healthcare

MDPI

Article

MedScrubCrew: A Medical Multi-Agent Framework for Automating Appointment Scheduling Based on Patient-Provider Profile Resource Matching

Jose M. Ruiz Mejia *,† **and Danda B. Rawat** *,†

Department of Electrical Engineering and Computer Science, Howard University, Washington, DC 20059, USA
* Correspondence: jose.mejia@bison.howard.edu (J.M.R.M.); danda.rawat@howard.edu (D.B.R.)
† These authors contributed equally to this work.

Abstract

Background: With advancements in Generative Artificial Intelligence, various industries have made substantial efforts to integrate this technology to enhance the efficiency and effectiveness of existing processes or identify potential weaknesses. Context, however, remains a crucial factor in leveraging intelligence, especially in high-stakes sectors such as healthcare, where contextual understanding can lead to life-changing outcomes. **Objective:** This research aims to develop a practical medical multi-agent system framework capable of automating appointment scheduling and triage classification, thus improving operational efficiency in healthcare settings. **Methods:** We present *MedScrubCrew*, a multi-agent framework integrating established technologies: Gale-Shapley stable matching algorithm for optimal patient-provider allocation, knowledge graphs for semantic compatibility profiling, and specialized large language model-based agents. The framework is designed to emulate the collaborative decision making processes typical of medical teams. **Results:** Our evaluation demonstrates that combining these components within a cohesive multi-agent architecture substantially enhances operational efficiency, task completeness, and contextual relevance in healthcare scheduling workflows. **Conclusions:** *MedScrubCrew*provides a practical, implementable blueprint for healthcare automation, addressing significant inefficiencies in real-world appointment scheduling and patient triage scenarios.

Keywords: clinical decision support system; large language model; multi-agent system; agentic workflows; Gale-Shapley Matching; patient-provider profiles

check for updates

Academic Editor: Daniele Giansanti

Received: 25 April 2025
Revised: 1 July 2025
Accepted: 2 July 2025
Published: 8 July 2025

Citation: Ruiz Mejia, J.M.; Rawat, D.B. *MedScrubCrew*: A Medical Multi-Agent Framework for Automating Appointment Scheduling Based on Patient-Provider Profile Resource Matching. *Healthcare* **2025**, *13*, 1649. https://doi.org/10.3390/healthcare13141649

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The ever-increasing complexity of healthcare presents a critical need for effective automation solutions to alleviate some of the profession's most time-consuming tasks, such as triaging in emergency room or family practice. Although considered a rudimentary task, triaging remains a persistent challenge across medical facilities in various domains. This challenge became particularly evident during the COVID-19 crisis, when medical resources, such as equipment, personnel, and time, were pushed to their limits. During this period, healthcare providers were often required to make rapid, high-stakes decisions to save lives and deliver optimal medical care to those in need.

Medical professionals must always make swift and informed decisions about their patients' health. In certain cases, these decisions may have only a minor impact on the overall health of the patient. However, in other situations, they can significantly influence quality

of care and, consequently, alter the expected outcomes of the patient. These decisions often depend on how medicine is practiced, which can vary significantly due to environmental factors and subjective interpretations. Among the many processes involved in patient care, triaging stands out as one of the most common yet contentious aspects.

In the medical context, triaging refers to the classification of patients based on the urgency of their needs, allowing the efficient allocation of time, personnel, and resources [1,2]. Although this may seem straightforward, it requires input from multiple areas and roles within medical crews to provide a truly relevant context for decision-making. In order to stay ahead of possible resource allocation, medical crews work together to forecast the prioritization of resources for a patient typically a day before by reviewing their schedule and discussing potential needs. This process is colloquially known as "scrubbing" the patient's schedule [3]. This process becomes increasingly challenging when resources are scarce and time is limited. Implementing a multi-agent system with specialized knowledge in these respective areas involved in patient treatment improves visibility, enabling the better identification of potential treatment routes and more effective resource allocation [4].

This research makes the following system-level contributions:

- We develop *MedScrubCrew*, a multi-agent system framework operationalizing the collaborative roles of a medical crew through specialized LLM-based agents.
- We integrate knowledge graphs to model and retrieve semantically compatible patient-provider profiles.
- We implement Gale-Shapley stable matching within this framework to allocate patients to providers based on resource availability and compatibility.
- We conduct simulation-based evaluations on task completion, tool correctness, and contextual relevancy, demonstrating the feasibility of orchestrating such frameworks in real-world healthcare settings.

Together, these contributions address critical resource-dependent inefficiencies in healthcare systems, offering strong potential for real-world implementation and future innovation.

The structure of this paper is as follows: Section 1 provides an introduction to our research. Section 2 elaborates on the gaps in recent research on medical multi-agent systems. Section 3 presents background information on state-of-the-art components in multi-agent systems. Section 4 describes the dataset and methodologies used in this framework. Section 5 elaborates on the theoretical and experimental multi-agent design. Section 6 discusses the evaluation metrics. Section 7 summarizes the results from the experiment. Section 8 provides a discussion and limitations. Section 9 provides our conclusion. Appendix A offers the conclusion.

2. Related Works

Recent advancements in LLM-based frameworks and multi-agent systems have aimed to enhance healthcare reasoning, triage decision-making, and diagnostic accuracy. Tang et al. [5] proposed MEDAGENTS, a multi-agent collaborative reasoning framework utilizing role-playing LLM agents to improve the medical question-answering performance in zero-shot settings. While demonstrating improved QA accuracy compared to chain-of-thought prompting, MEDAGENTS is limited to single QA tasks without addressing operational challenges such as patient-provider scheduling, triage workflows, or resource allocation. Additionally, it lacks traditional scheduling or triage systems, does not integrate knowledge graphs for semantic compatibility profiling, and omits complexity analysis for deployment feasibility.

Schmidgall et al. [6] introduced AgentClinic, a multimodal agent benchmark simulating clinical environments to evaluate LLM diagnostic reasoning under interactive settings, biases, and multilingual constraints. However, AgentClinic is purely a benchmarking plat-

form focused on evaluating diagnostic decision-making accuracy. It does not implement scheduling, triage matching, or resource optimization frameworks, and it does not integrate knowledge graphs or matching algorithms for operational systems. Its emphasis remains on agent evaluation rather than system implementation for healthcare workflows.

Han and Choi [7] developed a multi-agent CDSS integrating LLM agents for KTAS-based triage classification and treatment planning in emergency departments. Their system demonstrated improved triage classification accuracy by emulating ED staff roles, such as triage nurse, physicians, and coordinators. Nevertheless, it is confined to triage-level classification within KTAS protocols [8] and does not address appointment scheduling or patient-provider matching. Moreover, their framework does not employ stable matching algorithms for optimal resource allocation and lacks integration with knowledge graphs for semantic retrieval, limiting explainability and operational optimization. It also does not consider dynamic rescheduling or real-time resource optimization.

Other researchers have integrated the Korean Triage and Acuity Scale (KTAS) [8] for the assessment of triage and the RxNorm API in medication management. This research highlights the potential of LLM-driven multi-agent CDSS in improving emergency care management, reducing ED overcrowding, and improving patient outcomes.

Overall, while prior works focus on diagnostic reasoning [5] and benchmarking agentic performance [6] or triage classification [7], none integrate agentic LLM reasoning, knowledge graphs for semantic compatibility profiling, and stable matching algorithms into a single framework for operational appointment scheduling and triage workflows. Additionally, existing works lack traditional scheduling or triage heuristics, complexity analysis for scalability, and system-level evaluation.

In this work, we present *MedScrubCrew*, a practical multi-agent medical scheduling framework that integrates stable matching, knowledge graphs, and role-based LLM agents to automate appointment scheduling and triage classification. Our system addresses these gaps by demonstrating an orchestrated solution for real-world healthcare operations.

3. Brief Overview of Multi-Agent System for Healthcare

3.1. Agentic Artificial Intelligence (Agentic AI)

Agentic AI refers to AI systems designed to act autonomously by independently analyzing information, making informed decisions, and performing tasks in complex environments, thus enhancing operational efficiency in various domains [9]. In recent years, the pursuit of Artificial General Intelligence (AGI) has sparked discussions about leveraging specialized agents in specific fields to improve performance and outcomes.

As noted in [10], their study emphasizes collaboration with subject matter experts, identifying real-time scenarios, and acquiring relevant knowledge to optimize agent functionality. In the healthcare domain, agents can assume roles in specialized medical areas, potentially reducing context learning errors that might arise with general-purpose LLMs [4,10]. This specialization enables agents to focus on domain-specific tasks, improving accuracy and reliability in critical decision-making processes.

3.2. Agentic Prompt Engineering

Prompt engineering involves carefully designing input prompts to guide AI models in generating desired outputs, thus improving the precision and relevance of their responses [11]. This approach is similarly applied to agents that are instances of the same or different LLMs. However, agents' prompts are tailored differently because they include additional information or context that helps agents understand their purpose and develop a chain of thought to determine the best way to achieve their goals. These tailored prompts typically consist of a role, goal, and background to facilitate an agent's in-context learning.

3.3. In-Context Learning

In-context learning allows AI models to adapt to new tasks by utilizing information provided within the input context, enabling flexible and dynamic responses without the need for extensive retraining. According to recent studies [12,13], in-context learning does not require weight updates, training pipelines, and, in some cases, prompting in a few shots. It simply involves gathering inference from previous inputs and the current input context.

