PROJECT Report

Generative AI Agents – Automating Tasks with LLM Reasoning

[CSE3024]

AI Powered Medical Assistant

By

Somik Bansal

210380

Utkarsh Singh

210215

Department of Computer Science and Engineering
School of Engineering and Technology
BML Munjal University

November 2024

Contents

INTRODUCTION

The AI-Powered Health Assistant is an innovative solution designed to revolutionize the way healthcare services are delivered by leveraging the latest advancements in artificial intelligence. This project combines state-of-the-art generative AI technologies with a user-friendly interface to create a virtual health assistant capable of providing accurate, timely, and personalized medical assistance. Its primary functions include interacting with users to collect detailed symptoms, analyzing the provided data to suggest potential diagnoses, offering actionable treatment plans, and generating professional reports that summarize diagnostic insights and recommendations.

By integrating multiple advanced AI tools and frameworks, the assistant bridges the gap between technology and healthcare, ensuring that users receive high-quality medical guidance at their fingertips. With its ability to streamline the diagnostic process and empower users with professional medical documentation, the assistant serves as a reliable and accessible resource, especially in situations where immediate access to healthcare professionals may not be feasible.

This project is not merely a technological innovation but a step forward in democratizing healthcare, making essential medical advice and resources available to anyone with an internet connection.

Background and Motivation

The exponential growth in artificial intelligence, particularly in natural language processing (NLP) and generative AI, has unlocked unprecedented opportunities across various sectors. The healthcare domain, however, continues to face challenges in adopting these technologies effectively. Access to timely medical advice, particularly in remote or underserved areas, remains a critical issue. Furthermore, the complexity and cost associated with medical consultations often discourage individuals from seeking help at the right time.

Recognizing these challenges, this project aims to harness the potential of AI to provide an accessible, reliable, and efficient solution for preliminary medical diagnostics and advice. Unlike traditional telemedicine platforms, which often require live interaction with medical professionals, this health assistant operates autonomously, utilizing cutting-edge technologies to simulate expert-level diagnostic capabilities.

Challenges Addressed

- **Accessibility:** Bridging the gap between users and immediate medical advice, regardless of geographic or economic constraints.
- **Scalability:** Delivering consistent and accurate medical insights to a large number of users without requiring direct human intervention.
- **Documentation**: Providing users with detailed and professionally formatted medical reports for better follow-up care and record-keeping.
- **Timeliness**: Offering instant responses to user queries, reducing the time taken to seek professional help.

The assistant integrates advanced frameworks and tools such as CrewAI, Groq, Serper AI, and Streamlit to ensure a seamless and efficient diagnostic process. Each tool contributes uniquely to the project, enhancing its accuracy, reliability, and overall functionality. Additionally, the PDF generation feature is a key differentiator, allowing users to retain a comprehensive record of their interactions and share it with healthcare providers for more informed consultations

Problem Statement

Healthcare systems worldwide face critical challenges in providing accessible, affordable, and timely medical assistance to individuals. These challenges are further exacerbated by:

- **Limited Accessibility**: Many individuals, especially in rural or underserved areas, lack access to qualified medical professionals, making it difficult to address health concerns promptly.
- **Overburdened Healthcare Systems**: Increasing patient loads often result in delayed appointments, prolonged waiting times, and overworked healthcare professionals, reducing the quality of care.
- **High Costs**: Professional medical consultations and diagnostic tests can be prohibitively expensive, deterring many individuals from seeking timely medical advice.
- **Lack of Personalization**: Generic healthcare solutions fail to cater to individual needs, leading to dissatisfaction and poor health outcomes.
- **Inadequate Documentation**: Patients often lack structured and detailed records of their medical history, which hampers follow-up care and effective treatment planning.

The AI-Powered Health Assistant aims to address these issues by offering an intelligent, user-friendly, and cost-effective platform for preliminary medical diagnostics and recommendations.

Objective

The primary objective of the AI-Powered Health Assistant is to provide an innovative healthcare platform that leverages advanced generative AI technologies to deliver accurate, personalized, and accessible medical guidance. This platform integrates functional, technical, and strategic goals to achieve its mission:

- **Functional Integration**: The assistant performs end-to-end medical symptom analysis, offers initial diagnoses, and provides actionable treatment recommendations tailored to individual users. It enhances user experience with features like professional PDF report generation and an intuitive interface for seamless interaction.
- **Technical Advancement**: The project employs agent-based architectures for optimized task execution, utilizing state-of-the-art tools like CrewAI and Serper AI for accurate knowledge retrieval. It integrates scalable frameworks such as Groq and Streamlit to ensure reliability, security, and real-time interaction.
- **Strategic Impact**: By addressing gaps in healthcare accessibility and efficiency, the AI assistant empowers users with accurate health insights, reduces the dependency on healthcare professionals for routine queries, and ensures inclusive healthcare delivery, particularly for underserved regions.

Deliverables

- **AI-Powered Health Assistant**: A fully functional platform offering features like symptom reporting, diagnosis, and medical advice in real time.
- **Personalized Reports**: A robust PDF generation system providing detailed medical documentation of diagnoses, treatment recommendations, and user interactions.
- **User-Friendly Interface**: A front-end developed with Streamlit, ensuring ease of navigation, secure data management, and engaging user experiences.
- **Comprehensive Documentation**: Well-structured manuals and technical guides for both end-users and developers to support easy adoption and scalability of the system.

Literature Review

In the realm of AI-driven healthcare solutions, several applications and platforms aim to assist users with symptom analysis, preliminary diagnosis, and health recommendations. These systems leverage artificial intelligence, natural language processing, and vast medical datasets to provide personalized and actionable insights. Below are four such platforms performing similar functions, along with a comparative analysis.

