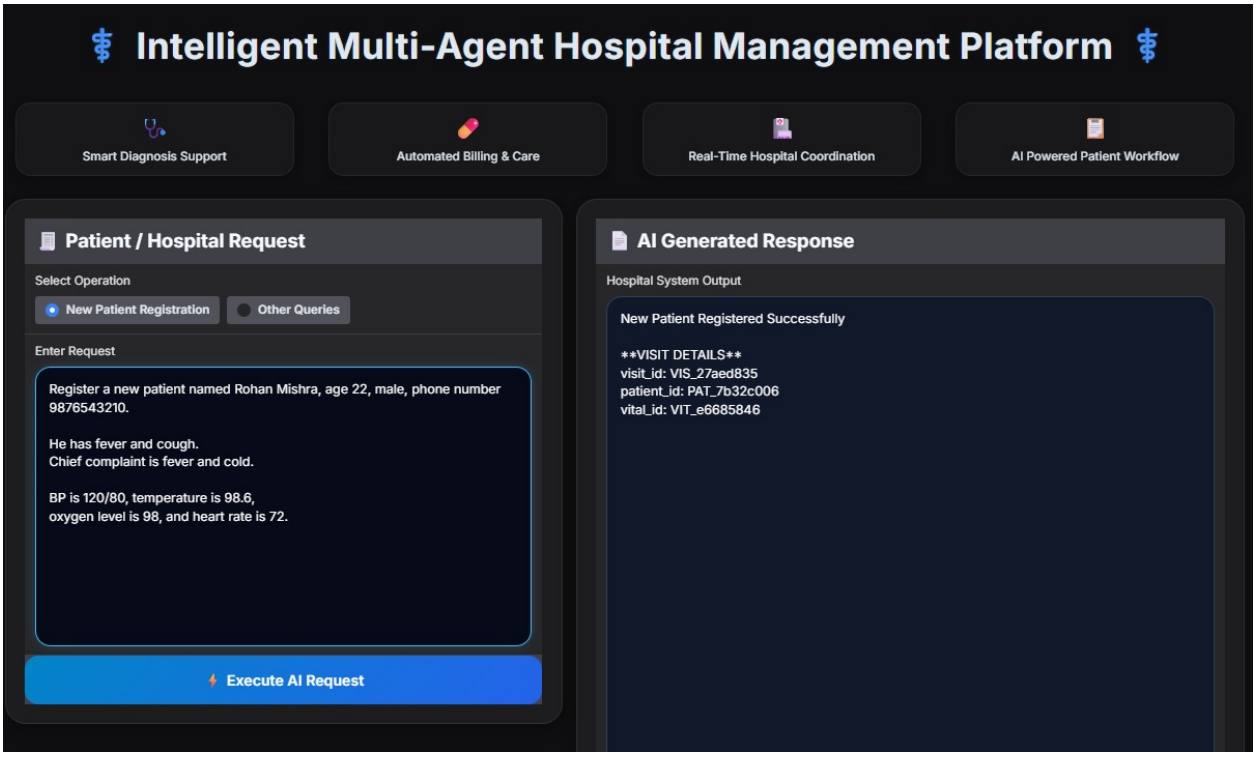


Figure 6: Patient entry

imaging radiology pictures with clinical questions and answers related to them, including X-ray, CT, and MRI imaging techniques for different body parts. The English version of the SLAKE dataset included in this research encompasses the train and test datasets. These include closed questions (yes/no type) and open-ended questions requiring identification of some part or pathology. To reduce the problem of yes/no prediction bias due to the imbalanced classes, a factor of 2 oversampling was applied in the last training process.

5.2. Workflow

Step 1: Patient Entry

The procedure begins with the interaction between the patient and the healthcare service using an app either from the phone or online platform. In this case, the patient gives relevant details such as his/her name, age, symptoms, and even history (if possible). Later on, the patient's data will be authenticated to get a unique ID. The example is provided in Fig. 6

Step 2: Intake via Patient Interaction Agent

After the patient is registered, the Patient Interaction Agent uses the natural language processing approach to gather and process the entered data. The agent translates the free-text description of symptoms into a structured form that can be analyzed by the system. The identification of important symptoms, mapping them to appropriate health categories, and ensuring that all necessary data about the patient has been entered are some of its functions.

Step 3: Triage and Severity Assessment

At this stage, triage is performed in order to determine how urgent or serious the health state of the patient is. Depending on their severity, patients can be divided into several categories, such as critical, moderate, or mild using medical criteria or machine learning methods. Then, the priority rating of each patient is estimated depending on the severity of their symptoms, age, and previous history. Such categorization allows hospitals to prioritize and optimize resources to provide faster and higher quality services to their clients. An example to triage method is shown in Fig. 7

Step 4: Doctor Assignment via Care Coordination Agent

After the evaluation of the patients' severity, the Care Coordination Agent assigns a doctor who should work with them. In addition to the patient's severity, there are several other aspects which should be taken into account at this stage. They include required qualifications of physicians, workload, and physician availability. In order to assign a doctor and reduce wait times, optimization methods such as reinforcement learning or simple algorithms can be used.