In [14], the authors claimed that in-context learning is prompt-dependent. As long as the prompt is aligned with the data it is trying to learn, the model can establish contextual patterns. In [13], the authors present how in-context learning can allow scene understanding, which is something LLM tends to struggle with due to trained knowledge. However, even with a large context window, some of the context may become skewed or lost during the generation process. Due to this sensitivity to the potential loss of context during learning or generation, agentic systems have gained prominence for their ability to preprocess data and retrieve relevant high-quality inputs for in-context learning.

4. Dataset and Proposed Method

In this section, we focus on identifying a high-quality dataset and developing a framework that simulates an electronic health record (EHR) system that is suitable for modeling semantic relationships.

In this section, we describe the system implementation of *MedScrubCrew*, detailing how we integrate existing components such as knowledge graphs, LLM-based specialized agents, and Gale-Shapley stable matching into a unified framework to automate medical appointment scheduling and triage classification. This approach aims to closely replicate the processes of a medical crew. In a recent study [15], the authors created a graphical clinical database that simulates an EHR together with metadata consisting of a holistic profile of the patient, the provider, and the medical facility.

4.1. Medical Information Mart for Intensive Care IV (MIMIC-IV)

MIMIC-IV database [16] is widely utilized and publicly accessible database that offers comprehensive clinical data from intensive care units, supporting a range of medical research and machine learning projects.

4.1.1. Patient Dataset

The patient dataset contains detailed demographic and clinical information, facilitating the comprehensive analyses of patient characteristics and health outcomes across different populations [17].

4.1.2. Triage Dataset

The triage dataset contains data related to the initial assessment and prioritization of patients, which are essential for evaluating and enhancing triage procedures in healthcare settings [16].

4.1.3. Admission Dataset

The admission dataset records information about patient admissions, including admission times, reasons for hospitalization, and associated medical interventions, aiding studies on hospital utilization and patient flow [17].

4.1.4. Notes Dataset

The notes dataset comprises clinical documentation and physician notes, providing qualitative insights into patient care, treatment plans, and medical observations [18].

4.1.5. Vital Signs Dataset

The vital signs dataset includes the time-series measurements of patients' vital statistics, such as heart rate, blood pressure, and temperature, crucial for monitoring patient health and predicting clinical outcomes [16].

4.2. Multi-Agent Components

The multi-agent components serve as the foundation for agents to collaborate and delegate tasks effectively. The system integrates several specialized AI agents, each assigned distinct roles to collaboratively manage various aspects of the healthcare workflow, mimicking the functionality of a medical crew [19]. This is achieved by providing specific and actionable tasks utilizing advance tools.

4.2.1. Tasks

The tasks managed by the multi-agent system are designed to replicate real-world medical processes, including scheduling appointments, classifying triage levels, and managing patient data effectively. Table 1 presents the details of the tasks and how they are structured for utilization in agentic systems.

Table 1. Agentic tasks.

Task Name	Prompt Description	Expected Output	Agent
`triage_assessment_task`	Determine triage method: use ESI (Levels 1–5) for large hospitals; START (Green, Yellow, Red, and Black) for smaller clinics.	Selected triage level.	Triage Nurse Agent
`pcm_prognosis_task`	Given triage level and chief complaint, produce a prognosis with outcomes and treatment recommendations.	Prognosis plan (explanation, treatment options, care recommendations).	Primary Care Manager
`pcm_resource_task`	List required medical resources (labs, exams, equipment) for patient care.	Detailed resource list.	Primary Care Manager
`ehr_data_retrieval_task`	Build a Neo4j Cypher query to fetch a patient's EHR data (visit frequency, history, interventions) without extraneous nodes.	Summary of EHR data.	EHR Specialist
`ehr_data_similarity_task`	From retrieved EHR notes, find the one most similar to the patient's chief complaint.	Comparative summary of best-matching note vs. complaint.	EHR Specialist
`logistical_query_task`	Generate a Cypher query to pull facility and provider info for a given healthcare setting.	Facility/provider details.	Medical Logistic Specialist
`patient_profiling_task`	Create a patient profile (needs, preferences, resources with quantities).	Structured patient-profile dictionary.	Medical Logistic Specialist
`provider_profiling_task`	Build a provider profile (specialties, availability, resources).	Structured provider-profile dictionary.	Medical Logistic Specialist
`matching_task`	Retrieve schedules from PostgreSQL and run Gale-Shapley to match patients and providers.	JSON of optimal patient-provider matching.	Medical Logistic Specialist

4.2.2. Tools

Tools are specialized functions that allow an agent to complete certain tasks, such as searching the web, generating an image, or interacting with certain files. Table 2 describes the arsenal of tools available for an agentic system. The benefit of using tools is that they are attached to agents but are not necessarily used if tasks can be completed and the goal can be achieved without them.

Table 2. Agentic medical process tools.

Tool Name	Description	Input Details
`Neo4j Query Tool`	Connects to a Neo4j database and executes one or multiple Cypher queries, enabling efficient extraction of medical information.	Query: A Cypher query string or list of strings; URI: Connection URI (defaults to `NEO4J-URI`); Username: Username (defaults to `NEO4J-USERNAME`); Password: Password (defaults to `NEO4J-PASSWORD`).
`Auto Regression Tool`	Performs regression using scikit-learn or XGBoost, either predicting from feature values or training on a dataset of medical records.	sbp: optional systolic blood pressure; pain: optional pain level; dbp: optional diastolic blood pressure; heart rate: optional; O_2sat: optional oxygen saturation; resp. rate: optional; Temperature: optional body temperature; acuity: optional acuity level; Data: optional CSV path or DataFrame for training.
`Gale-Shapley Matching Tool`	Matches patients to providers via an applied Gale–Shapley algorithm, optimizing based on mutual preferences and compatibility scores.	Patients: {ID → (preferences list; characteristics dict)}; Providers: {ID → (preferences list; characteristics dict)}.
`Appointment Schedule Tool`	Retrieves a provider's current schedule, along with associated patient profiles.	Provider: Identifier for the provider; Date: Schedule date.

4.2.3. Hierarchical Process

The hierarchical process outlines the structured workflow of the multi-agent system, detailing how tasks are prioritized, delegated, and coordinated among different agent levels to ensure efficient operation [20]. This process allows for an agent to assume the role of management, providing delegation and collaboration between other agents. It is useful for review and revision-dependent tasks.

4.3. Knowledge Graph

The knowledge graph integrates various sources of patient data, establishing semantic relationships between entities to support informed decision-making and effective patient-provider matching [21]. Semantics enable the system to interpret and use medical data accurately, while entities such as patients, healthcare providers, and medical conditions form the core of the graph structure. Relationships illustrate connections between entities, such as patient-provider links or treatment associations, creating a cohesive healthcare ecosystem [22]. The system retrieves and manipulates data efficiently, enabling complex queries and comprehensive analysis.

In [15], the authors employed a hybrid natural language processing approach using a fine-tuned DeBERTa model (decoding-enhanced BERT with disentangled attention) trained to perform named entity recognition tasks. Subsequently, they utilized a generative LLM to establish relationships between the identified entities. This approach was shown to be effective in facilitating research on the creation of patient-provider profiles.

The database agent uses a pre-populate knowledge graph containing 999 nodes and 2387 relationships from a 100-patient sample. It contains nodes ranging from demographics, medical facilities, biomarkers, pharmaceutical, and past medical history, among others [15].

4.4. Gale-Shapley Stable Matching

In this research, we applied stable matching to optimize patient-provider profile matching based on available resources and the services of both sides. We decided to select the Gale-Shapley algorithm because it allows for stable matching, meaning that matching is carried out not according to best preferences but according to the optimal choice [23]. In a scenario where a patient does not have all their preferences matched, they can still

have the optimal choice. As shown in Algorithm 1, this allows priority to be assigned to the provider's schedule if the patient meets the criteria set by the provider.

Algorithm 1 Gale-Shapley Matching Algorithm

INPUT: U: set of unmatched patients; P: set of patients; R: set of providers; $S(p, r)$: compatibility score function
OUTPUT: M, a matching of patients to providers
Initialize $U \leftarrow P$
Initialize $M(p) \leftarrow \emptyset$, $\forall p \in P$
Initialize $\Pi(p) \leftarrow \emptyset$, $\forall p \in P$
while $U \neq \emptyset$ **do**
  for $p \in U$ **do**
    $r \leftarrow$ highest-ranked $r' \in R \setminus \Pi(p)$
    $\Pi(p) \leftarrow \Pi(p) \cup \{r\}$
    if $M^{-1}(r) = \emptyset$ **then**
      $M(p) \leftarrow r$
      $U \leftarrow U \setminus \{p\}$
    else
      Let $\tilde{p}$ be the current match of r
      Compute $\text{rank}(p, r)$ and $\text{rank}(\tilde{p}, r)$ in r's preference list
      Compute scores $S_p = S(p, r)$ and $S_{\tilde{p}} = S(\tilde{p}, r)$
      if $\text{rank}(p, r) < \text{rank}(\tilde{p}, r) \ \vee \ \left(\text{rank}(p, r) = \text{rank}(\tilde{p}, r) \wedge S_p > S_q\right)$ **then**
        $M(q) \leftarrow \emptyset$
        $U \leftarrow U \cup \{\tilde{p}\}$
        $M(p) \leftarrow r$
        $U \leftarrow U \setminus \{p\}$
      end if
    end if
  end for
end while
return M

In [24], the authors addressed the problem of patient-trial matching using patient records and clinical trial eligibility criteria through multiclass classification. Similarly, this research leverages a large language model (LLM) to process and integrate the outputs of various agents, using semantic similarity to calculate a compatibility score for each criterion.

