HealthAI: Intelligent Healthcare Assistant Using IBM Granite

APSCHE Short Term Virtual Internship Program

Generative AI with IBM Cloud

Team Members:

1. Boreddy Supriya Reddy
2. Kakarla Naganandini
3. Nimmakayala Himani

Team ID:

LTVIP2025TMID31711

Abstract

HealthAI is an intelligent healthcare assistant powered by IBM Watsonx and Granite Generative AI models, designed to enhance accessibility and efficiency in the healthcare domain. Leveraging advanced natural language processing and machine learning, HealthAI simulates real-time medical consultation through an intuitive Streamlit-based interface. The system enables users to interact with an AI-driven chatbot for addressing health-related queries, predicting potential diseases based on symptom input, and receiving personalized treatment recommendations.

In addition to medical support, HealthAI offers guidance on home remedies and first aid for common ailments, promoting selfcare and timely intervention. The platform also includes a health analytics module that derives meaningful insights from user data to support preventive healthcare strategies. By integrating multiple AI functionalities into a single platform, HealthAI aims to bridge the gap between patients and professional medical assistance, especially in underserved areas. This project demonstrates the transformative potential of generative AI in revolutionizing patient care and decision-making in the healthcare industry.

Contents

Chapter 1

Introduction

1.1 Project Overview

In recent years, the healthcare industry has witnessed a dramatic transformation due to the integration of artificial intelligence (AI). Patients increasingly seek digital tools that offer timely, accurate, and personalized health insights. Traditional health consultation systems are often time-consuming, limited by location, and constrained by human resource availability. To overcome these challenges, AI-driven solutions are being adopted to make healthcare more accessible, efficient, and intelligent.

1.2 Application

The application of HealthAI spans multiple domains within healthcare:

- **Patient Chat:** Enables real-time Q&A system for health-related queries.
- **Disease Prediction:** Analyzes reported symptoms to suggest possible conditions.
- **Treatment Planner:** Generates treatment strategies considering age, gender ..
- **Home Remedies and First Aid:** Offers effective remedies for minor ailments.
- **Health Analytics:** Delivers trend-based insights from user inputs for care.

1.3 Goal

With the increasing demand for smart healthcare solutions and the growing adoption of AI in medicine, there is a significant opportunity to bridge the gap between patients and doctors. The motivation behind HealthAI stems from the need to:

- Provide 24/7 access to medical assistance.
- Reduce the burden on overburdened healthcare systems.
- Empower users with knowledge about their health through intelligent systems.
- Promote early diagnosis and preventive healthcare.

By integrating IBM Granite models and deploying a Streamlit-based interface, HealthAI offers an innovative and scalable solution for modern healthcare needs.

1.4 Problem Statement

Despite advancements in healthcare technology, a significant portion of the population still lacks immediate access to reliable medical information and guidance. Key problems include: • **Delayed Diagnosis:** Due to limited availability of doctors in remote areas.

- **Lack of Personalized Advice:** Generic solutions often ignore patient-specific data like age, gender, and comorbidities.
- **Overcrowded Medical Facilities:** Leading to long waiting times and inefficiencies.
- **Limited Awareness:** Of symptoms, treatments, and home remedies among non-medical users.

There is a pressing need for an intelligent healthcare assistant that can address these issues by offering personalized, accessible, and real-time medical support to users.

HealthAI addresses this challenge by leveraging Generative AI to simulate expert-level consultations and deliver actionable insights instantly.

Chapter 2

Project Plan

2.1 Brainstorming & Ideation

We conducted multiple brainstorming sessions to determine the potential of AI in healthcare assistance. We analyzed real-world scenarios where patients need instant and reliable medical information. This led to the ideation of a unified platform capable of answering health-related questions, predicting conditions, and offering personalized treatment advice.

2.2 Requirement Analysis

Based on user feedback and market research, we identified the following needs:

- Medical query resolution
- Symptom-based disease prediction
- Localized and contextual suggestions

We studied pain points in current digital healthcare tools and finalized the system requirements.

