

Article

JAVIS Chat: A Seamless Open-Source Multi-LLM/VLM Deployment System to Be Utilized in Single Computers and Hospital-Wide Systems with Real-Time User Feedback

Javier Aguirre ¹  and Won Chul Cha ^{2,3,4,*} 

- ¹ Smart Health Lab, Department of Digital Health, Samsung Advanced Institute for Health Sciences & Technology (SAIHST), Sungkyunkwan University, Seoul 06351, Republic of Korea; javiagu13@gmail.com
- ² Department of Digital Health, Samsung Advanced Institute for Health Sciences & Technology (SAIHST), Sungkyunkwan University, Seoul 06351, Republic of Korea
- ³ Department of Emergency Medicine, Samsung Medical Center, Sungkyunkwan University School of Medicine, 81 Irwon-ro, Gangnam-gu, Seoul 06351, Republic of Korea
- ⁴ Digital Innovation, Samsung Medical Center, Seoul 06351, Republic of Korea
- * Correspondence: docchaster@gmail.com; Tel.: +82-234102053

Abstract: The rapid advancement of large language models (LLMs) and vision-language models (VLMs) holds enormous promise across industries, including healthcare but hospitals face unique barriers, such as stringent privacy regulations, heterogeneous IT infrastructures, and limited customization. To address these challenges, we present the joint AI versatile implementation system chat (JAVIS chat), an open-source framework for deploying LLMs and VLMs within secure hospital networks. JAVIS features a modular architecture, real-time feedback mechanisms, customizable components, and scalable containerized workflows. It integrates Ray for distributed computing and vLLM for optimized model inference, delivering smooth scaling from single workstations to hospital-wide systems. JAVIS consistently demonstrates robust scalability and significantly reduces response times on legacy servers through Ray-managed multiple-instance models, operating seamlessly across diverse hardware configurations and enabling real-time departmental customization. By ensuring compliance with global data protection laws and operating solely within closed networks, JAVIS safeguards patient data while facilitating AI adoption in clinical workflows. This paradigm shift supports patient care and operational efficiency by bridging AI potential with clinical utility, with future developments including speech-to-text integration, further enhancing its versatility.

Keywords: open-source; LLM; hospital on-premises; framework



Academic Editors: Enno van der Velde and Thomas Heston

Received: 14 January 2025

Revised: 31 January 2025

Accepted: 6 February 2025

Published: 10 February 2025

Citation: Aguirre, J.; Cha, W.C. JAVIS Chat: A Seamless Open-Source Multi-LLM/VLM Deployment System to Be Utilized in Single Computers and Hospital-Wide Systems with Real-Time User Feedback. *Appl. Sci.* **2025**, *15*, 1796. <https://doi.org/10.3390/app15041796>

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

1.1. Background

The healthcare industry is rapidly embracing large language models (LLMs) and vision-language models (VLMs) [1–3], due to their potential to significantly improve patient care, support medical professionals, and streamline a variety of clinical and administrative processes. By extracting critical insights from patient records, synthesizing knowledge from medical literature, and offering context-sensitive decision support, these advanced AI models can enhance diagnostic accuracy, personalize treatment plans, and ultimately improve patient outcomes. Recognizing this transformative potential, hospitals and medical institutions are increasingly looking to integrate LLM and VLM capabilities into their internal workflows to advance data-driven, patient-centered healthcare.

1.2. Context and Existing Limitations

Despite the promise of LLMs and VLMs, adopting these technologies within hospital environments poses unique challenges. Privacy regulations [4] (e.g., HIPAA in the U.S. or Data 3 Laws in Korea) demand that patient data remain strictly confidential and within internal, controlled networks. Proprietary solutions, such as closed-source, external LLM services, are not only costly but often violate these stringent requirements by sending sensitive data off-premises.

Although recent multimodal AI frameworks like HAIM [5] and MONAI [6] have gained traction for clinical decision support, they largely focus on traditional machine learning or specific imaging tasks (e.g., using XGBoost, DenseNet, or BERT-like text encoders). These approaches typically rely on smaller or specialized models that may not require the significant computational resources of modern LLMs and VLMs. Furthermore, such frameworks generally lack robust, hospital-wide deployment architectures capable of securely orchestrating multiple concurrent models at scale in an offline environment.

In contrast, current state-of-the-art open-source frameworks designed for LLM deployments, such as OpenLLM, H2O LLM Studio, or RayLLM, still suffer from critical shortcomings in healthcare scenarios. These include inadequate privacy compliance, limited customization options, difficulty in scaling to hospital-wide deployments, and insufficient orchestration tools for managing multimodal workflows within an internal network. As a result, hospitals remain in need of a comprehensive, on-premises solution that can flexibly integrate large-scale LLM/VLM deployments into existing infrastructure while maintaining data security, scalability, and adaptability to evolving clinical needs [7,8].

1.3. Novelty and Contributions

To the best of our knowledge, there currently exists no fully integrated, open-source framework that addresses all of these constraints simultaneously. This paper introduces JAVIS, a novel, healthcare-focused deployment framework for on-premises LLM and VLM solutions. Distinguishing itself from existing systems, JAVIS offers the following:

- **Privacy Compliance:** Unlike OpenLLM or H2O LLM Studio, which rely on external infrastructures or rigid cloud-based workflows, JAVIS enables fully internal deployments, maintaining strict privacy compliance.
- **Customization and Modularity:** JAVIS's architecture supports real-time feedback loops inside private offline networks, customizable components, and dynamic workflows, overcoming the inflexibility seen in other solutions. This flexibility enables hospitals to tailor every aspect, from user interfaces to back-end processing pipelines, to institution-specific requirements.
- **Advanced Scalability and Orchestration:** Through seamless integration with Ray, JAVIS orchestrates multiple replicas of LLMs, VLMs, and supplemental modules (e.g., text-to-speech and multimodal fusion) at scale. This capability extends beyond the static scaling of other frameworks, ensuring that hospitals can efficiently handle increasing workloads, from a single workstation prototype to full hospital-wide deployment.
- **Future-Proof Integration and Maintenance:** While RayLLM introduced certain baseline capabilities, its archival status and lack of a user-friendly interface make it unsuitable for long-term healthcare integration. JAVIS builds upon these core ideas with ongoing support, updated containers, and straightforward maintenance procedures, ensuring sustainability and simplicity over time.

- **Enhanced Compatibility and Containerization:** JAVIS leverages Docker to ensure uniform execution across diverse environments, including legacy systems. This containerization mitigates compatibility issues and streamlines deployment across varying infrastructures.
- **Seamless Deployment and Scaling:** JAVIS provides effortless transitions from small-scale development environments to full hospital-wide production setups. Through modular container configurations, scaling up or down, adding new LLMs, or switching between development and production modes requires minimal commands and no intrusive reconfigurations.
- **Comprehensive Monitoring and Analytics:** JAVIS integrates detailed analytics dashboards and Ray performance-monitoring tools, offering real-time insights into user interactions, resource utilization, and model performance. This enables administrators to make data-driven decisions for capacity planning, resource allocation, and system optimization.
- **Robust Privacy and Regulatory Compliance:** Operating within a secure, closed internal network, JAVIS adheres to stringent privacy and data protection laws (e.g., Korea's Data 3 Laws), ensuring safe, lawful handling of sensitive medical data. Its privacy-by-design approach aligns with healthcare regulations, providing a trusted foundation for secure LLM/VLM deployments.

2. Methods

2.1. Overview of Key Barriers and JAVIS Objectives

Deploying large language models (LLMs) and visual language models (VLMs) in hospital environments presents a range of interconnected challenges that complicate their integration and effective use. A key issue is system compatibility, as hospitals often operate on diverse hardware and software configurations, ranging from varying operating systems to different graphics processing unit (GPU) drivers and CUDA versions, making it difficult to ensure seamless implementation. Adding to this complexity are strict privacy requirements: safeguarding sensitive patient data mandates that AI systems function within secure, closed networks, which creates further obstacles for data management and processing. Moving from experimental prototypes to production-ready applications introduces yet another hurdle, as deployment across multiple clinical settings demands careful adaptation and coordination. Scaling these systems at a hospital level is equally challenging, as managing fluctuating user demands requires robust engineering to maintain consistent performance. Ensuring reliability and consistency is also critical, as AI tools must provide uninterrupted clinical support even under varying loads, while real-time monitoring becomes indispensable for optimizing performance and maintaining trust among healthcare professionals. Finally, hospitals require highly customizable AI solutions that align with their specific workflows and regulatory standards, necessitating adaptable and flexible frameworks. The following sections will explore the methodologies employed by JAVIS to address these critical challenges, showcasing its ability to deliver practical, scalable solutions for deploying LLMs in complex hospital environments.

In the following methods section, these challenges are systematically addressed, focusing on breaking down the barriers to the adoption of large language models (LLMs) and vision-language models (VLMs) in hospitals. Each obstacle, e.g., compatibility, privacy, deployment, scaling, reliability, monitoring, and customization, is tackled through a structured, practical, and scalable approach, enabling the effective use of advanced AI technologies in demanding healthcare environments.

2.2. Customization

Customization is imperative in hospital environments where AI systems must align seamlessly with diverse clinical workflows, regulatory requirements, and specific user preferences. To address these needs, JAVIS integrates a versatile front-end and back-end stack that can be easily modified, extended, and deployed to meet the unique demands of each institution. The following is the underlying JAVIS stack: Vue.js [9], Django [10], Nginx [11], Celery, and SQLite [12].

- **Vue.js (front-end)** enables the rapid development of custom user interfaces and workflows, allowing the system to be tailored to specific hospital department needs.
- **Django (back-end)** manages server-side logic and application programming interface (API) creation, providing a flexible framework for easy feature expansion and adaptation to evolving requirements.
- **SQLite (database)** offers a lightweight, zero-configuration relational database that accelerates development and can be easily scaled or replaced as needed.
- **Nginx (reverse proxy)** handles traffic management and load balancing, ensuring efficient and secure distribution of requests while maintaining system performance.
- **Celery (task queue)** manages asynchronous background tasks, keeping the system responsive by processing long-running jobs without impacting the main application.

2.2.1. Deploying the Customization

JAVIS supports two operational modes for continuous customization and deployment:

- **JAVIS development mode** enables local customization and rapid iteration without downtime, allowing real-time enhancements based on feedback.
- **JAVIS production mode** uses a single Docker build to containerize components for quick, reliable deployment to production servers, ensuring consistency and stability.

Through this comprehensive customization framework, JAVIS empowers hospitals to tailor AI deployments to their specific needs, enhancing usability, compliance, and overall effectiveness.

2.2.2. Continuous Feedback Loop and Deployment Strategy in JAVIS

A continuous feedback loop is integral to the efficient development and deployment of customization requests within the JAVIS system. This loop ensures that all data within the internal network remains secure and confined while JAVIS is updated externally to leverage the latest libraries and tools. Upon completion of updates, a single Docker command transforms JAVIS into a production-ready state, allowing it to be seamlessly reintroduced into the internal network. Throughout this process, critical databases, including user data, analytics, and others, remain intact due to their volume mapping with Docker Compose, as illustrated in (Figure 1). This procedure ensures that data cannot go out due to network security, and new updates to the system are allowed to ensure improvement and customization.