Other studies [25,26] focused on the scenario of patients providing proposals to providers or medical facilities; however, in emergent situations, medical crews have the need to allocate appropriate resources depending on multiple factors that have little relevance to the health of patients but rather the ability of the medical crew to perform.

The Gale-Shapley algorithm facilitates the optimal matching of nodes in a bipartite graph [27,28]. However, because provider and patient characteristics can change due to new patients, updated patient criteria, or hourly adjustments in provider resources, it is essential that the nodes are able to dynamically adapt to these changes. This ensures that the matching process continues to achieve optimal results based on the updated preferences. We provide an example output in Appendix A, Listing A1.

4.5. Automated Machine Learning

Automated machine learning (AutoML) techniques are applied to streamline the selection and optimization of models, improving the predictive capabilities of a system with minimal manual intervention. In this study, we selected regression models to determine the triage level for vital signs and measured these models using the mean absolute error, mean squared error, and the coefficient of determination, and we performed cross-validation with a fold of 5 to correctly predict the level of acuity, as seen in Table 3. In Table 4, we provide

details of the hyperparameter used for the top three contending models. We utilized a list of regression models, such as Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regressor, Gradient Boosting Regressor, AdaBoost Regressor, Bagging Regressor, Extra Trees Regressor, and HistGradientBoosting Regressor. See Appendix A, Table A1, for additional model metrics.

Table 3. Top three models by performance metrics.

Model	R^2 (Mean)	MSE (Mean ± SE)	MAE (Mean ± SE)
HistGradientBoosting Regressor	0.373	-0.379 ± 0.001	-0.488 ± 0.001
GradientBoosting Regressor	0.367	-0.383 ± 0.001	-0.495 ± 0.000
RandomForest Regressor	0.331	-0.405 ± 0.001	-0.500 ± 0.001

Table 4. Hyperparameter settings for selected models.

Model	Hyperparameters
RandomForest Regressor	$n_{\text{estimators}} = 100$, random_state $= 42$
GradientBoosting Regressor	$n_{\text{estimators}} = 100$, learning_rate $= 0.1$
HistGradientBoosting Regressor	max_iter $= 100$

4.5.1. HistGradientBoosting Regressor

After evaluating the scoring of the models, we observed that the HistGradientBoosting Regressor model achieved satisfactory scores for the assignment, as seen in Table 3. The model can be mathematically represented as follows:

$$\hat{y}_i = \sum_{m=1}^{M} \eta \cdot h_m(x_i) \tag{1}$$

where

- $\hat{y}_i$ is the predicted value for the i-th observation;
- M is the total number of iterations (or weak learners);
- η is the learning rate, controlling the contribution of each weak learner;
- $h_m(x_i)$ is the prediction of the m-th weak learner for input x_i.

The HistGradientBoosting Regressor builds an ensemble of weak learners, typically decision trees, with each learner trained to minimize the residual errors of the preceding ensemble [29]. Its histogram-based optimization efficiently handles large datasets by discretizing continuous features into bins, enabling faster computation and reduced memory usage.

4.5.2. Mean Absolute Error

In the context of the regression model, the mean absolute error (MAE) quantifies the average magnitude of errors between the predicted acuity levels and the actual acuity levels without considering their direction. It is calculated as follows:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2}$$

where

- n is the total number of observations;
- y_i is the actual acuity level for the i-th observation;
- $\hat{y}_i$ is the predicted acuity level for the i-th observation.

A lower MAE indicates that the regression model's predictions are closer to the actual acuity levels on average, making it particularly useful for understanding the overall accuracy of the model in clinical settings [30].

4.5.3. Mean Squared Error

The mean squared error (MSE) evaluates the average squared difference between the predicted acuity levels and the actual acuity levels. It penalizes larger errors more than smaller ones, making it more sensitive to outliers. The formula for MSE is as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

where

- n is the total number of observations;
- y_i is the actual acuity level for the i-th observation;
- $\hat{y}_i$ is the predicted acuity level for the i-th observation.

In the AutoML Regression Model, a lower MSE reflects smaller average errors in predicting acuity levels, but the metric's sensitivity to outliers should be considered when interpreting the results [31].

4.5.4. Coefficient of Determination (R^2)

The coefficient of determination (R^2) measures how well the predicted acuity levels explain the variance in the actual acuity levels. It is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

where

- y_i is the actual acuity level for the i-th observation;
- $\hat{y}_i$ is the predicted acuity level for the i-th observation;
- $\bar{y}$ is the mean acuity level across all observations;
- $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the residual sum of squares (RSS);
- $\sum_{i=1}^{n}(y_i - \bar{y})^2$ is the total sum of squares (TSS).

For the AutoML Regression Model, an R^2 value close to 1 indicates that the tool's predictions explain most of the variance in acuity levels, while a value close to 0 suggests limited predictive power [32,33]. This metric is particularly valuable for evaluating the overall effectiveness of the model in clinical decision-making.

In summary, Figures 1–3 illustrate the comparative performance of various regression models evaluated in predicting the patient acuity levels. Figure 1 presents the HistGradientBoosting Regressor, which achieved the lowest MAE of approximately 0.488, suggesting it generates the most precise predictions compared to other evaluated models. Figure 2 shows again the HistGradientBoosting Regressor delivered the best performance with an MSE of around 0.379, demonstrating robustness in minimizing substantial prediction errors. It is important to note that while these metrics were recorded as negative values due to scikit-learn scoring convention, they are reported here in their absolute form. Lastly, Figure 3 illustrates R^2 in which the model attained the highest score of approximately 0.0373, indicating its superior ability to capture the underlying patterns within the data. Collectively, these metrics underscore the HistGradientBoosting Regressor as the appropriate model, providing robust and accurate predictions for patient triage acuity within the presented automated machine learning framework.

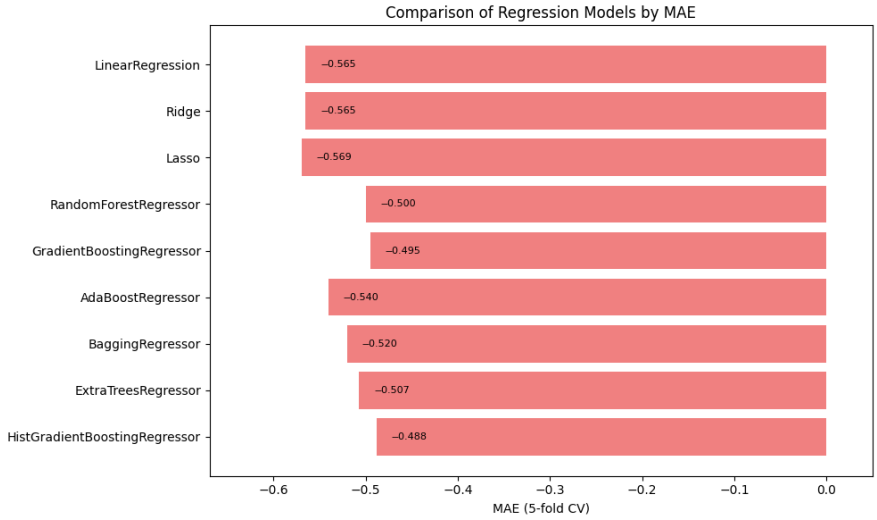


Figure 1. Automated regression MAE score: a bar plot showing the mean absolute error (MAE) scores across different regression models.

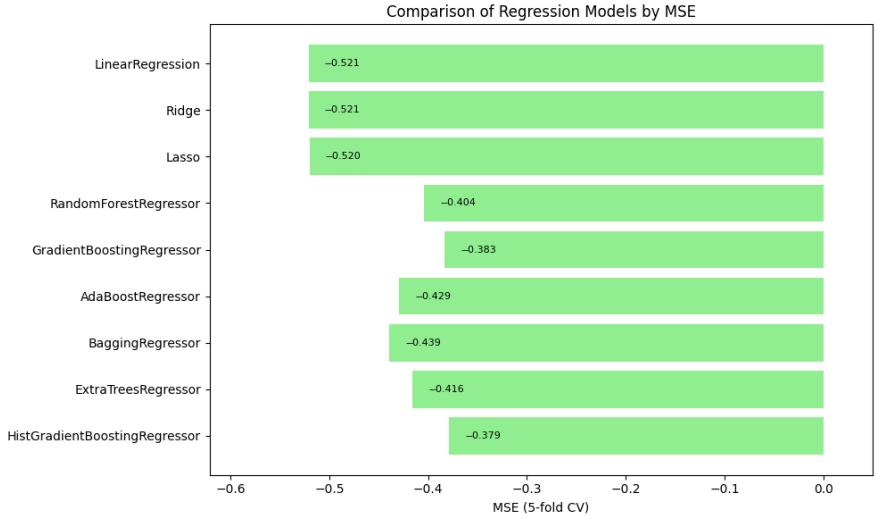


Figure 2. Automated regression MSE score: a bar plot showing the mean squared error (MSE) scores across different regression models.