They are :

- Natural language processing for queries
- Multilingual support
- Integration with a generative AI model
- Modules for prediction, chat, treatment, and analytics

2.3 Project Design

The system was designed with modular components to ensure flexibility and scalability:

- UI wireframes created using Streamlit for an intuitive interface
- Modular architecture ensures seamless integration between features
- Responsive design across devices
- Flow: Input → AI Response → Display → Optional Email Forwarding

2.4 Project Planning (Agile Methodologies)

To ensure timely development and continuous improvement, we adopted Agile methodology:

- Divided the project into 2-week sprints:
  - **Sprint 1:** Frontend UI setup and chat integration
  - **Sprint 2:** Disease prediction model integration
  - **Sprint 3:** Personalized treatment planner
  - **Sprint 4:** Analytics and multilingual features

- Conducted daily stand-up meetings to track progress and resolve issues promptly

2.5 Project Development

The application was developed using Python and modern frameworks:

- Built using **Python** and **Streamlit** for rapid development
- Integrated **IBM Granite** model via Hugging Face API for intelligent responses Core modules developed:
- Patient Chat
- Disease Prediction
- Personalized Treatment Planning
- Health Analytics Helper utilities:
- Translation module
- Speech-to-text input
- Email forwarding feature
- Location detection for localized care

2.6 Functional and Performance Testing

Comprehensive testing was performed to ensure reliability and performance:

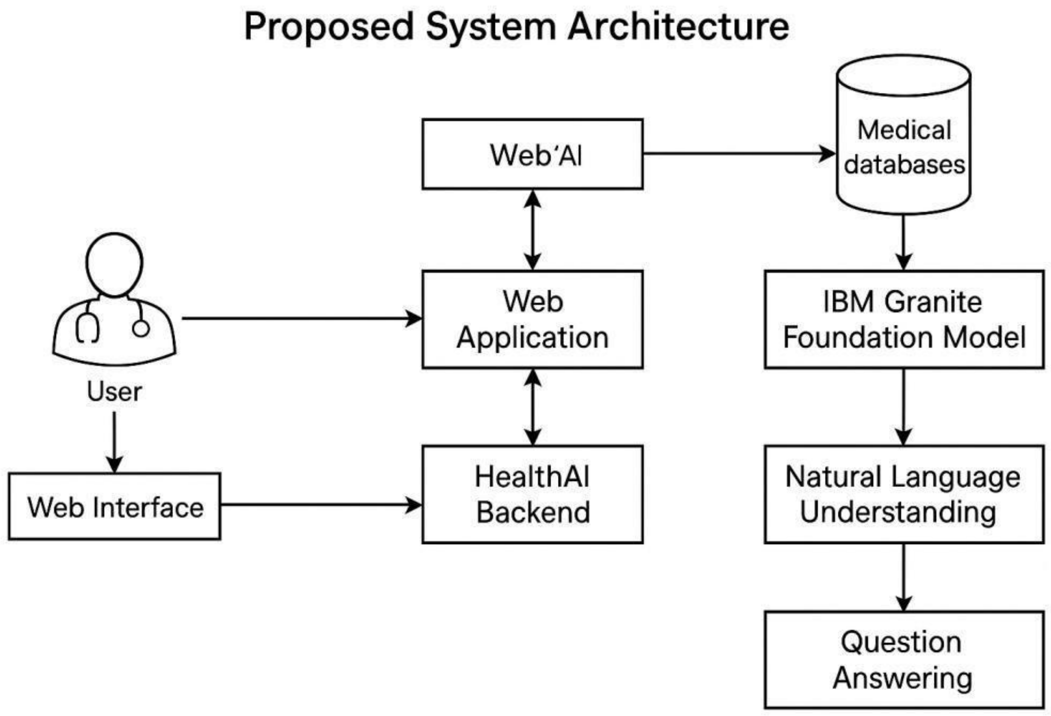
- Conducted functional testing for each feature/module
- Validated AI responses for accuracy and context alignment
- Optimized response time to ensure app responds within **1–3 seconds**

- Load tested API interactions under peak usage scenarios
- Confirmed compatibility across devices and browsers