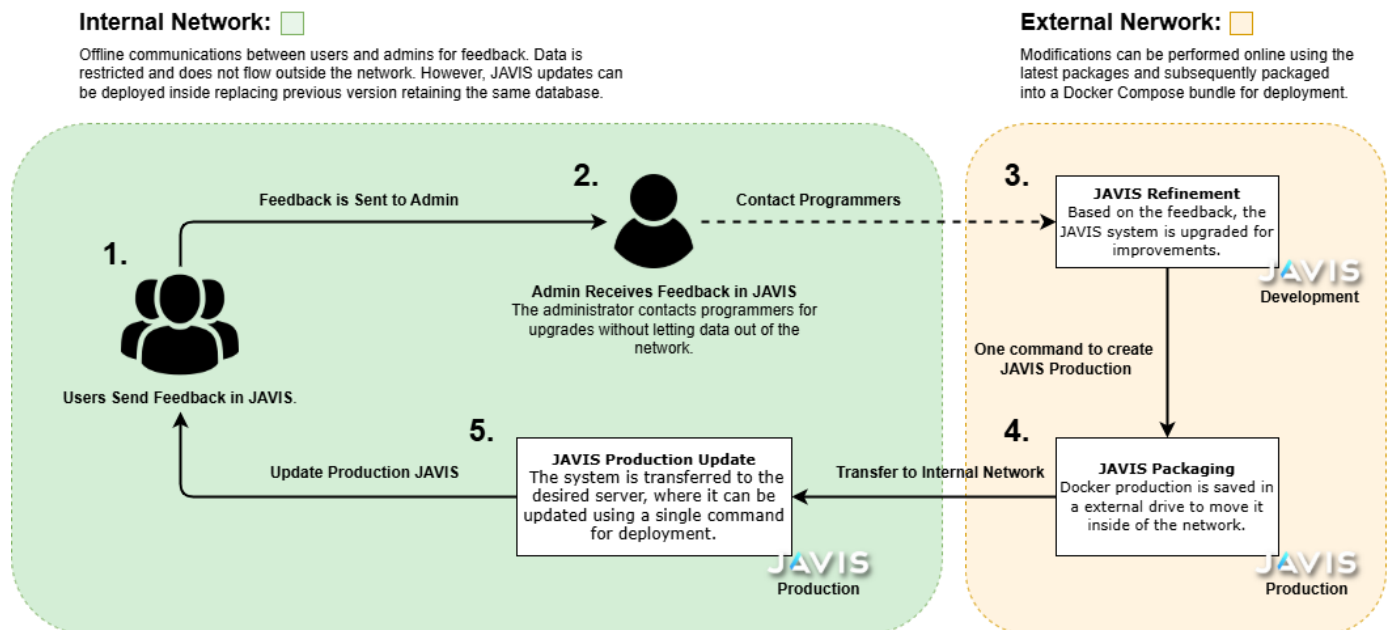


Figure 1. The workflow ensures a streamlined process for enhancing the JAVIS system through user feedback, maintaining a secure environment within the hospital network, and leveraging containerization technology for efficient deployment.

2.3. Scaling and Reliability

Ensuring the scalability and reliability of large language model (LLM) and vision-language model (VLM) deployments within hospital environments is paramount to accommodate varying user demands and maintain consistent performance. JAVIS achieves this through a combination of robust technologies and strategic containerization, enabling seamless scaling from individual workstations to extensive hospital-wide implementations.

2.4. Scaling Technological Foundations

JAVIS leverages a suite of advanced technologies to build a resilient and scalable infrastructure based on FastAPI [13], vLLM [14], and Ray [15]:

- **FastAPI** provides a high-performance framework for efficient model serving, enabling the rapid handling of incoming requests with minimal latency.
- **vLLM** optimizes LLM and VLM inference by enhancing speed and resource utilization, ensuring that models operate efficiently, even under heavy loads.
- **Ray** facilitates distributed computing and scalable orchestration, allowing JAVIS to manage and distribute workloads across multiple GPUs and nodes effectively.

These technologies collectively ensure that JAVIS can deliver high-performance LLM/VLM services reliably, regardless of the deployment scale.

2.4.1. Containerization Strategy

To address diverse deployment needs, JAVIS employs a container with two distinct modality templates: a **versatile mode** and a **scaling mode**. Each modality is tailored to specific operational contexts, providing optimized solutions for different scales of deployment that are accessed within the same container.

2.4.2. Versatile Mode (Local or Small-Scale Deployments)

Designed for environments with limited user bases, such as individual workstations or small laboratory servers, the versatile mode offers a lightweight yet powerful solution using vLLM and FastAPI:

- **Rapid deployment** facilitates the swift deployment of single model instances with minimal configuration, allowing researchers or clinicians to integrate LLMs into their workflows quickly without extensive technical overhead.
- **Easy duplication** supports hosting multiple models in parallel by simply copying the container definition and running additional instances. This feature is ideal for testing, demonstrations, or specialized use cases within small groups.
- **Robust performance** ensures consistent performance and low latency on personal computers or small servers, maintaining reliability for a limited number of users.

2.4.3. Scaling Mode (High-Volume Production)

For larger user bases and production environments, the scaling mode is engineered to handle high-volume demands with enhanced scalability and reliability using vLLM and Ray:

- **Distributed model serving** utilizes Ray to distribute model instances across multiple GPUs and nodes, enabling simultaneous handling of numerous requests. This distribution ensures high performance and low latency, even as concurrent user numbers increase.
- **Load balancing and horizontal scaling** leverages Ray's cluster orchestration capabilities to facilitate dynamic load balancing and horizontal scaling. As user demand grows, additional replicas can be spawned automatically to manage the increased load, maintaining consistent response times and system responsiveness.
- **Enhanced reliability** incorporates Ray's dynamic worker allocation and failover mechanisms to bolster deployment reliability. Comprehensive task scheduling ensures efficient resource utilization and minimizes downtime, even in the event of node failures.

2.5. Scalability and Reliability Test Setup

To evaluate the performance and scalability of JAVIS, we conducted two distinct experiments: an efficiency test on a standard consumer personal computer (PC) and a high-volume stress test using a multi-GPU setup. These experiments were designed to assess JAVIS's ability to handle varying computational demands and user loads, ensuring its robustness and reliability in clinical environments.

For all the experiments, the following models and quantizations were used in order to represent different size deployment, as well as LLM and VLM testing.

Models and Formats

- **LLMs:** We deployed Llama 3.2 with 1 billion (1b) and 3 billion (3b) parameters, as well as Llama 3.1 with 8 billion (8b) parameters.
- **VLMs:** The multimodal models included Qwen2 2b and Qwen2 7b parameters, along with Llava 1.5 with 13 billion (13b) parameters.
- **Quantization formats:** LLMs were deployed in a generalized post-training quantization (GPTQ) quantized [16] format to facilitate fast inference while maintaining high accuracy. VLMs utilized the activation-aware weight quantization (AWQ) [17] format, chosen for its availability and demonstrated ability to retain high accuracy.

2.6. Experimental Setup: High-Volume Stress Testing with a Multi-GPU Setup

The first experiment was designed to demonstrate JAVIS's capability to handle high volumes of concurrent users, thereby validating its scalability and reliability in production environments. This stress test was conducted using a setup with four NVIDIA A6000 GPUs, each with 50 GB of memory, providing substantial computational resources to support large-scale deployments.

2.6.1. Test Procedure

The stress test involved simulating varying numbers of concurrent users querying the system. For each LLM, we executed 20 clinically relevant prompts, as shown in (Table A2) in the Appendix A. For each VLM, 10 different images with 2 prompts per image resulting in a total of 20 prompts per model were tested. The specific prompts used are listed in (Table A3) in the Appendix A. We tested user loads of 10, 20, 30, 40, 50, 75, 100, 125, and 150 users. In order to measure the robustness of the container as well as the robustness of JAVIS, this experiment queried the JAVIS back-end to measure the complete roundtrip response times. This approach ensured that the test accurately reflected real-world usage scenarios, including the overhead of back-end processing and model orchestration.

2.6.2. Scalability Assessment

To assess scalability, we compared the performance of hosting a single model instance (1 replica) against multiple replicas orchestrated by Ray. For models with 7 billion parameters or larger, we could host up to 12 replicas, while smaller models (below 7 billion parameters) supported up to 20 replicas.

2.6.3. Performance Metrics

The key metric was the response time and standard deviation experienced by each simulated user under different load conditions. By measuring the response times across various user volumes, we evaluated whether the scaling strategies implemented by JAVIS could maintain consistent performance as demand increased. Additionally, we randomized the prompts for each query to simulate realistic usage patterns, ensuring that the system's performance was not biased by repetitive or predictable input sequences.

2.7. Experimental Setup: Efficiency Evaluation on a Standard Consumer PC

The second experiment aimed to determine whether JAVIS can operate efficiently on a typical consumer-grade PC equipped with a standard GPU. We utilized an NVIDIA RTX 3060 with 12 GB of GPU memory for this test. The experiment involved deploying various large language models (LLMs) and vision-language models (VLMs) and measuring their average response times across a set of clinically relevant prompts.

Test Procedure

We deployed each model using the versatile mode, which is optimized for local or small-scale deployments. The same 20 prompts of (Table A2) were sent to LLMs, and the 20 prompts of (Table A3) (10 images and 2 prompts per image) were sent to VLMs. The test was performed in a non-concurrent manner (iterative), and response time in seconds and standard deviation were captured in order to assess the efficiency.

2.8. Compatibility Challenges

Hospitals typically operate a diverse array of systems, each with distinct operating systems, drivers, CUDA versions, and software dependencies. This diversity complicates the deployment of complex AI models, as inconsistencies between environments can lead to conflicts, reduced performance, and increased maintenance efforts. Ensuring that large language models (LLMs) function uniformly across all these varied systems requires a robust solution that can encapsulate all necessary dependencies and configurations.

2.9. Docker and Docker Compose as Solutions

To overcome these compatibility challenges, JAVIS utilizes **Docker** [18] and **Docker Compose**. Docker containerizes applications by bundling them with their own operating systems, CUDA libraries, and all required dependencies. This encapsulation ensures that each component of JAVIS operates within a controlled and consistent environment independent of the host system's configuration.

2.9.1. JAVIS Deployment Stack: Docker Compose

The core components of the JAVIS system, including Nginx, Django, SQLite, Celery, and Vue.js, are managed using Docker Compose. This tool enables seamless orchestration and management of the entire stack, ensuring that all services work together within a consistent environment.

2.9.2. Benefits

- **Isolation:** Each Docker container runs its own instance of the operating system and dependencies, preventing conflicts between different components and host environments.
- **Portability:** Docker images can be deployed across any system that supports Docker, ensuring consistent behavior regardless of underlying hardware or software variations.
- **Simplified deployment:** Docker Compose orchestrates multiple containers, allowing the entire JAVIS stack, including Nginx, Django, SQLite, Celery, and Vue.js, to be deployed with a single configuration file. This streamlines the setup process and reduces the potential for configuration errors.

2.9.3. Flexible LLM Deployment: Docker Containers

In addition to the core JAVIS stack, LLMs are deployed in a separate, isolated Docker container. These containers communicate with the JAVIS core via defined APIs, offering several benefits.

2.9.4. Benefits

- **Modularity:** LLMs can be deployed independently of JAVIS, allowing hospitals to integrate them with existing systems and workflows without disrupting the core infrastructure.
- **Scalability:** Containerized LLMs can be easily scaled horizontally by deploying additional instances by utilizing the versatile mode or switching to the scaling container as needed to handle increased demand.
- **Versatility:** Whether deployed locally on individual workstations or distributed across multiple servers, the isolated container ensures consistent performance and ease of management.

Furthermore, containerized LLMs can function as standalone APIs, providing hospitals with the flexibility to connect them with pre-existing systems without necessitating full JAVIS deployment. This adaptability is crucial for integrating AI capabilities into diverse and already-established healthcare IT ecosystems.

2.9.5. JAVIS Architecture for Compatibility

The development and production modes of JAVIS facilitate customization and streamline system deployment, while the LLM/VLM container simplifies LLM deployment. (Figure 2) illustrates the overall architecture.