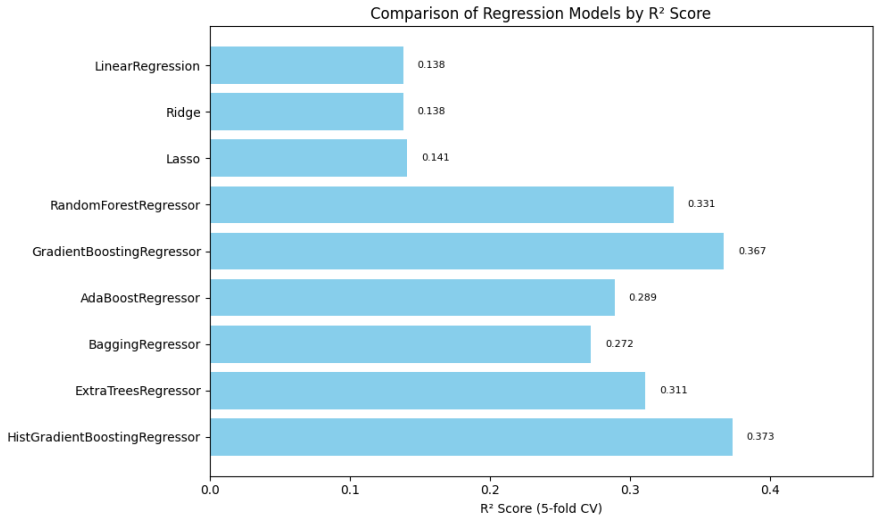


Figure 3. Automated regression R^2 score: a bar plot showing the R^2 scores across different regression models, representing the proportion of variance explained by each model.

5. Medical Multi-Agent Experimental System Design

This section proposes a theoretical and experimental design in a multi-agent medical system. In Figure 4, an automated approach is shown to gather relevant patient data based on prognoses conditioned from the initial complaints. The agents collaborate in order to create the patient-provider profile utilizing LLM to interpret in-context learning from each agent output [10]. Table 1 provides descriptions of the tasks for the specialized agents.

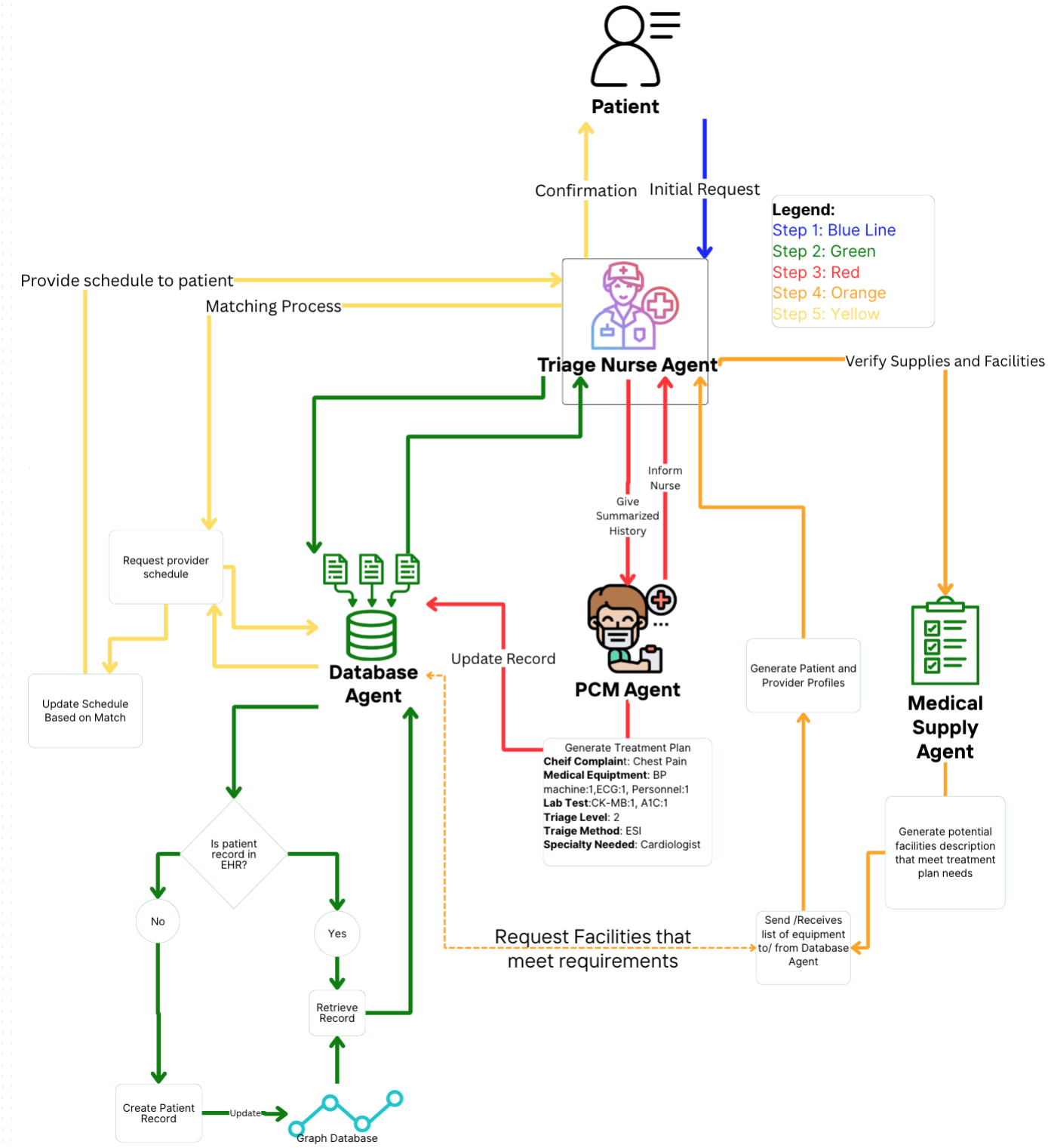


Figure 4. Medical multi-agent design: A schematic representation of the multi-agent system used for medical processes. Patient information is hypothetical.

5.1. Patient Scheduling and Triage Process

We modeled the patient scheduling and triage process as a discrete-time-state machine that follows a set of protocols typically used to re-assess and increase visibility on a patient's status [34]. According to [35], to automate an EHR, it must follow a set of processes that resembles the manual process a medical professional undergoes in order to achieve a proper schedule and triage status. The state machine governing this process explicitly accounts for new patient requests and involves a sequence of agent-performed tasks. Let $S(t)$ denote the system's state at the time t, which captures the current patient schedule. Let $R(t)$ be a new patient request arriving at time t, or $R(t) = \text{NoReq}$ if no request arrives. By convention, each subroutine $\mathcal{F}_*(S, \text{NoReq})$ simply propagates S unchanged. Thus, we define the following:

$$S(t+1) = \begin{cases} \mathcal{F}\big(S(t),\, R(t)\big), & \text{if } R(t) \neq \text{NoReq}, \\ \mathcal{F}\big(S(t),\, \text{NoReq}\big), & \text{otherwise} \end{cases} \tag{5}$$

$$\mathcal{F}(S,R) = \big(\mathcal{F}_{\text{appointment}} \circ \mathcal{F}_{\text{logistic}} \circ \mathcal{F}_{\text{prognosis}} \circ \mathcal{F}_{\text{profile}} \circ \mathcal{F}_{\text{vital}} \circ \mathcal{F}_{\text{history}}\big)(S,R). \tag{6}$$

In Equation (6), this composition operates from the right to the left, beginning with the evaluation of the patient's history and culminating in appointment scheduling.

The following are defined:

$S(t)$**:** Current state of the system at time t.

$S(t+1)$**:** Next state of the system at time $t+1$.

$R(t)$**:** New patient request submitted at time t (if any).

$\mathcal{F}(S,R)$**:** Composite function representing the sequence of agent tasks:

- $F_{\text{history}}(S,R)$: Triage nurse agent assesses patient history based on S and R.
- $F_{\text{vital}}(\cdot)$: Triage nurse agent evaluates new vital signs.
- $F_{\text{profile}}(\cdot)$: Primary care provider agent creates the patient's illness profile.
- $F_{\text{prognosis}}(\cdot)$: Primary care provider agent develops the prognosis plan with required resources.
- $F_{\text{logistic}}(\cdot)$: Medical logistic agent reviews the resource plan and identifies the top three providers with available resources.
- $F_{\text{appointment}}(\cdot)$: Appointment scheduling agent matches patient-provider profiles and schedules the appointment.

$\varnothing$ represents the absence of a new patient request.

5.2. Database Agent

A database agent specializes in retrieving and updating the graphical knowledge EHR database. This type of agent functions as a retrieval-augmented generation component, providing context for the framework and supporting the goals of other agents [36,37]. It leverages an instance of a LLM to generate valuable patient insights and enhance search queries by translating natural language into database-specific languages, such as Cypher [38,39].

