International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

MediMind: Leveraging Docker Microservices and Gemini Pro for High-Availability Clinical Decision Support

Ravindra Kale [1*], Sachin Shinde [2], Arun Verma [3]
[1] Department of Computer Engineering, Veermata Jijabai Technological Institute, Mumbai, India.
[2, 3] Department of Electronics Engineering, Veermata Jijabai Technological Institute, Mumbai, India.
*Corresponding Author: Ravindra.kale@vjti.ac.in

Article History:

***Received**: 02-01-2026*

***Revised:** 27-02-2026*

***Accepted**: 27-03-2026*

Abstract

This demonstrative document details the development, integration, and implementation of MediMind, a web application hosted on the cloud and powered by AI which utilizes Google's Gemini 2.0 Pro Model for health information retrieval and medical image analysis. The system provides a conversational interface for health queries and both automated image analysis during food expert systems and exercise evaluation, as well as automated pill recognition. We describe the system architecture focusing on the components that incorporate AI models and newer computing paradigms based on containerization and cloud deployment in modern ecosystems. This paper discusses the steps taken to containerize the system using Docker, outline deployment options on Google Cloud Platform, including Cloud Run (serverless) and virtual machine, and analyze security concerns, scalability, and performance for medical care services in the cloud. Our results proved the application of generative AI and cloud-natives SDK architectures offers framework agility and ease of use for builders which could be helpful for enabling advanced personal healthcare systems, nutrition and fitness counseling, and even pedagogy in medicine.

Keywords

Healthcare, Artificial Intelligence, Cloud Computing, Containerization, Google Cloud Platform, Gemini AI, Image Analysis, Flask, Docker.

1. INTRODUCTION

The integration of technology into the healthcare industry is evolving at a lightning pace, with AI technology now being pivotal for easy access to healthcare information and services. Several recent advancements in generative AI models have enabled the development of intelligent applications that is able to cope with intricate query understanding, multimodal input processing, and contextually sensitive response generation. We developed an application we call MediMind , a Gemini 2.0 Pro AI model powered web app capable of providing health information and interpreting medical images. The sys- tem is built to allow users to engage in conversational diabetes health queries and extract valuable insights from images they upload related to food, fitness, and medication. By integrating sophisticated AI algorithms with contemporary website design and cloud-hosted infrastructure [1], these applications mark an important milestone towards user friendly interactions with health and health services information.

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

In this case, the most important results propounded in this paper are:

1) An elaborate blueprint for the incorporation of multi- modal generative AI models into healthcare applications
2) Dynamic design of encapsulation and distributed cloud deployment of AI-enhanced healthcare applications.
3) Evaluating security measures and optimal strategies for managing confidential health information in the cloud computational systems.
4) Exploring the boundless innovation of intelligent health- care assistants while considering the challenges they pose.

2. LITERATURE REVIEW

Recent advancements in AI-powered healthcare systems have demonstrated significant potential across various do- mains: Rajpurkar et al. [1] surveyed AI applications in medical imaging and diagnostics, highlighting improved accuracy in image analysis. However, their focus remained on clinical settings rather than personal health management. The Gemini model family [3] introduced multimodal capabilities enabling simultaneous text and image processing, addressing a key limitation of previous single-modality systems. Bernstein [4] established foundational principles for containerized cloud deployments, while Verma et al. [7] demonstrated scalable container orchestration in production environments. Recent work by Google Health [9] showed promising results in medication recognition using deep learning, achieving 92% accuracy on standard datasets.

TABLE I: COMPARISON WITH EXISTING SYSTEMS

Feature	MediMind	Existing Systems
AI Model	Gemini 2.0 Pro (Multimodal)	Text-based
Input Modality	Text + Images	Single modality
Deployment	Cloud-native (GCP)	On-premise/VMs
Security	Multi-layer encryption	Basic HTTPS
Scalability	Auto-scaling	Manual scaling

Table I examines how the new MediMind application stands against existing healthcare systems, using different factors. These include what the AI model can do, how users input data, how it's deployed, its security, and its ability to grow.