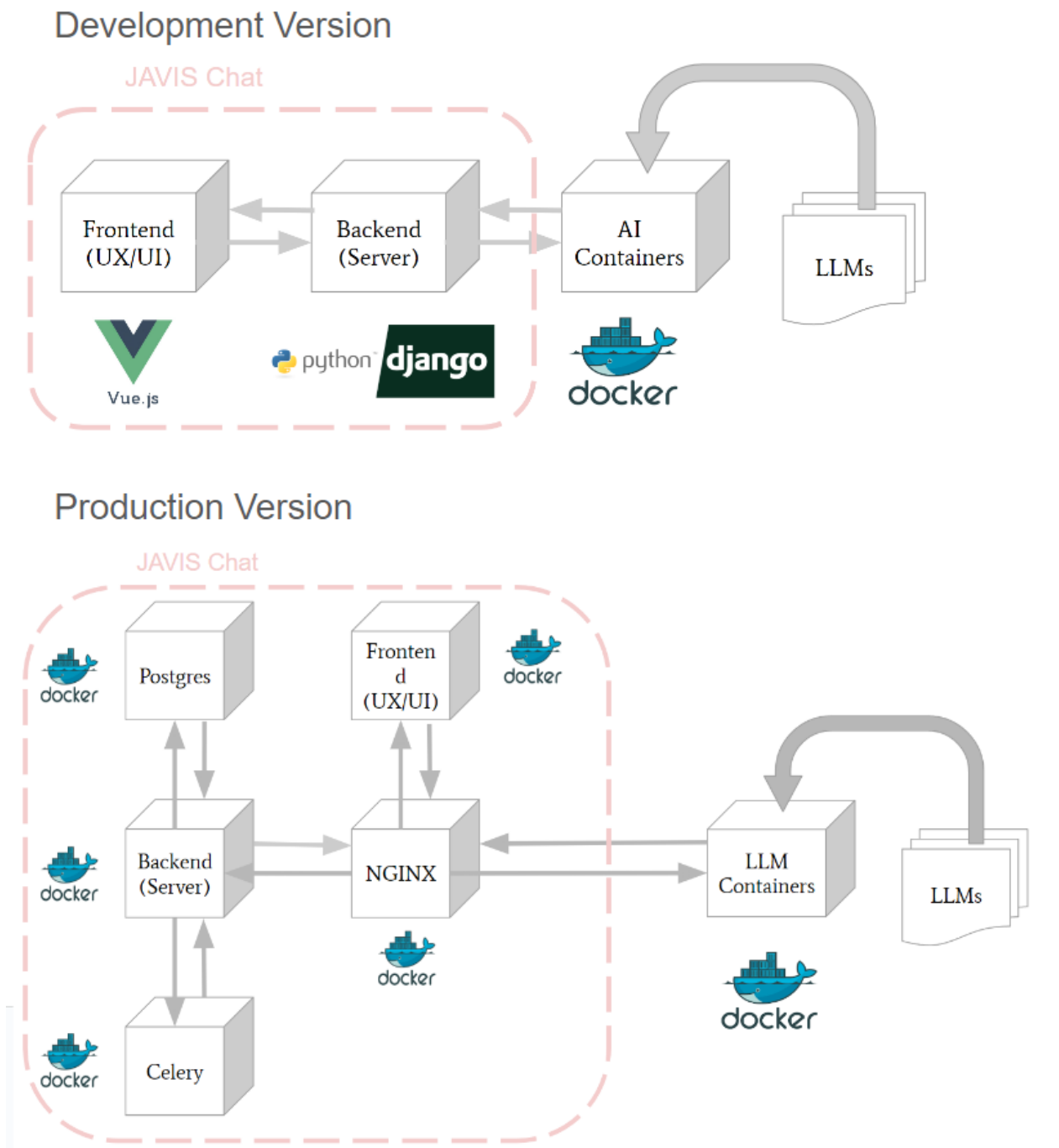


Figure 2. In the upper part, **Development JAVIS** can be seen. Its front-end in Vue.js and back-end with Django communicate through an API endpoint to locally hosted LLMs inside the Docker container. Below, **Production JAVIS** shows Docker Compose powered by the Nginx container, which acts as the central routing hub, directing traffic between the front-end and back-end containers. The front-end container communicates with the back-end container via Nginx to handle dynamic content and API requests. The back-end container, which manages server-side logic, interacts directly with the PostgreSQL container for database operations and with the Celery container for executing asynchronous tasks. Celery may also access PostgreSQL if needed for specific background tasks.

2.10. Experimental Setup: Compatibility Testing

To assess compatibility, a range of configurations representative of real-world conditions were evaluated. The operating systems tested included both older and newer versions of Linux, specifically versions 18.04 and 24.04, due to the predominant use of Linux in server environments. Additionally, deployments were carried out on Windows 10.

In terms of graphics hardware, both data center and consumer-grade GPUs were examined, including the NVIDIA A6000 with 50 GB of memory, NVIDIA GeForce RTX 3050 with 8 GB, and NVIDIA GeForce RTX 3060 with 12 GB. Various driver versions were also tested, spanning from the older 535 series to the latest 561 series. Moreover, different CUDA versions, namely 12.2 and 12.6, were utilized in the evaluations.

For each configuration, the system was deployed, and the installation time was recorded to ensure compatibility and evaluate the speed of setup.

2.11. Seamless Deployment

JAVIS was designed to facilitate effortless transitions across multiple levels of deployment, from a **personal/small** team setup to full **hospital/institution**-wide integration. This scalability has been achieved through a carefully orchestrated, containerized workflow using Docker Compose for JAVIS and Docker for LLM/VLM containers, ensuring that libraries, dependencies, databases, and core components are consistently configured at every stage.

Transition of scale levels: Transitioning JAVIS from a development environment to a production setting is as simple as running a single Docker build command. Furthermore, JAVIS integrates smoothly with the previously described versatile mode (for small user bases such as personal computers or small teams) and with the scaling mode (for hospital or institution-level user loads). Replacing the modality of containers is as simple as using a different command. For the versatile mode, `uvicorn run_llm:app -host 8000 -port 0.0.0.0` is used, and for running the scaling mode, `python run_ray_llms_vlms.py` command is employed.

Addition of LLMs: Integrating additional large language models (LLMs) is straightforward, as it involves updating the container configuration and launching new containers. JAVIS automatically routes queries to the newly added models through predefined APIs, facilitating easy expansion and experimentation.

Upgrading system capabilities: By decoupling the JAVIS core stack from the container, you can effortlessly upgrade the system's capabilities to handle large user volumes without affecting the primary JAVIS services. Moreover, these decoupling capabilities are especially valuable when the hospital requires a direct API for LLM/VLM deployment for any other use case, such as a common data warehouse (CDW) connection or powering any other hospital service. By leveraging a modular, container-based solution, we can initially support a small group of users for experimentation. Then, with just a simple two-command scaling process, the container can be expanded to serve a broad user base at the institutional level, all within the hospital's internal network.

Rapid Container Modality Switching for Scaling

JAVIS supports two primary container templates to accommodate varying operational scales:

- **A versatile mode**, which is ideal for single-machine or small lab setups.
- **A scaling mode**, which is optimized for large, institution-wide deployments.

Furthermore, JAVIS is split into two versions:

- **Development JAVIS**, which is ideal for tailoring the system to a given hospital or institution.
- **Production JAVIS**, which was built for large, institution-wide deployments.

In summary, each of the four components is designed with a modular approach and combining them as needed offers a solution to satisfy varying needs, from system tailoring and serving a small user base to scaling to an institutional level.

2.12. Monitoring

Effective monitoring is essential to ensure the reliability and performance of JAVIS in dynamic hospital environments. JAVIS incorporates comprehensive analytics and performance monitoring tools to provide valuable insights and facilitate proactive management.

2.12.1. JAVIS Analytics Dashboard

The **JAVIS analytics dashboard** is an integrated tool designed to track and analyze user interactions with deployed large language models (LLMs). Key features include the following:

- **User interaction tracking**, which logs detailed information on user interactions, including timestamps, selected models, and query volumes.
- **Aggregated metrics**, which compile usage metrics on an hourly, daily, weekly, and monthly basis, allowing administrators to observe trends and patterns over various timeframes.
- **Filtering capabilities**, which enable the filtering of metrics by individual users or user groups, providing granular insights into specific usage behaviors.
- **Informative insights**, which utilize the collected data to inform capacity planning, optimize resource allocation, and prioritize feature development based on actual usage patterns.

These analytics empower administrators to make data-driven decisions, enhancing the overall efficiency and effectiveness of JAVIS deployment.

2.12.2. Ray Performance Dashboard

For deployments utilizing the **scaling mode**, JAVIS integrates the **Ray performance dashboard**, which offers real-time monitoring of system performance and resource utilization. Key functionalities include the following:

- **Replica status monitoring**, which displays the status of all active replicas, ensuring that each instance of LLMs is functioning correctly.
- **Resource utilization metrics**, which provide real-time data on CPU and GPU usage, as well as memory consumption, allowing for immediate identification of potential bottlenecks.
- **A cluster health overview**, which offers a comprehensive view of the overall health of the computing cluster, including network performance and node status.
- **Model serving loads**, which track the load on each model serving instance, facilitating the balanced distribution of requests and preventing overloading of individual nodes.

The Ray performance dashboard enables administrators to perform prompt interventions, optimize resource allocation, and make informed scaling decisions to maintain optimal system performance under varying demand conditions.

2.12.3. Privacy and Regulatory Compliance

The JAVIS system is fully compliant with the Data 3 Laws in South Korea, which collectively govern data privacy, utilization, and protection. These laws, comprising the *Personal Information Protection Act (PIPA)*, the *Act on Promotion of Information and Communications Network Utilization and Information Protection (Information and Communications Network Act)*, and the *Credit Information Use and Protection Act*, serve as the Korean equivalent of the Health Insurance Portability and Accountability Act (HIPAA) in the United States. Together, they establish comprehensive guidelines for safeguarding sensitive personal and health-related data, emphasizing principles such as data minimization, secure processing, and accountability.

To ensure strict compliance with these laws, the JAVIS system operates exclusively within a secure, closed internal network environment. This architecture minimizes ex-

posure to external threats and unauthorized access while providing multiple layers of protection, including encryption, firewalls, intrusion detection systems, and role-based access control (RBAC).

The system adheres to the following principles of privacy and security:

- **Data minimization and protection:** Sensitive information is processed and stored in compliance with the Data 3 Laws at secure, PIPA-compliant domestic facilities. Encryption and secure access controls ensure data remains protected against breaches and misuse.
- **Pseudonymization and anonymization:** all data received through JAVIS is already anonymized due to the network's security, allowing the secure utilization of data for purposes such as statistical analysis, scientific research, and service optimization without compromising individual privacy.
- **Monitoring and accountability:** The system maintains comprehensive audit logs and monitoring mechanisms, enabling real-time detection of unauthorized access attempts. Regular vulnerability assessments, penetration tests, and compliance audits are conducted to ensure the system remains robust and secure.

By aligning with the Data 3 Laws, the JAVIS system embodies a *privacy-by-design* approach. Its secure infrastructure, advanced safeguards, and adherence to regulatory standards ensure the safe and lawful processing of sensitive information. This makes JAVIS a model for secure, regulatory-compliant software systems in the healthcare and data industries.

3. Results

3.1. Customization

The deployment of JAVIS within hospital environments has demonstrated its capability to rapidly customize and adapt to the specific needs of various departments. Leveraging the integrated feedback mechanism, different users submitted customization requests directly within the system, and the administrator received this feedback and prioritized feature development accordingly.

JAVIS features are separated into foundational features and requested features, where the foundational represent the functions built within JAVIS, and the requested are made possible due to the feedback loop between different departments and the admin. The foundational features are explained first, followed by the requested features made possible through the feedback loop.

In this case, since the administrator was the main programmer, most of the requests were able to be deployed within the same day of the request. Figure 3 displays the chat interface and the feedback panel that enabled the following results.

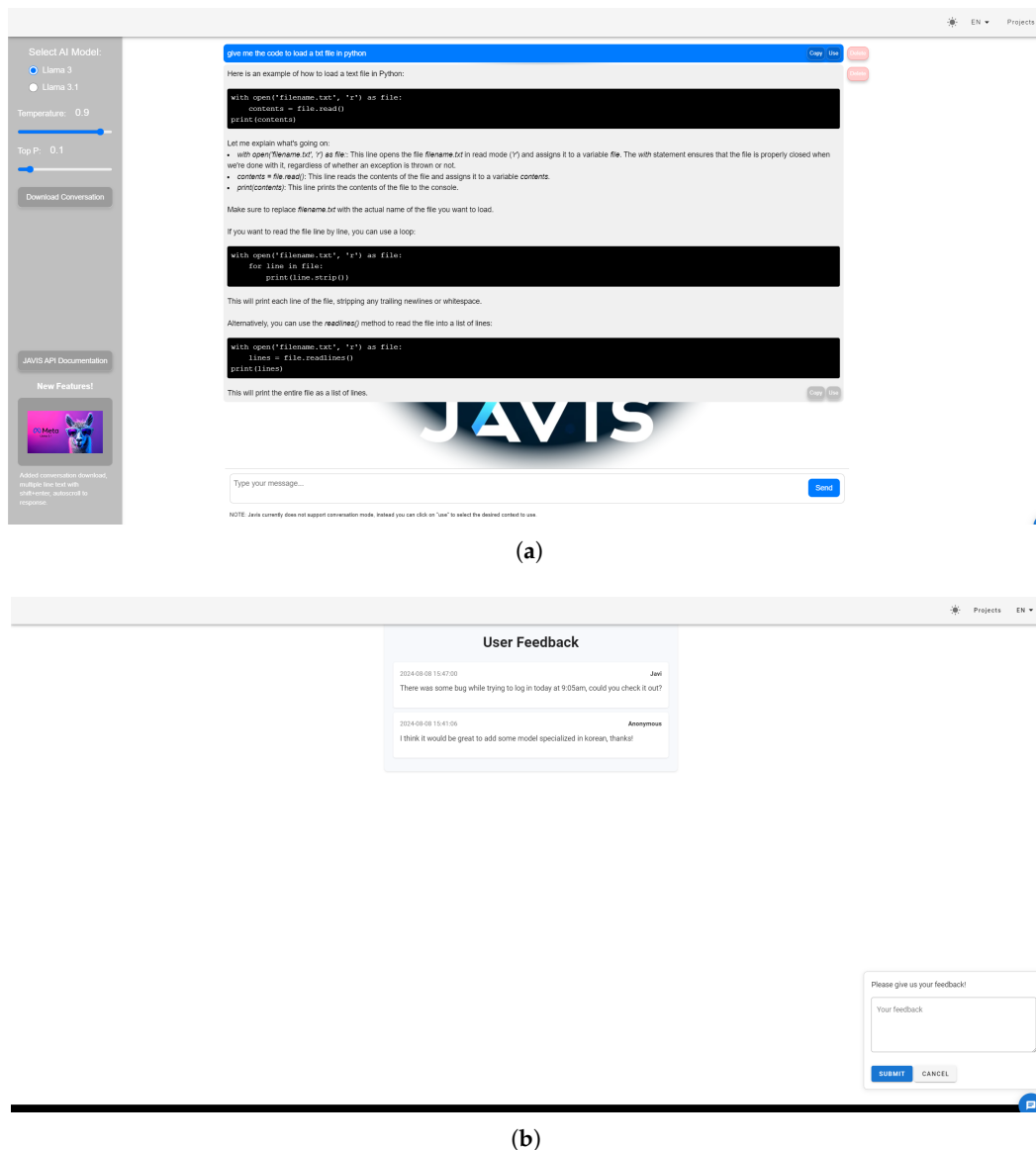


Figure 3. JAVIS system interfaces: chat and feedback modules. **(a)** JAVIS chat interface. On the left, model selection is shown, including top p and top k selection, as well as the download conversation feature. In the middle, the JAVIS chat with a sample code is displayed in code snippets. **(b)** JAVIS admin feedback panel. The right bottom icon is available through JAVIS to send feedback at any time.