5.3. Primary Care Provider Agent

The primary care provider (PCP) agent assumes the role of a PCP, which can be a physician or nurse practitioner with general medical knowledge in clinical medicine. PCPs are responsible for maintaining a patient's overall health [40,41]. The goal of the PCP agent is to accurately assign a prognosis and propose a potential treatment plan, which can then be used to create the patient profile.

5.4. Medical Logistic Agent

The goal of the medical logistic agent is to perform semantic comparisons between the preferences and characteristics of the patient profile and the provider profile. It collaborates with the database agent to query updated medical logistics, including resources, time, personnel, and provider services [42–44]. We acknowledge that semantic differences may arise based on facility-specific terminology and resource naming conventions. For instance, cardiologists may use different terms for equipment or services that are functionally similar.

To address this, the medical logistic agent is specialized in medical logistics, medical services, and equipment terminology, enabling it to accurately compare terms semantically. Additionally, it structures the patient-provider profile as the expected output, ensuring compatibility and alignment between patient needs and provider capabilities.

5.5. Triage Nurse

The triage nurse acts as the manager agent in this hierarchical framework. A triage nurse can be described as the nurse in charge of the patient process. In contrast to the PCP agent, which is responsible for the prognosis of illness and the services needed, the triage nurse is responsible for the overall process, which involves patient EHR review, resource allocation, and appointment scheduling [45]. This is portrayed in Figure 4, where the patient-provider profiles is given to triage nurse agent to assign a scheduling slot. In addition to reviewing, it delegates tasks such as retrieving historical information or creating a new patient inquiry.

5.6. Profile-Based Compatibility

To support patient-provider matching, we define a numerical compatibility score that quantifies how well a patient's characteristics align with those of a provider. Let π_p and π_r denote the structural profiles of the patient p and provider, respectively. These profiles may include attributes such as the medical condition, urgency, medical equipment needed, or specialty. The resulting score reflects how well a provider matches a patient's needs.

We define the compatibility score using an abstract function f:

$$S(p, r) = f(\pi_p, \pi_r), \tag{7}$$

where f is a domain-specific function that returns a real-valued compatibility score based on the alignment between retrieved patient and provider profiles. While the implementation details of f are abstracted in this work, it may represent a learned scoring model or a rule-based similarity metric tailored to the medical triage and scheduling context. The profiles may be composed of structured features:

$$\pi_p = (\text{condition}_p, \text{urgency}_p, \text{location}_p, \dots), \ \pi_r = (\text{specialty}_r, \text{capacity}_r, \text{location}_r, \dots).$$

Higher values of $S(p, r)$ represent better compatibility between the two. The similarity function f may be implemented using cosine similarity, a dot product, or a trained neural metric.

To convert these scores into ordinal preferences, we define patient and provider rankings based on sorted compatibility scores. Specifically, for each patient p, the rank of provider r is defined as follows:

$$\text{rank}_p(r) = 1 + \left|\{ r' \in R : S(p, r') > S(p, r)\}\right|, \tag{8}$$

Similarly, for each provider r, the rank of patient p is as follows:

$$\text{rank}_r(p) = 1 + \left|\{\, p' \in P : S(p', r) > S(p, r)\}\right|. \tag{9}$$

In this formulation, a rank of 1 indicates the most preferred match, and ties are implicitly handled via strict inequalities in the set definitions.

5.7. Patient-Provider Profile Matching Process

The patient-provider matching process is executed using the Gale-Shapley matching algorithm. We first match each patient in P to a provider in R based on mutually ranked preferences. Before the algorithms begin, every patient is unmatched, and no proposals are made. For each $p \in P$, we set the match and proposal sets as follows:

$$M(p) = \varnothing, \quad \Pi(p) = \varnothing, \quad \forall p \in P \tag{10}$$

At each iteration, an unmatched patient makes proposals to its most preferred provider, which has not yet been approached. We let $\text{rank}_p(r')$ denote the position of provider r' in patient p's preferences list, with 1 indicating the highest preferred choice. The next provider to receive a proposal is given by

$$r = \arg \min_{r' \in R \setminus \Pi(p)} (\text{rank}_p(r')) \tag{11}$$

which ensures that p always makes proposals to the highest ranked available provider.

When a patient p proposes to provider r, r compares p against its current match $\tilde{p}$. If r prefers p over $\tilde{p}$, then p becomes matched to $\tilde{p}$; otherwise, p remains unmatched. This update rule is expressed as

$$M(p) = \begin{cases} r, & \text{if provider } r \text{ prefers patient } p \text{ over its current match } \tilde{p} \\ \varnothing, & \text{otherwise.} \end{cases} \tag{12}$$

To ensure stable matching so that no patient-provider pair will prefer each other over their assigned matches, we enforce the following acceptance and rejection criteria. A provider r only accepts a proposal from p if r prefers p to any alternative r' that p might have preferred, and r rejects an existing match $\tilde{p}$ in favor of p only if $p \succ_r \tilde{p}$:

$$r \succ_p r' \implies M(p) = r \quad \text{only if provider } r \text{ accepts patient } p \tag{13}$$

$$p \succ_r \tilde{p} \implies \text{provider } r \text{ rejects patient } \tilde{p} \text{ in favor of patient } p \tag{14}$$

Finally, the matching algorithm terminates after at most $n \cdot m$ proposals because each of the $n = |P|$ patients can make proposals to at most $m = |R|$ providers.

The computation complexity of the Gale-Shapley matching algorithm utilized in *MedScrubCrew* is $\mathcal{O}(nm)$, where n is the number of patients and m is the number of providers. This is because, in the worst-case scenario, each patient may make proposals to every provider before finding a stable match. For typical scheduling scenarios involving hundreds of patients and providers, the algorithm remains computationally feasible, with runtime increasing linearly with the number of proposals. Additionally, since *MedScrubCrew* integrates dynamic patient and provider updates, periodic recomputation may be required. However, the deterministic nature of the algorithms and the complexity of the polynomial time ensure that scheduling can be executed efficiently within the practical operational time frame for daily appointment allocation tasks in clinical settings.

$$\text{Total Proposals} \;\leq\; n \cdot m. \tag{15}$$

Table 5 provides a description of the notation used in this matching process.

Table 5. Notations used in the Gale-Shapley matching algorithm.

Notation	Description
P	Set of patients with preferences and characteristics
R	Set of providers with preferences and characteristics
$S(p, r)$	Compatibility score function for patient p and provider r
M	Matching of patients to providers
U	Set of unmatched patients
$M(p)$	Current match for patient p (initially empty)
$\Pi(p)$	Set of providers patient p has proposed to
r	Provider under consideration during a patient's proposal
q	Current patient matched to a provider
$\text{rank}(p, r)$	Position of patient p in provider r's preferences
$\text{rank}_p(r)$	Position of provider r in patient p's preference list
S_p	Compatibility score of patient p with provider r
$S_{\tilde{p}}$	Compatibility score of patient $\tilde{p}$ with provider r

6. Evaluation and Metrics

During this research, we employed an LLM for the Judge method. In essence, the LLM-as-a-Judge framework utilizes an LLM instance to evaluate responses based on predefined metrics [46]. We evaluated the agents' responses based on the following criteria: the context provided (if any), the correct selection of tools, and task completeness.

6.1. Contextual Relevancy

Some tasks were provided with context in the form of previous agent responses. This allows the multi-agent system to maintain contextual awareness throughout the process. This metric calculates contextual relevancy by taking the number of relevant statements and dividing it by the total number of generated statements produced by the LLM. The resulting score ranges from 0, meaning no relevant context, to 1, meaning the response fully meets the relevant contextual criteria:

$$CR = \frac{N_{\text{relevant}}}{N_{\text{total}}}, \tag{16}$$

where CR represents the contextual relevancy score, N_{relevant} is the number of relevant statements, and N_{total} is the total number of statements generated, as seen in Appendix A, Listing A2.

The contextual relevance evaluation is defined as follows:

$$M(C) = \begin{cases} 1 & \text{if } CR \geq \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

where $M(C)$ represents whether the response maintains contextual relevancy, and τ is the predefined threshold.

The preference condition for maintaining context is given by the following:

$$C_{\text{relevant}} \succ C_{\text{irrelevant}} \implies M(C) = 1 \text{ only if context is maintained,} \tag{18}$$

$$C_{\text{incorrect}} \prec C_{\text{correct}} \implies C_{\text{incorrect}} \text{ is refined to improve relevancy,} \tag{19}$$

where $C_{\text{relevant}} \succ C_{\text{irrelevant}}$ means that the relevant context is preferred, and $C_{\text{incorrect}} \prec C_{\text{correct}}$ means that the incorrect context is adjusted.

The termination condition for contextual evaluation is as follows:

$$\text{Total Context Evaluations} \leq k \cdot m, \tag{20}$$

where k is the number of statements per evaluation, and m is the number of iterations performed.

6.2. Task Completeness

In a multi-agent system, each agent is assigned a variety of tasks. In some cases, a task may not be fully completed based on the expected input and output. The system evaluates whether the task description aligns with the generated output. This metric is based on binary scoring and assesses how closely the generated output satisfies the given task requirements.

$$TCS = \text{AlignmentScore}(\mathcal{T}, \mathcal{O}), \tag{21}$$

where TCS represents the task completion score, $\mathcal{T}$ is the assigned task, and $\mathcal{O}$ is the generated output.