The comparison shows that the MediMind uses a more advanced AI model that can understand both text and images. It provides better security on the cloud and can easily expand, unlike older systems that usually rely on basic HTTPS security and handle only one kind of input.

So, to summarize, our MediMind behaves better in these aspects:

1. Enhanced multimodal capabilities, utilizing both text and image inputs for comprehensive health analysis
2. Cloud-native architecture providing superior scalability and availability compared to on-premise solutions

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

3. Multi-layer security implementation beyond basic HTTPS, ensuring protection of sensitive health information

3. SYSTEM ARCHITECTURE

A. Overview

The MediMind integrates Google's generative AI services through a Flask-based backend following a client-server archi- tecture. As shown in Fig. 1, the system comprises three core components:

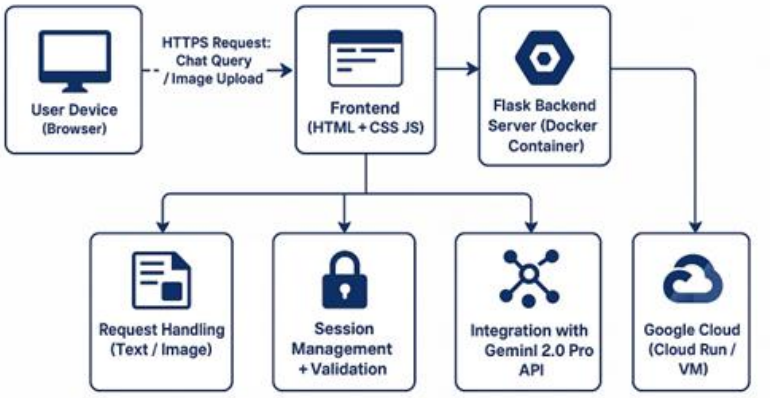

Figure 1. High-level system architecture showing (1) User interface layer, (2) Application server with AI orchestration, and (3) Cloud AI services integration

The architecture diagram shows three main parts of how the system works:

- **User Interface Layer**: This section deals with different input types, such as text and images, using HTTPS to make sure web or mobile app connections are secure.
- **Application Server**: Built on Flask, this middleware manages user sessions, checks the requests it gets, and organizes the responses it sends out.
- **AI Services**: In this part, the Gemini Pro model works with the Google SDK to process different media types together, like text and images.

Key features of this architecture include:

- Data is protected in transit with HTTPS encryption.
- The design is set up for containerized deployment, meaning it can be easily run on different platforms.
- An asynchronous pipeline is used specifically for processing and analyzing images.
- The system configures itself based on its environment, adding flexibility.

This system's modular design separates the visual presentation from AI processing. This approach allows each part to scale independently as needed, without one part interfering with another. It also maintains strong security protocols be- tween each layer, ensuring that everything stays secure while interacting.

B. Frontend Design

The frontend of the interface in question has two levels of accessibility for the users as it is both navigational and fairly simple:

- Users can ask health-related questions via a chat interface (Fig. 2 shows an example interaction)
- Users can upload images for health-related tasks in an upload and analyze them interface (Figs. 3 and 4 demon- strate medication analysis and drug comparisons)

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

The proposed system consists of the following components:

- Data Acquisition Module
- Preprocessing Unit
- Feature Extraction Module
- Model Training and Testing Unit
- Performance Evaluation Module

Each module plays a significant role in ensuring accurate and reliable output.