Step 5: Doctor Interaction Agent Processing

Doctor Interaction Agent represents a platform which provides interaction between the physician and the patient. Within this module, doctors evaluate patients' medical history, triage result, and make a decision about diagnosis and required measures for recovery. When needed, they will

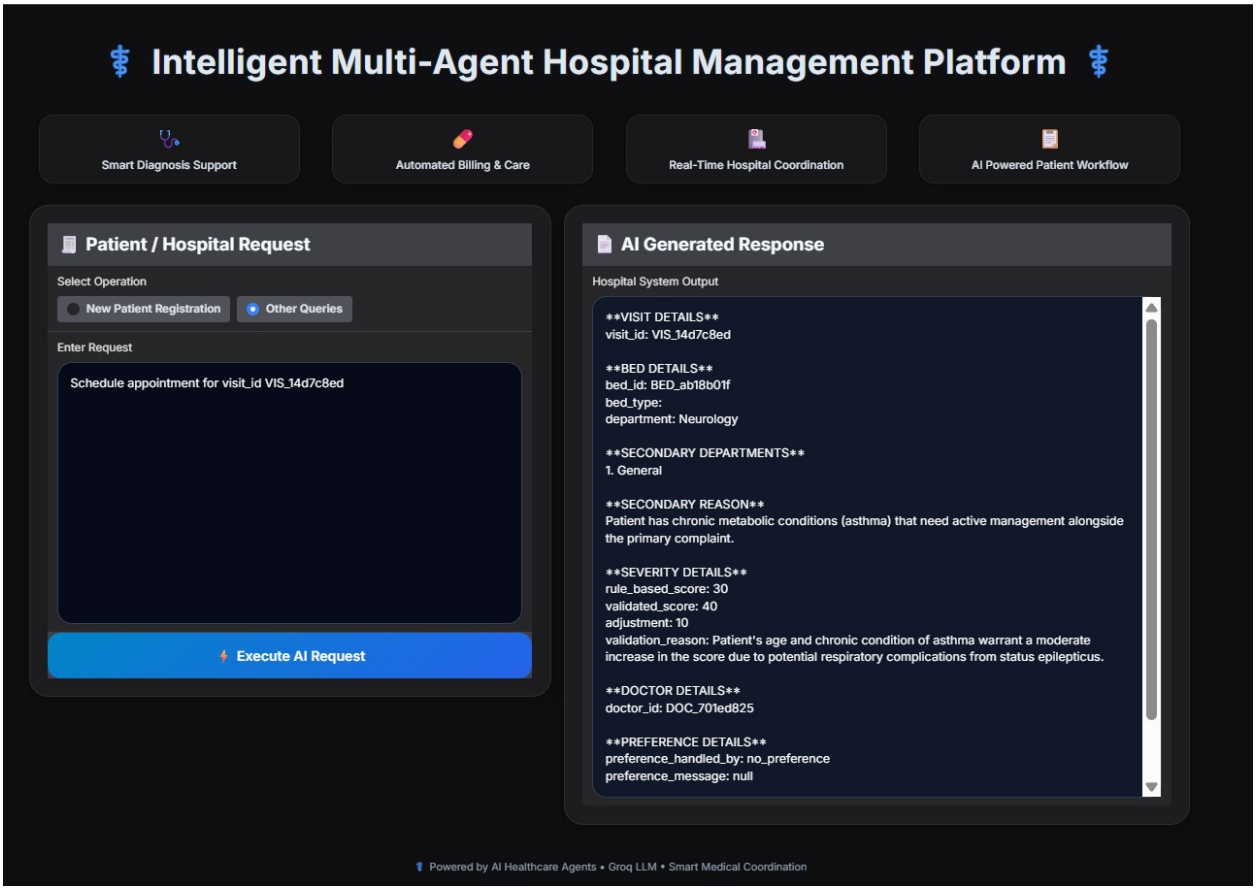


Figure 7: Triage scheduling

decide whether it is required to provide hospitalization to the patient and conduct some procedures. Information regarding diagnosis and recommended actions will be securely stored in the system.

Step 6: Bed Allocation via Bed Allocation Agent

In case hospitalization is needed, the Bed Allocation Agent is used to facilitate the process of assigning beds in the hospital. In the process of allocating beds, the agent keeps track of all available beds in different wards of the hospital. According to the patient's condition, medical needs, and available hospital resources, a suitable bed will be assigned. In addition to that, optimization techniques are applied to ensure minimal waiting time and efficient allocation of hospital beds.

Step 7: Continuous Monitoring and Updates

Continuous monitoring and updating of patient information is the final step of the proposed workflow. Data collected from each and every agent within the multi-agent system is always in sync, hence facilitating coordination between multiple hospital departments. In this way, the patient's status and his or her medical treatment is constantly monitored and updated. This is also helpful in providing feedback and reinforcement to improve the future decision-making processes.