The task completion evaluation is defined as follows:

$$M(\mathcal{T}) = \begin{cases} 1, & \text{if AlignmentScore}(\mathcal{T}, \mathcal{O}) \geq \tau, \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where $M(\mathcal{T})$ represents whether a task $\mathcal{T}$ is successfully completed, and τ is the threshold for considering a task as complete.

The preference criterion for task alignment is given by the following:

$$T_{\text{completed}} \succ T_{\text{incomplete}} \Rightarrow M(\mathcal{T}) = 1 \quad \text{only if } \mathcal{O} = \mathcal{T}, \tag{23}$$

$$T_{\text{incorrect}} \prec T_{\text{correct}} \Rightarrow T_{\text{incorrect}} \text{ is refined until completion,} \tag{24}$$

where $T_{\text{completed}} \succ T_{\text{incomplete}}$ denotes that a completed task is preferred over an incomplete one, and $T_{\text{incorrect}} \prec T_{\text{correct}}$ signifies that an incorrect task undergoes refinement until completion.

The termination of the task execution process is guaranteed as follows:

$$\text{Total Task Attempts} \leq k \cdot n, \tag{25}$$

where k is the number of tasks, and n represents the number of iterations required for task completion. We present an example prompt for how it calculates the scores in Appendix A, Listing A3.

6.3. Tool Correctness

Agents can be provided a multitude of tools in order to complete their tasks. These tools can range from a calculator to web searches. However, in some cases, the agent might have either similar tools that provide different responses or a wide variety of tools. In this metric, the tool selection will be evaluated to see if the selection is correct for the actual tool indicated:

$$TC = \frac{N_{\text{correct}}}{N_{\text{total}}}, \tag{26}$$

where TC represents tool correctness, N_{correct} represents the number of correctly used tools (or correct input parameters/outputs), and N_{total} represents the total number of tools used.

The tool selection evaluation is defined as follows:

$$M(T) = \begin{cases} 1 & \text{if } T \text{ is used correctly,} \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

where $M(T)$ represents whether a tool T is used correctly.

The correctness criterion for tool usage is given by the following:

$$T_{\text{selected}} \succ T_{\text{alternative}} \implies M(T) = 1 \text{ only if } T \text{ is used correctly,} \tag{28}$$

$$T_{\text{incorrect}} \prec T_{\text{correct}} \implies T_{\text{incorrect}} \text{ is discarded in favor of } T_{\text{correct}}, \tag{29}$$

where $T_{\text{selected}} \succ T_{\text{alternative}}$ means that the selected tool is preferred over an alternative, and $T_{\text{incorrect}} \prec T_{\text{correct}}$ means that an incorrect tool is replaced with the correct one.

The termination of the tool selection process is ensured as follows:

$$\text{Total Tool Evaluations} \leq k \cdot m, \tag{30}$$

where k is the number of tasks, and $m$is the number of tools. The accuracy of the T_{correct} and $T_{\text{incorrect}}$ values in the tool correctness metric is ensured through a deterministic and objective evaluation of expected tools using exact matching by the tool's name and, optionally, by input parameters and T_{correct} if it fully aligns with these expectations; any tool that does not meet these criteria is counted as $T_{\text{incorrect}}$. This rule-based approach leaves no room for ambiguity or the subjective reflection of the number of correct usages, ensuring that the TC metric is a precise and reliable measure of agent tool selection and performance.

7. Results

Multi-agent systems enable a wide range of applications. However, understanding their behaviors and decision-making processes is essential for improving explainability, particularly in justifying their generated responses. For the GPT-4o-mini [47] model, the evaluation was conducted using its own instance, while all other models were evaluated using GPT-4o-mini due to its lower latency and faster response times for smaller evaluation tasks.

We utilized six state-of-the-art models as the engines that power these agents. The results were evaluated based on contextual relevancy, tool usage, and task completion. As shown in Figure 5, our observations indicate that QWEN-2.5-coder-32b effectively selected and utilized the provided tools to generate appropriate responses while maintaining context. For instance, in the triage assessment task, the following was the case.

In contrast, the other Qwen models exhibited inconsistent performance. While Qwen-2.5-32B was able to retrieve relevant EHR notes, its task completion scores fluctuated between 0.7 and 0.9, reflecting occasional misinterpretations of task objectives despite retrieving medically relevant data. However, its logistical decision-making performance was lower, as it often failed to complete medical logistics profiling tasks, resulting in lower effectiveness when structuring ICU requirements or streamlining supply allocation. Similarly, the DeepSeek-R1-Distilled model struggled in retrieving EHR data, scoring 0.1 in some retrieval tasks, largely due to syntax errors in graph database queries.

Interestingly, in the medical logistical matching task, all models experienced errors during tool execution, often due to improper input formatting or missing expected attributes even with a structured Pydantic input/output model. This underscores a broader challenge in medical logistics optimization, where model precision in structured query generation directly impacts system effectiveness.

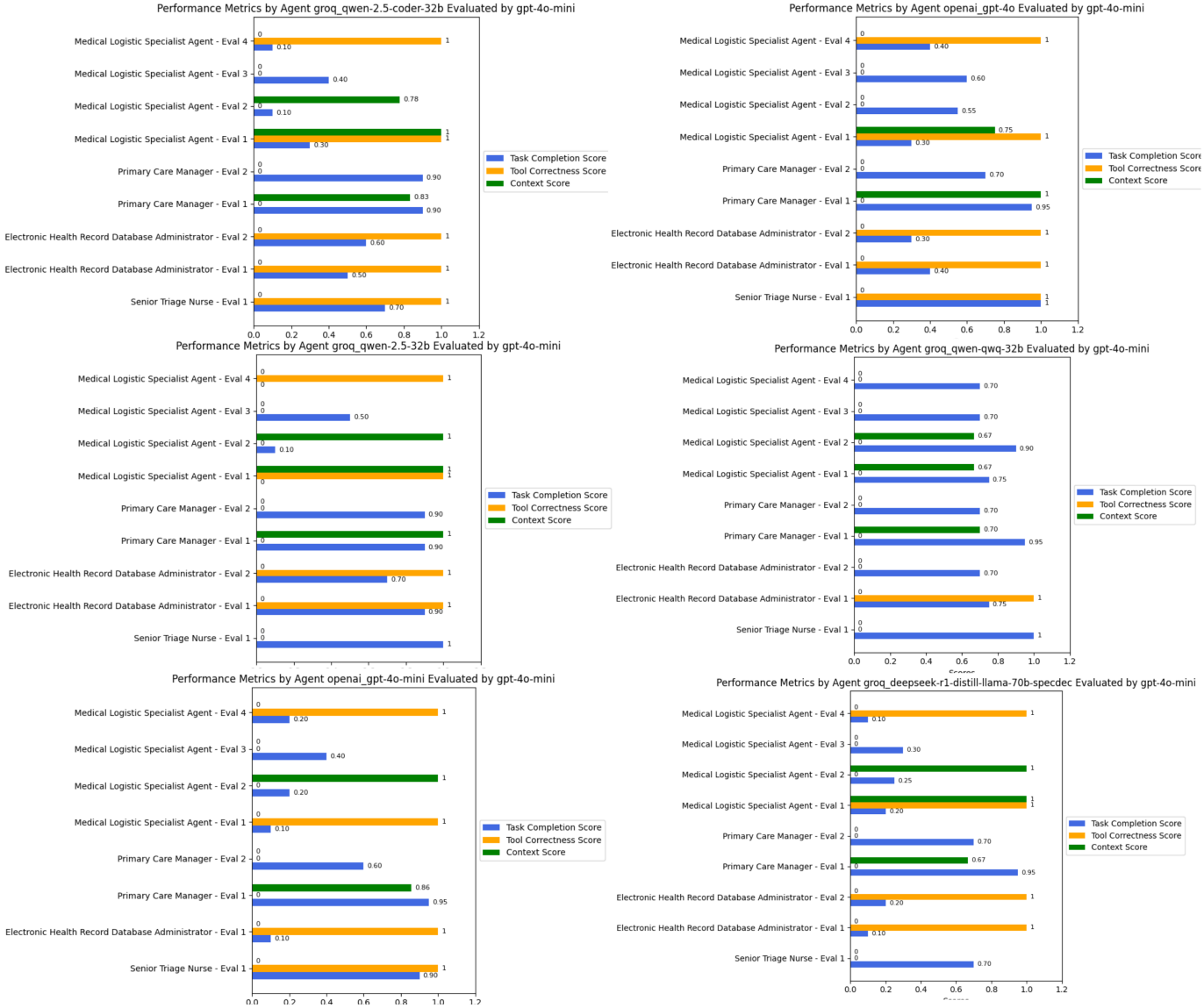


Figure 5. Performance comparison of six state-of-the-art large language models with function-calling capabilities across three evaluation metrics: contextual relevancy, tool usage accuracy, and task completion rate. The QWEN-2.5-coder-32b model demonstrates consistent suitable performance across all metrics with an overall average score of approximate 0.71.

Overall, QWEN-2.5-coder-32b demonstrated superior adaptability, particularly in decision-making tasks where it had access to well-structured tools. However, areas requiring deeper contextual synthesis, such as long-form EHR retrieval, highlighted incomplete data extraction across all models, suggesting the need for enhanced retrieval-augmented generation (RAG) capabilities for improved medical decision support.