3.2. Foundational Features

Here are the main features that are currently supported in JAVIS.

3.2.1. Multi-LLMs

The interface supports talking with different LLMs hosted in the containers; therefore, as many LLMs as the infrastructure allows can be hosted, and LLM switching is carried out in the left drawer with a simple click. This will redirect the call to the corresponding LLM API by passing the corresponding prompt template for that model in the back-end.

3.2.2. Multi-VLMs

JAVIS supports multimodal LLMs or VLMs, allowing the integration of various image types (JPG, PNG, etc.). This enhancement significantly increases the system's versatility in handling diverse clinical inputs, enabling more comprehensive and accurate patient interactions.

3.2.3. Chain of Thought

JAVIS includes a chain of thought feature that allows users to save and reuse their chains. This functionality enables any LLM to reason in a chain-of-thought manner and enables clinicians to build upon previous reasoning steps. A user can define a set of prompts, and they will be chained after each LLM response automatically to provide an advanced reasoning response. This allows accessibility to recurrent tasks and the ability to obtain higher accuracy for hard problems in a way that can be saved and reused in the future.

3.2.4. Rich Text Formatting

The interface automatically formats text, supporting tabs, bolding, code detection, and overall LLM output formatting, which enhances the clarity and organization of communication in clinical settings, as seen in (Figure 3).

3.2.5. Expandable Input Box and Auto-Scrolling

The input box is designed to expand, enabling users to view multiple lines of text simultaneously, which is particularly beneficial when reviewing detailed patient descriptions. Additionally, a smooth auto-scrolling function ensures seamless interaction by automatically adjusting the view as new messages are received.

3.2.6. File Uploads

The interface supports uploading common hospital file types (CSV, XLSX, TXT, DOC, and DOCX), transforming them into text on the front-end. This capability facilitates the integration of external data into conversation flows without overburdening the back-end system.

3.2.7. Contextual Question Answering

Users can select specific messages as context for question answering, improving accuracy and reducing the risk of information misinterpretation in sensitive clinical environments.

3.2.8. Temperature and Top-p Modifiers

For laboratory settings, the UI includes options to adjust temperature and top-p parameters, allowing experimentation with different response-generation styles and behaviors.

3.2.9. API Documentation

Comprehensive API documentation is readily accessible through the UI, guiding users on effectively utilizing the JAVIS API for connecting LLMs with other hospital systems and simplifying integration processes.

3.3. Requested Features

Several key customization requests were successfully implemented, showcasing JAVIS's flexibility and rapid development capabilities thanks to the previously described feedback loop.

3.3.1. EMR Team: Conversation Download

The EMR development team requested the ability to download conversation transcripts as .txt files for upload to their clinical data warehouse (CDW), along with the functionality to delete specific messages. Feedback was requested inside of JAVIS, leveraging Vue.js's flexible component design. These features were developed, containerized, updated, and tested within approximately five hours.

3.3.2. Researchers: Code Formatting

Researchers sought code outputs formatted similarly to ChatGPT, featuring a black background and copy-friendly snippets to enhance readability and usability. Adjusting the Vue.js front-end to incorporate this styling and formatting was accomplished in approximately two hours. The updated interface was quickly deployed across all users, and researchers benefited from a more comfortable user experience.

3.3.3. Physicians: Copy Button

Emergency room (ER) physicians requested a one-click “Copy” button to extract and reuse medical records in subsequent prompts. Implementing this user interface element required roughly thirty minutes of coding and testing. The new feature was efficiently rolled out to the ER department, allowing physicians to enhance their workflow with minimal disruption.

These outcomes highlight the efficiency and effectiveness of JAVIS in delivering customized AI solutions tailored to specific workflows and requirements of hospital environments. The seamless deployment enabled by a feedback loop ensures that these enhancements can be rapidly scaled and maintained.

3.4. *Scaling and Reliability*

This section presents the performance outcomes of deploying JAVIS using two distinct container configurations: the versatile mode on a consumer-grade GPU and the scaling mode on a legacy server equipped with four NVIDIA A6000 GPUs. These configurations were evaluated to assess JAVIS’s adaptability, scalability, and reliability in handling diverse workloads within hospital environments.

3.4.1. Scaling Mode: Scaling and Reliability at Scale

To evaluate JAVIS’s scalability and reliability under higher user loads, a container utilizing the scaling mode was deployed on a legacy server running Ubuntu 18.04, equipped with four NVIDIA A6000 GPUs, each possessing 50 GB of memory. Despite the A6000’s older and comparatively slower architecture relative to contemporary accelerators such as the A100 or H100, the container delivered robust performance. This outcome underscores JAVIS’s ability to efficiently produce results even on smaller and less advanced server configurations.

3.4.2. Stress Testing

Comprehensive stress tests were conducted to assess JAVIS’s robustness and scalability. Concurrent user interactions were simulated by dispatching simultaneous queries from increasing numbers of users, i.e., 10, 20, 30, 40, and 50, to evaluate system performance under scaling load conditions. Additional scenarios involving 75, 100, 125, and 150 concurrent users were analyzed and are presented in Appendix A (Table A1).

Preliminary results, summarized in (Table 1), demonstrate that JAVIS maintains efficient response times as user load increases. Notably, multiple-instance (Ray) models consistently outperform single-instance models across all tested concurrency levels, underscoring the effectiveness of scaling mode containers in managing distributed workloads. This disparity emphasizes the substantial advantages of using multiple replicas to distribute computational demand and maintain low-latency performance.

Table 1. Performance comparison of single-instance and replica models. Single instance refers to a single replica of each Llama model, whereas replica Models consist of 20 replicas for the Llama-1B and Llama-3B models and 12 replicas for the Llama-8B model. The table presents the mean response time (\pm standard deviation) in seconds for each user group (10, 20, 30, 40, and 50 concurrent users). These results demonstrate the robustness of the scaling mode container, illustrating that as the number of concurrent users increases, the response time per query becomes more pronounced, particularly for single-instance models.

Model Name	Type	10 Users	20 Users	30 Users	40 Users	50 Users
Single-Instance Models						
Qwen2 2b	VLM	5.559 \pm 2.696	10.435 \pm 5.554	15.412 \pm 9.321	22.001 \pm 12.731	24.624 \pm 14.281
Qwen2 7b	VLM	5.328 \pm 2.281	14.707 \pm 6.930	16.487 \pm 8.810	22.261 \pm 11.897	38.554 \pm 21.313
Llava 13b	VLM	8.936 \pm 4.228	15.412 \pm 7.147	21.148 \pm 12.046	29.623 \pm 17.567	29.830 \pm 18.053
Llama 1b	LLM	7.295 \pm 4.926	12.455 \pm 7.518	23.649 \pm 11.872	32.659 \pm 17.825	46.858 \pm 24.627
Llama 3b	LLM	14.966 \pm 7.089	23.895 \pm 11.596	34.184 \pm 18.992	50.214 \pm 28.914	55.470 \pm 31.084
Llama 8b	LLM	20.624 \pm 11.779	37.698 \pm 21.906	56.025 \pm 34.384	65.835 \pm 33.842	102.509 \pm 60.606
Multiple-Instance Models (Ray)						
Qwen2 2b	VLM	4.667 \pm 1.193	3.734 \pm 1.297	4.635 \pm 1.823	5.919 \pm 2.563	6.673 \pm 4.006
Qwen2 7b	VLM	3.122 \pm 1.379	4.478 \pm 2.458	6.667 \pm 3.532	10.054 \pm 6.426	8.461 \pm 3.814
Llava 13b	VLM	3.342 \pm 1.093	5.512 \pm 1.950	7.949 \pm 4.255	8.024 \pm 3.564	10.024 \pm 4.667
Llama 1b	LLM	5.092 \pm 3.495	4.664 \pm 3.667	6.172 \pm 3.961	9.492 \pm 5.306	9.169 \pm 7.175
Llama 3b	LLM	5.208 \pm 3.357	6.093 \pm 4.569	9.046 \pm 4.595	8.723 \pm 5.127	15.909 \pm 7.837
Llama 8b	LLM	5.889 \pm 3.477	12.563 \pm 6.528	16.241 \pm 7.456	14.167 \pm 8.645	16.938 \pm 11.953

3.4.3. Analysis of Stress Testing Results

Table 1 highlights the differences between single-instance (above) and multiple-instance (below) Ray-based model deployments. Single-instance deployments represent models run directly by VLMs without replication, while multiple-instance deployments leverage Ray to orchestrate and dynamically scale the number of VLM replicas across all available GPUs. By utilizing Ray's ability to create as many replicas as requested (in this case, up to 20 replicas for models under 7B parameters and up to 12 replicas for models above 7B), our system efficiently employs all four A6000 GPUs to handle increasing user loads. As user concurrency grows (from 10 to 20, 30, 40, 50, and beyond), Ray automatically scales the instances accordingly. The comparison in Table 1 consistently shows that multiple-instance models outperform their single-instance counterparts at every tested concurrency level. This performance gap widens as the number of concurrent users rises, demonstrating that distributing the workload across multiple replicas not only improves latency but also stabilizes response times under heavy load.

To illustrate this further, consider the case of the LLaMA 8B model at different user loads: 10, 30, 50, and 100 concurrent users. Under single-instance deployment, the model experiences significant latency increases as the user load intensifies. For example, at 10 concurrent users, a single-instance LLaMA 8B model averages about 20.6 s. At 50 users, it jumps to an average of 102.5 s. In contrast, when employing multiple Ray-managed replicas of the LLaMA 8B model, the response times are dramatically lower and much more stable across these conditions. At 10 concurrent users, multiple instances respond in roughly 5.9 s on average; even at 50 concurrent users, they maintain a reasonable 16.9 s. The disparity becomes even clearer at 100 users, where single-instance LLaMA 8B models exceed 146 s while multiple-instance models hold steady at around 35.6 s, showcasing Ray's ability to dynamically adjust resources and sustain performance as concurrency grows. This disparity is depicted in Figure 4, where the rate of growth of single-instance models accelerates as the user number grows, and the rate of multiple instances remains constant, even under heavy loads.

