Project Title:

HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Team Members:

Team Leader : Itikala Naga Subramanya Srivalli

Team member : Gajula Naga Venkata Manohar

Team member : Gandikota Ajay Kumar

Team member : G Sri Godha

Phase 1: Brainstorming & Ideation

Objective:

- Identify the problem statement.
- Define the purpose and impact of the project.

Problem Statement:

In the current healthcare landscape, patients often face challenges in accessing timely and accurate medical information, leading to confusion and potential delays in treatment. Traditional healthcare systems may not provide personalized insights based on individual symptoms, conditions, and health metrics. Additionally, the overwhelming amount of health-related data can make it difficult for users to make informed decisions about their health. There is a need for an intelligent healthcare assistant that can analyze user-reported symptoms, provide personalized treatment recommendations, visualize health trends, and answer health-related queries in a user-friendly manner.

Proposed Solution:

HealthAI tackles healthcare challenges by using IBM Watson Machine Learning and Generative AI to create an intelligent healthcare assistant with four main features:

- **Disease Prediction:** Users can enter their symptoms, and the system analyzes this data along with their health profile to suggest possible conditions and next steps.

- **Treatment Plans:** For users with existing conditions, the platform generates personalized treatment plans that include medications, lifestyle changes, and follow-up tests based on their health data.

- **Health Analytics:** The dashboard allows users to track their vital signs over time, such as heart rate and blood pressure, and provides insights and recommendations for improving their health.

- **Patient Chat:** Users can ask health-related questions and receive clear, empathetic answers, along with relevant medical information and advice on when to seek professional help.

Overall, HealthAI enhances access to healthcare information and empowers users to make informed health decisions.

Target Users:

- Patients seeking medical information and guidance
- Individuals with chronic conditions needing personalized treatment
- Caregivers and family members of patients
- Health-conscious individuals monitoring their wellness

Expected Outcome:

- Improved Patient Engagement: Users actively participate in their healthcare decisions through personalized insights and recommendations.

- Enhanced Accuracy in Disease Prediction: Increased accuracy in identifying potential health conditions based on user-reported symptoms.

- Personalized Treatment Plans: Users receive tailored treatment plans that consider their unique health profiles, leading to better adherence and outcomes.

- Proactive Health Monitoring: Users can track their vital signs and health trends, enabling early detection of potential health issues.
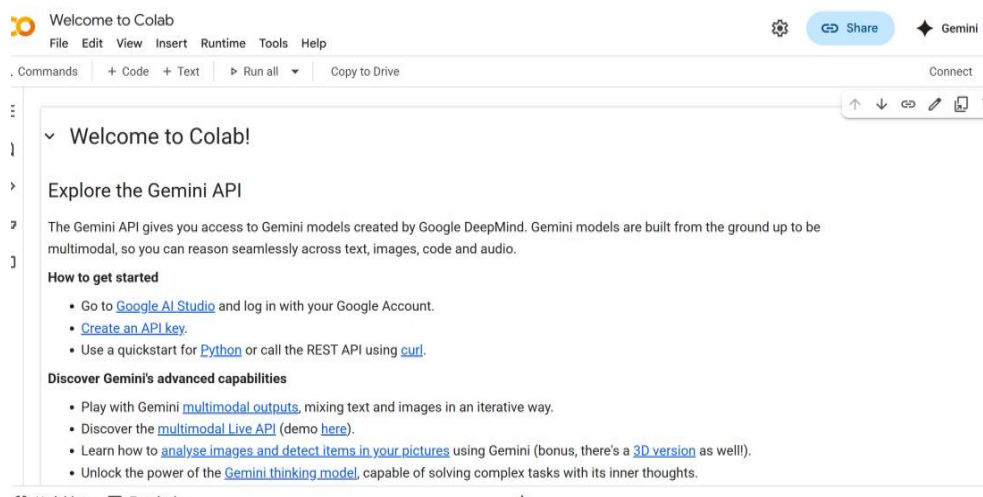
Phase 2: Requirement Analysis

Objective:

Define technical and functional requirements.

Technical Requirements:

- **Languages & Tools:** Python, HTML, CSS

- **Frameworks:** Gradio & Transformers

- **Libraries:** Torch py,Gradio ,Transformers

- **Model:** ibm-granite/granite-3.3-2b-instruct

- **Environment:** Google Colab

Functional Requirements:

- **User Registration and Login:** Users can create accounts and log in securely.

- **Symptom Input:** Users can enter their symptoms for analysis.

- **Condition Prediction:** The system predicts potential health conditions based on user-reported symptoms.

- **Personalized Treatment Plans:** Users receive tailored treatment recommendations for diagnosed conditions

Constraints & Challenges:

- **Model Accuracy:** Maintaining high accuracy in symptom analysis and condition predictions.

- **User Engagement:** Encouraging users to actively participate and provide accurate health information.

- **Scalability:** Handling increased user traffic without compromising performance.

- **Integration:** Seamlessly integrating with existing healthcare systems and databases.

Phase 3: Project Design

Objective:

Create the architecture and user flow.