8. Discussion and Limitations

MedScrubCrew demonstrates the feasibility of implementing a practical multi-agent system combining coordination, compatibility-based stable matching, and semantic knowl-

edge graphs. While it does not propose new algorithms, it provides a deployment-focused framework addressing operational inefficiencies in scheduling and triage workflows. In our experiments, Agent collaboration was occasionally disrupted by API rate-limit throttling and delayed tool calls, compromising responsiveness at scale. Lightweight reasoning models sometimes bypass specialized tool calls for trivial triaging tasks, still yielding correct outcomes via full-profile LLM processing but introducing potential bias by neglecting targeted, rule-based checks. Detailed agent collaboration also rapidly exhausts the LLM context window, increasing latency and risking the loss of earlier conversation states.

We also recognize the importance of prompt engineering in agentic workflows in providing clear and substantive role and task descriptions. To advance our research, we plan to implement an AI feedback loop to enhance the accuracy of generated outputs and review decisions made by the triage nurse during the scheduling of patient appointments.

Furthermore, we acknowledge several factors requiring further exploration: using other evaluation metrics such as Hallucination, Bias, and Prompt Alignment; the availability of medical testing results prior to scheduling; and the completion of referrals before appointment allocation. These features can significantly influence the position of a patient on the schedule and require additional considerations in the experimental design.

We also acknowledge the importance of security and privacy in a production environment. The data used is an open-source de-identified dataset that was used to create the needed information for the multi-agent crew to function. However, in a production environment, security and privacy would have to be taken into account, such as having self-hosted models, contractual agreements with LLM providers to self-host a specific model, or simply having guardrails in place for input verification.

While our evaluation tasks are structured and deterministic, future work will incorporate human clinician validation to complement LLM as a Judge evaluation and mitigate potential scoring biases.

Beyond these technical challenges, our framework omits several real-world considerations. We have not yet incorporated human-in-the-loop oversight. Clinicians cannot intervene during matching, which is critical for safety and trust. Additionally, dynamic resource constraints (e.g., fluctuating staff availability), telehealth scenarios, and interactions with IoT-enabled diagnostic devices remain unexplored.

9. Conclusions

In this research, we presented *MedScrubCrew*, an applied multi-agent system framework integrating specialized LLM agents, knowledge graphs, and stable matching to automate appointment scheduling and triage classification. While building on existing algorithms and AI tools, our work demonstrates practical system integration for healthcare automation. Future work will focus on clinician-in-the-loop evaluations and real-world pilot studies to assess clinical utility.

Author Contributions: These authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the DoD Center of Excellence in AI and Machine Learning (CoE-AIML) at Howard University under Contract W911NF-20-2-0277 with the U.S. Army Research Laboratory.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors express gratitude to the DoD Center of Excellence in AI and Machine Learning (CoE-AIML) at Howard University. Any opinions, findings, and conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the funding agencies.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Listing A1. Gale–Shapley output showing patient-provider matches with compatibility scores.

```
{
  "matches": {
    "patient1": {
      "provider_id": null,
      "compatibility_score": 0
    },
    "patient2": {
      "provider_id": "provider1",
      "compatibility_score": 25
    },
    "patient3": {
      "provider_id": "provider2",
      "compatibility_score": 20
    }
  }
}
```

Listing A2. Prompt for context relevance evaluation.

```
Based on the input and context, please generate a JSON object to
    indicate whether each statement found in the context is relevant
     to the provided input. The JSON will be a list of 'verdicts',
    with 2 mandatory fields: 'verdict' and 'statement', and 1
    optional field: 'reason'.
You should first extract statements found in the context, which are
    high level information found in the context, before deciding on
    a verdict and optionally a reason for each statement.
The 'verdict' key should STRICTLY be either 'yes' or 'no', and
    states whether the statement is relevant to the input.
Provide a 'reason' ONLY IF verdict is no. You MUST quote the
    irrelevant parts of the statement to back up your reason.

IMPORTANT: Please make sure to only return in JSON format.

Example Context: "Einstein won the Nobel Prize for his discovery of
    the photoelectric effect. He won the Nobel Prize in 1968. There
    was a cat."
Example Input: "What were some of Einstein's achievements?"

Example:
{
    "verdicts": [
        {
            "verdict": "yes",
            "statement": "Einstein won the Nobel Prize for his
                discovery of the photoelectric effect in 1968"
        },
        {
            "verdict": "no",
```

Listing A2. *Cont.*

```
            "statement": "There was a cat.",
            "reason": "The retrieval context contained the
                information 'There was a cat' when it has nothing to
                 do with Einstein's achievements."
        }
    ]
}

Input:
{input}

Context:
{context}
```

Listing A3. Prompt for task completeness evaluation.

```
"""Given the user goal (desired outcome) and the actual achieved
    outcome, compare how well the actual outcome aligns with the
    user's intended goal.

                Please return a JSON with two keys: `verdict` and `
                    reason`.
                - The `verdict` should be a score from 0 to 1, where
                     1 indicates the actual outcome perfectly
                    achieves the user's goal, and 0 indicates it
                    does not achieve the goal at all.
                - The `reason` should explain why the given verdict
                    was assigned.

                **
                IMPORTANT: Please make sure to only return in JSON
                    format, with `verdict` as a float between 0 and
                    1.
                Example:
                User goal: Have the system plan a weekend trip to
                    New York, including travel, accommodation, and
                    sightseeing.
                  Actual outcome: The system provided suggested
                      flights departing on Saturday and returning on
                       Sunday, identified hotels with check-in on
                      Saturday and check-out on Sunday, and
                      generated a list of sightseeing destinations
                      in New York City.
                Example JSON:
                {{
                    "verdict": 0.85,
                    "reason": "The system suggested flights,
                        accommodation, and sightseeing options but
                        did not fully plan the trip as expected."
                }}
                **

                User goal:
                {user_goal}

                Actual outcome:
                {actual_outcome}

                JSON:
            """
```

Table A1. Model evaluation metrics (mean and standard error) for all regressors.

Model	R^2	MSE		MAE	
	Mean	Mean	SE	Mean	SE
Linear Regression	0.138	−0.521	0.038	−0.565	0.001
Ridge	0.138	−0.521	0.038	−0.565	0.001
Lasso	0.141	−0.520	0.036	−0.569	0.001
RandomForest Regressor	0.331	−0.404	0.001	−0.500	0.000
GradientBoosting Regressor	0.367	−0.383	0.001	−0.495	0.000
AdaBoost Regressor	0.289	−0.429	0.002	−0.540	0.001
Bagging Regressor	0.272	−0.439	0.001	−0.520	0.000
ExtraTrees Regressor	0.311	−0.416	0.001	−0.507	0.001