Chapter 3

Code

3.1 Proposed System Architecture



3.2 Library / Technology Reference Table

Library / Technology	**Category**	**Purpose**	**Usage in Project**

streamlit	Frontend/Web UI	Build web apps using Python	Create UI, i/O, and deploy AI tools
langchain	AI Framework	Framework for LLM applications	Prompts, LLM chains and Workflows
langchain-ibm	IBM Integration Layer	LangChain plugin for IBM Watsonx	Connect LangChain with IBM Watsonx AI
ibm-watsonx-ai	IBM SDK	SDK for IBM Watsonx	Authentication and access from Watsonx
ibm watson machine	IBM SDK	Interface with Watson ML	Configure model parameters
pandas	Data Analysis	Tabular data handling	Reading, filtering, and displaying datasets
numpy	Numerical Computing	Efficient numerical operations	Support math and array operations
matplotlib	Data Visualization	2D static plotting	Visualize analysis from health/user data
plotly	Data Visualization	Interactive plotting	Dynamic, zoomable charts in Streamlit

bs4 (BeautifulSoup)	Web Scraping	HTML/XML parsing	Extract structured data from web pages

3.3 Project Implementation Phases

The project is divided into the following three main phases:

- **Phase 1: IBM Credentials Setup**

  Configure and authenticate with IBM Watsonx AI using appropriate API keys and credentials. This ensures secure access to foundation models for inference.

- **Phase 2: GitHub and Streamlit Integration**

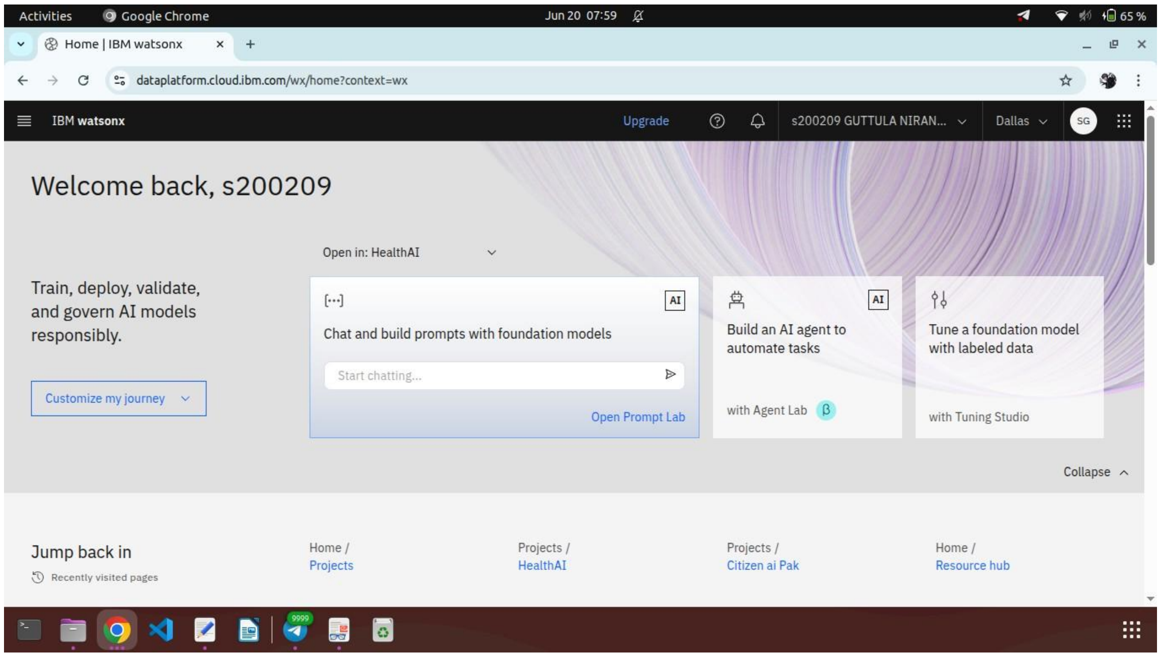
  Set up version control with GitHub and develop the web interface using Streamlit. Integrate LangChain and IBM Watsonx SDKs to build the core functionality of the application.

- **Phase 3: Web App Testing**

  Perform functional and usability testing of the deployed web application. Validate input/output handling, responsiveness, and ensure all AI features behave as expected.