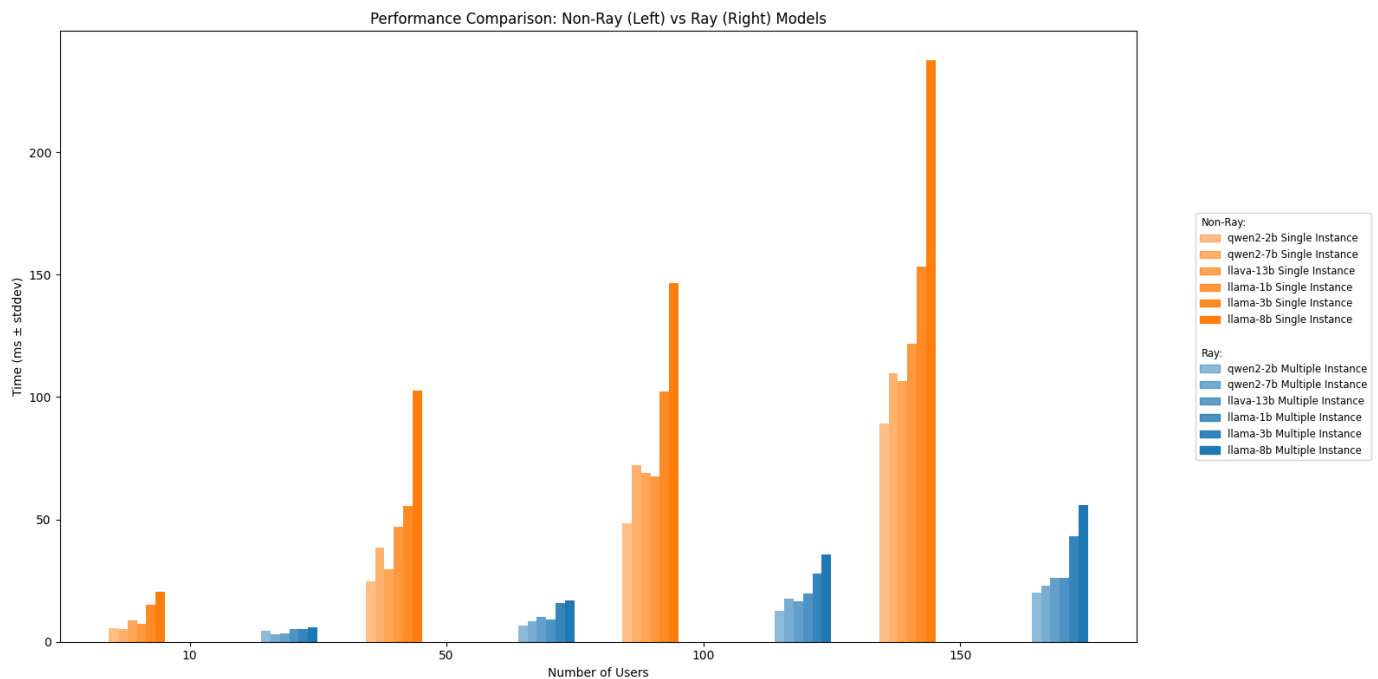


Figure 4. Single-instance (non-Ray) vs. multiple-instance (Ray) response time results per user. Each bar inside each bar plot group represents one model in the following order: qwen2 1b, qwen2 7b, llava 13b, llama 1b, llama 3b, and llama 8b. The same order is preserved for the blue plot represented by the models leveraged by autoscaling with Ray.

These results underscore JAVIS’s capacity to handle substantial user loads, even on a relatively modest four-GPU A6000 cluster. In a real-world scenario, it is rare that all users query the system simultaneously; there is usually “think time”, reading time, or idle periods between queries. As such, the effective number of users that can be supported in production scenarios is even higher. For instance, in our lab setup, with a single GPU card and just two instances, supporting around 40 researchers with acceptable latency is achievable. Stress tests, which were deliberately designed to push the system with simultaneous, synchronized queries, indicate that JAVIS can scale reliably. The back-end infrastructure has been validated to operate consistently under heavy loads, maintaining communication with the containers without failing or degrading catastrophically.

In summary, these experiments show that with a four-A6000 GPU cluster, even models at 8B parameters (**Llama 8B**) can serve up to **50 simultaneous users** in approximately **15 s** (multiple instances) versus over 100 s (single instance). For equivalent size VLM (**Qwen 7B**) at **100 users** load, multiple instances can respond in approximately **16 s**, while single instances may take around 68 s. Furthermore, for smaller 1B models (**Llama 1B**), we can serve as many as **150 concurrent users** with an approximately **20 s response time** compared to almost 90 s with a single instance. These outcomes highlight the robustness, scalability, and reliability of JAVIS. By leveraging Ray’s scaling capabilities, JAVIS ensures that even under substantial concurrency, performance remains stable and responsive, allowing for a significantly enhanced user experience.

It is important to note that these results were achieved on a relatively modest server with four A6000 GPUs. Therefore, increasing GPU availability would significantly enhance the capacity to handle larger user loads. Additionally, the results were based on simultaneous queries, meaning the actual user load that can be supported is likely much higher, as users typically have pauses between messages.

3.4.4. Versatile Mode: Performance on Consumer GPUs

The results of running 20 different prompts and evaluating the seconds and standard deviation elapsed are shown in (Table 2).

Table 2. Performance metrics for Llama GPTQ models on consumer GPUs.

Model Name	Type	RTX 3060 (12 GB)	A6000 (50 GB)
Llama 1B	LLM	3.36 ± 0.95	2.61 ± 0.61
Llama 3B	LLM	5.74 ± 1.58	3.90 ± 1.04
Llama 8B	LLM	9.21 ± 1.77	5.09 ± 1.14
Qwen2 2B	VLM	1.72 ± 1.28	1.30 ± 0.75
Qwen2 7B	VLM	4.68 ± 3.95	3.16 ± 1.76
Llama 13B	VLM	5.72 ± 4.53	3.50 ± 2.41

The performance metrics presented in Table 1 illustrate JAVIS's capability to efficiently handle a range of language and vision-language models on a consumer-grade NVIDIA GeForce RTX 3060 12 GB GPU. Notably, the response times increase with model size within the Llama GPTQ series, from an average of 3.36 s for the Llama-1B model to 9.21 s for the Llama-8B model. This trend highlights the expected scalability challenges as model complexity grows; however, the performance remains within practical limits for many hospital-related applications.

Conversely, vision-language models (VLMs) exhibit generally faster response times, with the Qwen2-2B model achieving an average of 1.72 s. Even larger Qwen2-7B and Llava-1.5 AWQ models maintain reasonable response times of 4.68 and 2.42 s, respectively, despite increased variability. These results demonstrate that JAVIS can effectively manage both smaller and moderately large models on consumer-grade hardware, ensuring versatility and reliability in diverse hospital environments where high-end computational resources may not be readily available. The **minimum average time** is **1.72 s** for Qwen 7B, whereas the **maximum time** is **9.21 s** for Llama 8B. In **all cases**, the **response time** is always **below 10 s**, concluding JAVIS's adaptability and robustness, making it a viable solution for a wide range of medical and research applications under typical laboratory constraints.

In addition to the findings, the same experiment was performed in an A6000 50 Gb GPU with a single-instance model in order to test the performance. The analysis showed that even using the full capacity of a big card, response times do not vary greatly, reinforcing that the versatile mode container is efficient for consumer-grade GPUs.

3.5. Compatibility

The compatibility and setup efficiency of the system were evaluated across various environments, GPU configurations, driver versions, and CUDA versions, as detailed in Table 3. The primary factor influencing setup time was the pre-installation status of the NVIDIA Container Toolkit and Docker rather than the specific GPU driver or CUDA versions used.

Table 3. System setup and compatibility across different environments.

Environment	GPU Configuration	GPU Driver	CUDA Version	Process Time (s)	Process Result	Additional Requirements
Windows 11	x1 RTX 3060 12 GB	561.09	12.6	37	Successful	No changes needed
Ubuntu 18.04	x4 A6000 50 GB	535.54.03	12.2	180	Successful	Container Toolkit + Docker
Ubuntu 24.04	x1 RTX 3050 8 GB	535.183.01	12.2	32	Successful	No changes needed
Ubuntu 24.04	x1 RTX 3050 8 GB	550.127.05	12.4	112	Successful	Container Toolkit

3.5.1. Setup Time Analysis

Setup times varied based on whether the NVIDIA Container Toolkit and Docker had already been installed. The fastest setup was achieved on Windows 11, completing the installation in approximately 37 s when containerization tools were pre-installed. In contrast, the longest setup time of 3 min occurred on Ubuntu 18.04 when the NVIDIA Container Toolkit and Docker needed to be installed during the deployment process. This demonstrates that even without pre-installed containerization tools, the full installation was accomplished swiftly, highlighting the system's speed and ease of use.

3.5.2. Compatibility and Issues

Across all tested environments and configurations, no compatibility issues were encountered. Deployments on Windows 11 and Ubuntu 24.04 with pre-installed containerization tools required no additional modifications, ensuring smooth and efficient setups. The Ubuntu 18.04 setup, while necessitating the installation of the Container Toolkit and Docker, was successfully completed without any compatibility concerns. The presence or absence of the NVIDIA Container Toolkit and Docker was the sole determinant of the setup time variations observed, with no adverse effects on system compatibility. The specific versions of GPU drivers and CUDA did not materially affect the setup times, underscoring that the primary factor was the presence of necessary containerization tools.

3.5.3. Overall Compatibility

The system demonstrated robust compatibility across all tested environments, GPU configurations, driver versions, and CUDA versions. The absence of setup issues and the ability to deploy efficiently in diverse setups, contingent on the installation status of containerization tools, underscore the system's versatility and reliability in real-world scenarios. Ensuring that the NVIDIA Container Toolkit and Docker are pre-installed can optimize setup efficiency, facilitating quicker and more seamless deployments.

3.6. Seamless Deployment

The results demonstrate JAVIS' ability to seamlessly adapt to varying operational requirements through the three recommended modality combinations. Each of them is allowed to upgrade from their current stage in a prompt manner by running a few commands in order to change modality, as shown in Appendix A (Figure A1).

- **JAVIS Tailoring (Development JAVIS + Versatile Mode):** In our case, the initial tailoring phase was performed on a Windows 10 workstation located in an external network environment. This setup provided convenient access to the latest libraries and enabled rapid experimentation with new large language models (LLMs). By coupling JAVIS development with the versatile mode, updates and tests were easily implemented, ensuring that LLM evaluation and refinements could be conducted with minimal effort.
- **Personal/Team Serving (Development/Production JAVIS + Versatile Mode):** Once tailored, JAVIS was shipped to a small group of users running Ubuntu 18.04. Here, the production variant was combined with the ersatile mode to allow straightforward

manual adjustments, such as the adjustment of GPU allocation and switching of LLMs as needed. Despite multiple users accessing the system, the resource overhead remained stable. Even a single-instance model using the versatile mode proved sufficient to support a modest yet efficient team workload.

- **Hospital/Institutional Serving (Production JAVIS + Scaling Mode):** To meet broader institutional demands, the same Ubuntu 18.04 configuration was used to allow connection to additional departments. JAVIS remained in production mode, and by leveraging the container in scaling mode, it was possible to autoscale system resources dynamically. Granting access to more departments or accommodating higher usage no longer required complex reconfiguration. Instead, a few simple adjustments, such as changing containers ensured seamless expansion to meet the institution's growing needs without incurring any noticeable performance overhead.

In summary, these results confirm the effectiveness of JAVIS' modular and flexible deployment approach. With minimal changes, the system is scaled smoothly from an individual workstation testing environment to serving multiple team members to institution-wide access, demonstrating its robust adaptability and efficiency in real-world scenarios.

3.7. Monitoring

The JAVIS chat analytics dashboard successfully tracks user activity and system performance over time. Figure 5 illustrates the aggregated chat usage metrics for JAVIS on a monthly basis. The blue bar plot highlights the total number of messages sent per day, showcasing peaks of high activity, particularly around 18 July, with over 90 messages. In contrast, subsequent days demonstrate reduced usage, punctuated by smaller activity spikes on 25 July, 1 August, and 7 August.

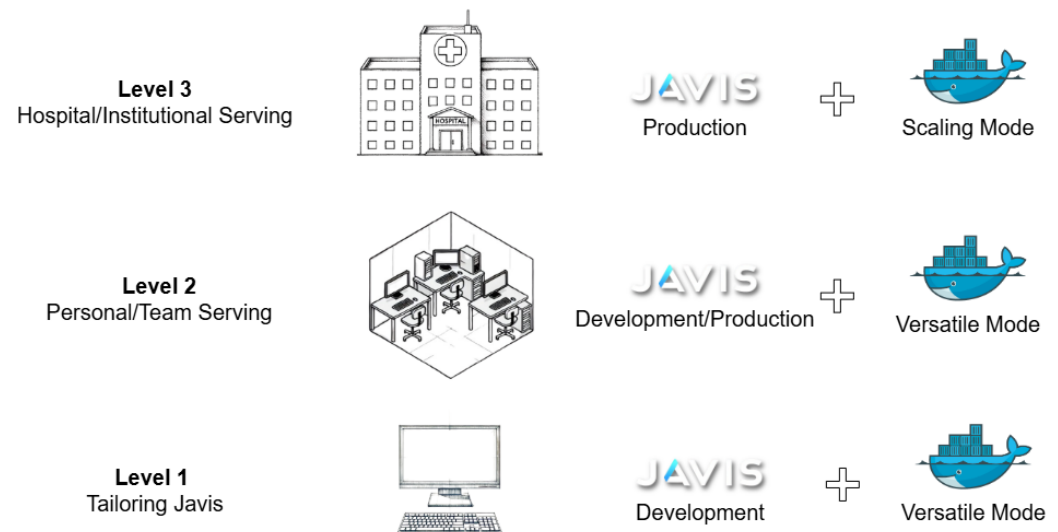


Figure 5. The figure illustrates JAVIS' modular deployment capabilities across three levels of operational requirements. At **Level 1 (Tailoring JAVIS)**, development JAVIS is combined with the versatile mode to enable initial development and experimentation on individual workstations. **Level 2 (Personal/Team Serving)** demonstrates the combination of development/production JAVIS with the versatile mode, supporting small teams with flexible resource adjustments and efficient manual tuning. Finally, **Level 3 (Hospital/Institutional Serving)** combines production JAVIS with the scaling Mode, allowing dynamic resource scaling to meet institutional demands seamlessly. These three levels highlight JAVIS' ability to transition smoothly from individual development environments to team collaboration and large-scale institutional deployment while maintaining adaptability and performance efficiency.