5.3. Communication Protocol

Inter-agent communication in this case does not depend on direct P2P messaging or common message bus. Instead, communication will be performed through database, using a function call approach. Function call interactions are implemented using 3 types of interaction patterns:

5.3.1. Pattern 1 – Direct Function Call (Synchronous Delegation):

If the Patient Query Agent needs to start the triage and scheduling process for a particular patient, it sends the *CareCoordinationAgent.handle(data)* function call with a dictionary containing the fields $visit_id$ and possible $patient_preference$. The CTSA goes through its complete pipeline synchronously and returns a resulting dictionary. Similarly, CTSA uses the *run_bed_allocation(conn, visit_id, severity, dept_short, patient_id, llm_override)* function call to start the Bed Allocation Agent workflow. This pattern supports loose coupling between agents, where each agent is given only the basic context token ($visit_id$) and then independently obtains any necessary data from the database.

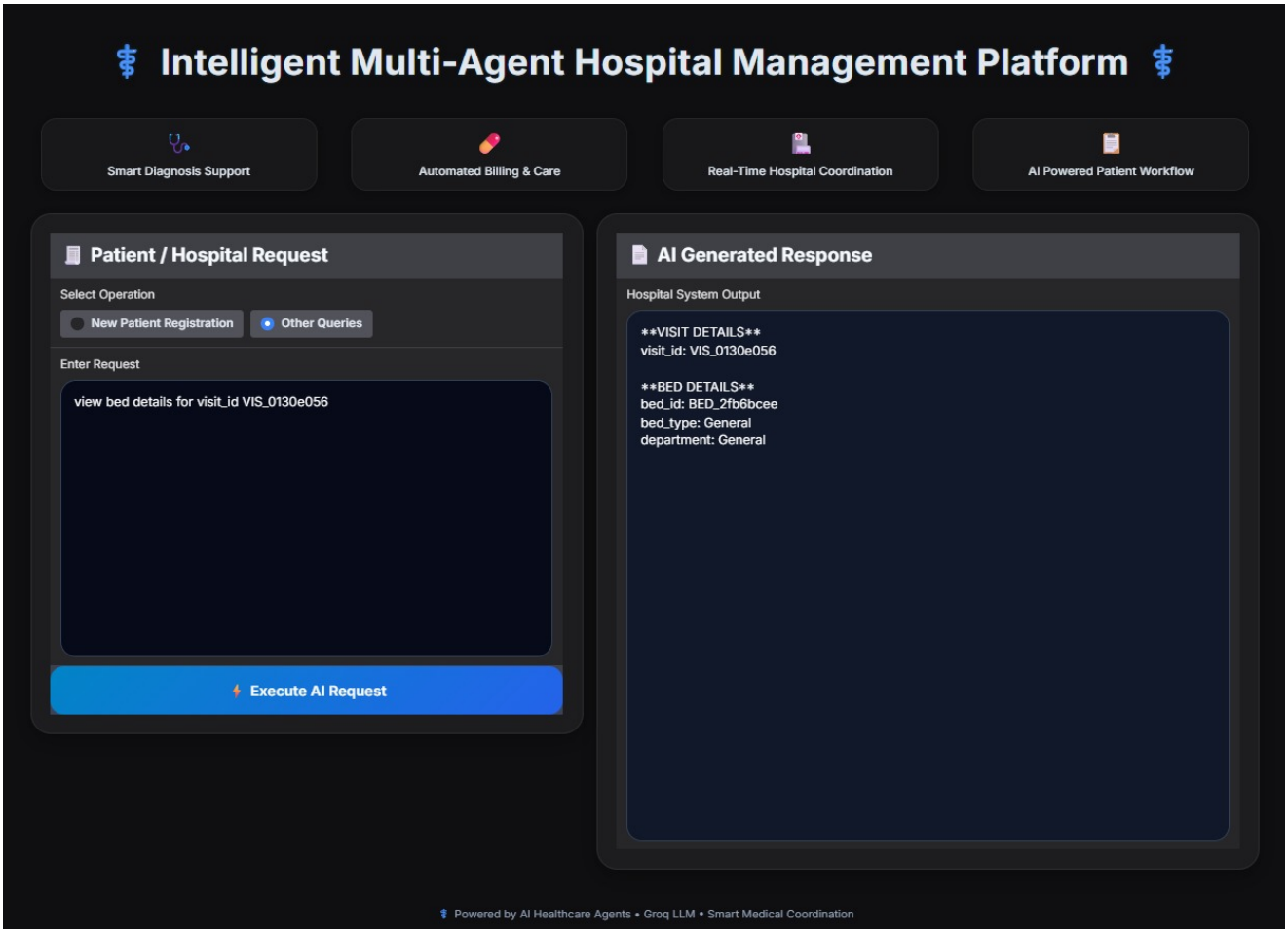


Figure 8: Bed Allocation

Pattern 2 – Database as Shared Blackboard:

MySQL database acts as a common blackboard, which all agents may read from and write to. For example, when Patient Intake Agent writes a visit record, it is immediately available to CTSA via SELECT query. When CTSA writes a triage record, then severity rating and department name instantly become available to Bed Allocation Agent. Similarly, when Bed Allocation Agent sets status field to 'Occupied', the billing system immediately sees the number of occupied beds in the ward. This pattern enables temporal decoupling – agents do not need to be located at the same machine nor run simultaneously, and results will be stored after program termination.