3.4 IBM CREDENTIALS

Step 1 : Login To IBM WatsonAX



Welcome back, s200209

Train, deploy, validate, and govern AI models responsibly.

Chat and build prompts with foundation models

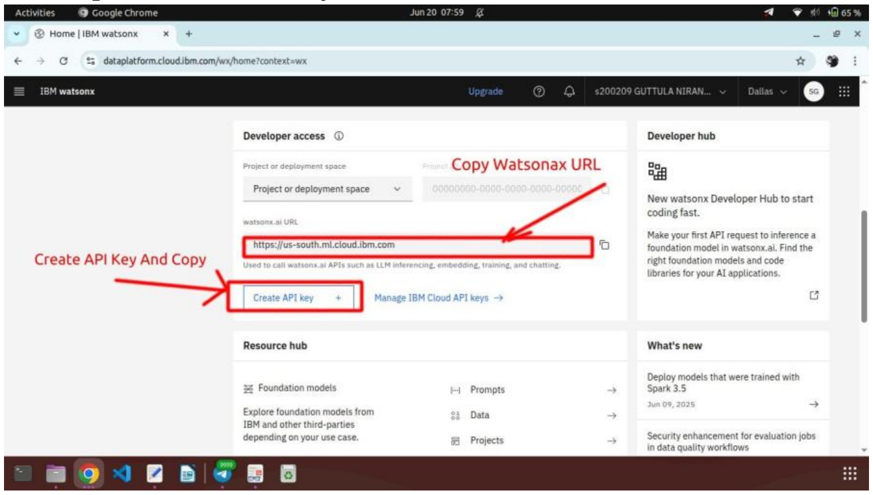
Build an AI agent to automate tasks

Tune a foundation model with labeled data

Jump back in

Recently visited pages

Step 2 : Create API key and URL



Project or deployment space: 00000000-0000-0000-0000-0000

watsonx.ai URL: https://us-south.ml.cloud.ibm.com

Used to call watsonx.ai APIs such as LLM inferencing, embedding, training, and chatting.

New watsonx Developer Hub to start coding fast.

Make your first API request to inference a foundation model in watsonx.ai. Find the right foundation models and code libraries for your AI applications.

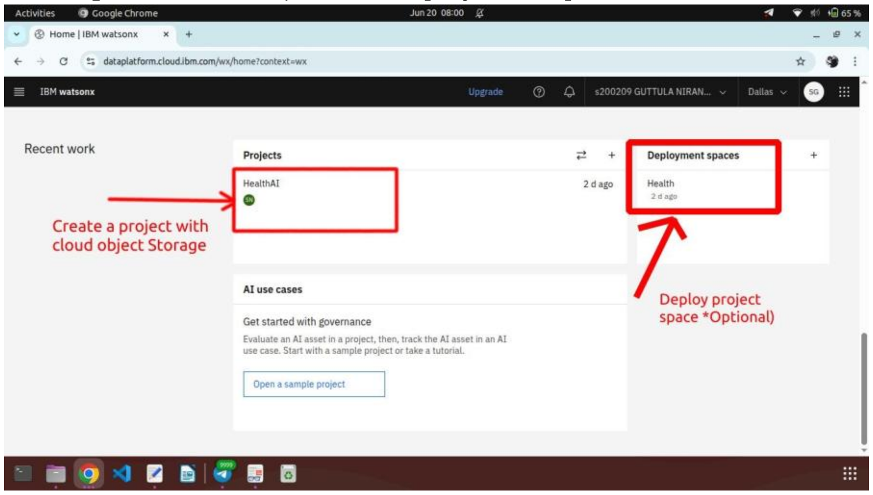
Explore foundation models from IBM and other third-parties depending on your use case.