In contrast, the line plot provides insights into each user's trends, allowing them to hide or show any specific user for better visibility, thus revealing consistent monitoring or activity during the same intervals. Weekly, daily, and hourly reports are also available.

Additionally, the Ray performance dashboard ensures real-time performance monitoring, such as resource utilization, replica health, and model-serving loads, which aid in maintaining optimal system performance.

These results demonstrate the system's ability to capture user interaction patterns while enabling proactive monitoring of infrastructure health and resource allocation.

3.8. Real-World Applications and the Impact of JAVIS in Hospital Workflows

JAVIS has played a critical role in enhancing physician efficiency, facilitating large-scale clinical data labeling, and enabling AI-assisted research within secure hospital environments. The following subsections outline key applications where JAVIS has been effectively utilized.

3.8.1. Improving Clinical Documentation and Decision Support

Efficient documentation and structuring of patient medical records are essential in clinical workflows, particularly in emergency room (ER) settings. JAVIS was deployed in ER discharge processes, where it facilitated the summarization of patient medical records, reducing the time required for manual documentation. Physicians reported that these AI-generated discharge summaries improved workflow efficiency, allowing them to focus more on patient care rather than administrative tasks.

In addition, due to hospital networks operating in isolated environments without Internet access, physicians reported struggling with medical knowledge retrieval, such as web searching for specific disease symptoms. JAVIS has been used to retrieve medical knowledge related to specific diseases and symptoms. While this was not used for diagnostic purposes, physicians reported that it was valuable for refreshing previously learned information, such as disease symptoms or general medical knowledge, i.e., using it as an internal network browser for non-critical situations.

Furthermore, the hospital's electronic medical records (EMR) development team leveraged JAVIS-hosted LLaMA 3.1 to reformat and structure unstructured EMR data, significantly improving the readability and accessibility of patient records. This enabled more efficient clinical decision-making, as physicians could more easily extract and interpret relevant patient information.

These applications demonstrate JAVIS's capacity to enhance documentation workflows, provide structured medical knowledge retrieval, and improve EMR usability, addressing key operational inefficiencies in clinical settings.

3.8.2. Scalable Clinical Data Labeling for AI Research

Developing AI-driven clinical models requires large-scale annotated datasets; however, the deployment of fine-tuned models for disease classification, symptom extraction, and test result identification has historically been constrained by hospital data privacy regulations and the absence of on-premises infrastructure for LLM hosting. JAVIS addressed this limitation by enabling the deployment of fine-tuned LLMs within the hospital network, facilitating the annotation of over 10,000 medical records for structured information extraction.

This capability has been instrumental in supporting AI research for clinical information retrieval and natural language processing (NLP) applications, as prior attempts to host large-scale models within hospital networks had been infeasible due to infrastructure constraints. JAVIS ensures that researchers can train, fine-tune, and deploy AI models in a privacy-compliant manner, allowing hospitals to develop AI-driven tools without reliance on external cloud services.

This use case highlights JAVIS's role in bridging the gap between AI development and hospital-based implementation, providing a secure, scalable solution for clinical data processing and annotation.

3.8.3. AI-Assisted Research and Software Development in Restricted Environments

A significant challenge in hospital-based research environments is the lack of Internet access within internal networks, which prevents researchers from utilizing external AI-powered tools such as StackOverflow, ChatGPT, or cloud-based LLMs. Before the deployment of JAVIS, researchers had to manually retrieve AI-generated code snippets from external computers and transfer them via text files, significantly slowing down the development process.

With the integration of JAVIS, researchers now have direct access to AI-powered coding assistance within the hospital network, enabling them to generate, refine, and debug code in real-time without reliance on external services. This has accelerated research productivity and streamlined software development workflows. Furthermore, JAVIS has been utilized in its own iterative improvement process, where LLM-generated code snippets have been used to facilitate non-critical system updates, demonstrating its practical utility as an AI-assisted development environment within secure hospital networks.

Additionally, the ability to host multiple models concurrently has enabled research teams to conduct comparative studies on AI models using sensitive hospital data, an approach that was previously impossible due to cloud-based restrictions. By providing a scalable, multi-model deployment infrastructure, JAVIS has facilitated research initiatives requiring strict data privacy compliance, allowing hospitals to conduct LLM benchmarking and validation studies without compromising patient confidentiality.

4. Discussion

The healthcare literature extensively explores the potential of large language models (LLMs) for improving patient care [19], diagnostics [20], and clinical workflows [21], with much of the focus on research and theoretical applications and proposed integrations into healthcare settings. However, practical deployment at scale within hospitals remains underexplored. Although there are numerous studies on LLM deployment, these are primarily non-healthcare-related and focus on purely technical aspects [22,23], such as serving infrastructure and containerization [24]. These papers generally concentrate on open-source software for LLM chat interfaces and do not address the specific challenges faced by healthcare environments, such as data privacy, integration with legacy systems, and the need for high reliability.

In contrast, the development and deployment of JAVIS address critical gaps in hospital-based AI frameworks, demonstrating its potential as a transformative tool for healthcare environments. While many existing frameworks focus on scalability or specific functionalities, JAVIS uniquely integrates scalability, privacy compliance, and customization into a holistic system tailored for hospitals. Its ability to operate entirely within closed internal networks ensures compliance with stringent privacy regulations while delivering robust AI solutions.

4.1. Addressing Existing Framework Limitations

Most open-source tools available for LLM deployment are designed for online environments, rendering them incompatible with hospital settings where offline operations are mandatory due to privacy concerns. Additionally, current hospital-deployable systems often address privacy but lack comprehensive capabilities such as conversation management, analytics, and seamless deployment across varying scales. Frameworks like RayLLM, while supporting scaling, have limited functionalities, are outdated, and fail to integrate critical elements like user-friendly interfaces, database connectivity, and feedback loops. JAVIS overcomes these limitations with its modular and hospital-centric architecture, offering a complete solution that is scalable, compliant, and customizable.

Comparative Analysis with Existing Frameworks

As part of our evaluation of JAVIS within hospital environments, we compared its core features against alternative LLM deployment solutions (see Table A3). These solutions include RayLLM, OpenLLM, and H2O LLM Studio, each of which exhibits certain limitations when considered for clinical use. In particular, OpenLLM and H2O LLM Studio rely heavily on external cloud setups, limiting their feasibility in strictly offline or privacy-sensitive settings. RayLLM partially addresses these concerns by enabling internal network deployment but lacks modern container maintenance and user interface support. In contrast, JAVIS not only maintains complete on-premises control over data and workloads but also integrates multi-LLM/VLM support and real-time monitoring, ensuring both scalability and compliance with regulations akin to HIPAA. This holistic approach, summarized in Table A3, positions JAVIS as the superior choice for healthcare institutions that require robust, private, and customizable AI solutions.

4.2. Scalability and Hardware Constraints

In addition to technical considerations, practical cost implications are a key concern for hospitals deploying large language models (LLMs) and vision-language models (VLMs). Although JAVIS itself remains agnostic regarding specific GPUs or model architectures, the computational requirements of underlying LLMs can be substantial. For instance, running a 7–8B model often requires approximately 6–10 GB of GPU memory under 4-bit quantization, an optimization strategy that helps reduce resource usage. As the number of active users grows, so does the demand for additional GPU replicas to maintain acceptable latency, thus influencing both the hardware outlay and operational expenditures (e.g., power and cooling).

Scalability tests demonstrated that JAVIS performs reliably even on older hardware configurations, such as a four-A6000 GPU cluster. In peak-performance trials, it supported up to 50 simultaneous users for 8B models, 100 users for 7B VLMs, and 150 users for 1B LLMs with acceptable latency. This highlights the system's robustness yet also underscores a broader limitation: larger user bases require proportionally more GPU resources. Notably, in practical usage scenarios, real concurrency levels tend to be lower. For instance, in our pilot environment for a 40-person lab, we utilized only half of a single A6000 card (effectively 24 GB of GPU memory) and still maintained responsive performance, as not all users simultaneously query the system at peak load.

Despite these demands, JAVIS's modular design and orchestration via Ray allow hospitals to scale resources incrementally. This approach mitigates the capital expenditure required for immediate large-scale deployment, enabling a more gradual expansion as adoption increases. Additionally, cost-saving strategies can be employed by leveraging smaller or specialized models (e.g., 3B parameters) for routine inquiries, substantially reducing memory and power consumption. We observed that a single 7–8B model instance

can handle multiple concurrent users; in real-world conditions, user queries are typically staggered, further lowering the need for constant peak performance.

Looking ahead, ongoing innovations in model distillation and quantization will continue to reduce the computational overhead associated with LLMs and VLMs. By maintaining a container-based, software-centric architecture, JAVIS can seamlessly integrate these more efficient models as they emerge. Consequently, hospitals can adopt a cost-effective, future-proof strategy that aligns with both existing resource constraints and evolving AI technologies.

4.3. Customization and Modularity

Customization within JAVIS is straightforward, facilitated by its stack of Vue.js for front-end updates and Django for back-end management. The integrated feedback loop enables departments to request specific functionalities, which can often be implemented swiftly. For example, front-end and back-end changes for multimodality support required approximately one week of development. Such flexibility illustrates JAVIS's capability to adapt to diverse departmental needs efficiently, although larger functionalities may necessitate additional development time. Once deployed, the containerization ensures seamless transitions from development to production environments, minimizing potential disruptions.

4.4. Privacy and Security

JAVIS adheres to strict privacy standards by operating entirely within a secure hospital network, ensuring compliance with laws such as Korea's Data 3 Laws and similar regulations globally. Its reliance on existing hospital security measures, including encryption and network isolation, guarantees the safeguarding of sensitive data. This approach has proven effective in addressing privacy concerns without introducing vulnerabilities, making JAVIS a reliable solution for data-sensitive environments.

4.5. Future Directions

JAVIS's architecture is designed to remain future-proof, with minimal dependencies in its containers to prevent compatibility issues. Its current integration of vLLM and Ray orchestration ensures ongoing compatibility with emerging inference engines and scalable AI technologies. Building on these foundations, several initiatives and research directions are underway to further enhance JAVIS's role in modern healthcare settings.

4.5.1. Multimodal and Specialized Models

One major focus involves training multimodal models capable of interpreting medical images such as X-rays or MRIs and integrating these findings with textual data for more comprehensive clinical decision support. By coordinating multiple specialized models, e.g., a retriever for information extraction, a summarizer for condensing relevant data, and a rule-based component for validation, JAVIS can deliver robust and safe results. Early efforts demonstrate promise; for instance, over 10,000 medical records have already been labeled using the JAVIS API, showcasing the framework's potential for large-scale text labeling and retrieval.

4.5.2. Advanced Orchestration and Agentic Pipelines

Moving forward, JAVIS aims to coordinate agentic pipelines that handle tasks in sequence: summarization, prediction, data extraction, and structured formatting. By connecting specialized LLMs and VLMs for distinct sub-tasks, JAVIS will support complex workflows tailored to various clinical or research scenarios. Future expansions of these pipelines include combining LLM-based extraction of key medical terms with rule-based

systems to mitigate the risk of hallucinations, ensuring higher accuracy and reliability in hospital environments.

4.5.3. Continuous Adaptation and Resource Optimization

As GPU technologies evolve and LLMs become more energy-efficient, JAVIS will adapt accordingly, offering a flexible and resource-conscious platform for hospital-scale AI deployments. The framework's modular design allows hospitals to incrementally integrate newer or smaller models without restructuring their entire AI infrastructure. This adaptability is particularly beneficial for safeguarding long-term investments in hardware and staff training.