References

1. Dippenaar, E. Triage systems around the world: A historical evolution. *Int. Paramed. Pract.* **2019**, *9*, 61–66. [CrossRef]
2. Ruiz Mejia, J.M.; Rawat, D.B. Exploring the Advancements of AI-enabled Clinical Decision Support Systems for Patient Triage in Healthcare. In Proceedings of the IEEE International Conference on E-Health Networking, Application & Services, Nara, Japan, 8–20 November 2024.
3. Ghorob, A. Healthy Huddle Warm Up Tool. © 2014 The Regents of the University of California. 2014. Available online: https://cepc.ucsf.edu/sites/cepc.ucsf.edu/files/Healthy_Huddles_Warm_Up_14-0602.pdf (accessed on 5 February 2025).
4. Singh, A.; Ehtesham, A.; Kumar, S.; Khoei, T.T. Enhancing AI Systems with Agentic Workflows Patterns in Large Language Model. In Proceedings of the 2024 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 29–31 May 2024; pp. 527–532. [CrossRef]
5. Tang, X.; Zou, A.; Zhang, Z.; Li, Z.; Zhao, Y.; Zhang, X.; Cohan, A.; Gerstein, M. MedAgents: Large Language Models as Collaborators for Zero-shot Medical Reasoning. *arXiv* **2024**, arXiv:2311.10537. [CrossRef]
6. Schmidgall, S.; Ziaei, R.; Harris, C.; Reis, E.; Jopling, J.; Moor, M. AgentClinic: A multimodal agent benchmark to evaluate AI in simulated clinical environments. *arXiv* **2024**, arXiv:2405.07960. [CrossRef]
7. Han, S.; Choi, W. Development of a Large Language Model-based Multi-Agent Clinical Decision Support System for Korean Triage and Acuity Scale (KTAS)-Based Triage and Treatment Planning in Emergency Departments. *arXiv* **2024**, arXiv:2408.07531. [CrossRef]
8. Moon, S.H.; Shim, J.L.; Park, K.S.; Park, C.S. Triage accuracy and causes of mistriage using the Korean Triage and Acuity Scale. *PLoS ONE* **2019**, *14*, e0216972. [CrossRef]
9. Hu, S.; Lu, C.; Clune, J. Automated Design of Agentic Systems. *arXiv* **2024**, arXiv:2408.08435. [CrossRef]
10. Aguilera, A.; Subero, A. A Multi-agent Architecture and System for Dynamic Medical Knowledge Acquisition. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, Austria, 10–12 December 2008; pp. 1165–1170. [CrossRef]
11. Hagos, D.H.; Battle, R.; Rawat, D.B. Recent Advances in Generative AI and Large Language Models: Current Status, Challenges, and Perspectives. *arXiv* **2024**, arXiv:2407.14962. [CrossRef]
12. Zhou, H.; Nova, A.; Larochelle, H.; Courville, A.; Neyshabur, B.; Sedghi, H. Teaching Algorithmic Reasoning via In-context Learning. *arXiv* **2022**, arXiv:2211.09066. [CrossRef]
13. Balažević, I.; Steiner, D.; Parthasarathy, N.; Arandjelović, R.; Hénaff, O.J. Towards In-context Scene Understanding. *arXiv* **2023**, arXiv:2306.01667. [CrossRef]
14. Machlab, D.; Battle, R. LLM In-Context Recall is Prompt Dependent. *arXiv* **2024**, arXiv:2404.08865. [CrossRef]
15. Ruiz Mejia, J.M.; Rawat, D.B. ClinicalGraph: An Applied Approach in Clinical EHR Knowledge Graph Generation for Optimized Clinical Decision Support System. In Proceedings of the IEEE International Conference on E-health Networking, Application & Services, Nara, Japan, 18–20 November 2024.
16. Johnson, A.; Bulgarelli, L.; Pollard, T.; Gow, B.; Moody, B.; Horng, S.; Celi, L.A.; Mark, R. MIMIC-IV-ED (version 3.0). PhysioNet. RRID:SCR_007345. 2024. Available online: https://physionet.org/content/mimic-iv-ed/2.2/ (accessed on 5 February 2025).
17. Johnson, A.; Bulgarelli, L.; Pollard, T.; Gow, B.; Moody, B.; Horng, S.; Celi, L.A.; Mark, R. MIMIC-IV (version 3.0). PhysioNet. RRID:SCR_007345. 2024. Available online: https://physionet.org/content/mimiciv/3.0/ (accessed on 5 February 2025).
18. Johnson, A.; Bulgarelli, L.; Pollard, T.; Gow, B.; Moody, B.; Horng, S.; Celi, L.A.; Mark, R. MIMIC-IV-Note (version 3.0). PhysioNet. RRID:SCR_007345. 2024. Available online: https://physionet.org/content/mimic-iv-note/ (accessed on 5 February 2025).
19. White, J. Building Living Software Systems with Generative & Agentic AI. *arXiv* **2024**, arXiv:2408.01768. [CrossRef]
20. Girault, A.; Lee, B.; Lee, E. Hierarchical finite state machines with multiple concurrency models. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **1999**, *18*, 742–760. [CrossRef]

21. Timilsina, M.; Alsamhi, S.; Haque, R.; Judge, C.; Curry, E. Knowledge Graphs, Clinical Trials, Dataspace, and AI: Uniting for Progressive Healthcare Innovation. In Proceedings of the 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 15–18 December 2023; pp. 4997–5006. [CrossRef]
22. Tirpude, S.C.; Shahu, V.R.; Neema, M.G.; Chatterjee, R.K.; Wanve, P.H. Relationship Extraction from Electronic Health Records & Generating Knowledge Graph. In Proceedings of the 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, 9–10 February 2024; pp. 290–295. [CrossRef]
23. Abououf, M.; Singh, S.; Otrok, H.; Mizouni, R.; Ouali, A. Gale-Shapley Matching Game Selection—A Framework for User Satisfaction. *IEEE Access* **2019**, *7*, 3694–3703. [CrossRef]
24. Yuan, J.; Tang, R.; Jiang, X.; Hu, X. LLM for Patient-Trial Matching: Privacy-Aware Data Augmentation Towards Better Performance and Generalizability. American Medical Informatics Association (AMIA) Annual Symposium. 2023. Available online: https://par.nsf.gov/biblio/10448809-llm-patient-trial-matching-privacy-aware-data-augmentation-towards-better-performance-generalizability (accessed on 5 February 2025).
25. Knight, D.R.T.; Aakre, C.A.; Anstine, C.V.; Munipalli, B.; Biazar, P.; Mitri, G.; Valery, J.R.; Brigham, T.; Niazi, S.K.; Perlman, A.I.; et al. Artificial intelligence for patient scheduling in the real-world health care setting: A metanarrative review. *Health Policy Technol.* **2023**, *12*, 100824. [CrossRef]
26. Waruhari, P.; Babic, A.; Nderu, L.; Were, M.C. A Review of Current Patient Matching Techniques. *Stud. Health Technol. Inform.* **2017**, *238*, 205–208.
27. Li, K.; Zhang, Q.; Kwong, S.; Li, M.; Wang, R. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **2013**, *18*, 909–923.
28. Ren, J.; Xia, F.; Chen, X.; Liu, J.; Hou, M.; Shehzad, A.; Sultanova, N.; Kong, X. Matching algorithms: Fundamentals, applications and challenges. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *5*, 332–350. [CrossRef]
29. Elhadjamor, E.A.; Harbaoui, H. A Comparison Analysis of Heart Disease Prediction Using Supervised Machine Learning Techniques. In Proceedings of the 2024 IEEE Symposium on Computers and Communications (ISCC), Paris, France, 26–29 June 2024; pp. 1–6. [CrossRef]
30. Myttenaere, A.D.; Golden, B.; Grand, B.L.; Rossi, F. Mean Absolute Percentage Error for regression models. *Neurocomputing* **2016**, *192*, 38–48 . [CrossRef]
31. Willmott, C.J.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Clim. Res.* **2005**, *30*, 79–82. [CrossRef]
32. Jadon, A.; Patil, A.; Jadon, S. A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting. *arXiv* **2022**, arXiv:2211.02989. [CrossRef]
33. Tjur, T. Coefficients of Determination in Logistic Regression Models—A New Proposal: The Coefficient of Discrimination. *Am. Stat.* **2009**, *63*, 366–372. [CrossRef]
34. Daniels, J.F. Purposeful and timely nursing rounds: A best practice implementation project. *JBI Database Syst. Rev. Implement. Rep.* **2016**, *14*, 248–267. [CrossRef] [PubMed]
35. Mejia, J.M.R.; Rawal, A.; Rawat, D.B. Finite State Automata for Real-Time Health Electronic Record Update: A survey. In Proceedings of the 2022 IEEE Global Humanitarian Technology Conference (GHTC), Santa Clara, CA, USA, 8–11 September 2022; pp. 154–161. [CrossRef]
36. Ramanujam, S.; Capretz, M. Design of a multi-agent system for autonomous database administration. In Proceedings of the Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513), Niagara Falls, ON, Canada, 2–5 May 2004; Volume 2, pp. 1167–1170. [CrossRef]
37. Srinivas Sajja, P. Multi-Agent System for Knowledge-Based Access to Distributed Databases. *Interdiscip. J. Inf. Knowl. Manag.* **2008**, *3*, 001–009. [CrossRef]
38. Wang, B.; Ren, C.; Yang, J.; Liang, X.; Bai, J.; Chai, L.; Yan, Z.; Zhang, Q.W.; Yin, D.; Sun, X.; et al. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. *arXiv* **2024**, arXiv:2312.11242. [CrossRef]
39. Xie, W.; Wu, G.; Zhou, B. MAG-SQL: Multi-Agent Generative Approach with Soft Schema Linking and Iterative Sub-SQL Refinement for Text-to-SQL. *arXiv* **2024**, arXiv:2408.07930. [CrossRef]
40. Nguyen, M.L.T.; Honcharov, V.; Ballard, D.; Satterwhite, S.; McDermott, A.M.; Sarkar, U. Primary Care Physicians' Experiences with and Adaptations to Time Constraints. *Jama Netw. Open* **2024**, *7*, e248827. [CrossRef]
41. Shi, L. The Impact of Primary Care: A Focused Review. *Scientifica* **2012**, *2012*, 432892. [CrossRef]
42. Jackson, I.; Jesus Saenz, M.; Ivanov, D. From natural language to simulations: Applying AI to automate simulation modelling of logistics systems. *Int. J. Prod. Res.* **2024**, *62*, 1434–1457. [CrossRef]
43. Dossou, P.E.; Foreste, L.; Misumi, E. Intelligent Support System for Healthcare Logistics 4.0 Optimization in the Covid Pandemic Context. *J. Softw. Eng. Appl.* **2021**, *14*, 233–256. [CrossRef]
44. Samaridi, N.; Papakitsos, E.; Papoutsidakis, M.; Mouzala, M.; Karanikolas, N. Developing a Logistics Ontology for Natural Language Processing. *WSEAS Trans. Inf. Sci. Appl.* **2024**, *21*, 385–397. [CrossRef]

45. Lu, M.; Ho, B.; Ren, D.; Wang, X. TriageAgent: Towards Better Multi-Agents Collaborations for Large Language Model-Based Clinical Triage. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024*; Al-Onaizan, Y., Bansal, M., Chen, Y.N., Eds.; Association for Computational Linguistics: Miami, FL, USA, 2024; pp. 5747–5764. [CrossRef]
46. Available online: https://deepeval.com/docs/metrics-contextual-relevancy (accessed on 6 March 2025).
47. OpenAI. GPT-4o mini: Advancing Cost-Efficient Intelligence, 2024. Available online: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (accessed on 6 March 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.