Pattern 3 - LLM Tool Call Protocol (JSON Function Dispatch):

In each individual agent, the large language model (LLM) communicates with the backend tools using a JSON function dispatch protocol. In essence, the LLM is first presented with a set of tool definitions (name, description, and parameter schema) and responds with a JSON structure that denotes the tool to be used and the parameters associated with it. The agent runtime then checks whether the tool is defined in the registry, deserializes the arguments from JSON, invokes the Python function specified in the definition, and appends the result as a tool-result message to the conversation history. The cycle continues until a response is generated that contains no tool calls. The function dispatch protocol has been implemented using Groq tool-use API for Groq Agents and Ollama native format for local deployments. JSON-based de-serialization of tool call parameters and registry check prevent hallucinated function dispatch.

5.4. Evaluation Metrics

The multi-agent hospital management system developed by him has been assessed through several performance indicators based on coordination for evaluating efficiency and reliability in inter-agent communication and decision-making process. As evidenced from the evaluation outcomes, the developed model has achieved the Task Completion Rate equal to 82.50%, illustrating the capability of the system to effectively manage hospital administration processes. The framework sustained the Average Processing Time equal to 5.12 seconds, signifying efficient performance and effective decision-making. In addition, the developed

framework achieved the Query Handling Success Rate of 79.15%, which confirms its effectiveness in handling interactions associated with hospital and patients. Regarding the assessment of collaborative intelligence among the agents, it is worth mentioning that the framework has shown the Coordination Efficiency of 87.50%, which illustrates synchronized and coordinated cooperation between the agents. Besides, the relatively low Agent Conflict Rate equal to 5.00% indicates that there was no conflict among the agents in the process of decentralized resources allocation and scheduling decision-making. Moreover, the achieved Communication Overhead of 0.88 signifies efficient communication among the agents with low information transfer cost, whereas the Convergence Rate of 96.4% confirms the learning stability and adaptation potential of reinforcement learning architecture. The reported evaluation metrics were obtained using manually generated multiple patient-query scenarios, and as the proposed framework operates in a dynamic environment, performance values may vary depending on patient load and operational complexity. These evaluation outcomes collectively demonstrate the effectiveness, scalability, and coordination capability of the proposed intelligent hospital administration framework.

6. Results and Discussion

6.1. Experimental Setup :

The multi-agent-based healthcare management system suggested was analyzed based on simulated workflow data representing a variety of healthcare situations. In particular, the daily arrival rate of patients ranged between 100 and 500, demonstrating the scalability and adaptability of the suggested solution. The simulated hospital contained 10 to 30 doctors assigned to different departments such as General, Emergency, and Specialists departments, which represented various scenarios and priorities that might be encountered in real-life conditions.In order to determine whether the developed healthcare management system is effective and efficient, some performance indicators were considered during the experiment. First, Average Waiting Time (W) helped to find out how fast patients received medical treatment once registered. Second, the Doctor Utilization Rate (U) was chosen to identify how efficiently healthcare specialists worked in patient care. Third, Patient Satisfaction Score (S) was used to measure the satisfaction level among patients and reflected their experience in waiting for doctors and receiving medical treatment. Besides, two more performance criteria – System Throughput (T) and Scheduling Efficiency (E) – measured the overall system efficiency in processing patients and allocating resources.

6.2. Performance Comparison :

The performance comparison is discussed in Table 8. Here, "traditional healthcare systems" refer to those systems within hospitals that have been operating under conventional systems where operations have primarily been managed manually, with static processes, centralized decision making, and minimal automation. The traditional healthcare system does not have intelligent decision support, intelligent inter-departmental communication, adaptability for resource management, and patient management capabilities. They thus experience issues like slow patient reactions, poor bed allocation, communication breakdowns between different departments, and lack of scalability during emergencies or peak times.