Deploy models that were trained with Spark 3.5

Jun 09, 2025

Security enhancement for evaluation jobs in data quality workflows

Step 3 : Create Project and Deployment Space



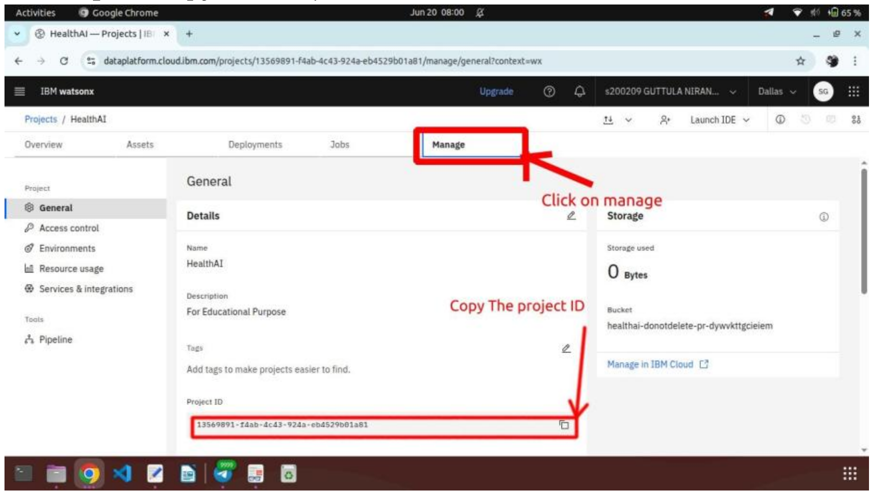
HealthAI — 2 d ago

Health — 2 d ago

Get started with governance

Evaluate an AI asset in a project, then, track the AI asset in an AI use case. Start with a sample project or take a tutorial.

Step 4 : Copy the Project ID



Projects / HealthAI

General

Details

Name
HealthAI

Description
For Educational Purpose

Tags
Add tags to make projects easier to find.

Project ID
13569891-f4ab-4c43-924a-eb4529b01a81

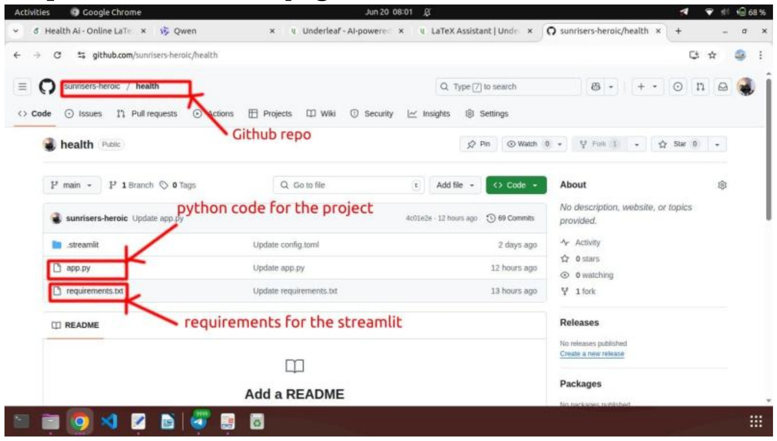
Storage

Storage used
0 Bytes

Bucket
healthai-donotdelete-pr-dywvkttgcieiem

3.5 Github and Code

Step 1 : Githhub Main page



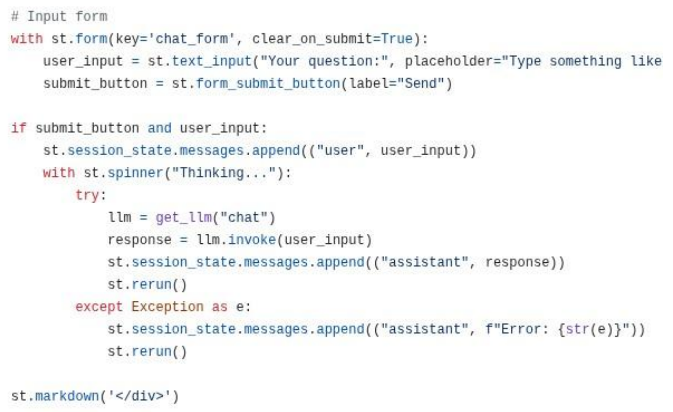
sunrisers-heroic / health

.streamlit	Update config.toml	2 days ago
app.py	Update app.py	12 hours ago
requirements.txt	Update requirements.txt	13 hours ago