4.5.4. Speech-to-Text and Interactive Innovations

Future enhancements also include speech-to-text (STT) capabilities, enabling practitioners to record and process doctor–patient interactions more interactively. Coupled with chain-of-thought features, STT integration will support the automatic generation of structured summaries or patient notes, further optimizing clinical workflows.

4.5.5. Envisioning JAVIS as a Hospital AI Powerhouse

Ultimately, we envision JAVIS as the central AI hub in modern hospitals, facilitating the easy installation and adoption of future AI systems. By supporting new research on specialized LLMs and VLMs, JAVIS will continue to deliver advanced reasoning, robust data management, and intuitive interfaces. For instance, the ongoing development of English–Korean retrieval systems tailored for medical use, where symptoms, diagnoses, and test results can be accurately extracted, illustrates the platform's potential to power next-generation retrieval-augmented generation (RAG) architectures. As these projects evolve, JAVIS's user interface and container-based deployment model will remain easily configurable, ensuring that feedback loops and continuous improvements keep pace with hospital demands.

By leveraging multimodal capabilities, scalable orchestration, and a forward-compatible design, JAVIS positions itself as a flexible and indispensable tool for healthcare institutions seeking to harness the transformative potential of AI in a secure, private, and continually evolving environment.

4.6. Real-World Insights and Challenges

Deploying JAVIS in a hospital environment revealed significant insights. The strict approval processes required for enabling IP access initially delayed adoption but underscored the necessity of LLMs in strictly offline settings, where internet access is unavailable or highly restricted. In such environments, LLMs became indispensable resources for addressing coding issues, solving EMR-related tasks, and providing quick access to critical knowledge. Physicians, in particular, benefited from the system's capacity to assist with clinical queries and memory recall, thereby enhancing productivity and informed decision-making across multiple departments.

Despite the clear value offered by offline LLMs, user adaptation and staff training also emerged as key considerations during JAVIS's pilot deployments. Early prototypes presented usability challenges, including a chat interface that did not auto-scroll to new messages, a non-expandable input field, and a lack of clarity around API integration. These design limitations initially hindered seamless integration into day-to-day workflows and required additional support from IT teams. However, real-time feedback mechanisms led to continuous improvements: modern interfaces, intuitive drag-and-drop file upload, and clearer documentation for API usage. Consequently, new users now report minimal onboarding time, thanks to an interface that closely resembles common chat platforms.

Formal training programs or lengthy onboarding manuals have proved largely unnecessary, reducing both the time and costs typically associated with user support. Most inquiries today focus on file format compatibility and specialized features rather than navigating the system itself. Nonetheless, IT support and concise documentation remain essential for advanced functionalities, such as custom integrations or specialized data parsing, which some researchers and clinicians require. By maintaining a user-centric design philosophy and accessible support resources, JAVIS ensures long-term adoption and sustained engagement across diverse hospital departments.

4.7. Conclusions

JAVIS represents a paradigm shift in hospital-based AI deployment by bridging the gap between scalability, privacy, and customization. Its ability to operate offline while delivering state-of-the-art AI functionalities positions it as a critical tool for modern healthcare institutions. Although challenges such as hardware limitations and time-intensive feature additions remain, JAVIS's robust architecture and future-oriented design ensure its adaptability and sustainability as AI technologies continue to advance.

5. Conclusions

Summary

This paper introduces JAVIS, a robust and scalable deployment framework specifically designed to address the unique challenges of integrating large language models (LLMs) and vision-language models (VLMs) into healthcare environments. By ensuring complete privacy compliance and delivering reliable customization and scalability capabilities, JAVIS bridges significant gaps in existing solutions. Its modular architecture and seamless deployment process enable it to adapt to varying operational scales, from individual workstations to hospital-wide systems, while maintaining high performance and reliability.

JAVIS has demonstrated its utility in enhancing hospital workflows through real-time user feedback and enabling analytics-driven optimization. With a future-oriented design, it is poised to integrate emerging AI advancements, including speech-to-text capabilities, ensuring its continued relevance and impact. By providing an open-source, hospital-centric solution, JAVIS sets a new benchmark for deploying advanced AI in the healthcare sector, ultimately contributing to improved patient outcomes and operational efficiency.

Author Contributions: Conceptualization, J.A. and W.C.C.; methodology, J.A. and W.C.C.; software, J.A.; validation, J.A. and W.C.C.; formal analysis, J.A.; investigation, J.A.; resources, W.C.C.; writing—original draft preparation, J.A.; writing—review and editing, J.A.; visualization, J.A.; supervision, W.C.C.; project administration, W.C.C.; funding acquisition, W.C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number : HI22C0452).

Data Availability Statement: The data used in this research was included in the Appendix A in Tables A2 and A3.

Acknowledgments: This study was possible due to the grant of Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI).

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

Abbreviations

JAVIS	Joint AI Versatile Implementation System
LLM	Large Language Model
VLM	Vision-Language Model
GPTQ	Generalized Post-Training Quantization
AWQ	Activation-aware Weight Quantization
HIPAA	Health Insurance Portability and Accountability Act
CDW	Common Data Warehouse
GPU	Graphics Processing Unit
PC	Personal Computer
AI	Artificial Intelligence
API	Application Programming Interface

Appendix A

Table A1. Performance comparison: single-instance vs. replica models.

Model Name	Type	75 Users	100 Users	125 Users	150 Users
Single-Instance Models					
qwen2-2b-1-replica	VLM	35.6471 ± 21.5337	48.3445 ± 27.6978	65.3267 ± 35.5868	89.1086 ± 48.6269
qwen2-7b-1-replica	VLM	57.4285 ± 30.0506	72.2567 ± 42.4952	82.1418 ± 46.4424	109.6259 ± 59.9394
llava-13b-1-replica	VLM	52.1012 ± 29.8732	68.8172 ± 38.2909	90.2259 ± 50.8535	106.5157 ± 60.0830
llama-1b-1-replica	LLM	62.6997 ± 35.7966	67.5163 ± 42.8536	97.8888 ± 55.8919	121.9497 ± 73.0799
llama-3b-1-replica	LLM	86.7498 ± 50.5465	102.4427 ± 55.4608	130.1832 ± 73.8914	153.1903 ± 85.6271
llama-8b-1-replica	LLM	119.6549 ± 66.4862	146.5137 ± 82.7816	188.2520 ± 115.2399	237.6892 ± 131.8673
Multiple-Instance Models (Ray)					
qwen2-2b-20-replica	VLM	9.5376 ± 4.9529	12.7513 ± 6.4241	15.4664 ± 6.9279	19.9774 ± 10.2574
qwen2-7b-12-replica	VLM	14.2411 ± 6.8963	17.4584 ± 10.2561	18.2418 ± 9.7270	22.9139 ± 12.3596
llava-13b-12-replica	VLM	12.9215 ± 5.9919	16.4164 ± 8.3392	23.0050 ± 12.2063	26.2898 ± 13.1351
llama-1b-20-replica	LLM	12.3633 ± 7.5692	19.8064 ± 10.1389	20.3427 ± 10.7773	26.2847 ± 13.4261
llama-3b-20-replica	LLM	22.6402 ± 10.9646	27.8081 ± 14.8628	36.1693 ± 16.2197	42.9934 ± 22.5223
llama-8b-12-replica	LLM	31.2869 ± 18.0103	35.6251 ± 19.3717	47.2479 ± 26.6813	55.7640 ± 31.7094

Table A2. Prompts used for evaluating consumer-grade GPUs.

ID	Prompt
1	What are the symptoms of diabetes?
2	Explain the difference between type 1 and type 2 diabetes.
3	What are the side effects of ibuprofen?
4	How does a vaccine work to protect against viruses?
5	A 45-year-old male patient presents with persistent fatigue, unexplained weight loss over the past three months, and occasional night sweats. He reports having a low-grade fever for the past two weeks. Upon physical examination, palpable lymph nodes were found in the cervical and axillary regions. Blood tests revealed mild anemia and an elevated ESR. What are the possible differential diagnoses, and what further investigations should be performed?
6	Explain the anatomy of the human heart.
7	A 35-year-old female presents to the emergency department with severe, sharp right lower abdominal pain that started suddenly 12 h ago. She also reports nausea and vomiting but denies diarrhea. She has no history of chronic conditions, but her last menstrual period was two weeks ago. Physical examination reveals tenderness in the right lower quadrant and positive rebound tenderness. Vital signs show a mild fever of 38°C. Based on this presentation, what is the most likely diagnosis, and what diagnostic tests should be ordered to confirm it?
8	Describe the process of organ transplantation.
9	A 60-year-old male with a history of poorly controlled type 2 diabetes presents with a non-healing ulcer on the sole of his foot. The ulcer has been present for three weeks and has started emitting a foul-smelling discharge. The patient also reports intermittent fever and swelling around the affected area. Physical examination reveals erythema and warmth around the ulcer, along with signs of cellulitis. What is the likely diagnosis, and what steps should be taken for immediate management and long-term care?
10	How does chemotherapy treat cancer?
11	List the primary functions of the liver.
12	A 22-year-old college student reports to the clinic with a 5-day history of fever, severe headache, and a stiff neck. The patient also complains of sensitivity to light and has vomited twice. No recent travel or history of infectious disease is noted, but he mentions being in close contact with a roommate who had flu-like symptoms a week ago. Physical examination reveals positive Kernig and Brudzinski signs. What is the suspected condition, and how should this case be managed urgently?
13	What is the difference between an MRI and a CT scan?
14	How is asthma diagnosed and treated?
15	Discuss the causes and treatments for chronic pain.
16	A 50-year-old woman comes in with complaints of chest tightness and shortness of breath that worsens on exertion and improves with rest. She has been experiencing this for the past three months. She has a history of hypertension and takes medication irregularly. Physical examination reveals a systolic murmur heard best at the apex and radiating to the axilla. An ECG shows left ventricular hypertrophy. What is the likely diagnosis, and what diagnostic tests and treatment options should be considered?
17	Describe the stages of wound healing.
18	What is the role of insulin in the body?
19	How do antibiotics work to fight infections?
20	What are the common causes of anemia?

Table A3. Images and their corresponding prompts used for testing reponse times in consumer GPUs.

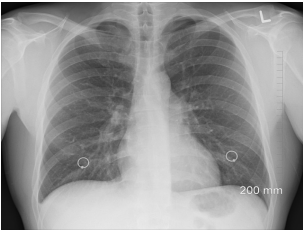
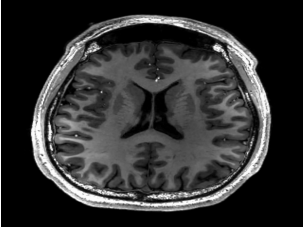





Image	Prompts
	<div><div>1.</div><div>Identify any abnormalities in this chest X-ray, such as pneumonia, tuberculosis, or lung cancer.</div><div>2.</div><div>Diagnose and describe the next steps for a patient with the following X-ray.</div></div>
	<div><div>1.</div><div>Locate and classify any brain tumors in this MRI scan.</div><div>2.</div><div>Outline the regions of the tumor in this MRI scan.</div></div>
	<div><div>1.</div><div>Determine the stage of the kidney stone visible in this CT scan.</div><div>2.</div><div>What do you see in the CT Scan?</div></div>
	<div><div>1.</div><div>Differentiate between melanoma, eczema, and psoriasis in this image.</div><div>2.</div><div>Does this skin condition suggest?</div></div>
	<div><div>1.</div><div>Detect signs of diabetic retinopathy or macular degeneration in this retinal image.</div><div>2.</div><div>Classify abnormalities in this retinal image, such as retinal detachment or vascular occlusion.</div></div>
	<div><div>1.</div><div>Locate cavities, impacted teeth, or signs of periodontal disease in this dental X-ray.</div><div>2.</div><div>Analyze this dental X-ray for signs of tooth decay or bone loss.</div></div>
	<div><div>1.</div><div>Analyze this low-quality ultrasound image and provide a diagnosis.</div><div>2.</div><div>Identify abnormalities in this fetal ultrasound image.</div></div>

Table A3. Cont.