Table 8
Performance Comparison of Scheduling Systems,

Metric	Traditional System [11]	AI Scheduling System [13], [18]	Proposed Agentic System
Avg Waiting Time (minutes)	65	38	18
Doctor Utilization (%)	58%	74%	89%
Patient Satisfaction (0–10)	5.8	7.2	9.1
Throughput (patients/day)	210	290	360
Emergency Handling Delay (min)	25	14	6
Appointment No-Show Handling	Poor	Moderate	Adaptive (Agent based)

Harrison et al. [11] highlights the drawbacks and inefficiencies associated with traditional approaches to handling data in healthcare information systems. Hick et al. [12] emphasizes the limitations in manually assigning resources in times of disasters and managing emergencies in healthcare management systems. Junaid [13] explains the transition from the traditional management systems of health care into modern intelligent systems. In a similar approach, Sutherland [18] highlights the importance of intelligent hospital operations in place of traditional methods.

Unlike the traditional approaches, the current model presents a healthcare management system based on a multi-agent approach that is empowered by the use of AI and comprises the following components: the Patient Interaction Agent, Doctor Interaction Agent, Care Coordination Agent, and Bed Allocation Agent. These components collaborate together in order to perform automated operations in the healthcare industry, improve communication and resource management, and provide intelligent assistance using artificial intelligence and language models. The move towards a more intelligent multi-agent model is also confirmed by references [4], [6], [10], and [14].

6.3. Performance Analysis and Agent Responsibilities :

The influences of each agent were determined based on the effect of each agent on the performance metrics of the entire system when running multiple simulations involving

Table 9
Agent Roles and Impact on System Metrics

Agent	Primary Role	Impact on Metrics
Patient Interaction Agent	Intake, triage, appointment requests	Reduces input delays, improves satisfaction
Doctor Assistant Agent	Doctor scheduling, load balancing	Improves utilization, reduces idle time
Care Coordination Agent	Global optimization & emergency handling	Minimizes waiting time, improves throughput

patients' queries. Agent executes its own specific function in the hospital process, and the influence of each agent is seen from improvements in the performance metrics of the entire system such as Task Completion Ratio (82.50%), Average Processing Time (5.12 s), Query Processing Rate (79.15%), Coordination Efficiency (87.50%), Agent Conflict Ratio (5.00%), Communication Overhead (0.88), and Convergence Rate (96.4%).

The Agent Contribution Analysis is discussed in Table 9.

6.4. Discussion :-

Based on the results obtained from the experiments described above, inefficient management in healthcare institutions utilizing traditional management approaches is evident due to various reasons. First, the traditional method largely relies on manual scheduling of processes, hence increasing the burden of work for the administrative department. Delays in providing timely treatment to patients become common because of inefficiency. Moreover, the lack of an optimization approach within the traditional approach makes the system less flexible when changes such as emergencies occur in the organization. Poor prioritization of emergencies causes delays and inefficiency in service delivery. Communication and coordination issues within departments hinder efficient use of resources.

In contrast to the traditional health system, there are several benefits associated with the newly designed multi-agent health system, and they all emanate from the unique design of agents used within the framework. The individual agents operate independently but have the capability to cooperate and coordinate their activities to ensure efficient management processes within the hospitals. Examples of some of these tasks include Patient Agent, which collects patient data, Doctor Agent responsible for managing workload of physicians, and the Coordination Agent responsible for optimizing decisions across the board.

Another important benefit of the proposed system relates to its adaptive nature. Multi-agents may quickly adjust their operations in cases of emergency to reorder patients' priority order and redistribute available doctors accordingly. Updating of schedules occurs continuously with consideration of current conditions in the hospital to ensure effective patient flow without any delays. Such flexibility contributes to increased efficiency of operations especially in those periods when there is high demand on patients.

In addition, it should be noted that the proposed architecture demonstrates good scalability. With a growing number of patients, operation efficiency will not decrease, since the process of work will be performed by several agents working together, which implies absence of a bottleneck phenomenon. This makes possible a further extension of the medical system in accordance with changing requirements of hospitals. Hence, the agentic approach provides better solutions compared to conventional hospital information systems.

6.5. Key Findings :-

Analytical assessment of the proposed multi-agent based healthcare management system reveals the importance of each component in the process of efficient optimization of the process. In order to reveal the role of different components, an ablation-style experiment was performed in which the system behavior was analyzed after disabling some components such as memory, reinforcement learning techniques, coordination and particular agent roles. The results suggest that all these components are necessary in order to reduce patient waiting time, even distribution of workload for physicians, efficient use of resources, and patient satisfaction.

6.5.1. Effect of Removing Memory Mechanism:

When there is no memory, the system will not be able to take into account previous records, diagnoses made, and previous treatment given by treating the patients' request individually. The doctors will find themselves unable to get full information, thus taking more time and asking for more information. There will be no personalization of care, repetitions will take place concerning scheduling, resource usage, leading to longer waiting periods and reduced satisfaction.

6.5.2. Effect of Removing Reinforcement Learning (RL):

The principle behind reinforcement learning forms the foundation of adaptive decision-making and optimization in the system, especially when it comes to operations such as assigning doctors to shifts, scheduling them for duties, and allocating medical resources. Without reinforcement learning, the system depends only on scheduling through fixed rule-based scheduling procedures that do not factor in changes in the dynamics of the hospital operations due to emergencies, varying rates of inflows of patients, and uneven workloads of the doctors. This can result in some doctors being overwhelmed with patients whereas other doctors have few to attend to, resulting in inefficient allocation of doctors' services as well as long patient wait times.