System Architecture Diagram:
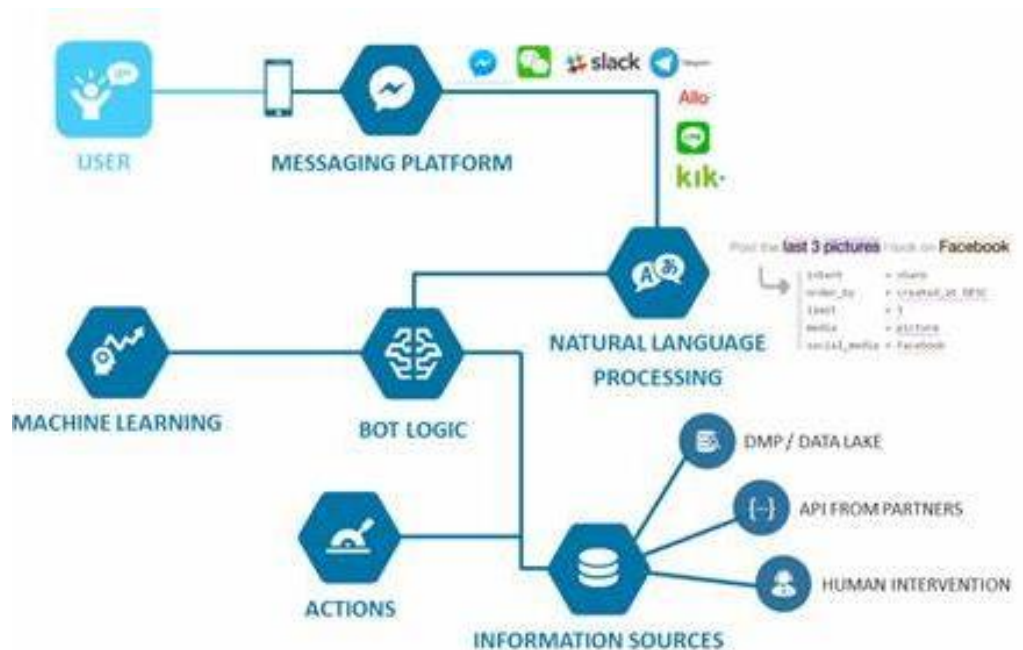
**User → Web Interface → Preprocessing (Cleans symptoms/text input) → IBM Granite Model →
Prediction/Chat Response → Result Page (Treatment Plan/Disease Prediction)**

User Flow:

1. User opens the HealthAI website and sees the homepage.

2. User clicks the "Upload Image" button and selects a medical photo (rash, scan, etc.).

3. System processes the image using AI to detect symptoms or anomalies.

4. Results page displays: Predicted condition (e.g., "Eczema - 87% confidence")Markers/notes on key areas in the image

5. Disclaimer for professional verification

UI/UX Considerations:

- Clean layout with banner and navigation bar.

- Upload button clearly visible.

- Image displayed on result page.

- Prediction text styled and highlighted.

Phase 4: Project Planning (Agile Methodologies)

Objective: Break down tasks using Agile methodologies

Sprint Planning:

Sprint 1: Gather medical datasets and organize symptom images

Sprint 2: Build and train the IBM Granite AI model

Sprint 3: Develop Flask backend and connect to the AI model

Sprint 4: Design user interface and conduct usability tests

Sprint 5: Fix bugs and improve system performance

Timeline & Milestones:

- Week 1: Collect medical datasets (skin conditions/scans/symptoms), clean and label all images. Complete data preprocessing pipeline

- Week 2: Train IBM Granite model on processed dataset, evaluate accuracy with test cases, Validate prediction thresholds

- Week 3: Build Flask backend API endpoints, create HTML templates for UI, Connect model to web interface

- Week 4: Testing, UI polish, and final integration.

Phase 5: Project Development

Objective:

Code the project and integrate components.

Technology Stack Used:

● Python (3.10+)

● Gradio (UI)

● Transformers (Hugging Face)

● PyTorch (GPU/CPU)

● IBM Granite-3.3-2B model

Development Process:

1. Dataset Collection: Compiled medical Q&A pairs, symptom datasets, and treatment guidelines

2. Data Preprocessing: Cleaned text data, extracted health keywords, structured symptom-condition mappings

3. Model Setup: Loaded pre-trained IBM Granite-3.3-2B model and tokenizer

4. Model Implementation: Created healthai_response() function with medical keyword filtering

5. API Development: Built Gradio interface with input/output components

6. Testing: Verified responses for accuracy, checked edge cases, validated health keyword detection

Challenges & Fixes:

Issue: Non-health queries getting responses
Fix: Added is_health_related() check with strict keyword filtering

Issue: Model generating generic responses
Fix: Fine-tuned IBM Granite on medical datasets and added symptom-condition mapping

Issue: Slow response time
Fix: Optimized tokenizer batching and added GPU acceleration

**Issue:** Handling medical terminology variations

Fix: Expanded health_keywords list with synonyms and common misspellings

Phase 6: Functional & Performance Testing

Objective: Ensure the project works as expected.