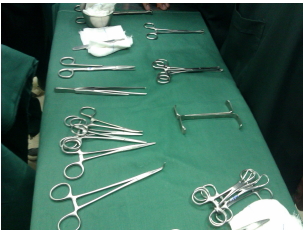


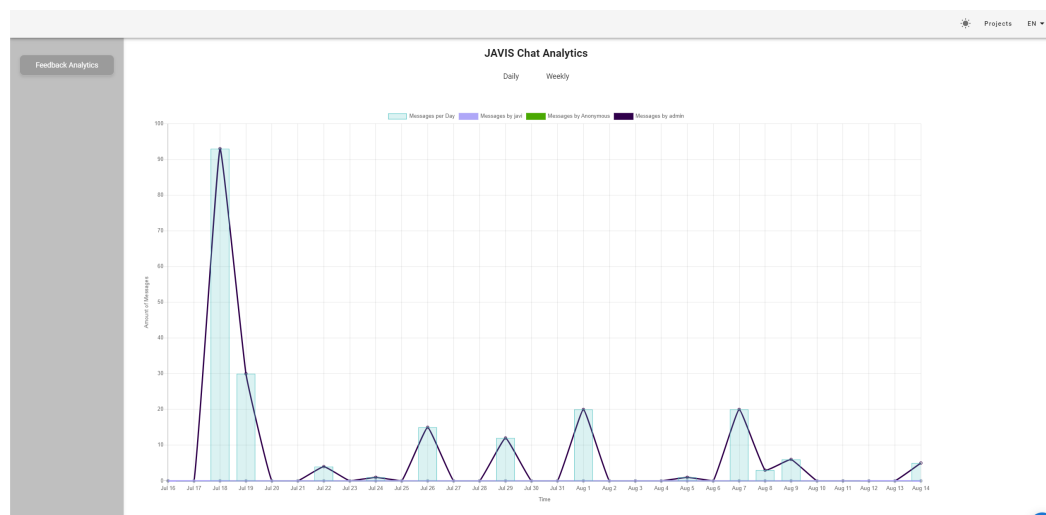
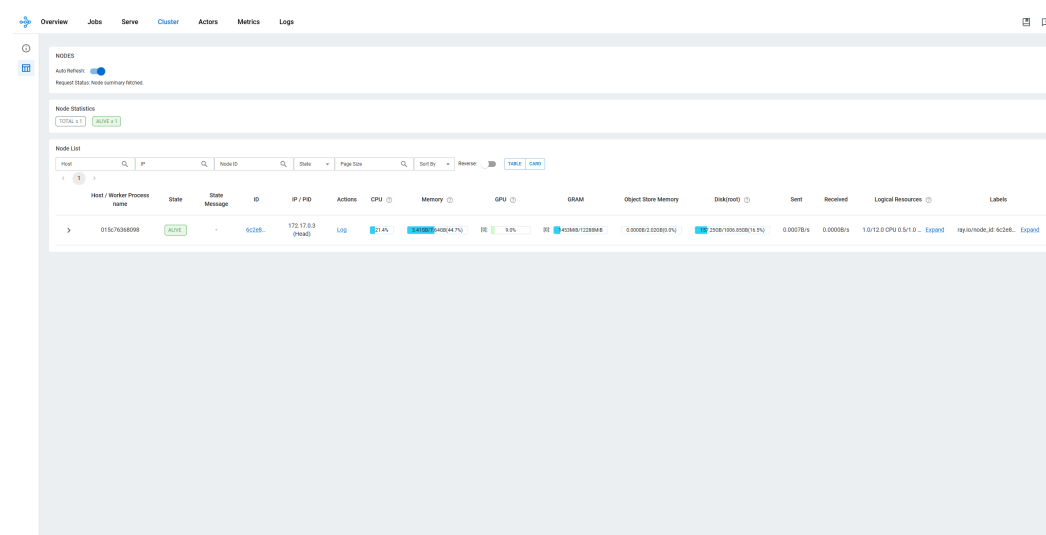
Image	Prompts
	<ol style="list-style-type: none"> 1. How many pairs of scissors are there in the image? 2. Classify the surgical instruments visible in this image and their use.
	<ol style="list-style-type: none"> 1. Identify fractures and classify their types in this trauma X-ray. 2. In which part of the body has the trauma happened? Please go into detail.
	<ol style="list-style-type: none"> 1. Classify this X-ray of a dog for signs of hip dysplasia. 2. Identify fractures or abnormalities in this veterinary X-ray.

Table A4. Comparative analysis of JAVIS vs. alternative LLM deployment solutions. JAVIS is the only solution that successfully integrates on-premises, privacy-compliant AI deployment with multi-LLM/VLM support, real-time monitoring, and scalable orchestration via Ray Serve and VLM. Unlike OpenLLM and H2O LLM Studio, which rely on external cloud setups and lack hospital-wide scalability, JAVIS ensures full data privacy while providing a flexible and customizable architecture tailored to healthcare needs. Compared to RayLLM, JAVIS not only maintains the reliability of Ray Serve and VLM but also modernizes container maintenance for long-term support while introducing a user-friendly UI for streamlined hospital integration. JAVIS is the superior choice for deploying AI within healthcare institutions, overcoming the privacy, scalability, and maintenance barriers faced by competing solutions.

Feature	JAVIS (Fully Compliant)	RayLLM	OpenLLM	H2O LLM Studio
Target Audience	Healthcare-focused	Internal network deployment	General-purpose developers	Developers/Researchers
Privacy and Compliance	✓ (Critical Feature) Closed-network, HIPAA-like security	✓ (Critical Feature) Closed-network, HIPAA-like security	✗ (Critical Limitation) Requires external cloud setups	✗ (Critical Limitation) Requires external cloud setups
Long-Term Support	✓ (Critical Feature) Actively Supported	✗ (Critical Limitation) Archived (May 2024)	✓ (Critical Feature) Actively Supported	✓ (Critical Feature) Actively Supported
Deployment Flexibility	✓ Scales from PCs to hospitals (Entire Software Stack)	✓ Scales from PCs to hospitals (Just Inference Engine)	✗ Only Available as Cloud Server Endpoint	✗ Only Available as Cloud Server Endpoint
Scaling	✓ Multi-replica/Multi-model	✓ Multi-replica/Multi-model	✓ Online Only Multi-replica/Multi-model	✗ Lacks Scaling Support
LLM/VLM Support	✓ Multi-LLM/VLM	✓ Multi-LLM/VLM	✗ Open-source LLMs only	✗ Limited to single LLM deploy
Monitoring	✓ Advanced analytics + Ray dashboard	≈ Just replica and status monitoring	≈ Just replica and status monitoring	≈ Just static experiment metrics
Tailoring and Customization	✓ Fully customizable for hospital tailoring	✗ No UI (just inference engine)	✗ Limited to pre-defined GUIs	✗ Limited to pre-defined GUIs
UI Availability	✓ Fully customizable UI	✗ No UI (CLI-based only)	✓ Basic Chat UI	✓ Basic Chat UI (pre-defined)



(a)



(b)

Figure A1. JAVIS monitoring tools. The upper image displays the JAVIS analytics regarding user usage. The lower image represents a Ray dashboard representing the cluster and replica status of each LLM and VLM. (a) JAVIS chat analytics of usage per user monthly. The blue bar plot describes the total messages per day accumulated between users. The line plot describes, in this case, the amount of usage by the administrator per day. (b) Ray analytics displaying the current loaded GPU and current usage.

References

1. Nazi, Z.A.; Peng, W. Large Language Models in Healthcare and Medical Domain: A Review. *Informatics* **2024**, *11*, 57. [\[CrossRef\]](#)
2. Eloundou, T.; Manning, S.; Mishkin, P.; Rock, D. GPTs are GPTs: Labor market impact potential of LLMs. *Science* **2024**, *384*, 1306–1308. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Cheong, I.; Xia, K.; Feng, K.K.; Chen, Q.Z.; Zhang, A.X. (A) I Am Not a Lawyer, But...: Engaging Legal Experts towards Responsible LLM Policies for Legal Advice. In Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, Rio de Janeiro Brazil, 3–6 June 2024; pp. 2454–2469.
4. Adeniyi, A.O.; Arowoogun, J.O.; Okolo, C.A.; Chidi, R.; Babawarun, O. Ethical considerations in healthcare IT: A review of data privacy and patient consent issues. *World J. Adv. Res. Rev.* **2024**, *21*, 1660–1668. [\[CrossRef\]](#)
5. Soenksen, L.R.; Ma, Y.; Zeng, C.; Boussioux, L.; Villalobos Carballo, K.; Na, L.; Wiberg, H.M.; Li, M.L.; Fuentes, I.; Bertsimas, D. Integrated multimodal artificial intelligence framework for healthcare applications. *NPJ Digit. Med.* **2022**, *5*, 149. [\[CrossRef\]](#) [\[PubMed\]](#)

6. Cardoso, M.J.; Li, W.; Brown, R.; Ma, N.; Kerfoot, E.; Wang, Y.; Murrey, B.; Myronenko, A.; Zhao, C.; Yang, D.; et al. Monai: An open-source framework for deep learning in healthcare. *arXiv* **2022**, arXiv:2211.02701.
7. Karabacak, M.; Margetis, K. Embracing large language models for medical applications: Opportunities and challenges. *Cureus* **2023**, *15*, e39305. [[CrossRef](#)] [[PubMed](#)]
8. Yang, R.; Tan, T.F.; Lu, W.; Thirunavukarasu, A.J.; Ting, D.S.W.; Liu, N. Large language models in health care: Development, applications, and challenges. *Health Care Sci.* **2023**, *2*, 255–263. [[CrossRef](#)] [[PubMed](#)]
9. bin Uzayr, S.; Cloud, N.; Ambler, T.; bin Uzayr, S.; Cloud, N.; Ambler, T. Vue. js. In *JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now*; Apress: New York, NY, USA, 2019; pp. 523–539.
10. Forcier, J.; Bissex, P.; Chun, W.J. *Python Web Development with Django*; Addison-Wesley Professional: Hoboken, NJ, USA, 2008.
11. DeJonghe, D. *Nginx Cookbook*; O'Reilly Media: Sebastopol, CA, USA, 2020.
12. Owens, M.; Allen, G. *SQLite*; Apress LP New York: New York, NY, USA, 2010.
13. Lubanovic, B. *FastAPI*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2023.
14. Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C.H.; Gonzalez, J.E.; Zhang, H.; Stoica, I. Efficient Memory Management for Large Language Model Serving with PagedAttention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, Koblenz, Germany, 23–26 October 2023.
15. Karau, H.; Lublinsky, B. *Scaling Python with Ray*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
16. Frantar, E.; Ashkboos, S.; Hoefler, T.; Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv* **2022**, arXiv:2210.17323.
17. Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.M.; Wang, W.C.; Xiao, G.; Dang, X.; Gan, C.; Han, S. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proc. Mach. Learn. Syst.* **2024**, *6*, 87–100. [[CrossRef](#)]
18. Miell, I.; Sayers, A. *Docker in Practice*; Simon and Schuster, Manning: Shelter Island, NY, USA, 2019.
19. Busch, F.; Hoffmann, L.; Rueger, C.; van Dijk, E.H.; Kader, R.; Ortiz-Prado, E.; Makowski, M.R.; Saba, L.; Hadamitzky, M.; Kather, J.N.; et al. Systematic Review of Large Language Models for Patient Care: Current Applications and Challenges. *medRxiv* **2024**. [[CrossRef](#)] [[PubMed](#)]
20. Hager, P.; Jungmann, F.; Holland, R.; Bhagat, K.; Hubrecht, I.; Knauer, M.; Vielhauer, J.; Makowski, M.; Braren, R.; Kaissis, G.; et al. Evaluation and mitigation of the limitations of large language models in clinical decision-making. *Nat. Med.* **2024**, *30*, 2613–2622. [[CrossRef](#)] [[PubMed](#)]
21. Rao, A.; Pang, M.; Kim, J.; Kamineni, M.; Lie, W.; Prasad, A.K.; Landman, A.; Dreyer, K.; Succi, M.D. Assessing the utility of ChatGPT throughout the entire clinical workflow: Development and usability study. *J. Med. Internet Res.* **2023**, *25*, e48659. [[CrossRef](#)] [[PubMed](#)]
22. Ye, Z.; Ying, R. An AI-aware Orchestration Framework for Cloud-based LLM Workloads. In Proceedings of the 2024 IEEE 10th International Conference on Edge Computing and Scalable Cloud (EdgeCom), Shanghai, China, 28–30 June 2024; pp. 22–24. [[CrossRef](#)]
23. Colombi, L.; Gilli, A.; Dahdal, S.; Boleac, I.; Tortonesi, M.; Stefanelli, C.; Vignoli, M. A Machine Learning Operations Platform for Streamlined Model Serving in Industry 5.0. In Proceedings of the NOMS 2024—2024 IEEE Network Operations and Management Symposium, Seoul, Republic of Korea, 6–10 May 2024; pp. 1–6. [[CrossRef](#)]
24. Turnbull, J. *The Docker Book: Containerization Is the New Virtualization*; James Turnbull, Lulu: Raleigh, NC, USA, 2014.

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.