6.5.3. Effect of Removing Patient Interaction Agent:

The Patient Interaction Agent handles the gathering of patient information and the conversion of the unstructured symptom descriptions to structured clinical information using natural language processing. Without the agent, the process of data entry would involve manual intervention and thus be subject to inconsistencies and an increased likelihood of erroneous data entry. This results in delays in the triage and diagnosis process because the clinician will require extra time to decode and verify the details of the patient's symptoms. The process of automatic extraction of symptoms by the agent also enables a faster and more accurate assessment of the severity of the condition.

6.5.4. Effect of Removing Care Coordination Agent:

The Care Coordination Agent helps in achieving effective physician assignment and coordination between various departments. Without this module, physicians' assignments may prove to be stagnant or manual, leading to inequity in their work distribution and poor resource management. The emergencies of patients may face a delay in seeing the specialists assigned to them, and there could be numerous scheduling conflicts. However, in the absence of coordination between departments, communication becomes ineffective, thus leading to inefficiencies in treatment schedules. On the other hand, implementation of the Care Coordination Agent module promotes effective and dynamic assignment of physicians depending on their specialization and availability.

6.5.5. Effect of Removing Doctor Interaction Agent:

The Doctor Interaction Agent is responsible for ensuring effective communication between doctors and their patients in a systematic manner, while documenting their consultations in an orderly fashion. Without this particular agent, the process of consultation becomes fragmented, and manual handling of the consultations is done. These factors might lead to ineffective documentation, late updating of data, and poor coordination during diagnosis, pharmacy, and admission stages. Clinicians can spend more time in paperwork than in patient care. The addition of the Doctor Interaction Agent will enhance communication, document consultations effectively, and improve treatment continuity within the system.

6.5.6. Effect of Removing Bed Allocation Agent:

The Bed Allocation Agent is one of the vital parts for ensuring bed resource allocation. Without the part of the Bed Allocation Agent, the process of allocating beds would be very lengthy for patients before being admitted since manual bed allocation would take place and would possibly lead to delay when the hospital is busiest. In addition, there may be ineffective bed allocation where the beds might be underutilized or crowded in certain places. Such cases arise due to the nature of bed allocation using the Beds Allocation Agent.

According to the research results, the suggested healthcare system, based on the agency approach, exhibits better performance due to collaboration of intelligent agents, adaptation, learning, and memory-driven decisions. Contribution of each of these elements to improved performance is illustrated through decreased waiting time for patients, increased efficiency of physicians, efficiency improvements, and higher satisfaction levels of patients. Exclusion of any of these components leads to lower performance levels, thus indicating improvements made in comparison with current hospital management systems.

7. Ethical Considerations

However, the implementation of such an AI-based multi-agent healthcare system will raise various issues from the ethical standpoint. This section will highlight those key aspects as patient confidentiality, fairness, accountability, and regulation compliance concerning the given agentic model of a healthcare system.

7.1. Patient Privacy and Data Security :-

The system operates using extremely sensitive data that consist of patient history, symptoms, scheduling, and interaction inputs, whereby the last is handled mainly by the Patient Interaction Agent. The following approaches are adopted to secure the data and maintain their privacy:

(a) Data minimization: the system will only collect and analyze data that are critical for the functionality of each individual agent. The Doctor Assistant Agent, for example, can access only scheduling data, without being able to access any medical history.

(b) Encryption: the data will be secured using encryption both while stored in the database and when transferred, using the TLS/HTTPS protocol.

(c) Role-based access control: access to relevant data will be limited depending on each role, as outlined below:

(1) Patient Interaction Agent: access to patient intake data

(2) Doctor Assistant Agent: access to scheduling and workload data

(3) Care Coordination Agent: access to consolidated and anonymized data

7.2. Fairness and Bias Mitigation :

AI algorithm deployment in the healthcare field might accidentally cause some form of bias when it comes to scheduling and prioritization processes, which would be performed by the Care Coordination Agent. This bias could arise due to hospital records, rewards used in reinforcement learning models, as well as varying use of the system among different patient demographics. In order to reduce this problem, it is essential that datasets used in training include patients of all age, social class, and illness types to ensure equality and fairness. Moreover, regular audits should be carried out to detect possible discrimination concerning waiting time or uneven physician distribution.

7.3. Transparency :

Any decisions related to healthcare should always be open and clear in order to promote trust between the patients and doctors. However, using AI technologies for healthcare poses many challenges because sometimes reinforcement learning algorithms exhibit behaviors described by the term "black box" which means that such algorithms are not easy to comprehend, and automated processes used for scheduling do not clarify how certain prioritizations are done. Thus, in order to increase the transparency of the decisions made by the suggested system, there are special transparency features introduced into the model. First, agents create extensive decision logs which allow explaining the reasons of why one patient is prioritized over another and why one doctor has been allocated for helping the patient. Moreover, physicians have an option to override any decision made by the Doctor Assistant Agent. Finally, the Care Coordination Agent keeps track of systemic and policy decisions in audit logs.