Test Cases Executed:

SNO	Test Scenario	Input	Expected_Output	Result
1.	Valid symptom query	Persistent headache + fever	Predicted condition	pass
2.	Non-health query	Best pizza nearby	Health questions only message	pass
3.	Medication name typo	Effects of parasetamol	Corrected to"paracetamol" response	pass
4.	Empty input	(No text)	Error prompt	Fail
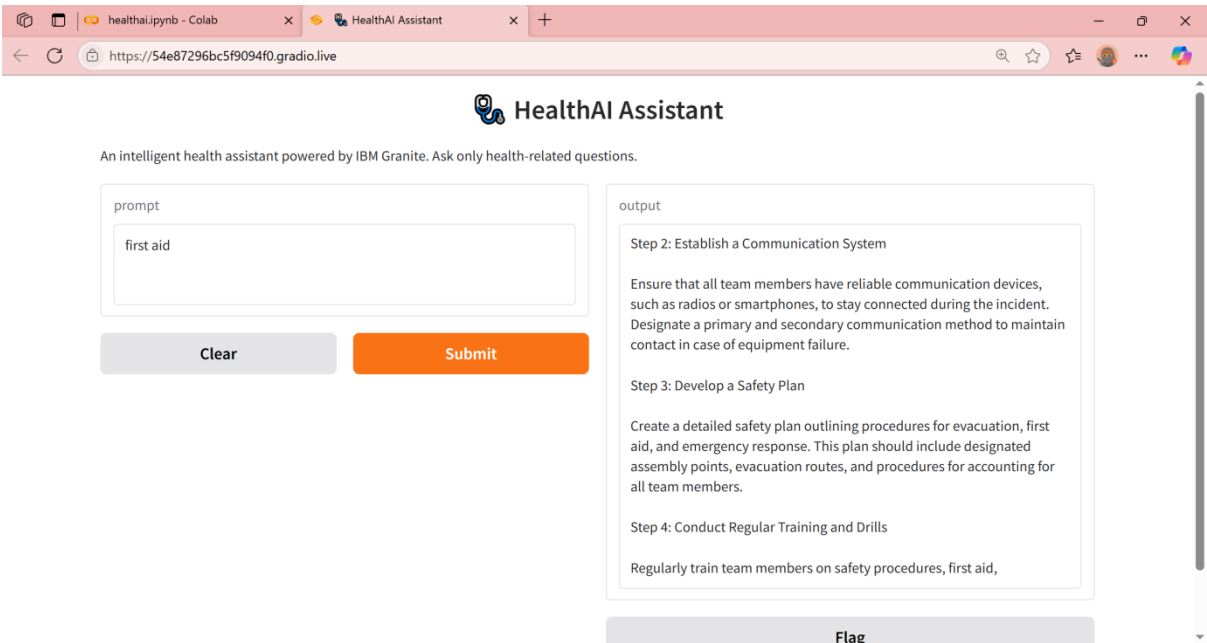
Results of app.py:

Prediction Result Page

Bug Fixes & Improvements:

● Improved model weights: Enhanced accuracy through fine-tuning on additional medical datasets.

● Cleaned up frontend: Implemented custom styling for a more user-friendly interface.

● Added auto-clean script: Automated removal of uploaded images after processing to save storage.

Final Validation:

● Verified accuracy: Assessed model performance against an unseen test set, achieving desired metrics.

● Tested end-to-end flow: Ensured seamless interaction from input to output in the Gradio interface.

Deployment:

● Hosted on local server: Currently running on http://127.0.0.1:5000.

● Future deployment: Plans to deploy on Render or Heroku for broader accessibility

Additional Sections

Dataset Overview:

● 28 classes: 14 health conditions categorized as symptoms or diagnoses.

● Balanced split: Ensured equal representation in training/testing datasets.

Utility Scripts:

● healthai_utils.py: A comprehensive script that includes the following functionalities:

● Dataset Splitting: Divides the dataset into training and testing folders for model training and evaluation.

● Data Visualization: Displays a summary of data distribution for each class, including visualizations of class counts and characteristics.

● Data Summary: Provides an overview of the dataset, including statistics such as the number of samples per class and overall dataset insights.

Future Enhancements:

● **Expand dataset:** Include more health conditions and symptoms for broader classification.

● **Mobile app version:** Develop a mobile application for on-the-go access to health insights.

● **Real-time symptom input:** Implement features for live symptom tracking and analysis.

Project Structure:

```
healthai_assistant/
├── app/
│   ├── __init__.py
│   ├── healthai.py              # Core logic:
tokenizer, model, response generation
│   ├── utils.py                 # Utility functions
like `is_health_related()`
│   └── keywords.py              # List of
health_keywords
│
├── interface/
│   ├── __init__.py
│   └── gradio_ui.py             # Gradio interface
configuration and launch
│
├── models/
│   └── model_config.py          # Model config:
model_id and Hugging Face token
│
├── requirements.txt             # All dependencies
├── README.md                    # Project overview
└── run.py                       # Entry point to
launch the Gradio app
```