IBM Model : Setup the Github Repository

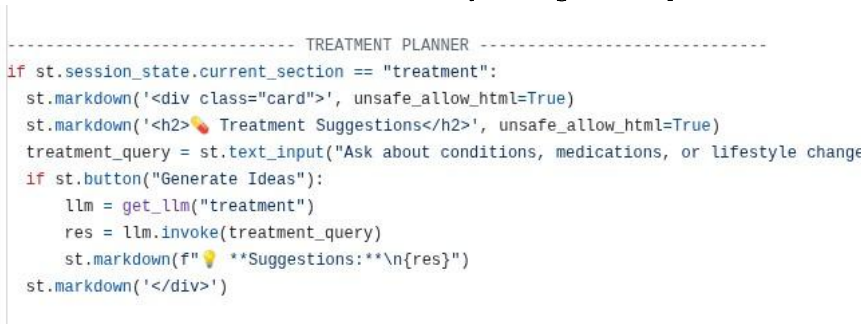
```
def get_llm(model_name):
    return WatsonxLLM(
        model_id=model_map[model_name],
        url=credentials.get("url"),
        apikey=credentials.get("apikey"),
        project_id=project_id,
        params={
            GenParams.DECODING_METHOD: "greedy",
            GenParams.TEMPERATURE: 0.7,
            GenParams.MIN_NEW_TOKENS: 5,
            GenParams.MAX_NEW_TOKENS: 300,
            GenParams.STOP_SEQUENCES: ["Human:", "Observation"],
        },
    )
```

Chat : Creating the patient Chat model



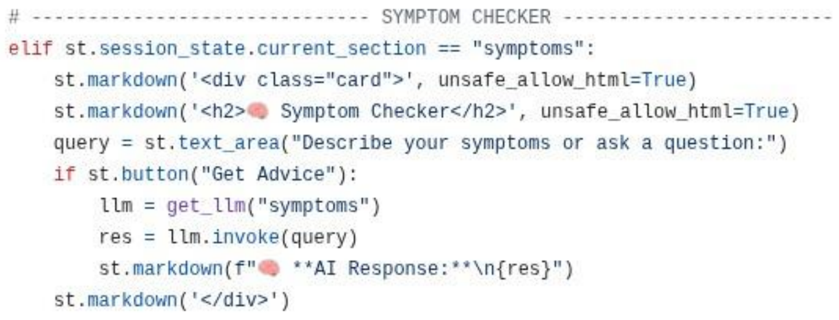
```
# Input form
with st.form(key='chat_form', clear_on_submit=True):
    user_input = st.text_input("Your question:", placeholder="Type something like
    submit_button = st.form_submit_button(label="Send")

if submit_button and user_input:
    st.session_state.messages.append(("user", user_input))
    with st.spinner("Thinking..."):
        try:
            llm = get_llm("chat")
            response = llm.invoke(user_input)
            st.session_state.messages.append(("assistant", response))
            st.rerun()
        except Exception as e:
            st.session_state.messages.append(("assistant", f"Error: {str(e)}"))
            st.rerun()

st.markdown('</div>')
```

Treatment : Treament Genrator by taking User Input



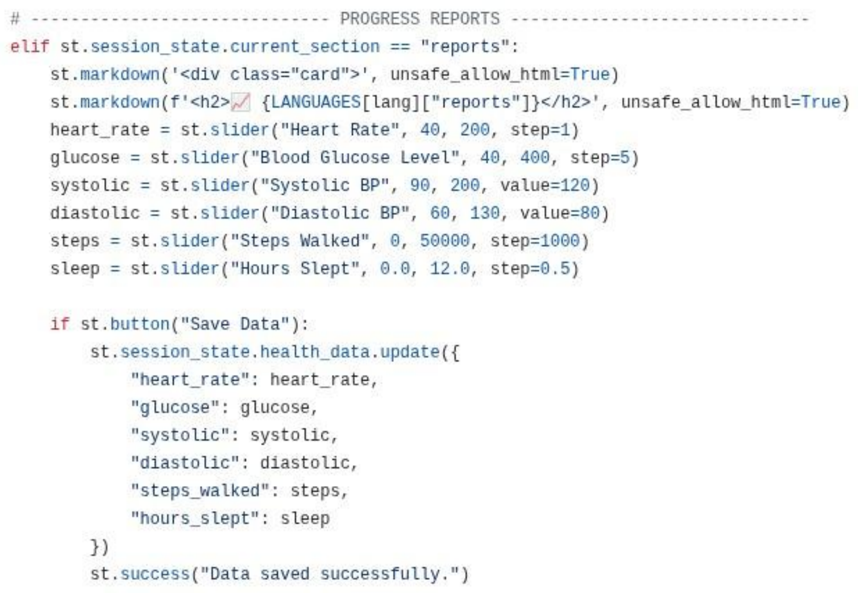
```
------------------------------ TREATMENT PLANNER ------------------------------
if st.session_state.current_section == "treatment":
  st.markdown('<div class="card">', unsafe_allow_html=True)
  st.markdown('<h2>💊 Treatment Suggestions</h2>', unsafe_allow_html=True)
  treatment_query = st.text_input("Ask about conditions, medications, or lifestyle change
  if st.button("Generate Ideas"):
      llm = get_llm("treatment")
      res = llm.invoke(treatment_query)
      st.markdown(f"💡 **Suggestions:**\n{res}")
  st.markdown('</div>')
```

