Health AI: Intelligent Healthcare Assistant

Generative AI with IBM

Transforming healthcare delivery through intelligent AI solutions that enhance patient care and clinical efficiency

The Healthcare Workforce Crisis Meets AI Innovation

18M
Worker Shortage
Global healthcare worker shortfall expected by 2030 according to IBM research

77%
Executive Confidence
Healthcare executives report AI delivers measurable competitive advantage

The healthcare industry faces an unprecedented workforce crisis, but IBM's AI assistants are emerging as a powerful solution. By automating administrative tasks and coordinating multidisciplinary teams, these intelligent systems are boosting productivity when it's needed most.

Healthcare organizations worldwide are discovering that AI doesn't replace human care—it amplifies it, allowing medical professionals to focus on what matters most: patient outcomes.

IBM's Agentic AI: From Productivity to Performance

AI Integration Services

End-to-end transformation with agentic apps combining AI assistants, orchestration, and optimized data management

Expert Workforce

Over 75,000 IBM consultants certified in generative AI, accelerating adoption with governance and ethical frameworks

Multi-Agent Orchestration

Empowers AI to autonomously execute complex healthcare workflows with precision and reliability

IBM's comprehensive approach ensures healthcare organizations don't just implement AI—they transform their entire operational ecosystem with intelligent, ethical, and scalable solutions.

Virtual Health Assistants: Personalized Patient Support

Smart Scheduling

AI-powered assistants handle appointment scheduling and send tailored medication reminders

Chronic Care Management

Real-time health monitoring with generative AI that adapts care plans dynamically

Mental Health Support

AI-driven therapeutic interactions enhance accessibility and privacy for mental wellness

Companies like iLink Digital are already demonstrating how AI-driven health plans can revolutionize patient engagement, making personalized healthcare accessible 24/7.

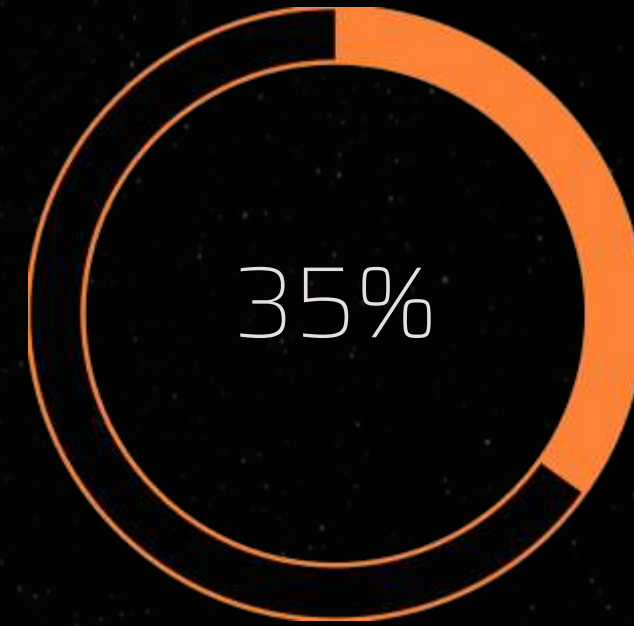# Clinical Impact: Early Detection & Decision Support

40%

Current AI Usage

Healthcare executives using AI for inpatient monitoring and early warnings

IBM Watson Health Excellence

Leveraging AI for precision oncology and predictive analytics that integrate genomics and wearable data for comprehensive patient insights.

35%

Faster Diagnosis

Reduction in diagnostic time, improving outcomes and lowering costs

Future Implementation

Full AI implementation for clinical monitoring is expected within 3 years, marking a transformative shift in healthcare delivery.

Overcoming Challenges: Security, Skills & Integration

Cybersecurity Priority

53% of healthcare leaders cite cybersecurity and patient data protection as their top challenges in AI implementation

Privacy-First AI

IBM promotes zero trust frameworks and rigorous data governance to ensure patient information remains secure

Workforce Development

Strategic workforce upskilling and change management programs are critical for scaling AI-powered healthcare solutions

Success in healthcare AI isn't just about technology—it's about building trust, ensuring security, and empowering healthcare professionals with the skills they need.

The Future of Healthcare with IBM AI

Liberated Clinicians

AI assistants free healthcare professionals to focus on patient care, dramatically improving quality and efficiency

Scalable Ecosystem

IBM's comprehensive AI Integration Services enable ethical, high-impact AI adoption across healthcare organizations

Compassionate Innovation

Where intelligent AI meets compassionate care—transforming healthcare delivery for patients and providers alike

Join IBM in revolutionizing healthcare delivery—creating a future where technology amplifies human compassion and clinical excellence.

```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
```

```
                do_sample=True,
                pad_token_id=tokenizer.eos_token_id
            )

        response = tokenizer.decode(outputs[0], skip_special_tokens=True)
        response = response.replace(prompt, "").strip()
        return response

def disease_prediction(symptoms):
    prompt = f"Based on the following symptoms, provide possible medical conditions and general medication suggestions. Always emphasize the importance of consulting a doctor for
    return generate_response(prompt, max_length=1200)

def treatment_plan(condition, age, gender, medical_history):
    prompt = f"Generate personalized treatment suggestions for the following patient information. Include home remedies and general medication guidelines.\n\nMedical Condition: {
    return generate_response(prompt, max_length=1200)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Medical AI Assistant")
    gr.Markdown("**Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.**")

    with gr.Tabs():
        with gr.TabItem("Disease Prediction"):
            with gr.Row():
                with gr.Column():
                    symptoms_input = gr.Textbox(
                        label="Enter Symptoms",
                        placeholder="e.g., fever, headache, cough, fatigue...",
```

```
                                              lines=4
                                         )
                                    predict_btn = gr.Button("Analyze Symptoms")

                             with gr.Column():
                                    prediction_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)

                        predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)

                  with gr.TabItem("Treatment Plans"):
                        with gr.Row():
                             with gr.Column():
                                    condition_input = gr.Textbox(
                                         label="Medical Condition",
                                         placeholder="e.g., diabetes, hypertension, migraine...",
                                         lines=2
                                    )
                                    age_input = gr.Number(label="Age", value=30)
                                    gender_input = gr.Dropdown(
                                         choices=["Male", "Female", "Other"],
                                         label="Gender",
                                         value="Male"
                                    )
                                    history_input = gr.Textbox(
                                         label="Medical History",
                                         placeholder="Previous conditions, allergies, medications or None",
                                         lines=3
                                    )
```

```python
predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)

        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
                        placeholder="e.g., diabetes, hypertension, migraine...",
                        lines=2
                    )
                    age_input = gr.Number(label="Age", value=30)
                    gender_input = gr.Dropdown(
                        choices=["Male", "Female", "Other"],
                        label="Gender",
                        value="Male"
                    )
                    history_input = gr.Textbox(
                        label="Medical History",
                        placeholder="Previous conditions, allergies, medications or None",
                        lines=3
                    )
                    plan_btn = gr.Button("Generate Treatment Plan")

                with gr.Column():
                    plan_output = gr.Textbox(label="Personalized Treatment Plan", lines=20)

            plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input, history_input], outputs=plan_output)

    app.launch(share=True)
```

Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Disease Prediction   Treatment Plans

Enter Symptoms

e.g., fever, headache, cough, fatigue...

Possible Conditions & Recommendations

Analyze Symptoms