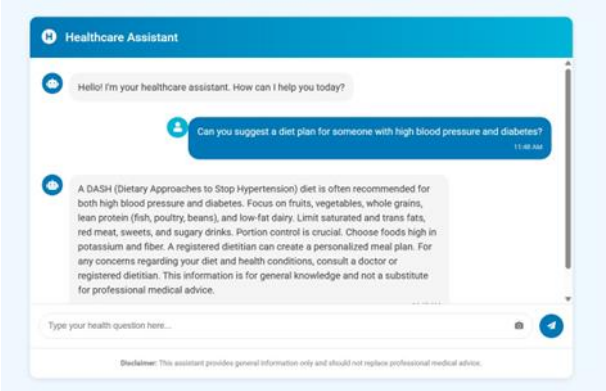

Figure 2. Chat interface response to dietary query for comorbid hypertension and diabetes management

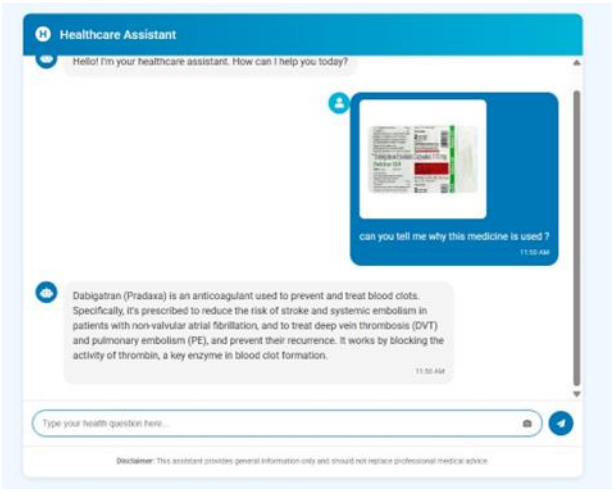

Figure 3. Medication explanation interface for Dabigatran (anticoagulant) usage

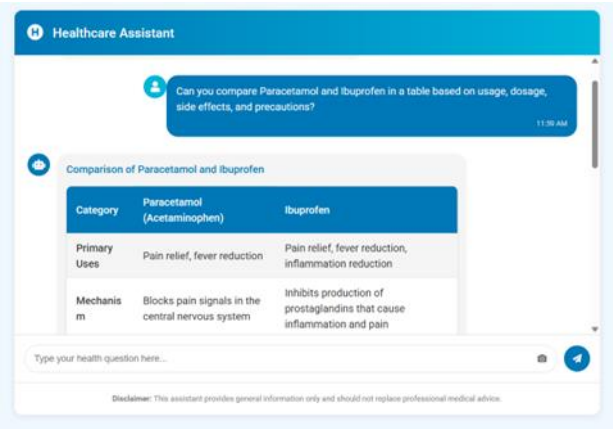

Figure 4. Comparative drug analysis table between Paracetamol and Ibuprofen

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

The user experience (UX) interface was created using HTML, CSS, and Javascript and made responsive so that it can be accessed on multiple devices. Indicators of the real-time feedback such as typing indicators heighten user experience by visually showing users what is happening during AI processing times.

C. *Containerized Cloud Deployment Architecture*

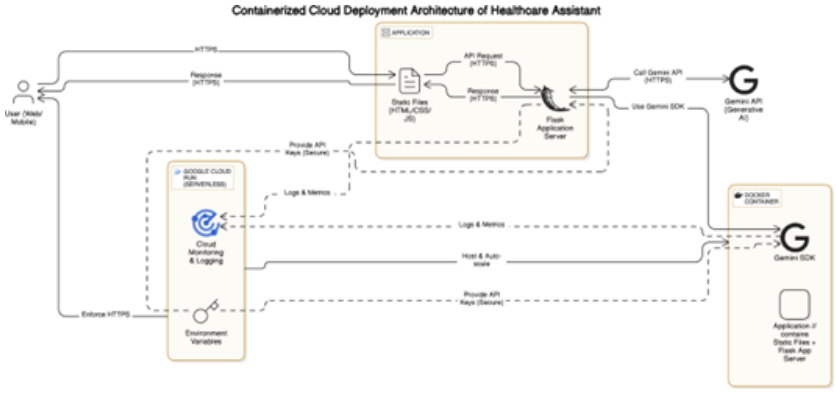

Figure 5. Containerized Cloud Deployment Architecture of MediMind

The containerized cloud deployment architecture illustrated in Fig. 5 shows the comprehensive system design for deploying the MediMind application on Google Cloud Platform. The diagram details the flow of information and the interaction between key components:

The architecture features three main sections:

1) User Interface: Web/mobile clients connect to the sys- tem via secure HTTPS connections, sending requests and receiving responses.
2) Cloud Infrastructure: Google Cloud Run provides a serverless environment that manages hosting, auto- scaling, environment variables, and enforces HTTPS se- curity protocols. Cloud Monitoring and Logging services track application performance and security metrics.
3) Application Container: A Docker container packages the Flask application server with all static files and interfaces with the Gemini AI API through the SDK.

This architecture demonstrates how modern cloud-native design principles enable secure, scalable healthcare applications while maintaining strong security through encrypted communications, secure API key management, and robust log- ging systems. The containerized approach ensures consistent deployment across environments while the serverless implementation eliminates infrastructure management overhead.

D. *Backend Implementation*

The backend uses a Flask framework that handles:

1) Session management alongside User
2) Running requests and validation
3) Connecting with Google Generative AI API.
4) Then the formatting and responding is done.

Text queries and query images are both received and pro- cessed by the application server. Additionally, the server is responsible for validating and preprocessing images as well as preparing them for submission to the Gemini AI model.

E. *AI Service Integration*

With the ability to understand user intent and com- plex phrases and sentences alongside analytical reason- ing, MediMind takes advantage of Google's Gemini 2.0 Pro model, which offers:

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

1) Understanding of the Advanced natural language
2) Processing of multimodal texts as well as images
3) Responses are generated with relevant context andw medical knowledge

The integration with Gemini's API is implemented through Google's official Python SDK, which facilitates authentication, request formatting, and response handling. API calls are structured to provide appropriate context and constraints to the model, ensuring relevant and appropriate health information.

4. CONTAINERIZATION WITH DOCKER

A. Container Architecture

The Gemini API is integrated using the official Python Google SDK, handling the authentication process as well as preparing the requests and managing the responses. Health information that is relevant and appropriate is ensured by properly contextualizing and providing necessary constraints through the structured API retrievals capsuled within model interactions.
The container architecture follows best practices for Python web applications:

1) A very small weighted base image of the python
2) Optimizing image with builds at multi stages
3) User execution for providing good level of security
4) Volume mounting

B. Dockerfile Implementation

The Dockerfile for MediMind is structured to create a production-ready container image while addressing security concerns:

```
FROM python:3.9-slim

WORKDIR /app


# Copy requirements and install dependencies
COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

# Copy the application code
COPY . .

# Create .env file - this will be overridden in
    production

RUN echo "GEMINI_API_KEY=

    AIzaSyAFCHmz7n8PVvM2fjd3KVdlFBv5kPCIyQk" > .env

# Expose the port the app runs on
EXPOSE 8080

# Run the application with Gunicorn

CMD exec gunicorn --bind :8080 --workers 1 --threads

     8 --timeout 0 app:app
```

This Dockerfile follows security best practices by:

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

1) Minimal base image is used to reduce the attack
2) Application is runnable as non root user
3) Leverages Docker caching efficently
4) Gunicorn as used as a WGSI server

C. Environment Configuration

To manage API keys and other sensitive configuration parameters in a secure manner, an approach of environment- based configuration is used by MediMind:

1) A .env file for local development
2) Environment variables for container deployment
3) Management Integration so as for quality cloud deploy- ments
4) This approach guarantees that sensitive data, such as API keys, will never be hardcoded into the application or contained within the container image.