Symptoms checker: symptom checker



```
# ------------------------------ SYMPTOM CHECKER ------------------------
elif st.session_state.current_section == "symptoms":
    st.markdown('<div class="card">', unsafe_allow_html=True)
    st.markdown('<h2>🧠 Symptom Checker</h2>', unsafe_allow_html=True)
    query = st.text_area("Describe your symptoms or ask a question:")
    if st.button("Get Advice"):
        llm = get_llm("symptoms")
        res = llm.invoke(query)
        st.markdown(f"🧠 **AI Response:**\n{res}")
    st.markdown('</div>')
```

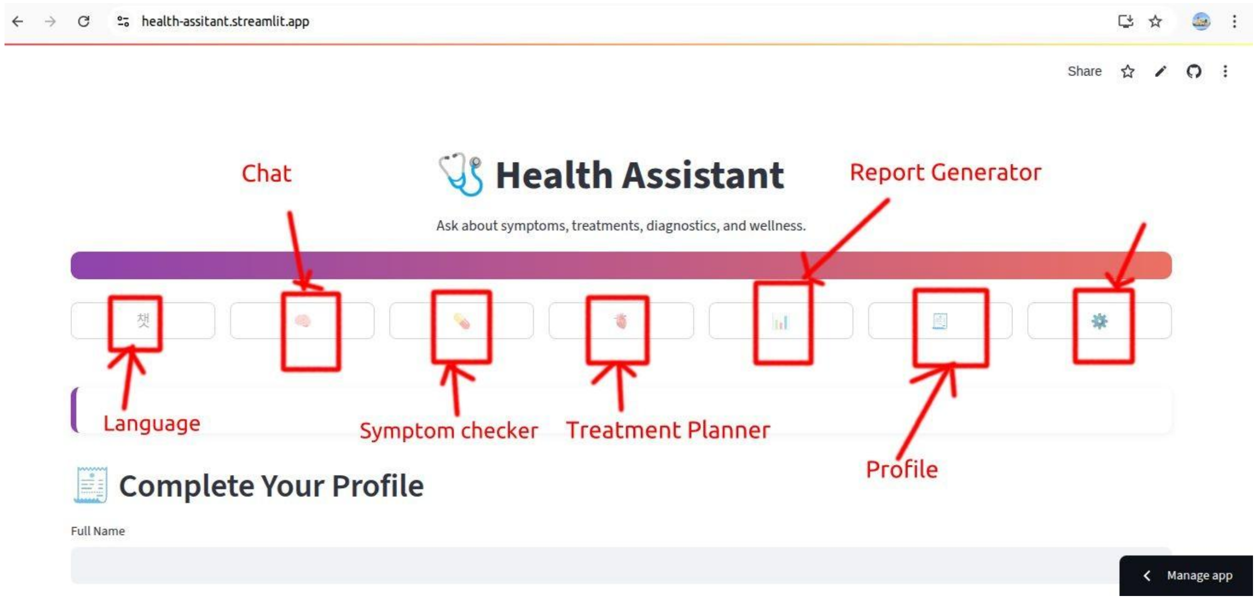
Reports Generation : report creation and pdf download



```
# ------------------------------ PROGRESS REPORTS ------------------------------
elif st.session_state.current_section == "reports":
    st.markdown('<div class="card">', unsafe_allow_html=True)
    st.markdown(f'<h2>📈 {LANGUAGES[lang]["reports"]}</h2>', unsafe_allow_html=True)
    heart_rate = st.slider("Heart Rate", 40, 200, step=1)
    glucose = st.slider("Blood Glucose Level", 40, 400, step=5)
    systolic = st.slider("Systolic BP", 90, 200, value=120)
    diastolic = st.slider("Diastolic BP", 60, 130, value=80)
    steps = st.slider("Steps Walked", 0, 50000, step=1000)
    sleep = st.slider("Hours Slept", 0.0, 12.0, step=0.5)

    if st.button("Save Data"):
        st.session_state.health_data.update({
            "heart_rate": heart_rate,
            "glucose": glucose,
            "systolic": systolic,
            "diastolic": diastolic,
            "steps_walked": steps,
            "hours_slept": sleep
        })
        st.success("Data saved successfully.")
```