7.4. Regulatory Compliance :

In order to be fully compliant with all healthcare regulations and laws within the regions where the system will be implemented, the design will ensure full compliance with all such relevant legislations, namely HIPAA in the USA, GDPR in Europe, and DISHA in India. In order to ensure that the regulatory compliance as well as the patients' privacy is protected, there are certain aspects that should be adhered to. One of those is obtaining the consent from the patient for accessing his/her information as well as for using the Artificial Intelligence for making decisions related to healthcare treatment. In order to grant such an access, the patients will have access and erasure rights, so that they can manage and delete any personal information that is saved on the platform. The system also ensures data localization, as patient information will be stored only within the country where the system will be used.

8. Conclusion and Future Work

8.1. Conclusion :-

In this research, the conceptual model for healthcare management using multiple AI agents is presented, consisting of three key components: the Patient Interaction Agent, the Doctor Assistant Agent, and the Care Coordination Agent. The architecture is designed to overcome the inefficiencies in conventional hospital structures through decentralization, flexibility, and reinforcement learning-enabled decision-making.

The Patient Interaction Agent streamlines patient registration, symptom evaluation, and appointment booking. The Doctor Assistant Agent maximizes physician availability and workload balance. Lastly, the Care Coordination Agent acts as the global optimizer, facilitating coordination between various agents, handling emergency situations, prioritizing essential cases, and maintaining overall system efficiency.

Compared to centralized AI models, the agentic approach enables scalable and flexible decision-making in which individual agents work independently but collaborate to achieve an optimal solution. Reinforcement learning enhances adaptability and continuous learning from real-world experiences. Overall, the results show that agentic AI technology has the potential to revolutionize healthcare operations.

8.2. Future Work :-

Despite the fact that the proposed system shows itself quite successfully in terms of automation and hospital management, there are also some ways to develop and enhance it in order to make it better perform in real environments and situations. First of all, the use of wearables (smartwatches, biosensors, etc.) would allow for real-time health tracking, implementing preventive healthcare measures and reducing the need to visit hospitals. Another interesting feature is related to the implementation of predictive diagnosis systems that will use Artificial Intelligence algorithms to analyze patients' symptoms, medical records, and other types of data in order to diagnose diseases at an early stage. With this

approach implemented, doctors will be able to recognize patients with heart diseases and diabetes.

Also, it might be interesting to implement federated hospital networks in which multiple hospitals work together while ensuring the privacy of patients and confidentiality of their personal information. This solution can help with improved resource allocation, decentralized exchange of information between hospitals, equal distribution of patients, and crisis management in case of pandemic. Moreover, the use of reinforcement learning and multi-criteria optimization techniques can result in improved decision-making processes in terms of waiting time, fairness, and cost-effectiveness of decisions. Also, the application of deep reinforcement learning and the use of reward shaping technique will definitely be interesting and applicable in this situation.

Integration of EHR systems with the proposed healthcare solution will allow for the access of the patients' complete medical histories, which will lead to better diagnosing process, treatment plan creation, and schedule making in the context of hospital management. Moreover, the use of Explainable AI techniques will increase the transparency and comprehensibility of the process for clinicians in terms of understanding how the software makes decisions.

In general, Agentic AI shows quite a lot of perspectives in modern medicine because of its ability to perform collaborative and distributed tasks and to create intelligent healthcare management system. Future healthcare systems built on the basis of this technique will become more self-regulating and adaptive, allowing us to allocate healthcare resources in a proper way and provide personalized treatments with minimal human interference.

8.3. Summary

The multi-agent system developed in this paper builds a strong basis for future healthcare systems. The use of AI, reinforcement learning, and the development of an agent-based approach provides evidence for the possibility of developing smart, effective, and ethical healthcare systems that will significantly improve efficiency and results.

Author Contribution

K.M., M.K.P., and S.T. conducted the literature review, designed and developed the proposed agentic AI-based hospital management system, performed the experiments, analyzed the results, prepared the figures and tables, and drafted the manuscript. A.K.P. conceived the research idea, supervised the overall study, provided technical guidance throughout the project, reviewed and refined the methodology and results, and critically revised the manuscript. All authors contributed to discussions, reviewed the manuscript, and approved the final version for publication.

Funding Information

We thank SOA University for sponsoring the article processing charge.

Data Availability Statement

The datasets used and generated during the current study are publicly available. The SLAKE dataset can be accessed at:

`https://huggingface.co/datasets/BoKelvin/SLAKE`

The synthetic hospital dataset generated for this study is available at:

`https://github.com/Shreya7tripathy/Agentic-AI-System-for-Hospital`

No real patient data or hospital records were used in this study.