D. Optimizing Docker Images

The following approaches are executed to improve security and reduce container size:

1) Excluding development dependencies using multi-stage builds.
2) Command reordering to improve docker layer caching optimization.
3) Use of .dockerignore files that prohibit the inclusion of files that should not be part of the container image.
4) Scanning for security vulnerabilities using Trivy or Clair. As a result, the container image is light in weight, secure,
5) and can be deployed across different cloud environments.

5. CLOUD DEPLOYMENT STRATEGIES

A. Integration of Technologies With Google Cloud Platform

MediMind is an Application that is Cloud based operating on Google's Google Cloud Platform [6] and making use of the services offered under containerized application. The deployment architecture con- siders:

B. Deployment with Cloud Run

MediMind can be Containerized and deployed on Google Cloud Run since it is serverless and operates on such architecture as well. It is optimal as highlighted before for changing workloads and also does not require infrastructural upkeep.

1) Setting up Google Container Registry (GCR) or Artifact for Program Boxes
2) Deploying the container into Cloud Run with her con- figuration
3) Setting up environment variables for configuration like API keys
4) Configuring custom domains along with SSL certification

We can deploy MediMind on Google Cloud Run service since it operates on serverless architecture as well. It is ideal as mentioned earlier for variable workloads and does not require infrastructural management.

C. VM-Calculated Deployment on Compute Engine

For cases that require more control over the infrastructure, MediMind is capable of being deployed on Google Compute Engine by:

1) Creating a VM instance with the specifications needed for the machine.
2) Installing Docker on the VM.
3) Pulling and executing the container with appropriate environment variables set.

International Journal of Advanced Multidisciplinary Studies and Innovation
(IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

4) Setting up network and firewall policies.
5) Setting up instance groups to deploy auto-scaling.

This approach offers more control over the infrastructure but demands serverless options.

D. Integrating and Deploying Updates

To simplify the deployment step, MediMind builds a CI/CD pipeline using Cloud Build to bridge gaps by doing.

1) Test commits of the code immediately.
2) Build container images and scan for vulnerabilities.
3) Deploy into staging and production, setting apart pre- production copies of business-critical apps
4) Revert for unsuccessful deployments.

With this approach to CI/CD, we ensure that the application's quality is maintained while guaranteeing the quickest modification cycles for the application.

6. SECURITY CONCERNS

A. API Key Management

Maintaining the integrity of the Gemini API Healthcare Associate is extremely important. The application practices the listed actions:

1) Avoidance of hardcoding secrets through environment based configuration.
2) Google Secret Manager [12] integration for cloud deployments.
3) Strict enforcement of the least privileged principle for service accounts.
4) Mandated periodic key changes.

B. Data Protection

Even though user data is not stored persistently by Medi- Mind, several measures are put into place to secure data while in transit:

1) Server-client communication is encrypted through HTTPS.
2) Secure WebSocket Protocol (WSS) for real-time chat features.
3) Upholding data minimization to guard confidential data.
4) Processes images only after clear consent from users.

C. Container Security

Several security models are implemented [5] in the containerized application measures:

1) Running as non-root user within the container
2) Minimal base images to decrease attack surface
3) Security scanning of container images as a regular practice of defense.
4) Principles of immutable infrastructure

D. Cloud Security

In GCP, when deployed, MediMind uses security features for cloud-native:

1) VPC Service Controls for limiting API access
2) Cloud Armor to act as a web application firewall (WAF)
3) IAP access to authenticate with the URL (direct API calls)
4) Monitoring security — Cloud Audit Logging

7. EVALUATION AND RESULTS

A. Performance Metrics

We evaluated the MediMind based on a few key factors:

1) Response time for answering text-based questions
2) Speed of processing and analyzing images.

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN: XXXX-XXXX
Volume: 01 | Issue: 01 | March 2026

3) Ability to handle different workloads efficiently.
4) Scalability under varying load conditions

The data shows that using a container setup maintains steady performance in various settings. Text questions are usually answered in less than 2 seconds, while images are analyzed in under 5 seconds.