3.6 Streamlit Deployment

Home Page :Main Page



Ask about symptoms, treatments, diagnostics, and wellness.

3.7 Modules Overview

1. **Patient Chat**

- **Function:** Allows users to ask health-related queries.

- **Example:** “What are the symptoms of diabetes?”
- **AI Response:** Provides an informative and medically appropriate answer.

2. **Disease Prediction**
    - **Function:** Predicts potential diseases based on symptoms input by the user.
    - **Input Example:** Fever, cough, body aches
    - **Output Example:** You may have Influenza or COVID-19. Please consult a doctor for confirmation.
3. **Personalized Treatment Planner**
    - **Function:** Suggests treatments based on condition, age, gender, and comorbidities.
    - **Input Example:**
        - Condition: Diabetes
        - Age: 50
        - Gender: Female
        - Comorbidities: Hypertension
    - **Output Example:**
        - Treatment Plan: Metformin (after food), low-carb diet, daily 30-min walk, blood sugar monitoring.
        - Note: Consult your doctor before starting any medication.
4. **Health Analytics**
    - **Function:** Analyzes basic data to provide insights.
    - **Example Inputs:** Age, weight, height
    - **Outputs:** BMI, risk level, general recommendations (e.g., risk of heart disease, suggestions to reduce it).

3.8 Sample Use Scenarios

- **Scenario 1: Chat Query**

  User: What are the early signs of dengue?

  AI: The early symptoms of dengue include high fever, severe headache, pain behind the eyes, joint and muscle pain, fatigue, and skin rash.

- **Scenario 2: Disease Prediction**

  User: I have fever, chills, and sweating.

  AI: Based on your symptoms, you might be experiencing Malaria. Please consult a medical professional for testing and confirmation.

- **Scenario 3: Personalized Treatment Plan**

  Condition: Asthma

  Patient Info: Age 35, Male, No comorbidities

  AI: Suggested Plan – Use an inhaler with salbutamol for relief, avoid allergens, take prescribed corticosteroids daily, and monitor breathing using a peak flow meter.

- **Scenario 4: Health Analytics**

  Input: Age: 45, Weight: 90 kg, Height: 1.6 m

  AI: Your BMI is 35.2 (Obese). You are at increased risk for heart disease, type 2 diabetes, and hypertension. Recommended: Weight loss, exercise, and dietary modifications.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

HealthAI is a significant healthcare assistance.

- **24/7 Availability:** Always ready to respond to queries.
- **Multilingual Support:** Users can interact in regional languages.
- **Fast and Accurate:** Uses IBM's powerful Granite LLM for real-time responses.
- **User-Friendly Interface:** Streamlit provides a simple and clean user interface.

4.2 Future Work

- Integrate with hospital databases and EHRs for better recommendations.
- Add voice interaction using speech recognition and TTS engines.
- Provide real-time doctor consultation scheduling.
- Expand medical database for rural and regional disease coverage.

4.3 References

• IBM Granite Models – ibmcloud