Health AI-Project Planning:

# 1. Abstract

This project presents a Medical AI Assistant developed using Python, Gradio, and IBM Granite-3.2-2B-Instruct model. The system is designed to provide informational health guidance, including disease prediction from symptoms and generating personalized treatment plans. The application uses Large Language Model (LLM) capabilities to generate natural language responses and offers an interactive interface for users. A strong emphasis is placed on disclaimers, ensuring users understand the tool is for informational purposes only and does not replace medical professionals.

# 2. Introduction

Healthcare information is vital for patients, especially for identifying symptoms and possible conditions. This project integrates AI with healthcare by developing a Medical AI Assistant that predicts possible conditions from symptoms and generates treatment plans based on patient information. The project demonstrates how Large Language Models can be applied in healthcare applications.

# 3. Objectives

- Develop an easy-to-use interface for health-related queries.  
- Assist users in identifying possible conditions from symptoms.  
- Provide general treatment suggestions and home remedies.  
- Emphasize responsible AI usage with disclaimers.

# 4. Tools and Technologies

Programming Language: Python 3.x  
Libraries: Gradio, Transformers, PyTorch  
Model: IBM Granite-3.2-2B-Instruct  
Deployment: Gradio interface with shareable link

# 5. System Architecture

The system follows a structured workflow:  
  
User Input → Tokenizer → Granite LLM → Response Decoder → Gradio UI Output  
  
1. User provides symptoms or condition details.  
2. Input is tokenized using AutoTokenizer.  
3. Model generates a response using causal language modeling.  
4. Response is decoded and displayed in the interface.

# 6. Functional Requirements

1. Disease Prediction: Input symptoms → Output possible conditions & recommendations.  
2. Treatment Plan: Input condition, age, gender, medical history → Output personalized plan.  
3. User Interface: Tab-based, interactive, and easy-to-use.

# 7. Non-Functional Requirements

- Performance: Responses within 5–10 seconds.  
- Usability: Simple and clean design.  
- Reliability: Handles invalid inputs gracefully.  
- Ethics: Always includes disclaimers.

# 8. Implementation Plan

Phase 1: Requirements Gathering & Model Selection  
Phase 2: Model Integration  
Phase 3: Interface Development  
Phase 4: Testing  
Phase 5: Deployment

# 9. Testing

- Unit Testing: Test each function separately.  
- Integration Testing: Validate flow between UI and model.  
- Validation Testing: Ensure disclaimers are always shown.

# 10. Advantages

- Quick health-related information.  
- Interactive and accessible.  
- Patient-specific personalization.  
- Demonstrates AI use in healthcare.

# 11. Limitations

- Not a substitute for real diagnosis.  
- Accuracy depends on model training.  
- Internet required for model download.  
- Response speed depends on hardware.

# 12. Applications

- Educational healthcare tool.  
- Informational assistant for patients.  
- Demonstration of AI in healthcare.  
- Extendable to telemedicine with oversight.

# 13. Conclusion

The Medical AI Assistant demonstrates how AI can contribute to healthcare information systems. By predicting possible conditions and suggesting general treatment plans, it provides valuable insights while emphasizing professional consultation. With further enhancements and expert supervision, this system can be integrated into healthcare platforms for broader applications.