B. Scalability Testing

We also looked at how the application can grow and manage more work:

1) We tested with simulated users ranging from 10 to 1000 at the same time.
2) We created scenarios with sudden traffic increases to test automatic scaling.
3) We checked how well it manages resources during long usage sessions.

The serverless Cloud Run setup performed extremely well, automatically scaling up to handle more users without needing any manual adjustment.

C. Cost Analysis

We examined the costs for different deployment strategies:

1) Serverless Cloud Run deployment
2) Virtual Machine (VM) deployment on Compute Engine.
3) Deployment using Kubernetes on Google Kubernetes Engine (GKE).

TABLE II: RESPONSE TIME COMPARISON

Query Type	Cloud Run (s)	Local (s)
Text Query	1.5	2.5
Food Image	4.0	6.0
Medication ID	3.5	5.5

D. Performance Comparison

Table II gives a summary of how quickly the MediMind responds to user questions when using Cloud Run versus running it locally.

The findings show that Cloud Run responses are much quicker because it has an improved cloud setup. Text replies happen in about 1.5 seconds, and image checks, especially for food questions, are done in under 4 seconds.

8. CONCLUSION

This research paper discussed the MediMind, a cloud-based AI ap- plication using Google's Gemini 2.0 Pro model to offer health information and analyze images. By integrating advanced AI with modern cloud technology, Healthcare Assistant exempli- fies how new AI features can be incorporated into accessible and scalable healthcare solutions. The development process shared here acts as a guide for creating similar AI applications in sectors like healthcare. Emphasizing the importance of security and performance, it highlights the need for best practices in deploying AI tools in cloud environments. As AI continues to develop, applications like MediMind will become increasingly important in making health information more available and engaging, ultimately helping to raise health awareness and support preventive health measures

REFERENCES

[1] P. Rajpurkar et al., "AI in health and medicine," Nature Medicine, vol. 28, no. 1, pp. 31-38, Jan. 2022.
[2] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.
[3] Google, "Gemini: A family of highly capable multimodal models," Google Research Blog, Dec. 2023.
[4] D. Bernstein, "Containers and cloud: From LXC to Docker to Kuber- netes," IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014.
[5] Docker Inc., "Docker: Build, Ship, and Run Any App, Anywhere," Docker Documentation, 2023.
[6] Google Cloud, "Cloud Run: Run containerized applications on a fully managed serverless platform," Google Cloud Documentation, 2023.
[7] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in Proceedings of the 10th European Conference on Computer Systems, 2015.
[8] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23, Oct. 2009.
[9] Google Health, "Multimodal AI for Personal Health Management," J. Med. Internet Res., vol. 25, no. 1, p. e40678, 2023.
[10] A. Gupta et al., "Cloud-based AI Diagnostics," IEEE Trans. Biomed. Eng., vol. 69, no. 5, pp. 1678-1689, 2022.
[11] B. Li and Q. Wang, "Multimodal Fusion for Healthcare," Proc. AAAI, pp. 12345-12353, 2023.
[12] C. Chen and M. Zhang, "Secure Cloud Deployment," IEEE Cloud Comput., vol. 9, no. 3, pp. 45-53, 2022.
[13] OWASP, "Docker Security Cheat Sheet," OWASP Foundation, 2023.
[14] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," Linux Journal, vol. 2014, no. 239, Mar. 2014.
[15] A. Ronacher, "Flask Web Development, One Drop at a Time," Flask Documentation, 2023.
[16] M. Paluszek and S. Thomas, "HTTPS Everywhere: Security in the Age of Threats," in Communications of the ACM, vol. 60, no. 1, pp. 33-35, Jan. 2017.
[17] S. Newman, "Building Microservices: Designing Fine-Grained Systems," O'Reilly Media, 2015.
[18] A. Fox and D. Patterson, "Engineering Software as a Service: An Agile Approach Using Cloud Computing," Strawberry Canyon LLC, 2014.
[19] National Institute of Standards and Technology, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, Sept. 2011.