References

[1] Abdul, S., Adeghe, E.P., Adegoke, B.O., Adegoke, A.A., Udedeh, E.H., 2024. Ai-enhanced healthcare management during natural disasters: conceptual insights. Engineering Science & Technology Journal 5, 1794–1816.

[2] Adawadkar, M., 2025. The evolution of generative and agentic ai: From rule-based systems to autonomous intelligence. American Journal of Technology Advancement Vol 2.

[3] Ahmed Kamal, M., Ismail, Z., Shehata, I.M., Djirar, S., Talbot, N.C., Ahmadzadeh, S., Shekoohi, S., Cornett, E.M., Fox, C.J., Kaye, A.D., 2023. Telemedicine, e-health, and multi-agent systems for chronic pain management. Clinics and Practice 13, 470–482.

[4] Alaskar, K., Tamboli, F.A., Memon, S., Gulavani, S.S., Mudalkar, P.K., Desai, V., Rasal, P.R., Tandale, P.G., Gaikwad, D.T., 2022. Artificial intelligence (ai) in healthcare management. J Pharm Negat Results 10.

[5] Amiri, Z., 2024. Leveraging ai-enabled information systems for healthcare management. Journal of Computer Information Systems , 1–28.

[6] Bao, Z., Liu, Q., Guo, Y., Ye, Z., Shen, J., Xie, S., Peng, J., Huang, X., Wei, Z., 2024. Piors: Personalized intelligent outpatient reception based on large language model with multi-agents medical scenario simulation. arXiv preprint arXiv:2411.13902 .

[7] Cao, W., Zhang, Q., Liu, J., Liu, S., 2026. From agents to governance: Essential ai skills for clinicians in the large language model era. Journal of Medical Internet Research 28, e86550.

[8] Dhasarathan, C., Shanmugam, M., Kumar, M., Tripathi, D., Khapre, S., Shankar, A., 2024. A nomadic multi-agent based privacy metrics for e-health care: a deep learning approach. Multimedia Tools and Applications 83, 7249–7272.

[9] Dhatterwal, J.S., Naruka, M.S., Kaswan, K.S., 2023. Multi-agent system based medical diagnosis using particle swarm optimization in healthcare, in: 2023 International Conference on Artificial Intelligence and Smart Communication (AISC), IEEE. pp. 889–893.

[10] Fan, Z., Wei, L., Tang, J., Chen, W., Siyuan, W., Wei, Z., Huang, F., 2025. Ai hospital: Benchmarking large language models in a multi-agent medical interaction simulator, in: Proceedings of the 31st International Conference on Computational Linguistics, pp. 10183–10213.

[11] Harrison, M.I., Koppel, R., Bar-Lev, S., 2007. Unintended consequences of information technologies in health care—an interactive sociotechnical analysis. Journal of the American medical informatics Association 14, 542–549.

[12] Hick, J.L., Hanfling, D., Cantrill, S.V., 2012. Allocating scarce resources in disasters: emergency department principles. Annals of emergency medicine 59, 177–187.

[13] Junaid, S.B., Imam, A.A., Balogun, A.O., De Silva, L.C., Surakat, Y.A., Kumar, G., Abdulkarim, M., Shuaibu, A.N., Garba, A., Sahalu, Y., et al., 2022. Recent advancements in emerging technologies for healthcare management systems: a survey, in: Healthcare, MDPI. p. 1940.

[14] Li, J., Lai, Y., Li, W., Ren, J., Zhang, M., Kang, X., Wang, S., Li, P., Zhang, Y.Q., Ma, W., et al., 2024. Agent hospital: A simulacrum of hospital with evolvable medical agents. arXiv preprint

arXiv:2405.02957 .

[15] Pandey, H.G., Amod, A., Kumar, S., 2024. Advancing healthcare automation: Multi-agent system for medical necessity justification, in: Proceedings of the 23rd Workshop on Biomedical Natural Language Processing, pp. 39–49.

[16] Reffad, H., Alti, A., Almuhirat, A., 2023. A dynamic adaptive bio-inspired multi-agent system for healthcare task deployment. Engineering, Technology & Applied Science Research 13, 10192–10198.

[17] Sehrawat, S.K., 2023. Intelligent healthcare management: Advancing healthcare with integrated ai and ml solutions. International Journal of Research in Medical Sciences and Technology 17, v16i01.

[18] Sutherland, J., van den Heuvel, W., 2006. Towards an intelligent hospital environment: adaptive workflow in the or of the future, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), IEEE. pp. 100b–100b.

[19] Wang, Z., Zhu, Y., Zhao, H., Zheng, X., Sui, D., Wang, T., Tang, W., Wang, Y., Harrison, E., Pan, C., et al., 2025. Colacare: Enhancing electronic health record modeling through large language model-driven multi-agent collaboration, in: Proceedings of the ACM on Web Conference 2025, pp. 2250–2261.