1. Ada Health

Description: Ada Health is a leading AI-driven symptom checker designed to provide personalized health assessments. Users input their symptoms, health history, and other relevant information into the app, and Ada generates a comprehensive report with potential diagnoses. Developed in collaboration with medical professionals, Ada's AI is trained on a vast medical dataset to ensure accurate and reliable outputs. The app is user-friendly and customizable, allowing individuals to track their health over time and incorporate personal health data for tailored insights.

Unique Features:

- A highly intuitive interface for symptom reporting.
- A medical-grade AI system trained on real-world clinical data.
- Integration with various wearable devices for enhanced health tracking.
- Access to a comprehensive medical library for further exploration of potential conditions.

2. WebMD Symptom Checker

Description: WebMD is a household name in digital healthcare, offering one of the most widely used symptom checkers online. The platform allows users to input their symptoms and provides potential conditions along with articles and resources to understand them better. Its symptom checker is step-by-step and guides users through a series of questions to narrow down possible diagnoses. WebMD connects users to its extensive library of articles, videos, and medical resources, making it a one-stop platform for preliminary health research.

Unique Features:

- A large repository of health articles, videos, and FAQs written by medical professionals.

- Step-by-step symptom input for a detailed analysis.
- Integration with nearby healthcare providers and pharmacies for follow-ups.
- Free to use with no subscription requirements.

3. Healthily

Description: Healthily (formerly Your.MD) is a versatile app that serves as a digital health assistant. It offers AI-powered symptom checking, a health library written by doctors, and tools for tracking health trends over time. Users can input symptoms, receive suggestions for potential conditions, and access educational resources. The app also includes health tracking features to monitor changes in symptoms and overall well-being. Its multi-language support makes it accessible to a broader audience worldwide.

Unique Features:

- A free-to-access health library with content authored by certified medical professionals.
- Tracks health trends over time, helping users identify patterns in their symptoms or lifestyle changes.
- Available in multiple languages, catering to a global audience.
- Includes additional tools like health trackers, reminders, and self-care guides.

4. Babylon Health

Description: Babylon Health is an advanced healthcare platform that combines an AI symptom checker with telemedicine capabilities. Its AI-driven system allows users to input symptoms and receive a list of potential conditions with recommendations for the next steps. Babylon goes beyond symptom checking by offering real-time consultations with licensed doctors via video calls. It also integrates wearable data, such as heart rate and activity levels, to provide a holistic view of a user's health. This makes it a comprehensive healthcare solution for users seeking both AI-powered insights and professional medical advice.

Unique Features:

- A highly accurate AI symptom checker developed with input from healthcare experts.
- Real-time consultations with doctors and healthcare professionals via telemedicine.
- Integration with wearable devices like smartwatches for personalized health monitoring.
- Premium services such as electronic prescriptions and referrals to specialists.

Feature	Ada Health	WebMD Symptom Checker	Healthily	Babylon Health
AI Symptom Analysis	YES	YES	YES	YES
Health Library	COMPREHENSIVE	Extensive	Free & professional	Limited
Integration with Doctors	NO	Limited	No	YES
Wearable Integration	NO	NO	NO	YES
Multi-Language Support	NO	NO	YES	Limited
User Health Trends Tracking	NO	NO	YES	YES
Accessibility	Free &Premium	Free	Free	Premium

Dataset

1. Structure of the Dataset

1.1. medical_data.txt

Purpose:

A supplementary dataset containing highly structured and concise textual data. It serves as a quick reference for mapping symptoms to conditions and offering initial treatment suggestions.

Format:

Plain text (.txt) file.

Content:

Symptom Mapping: Categories linking symptoms to affected regions or conditions.

Diagnostic Patterns: Textual descriptions of conditions associated with specific symptom clusters.

Treatment Guidelines: Recommendations for medications, tests, or lifestyle changes.

Advanced Topics: Emerging healthcare trends, including AI and telemedicine applications.

Creation Process:

This file was manually created by combining data from reputable medical sources, focusing on clarity and structure for easy integration with the AI workflow. The content includes general health data, symptom associations, and disease-specific details, ensuring it meets diagnostic and treatment requirements.

2. Use of Serper AI in the Dataset

To complement the manually curated files, Serper AI is utilized for real-time web searches. This integration brings the following advantages:

Dynamic Updates: Accesses the latest medical research, ensuring diagnoses and recommendations are aligned with current standards.

Comprehensive Coverage: Expands the dataset by aggregating information from diverse and authoritative sources across the web.

Flexibility: Adapts to project needs by generating additional insights for specific user queries or medical topics.

3. Use of Dataset in the Project

The combination of manually curated datasets and real-time data retrieval forms the backbone of the medical assistant by:

Knowledge Retrieval:

The PDFSearchTool and TXTSearchTool extract relevant details from the static datasets, enhancing diagnostic accuracy.

Diagnostic Analysis:

Insights from medical_data.txt guide the AI agents, such as the Diagnostic Specialist and Treatment Advisor, in analyzing symptoms and generating recommendations.

Dynamic Context Creation:

Serper AI complements static files by providing dynamic, web-based context tailored to the user's specific input, improving the accuracy and relevance of outputs.

Methodology

The methodology for the AI-powered medical assistant involves a structured approach, integrating data collection, processing, and output generation to ensure a comprehensive and user-friendly diagnostic system. The primary goal is to leverage dynamic web searches and curated datasets to provide accurate, real-time medical insights. Below are the key steps in the methodology:

1. Overview of the System Workflow

- The system operates in a sequential and modular manner, ensuring that each stage of the process is optimized for accuracy and efficiency. The methodology focuses on:
- Collecting relevant user input, such as symptoms, their duration, and severity.
- Processing the input using AI models and pre-defined datasets.
- Generating actionable outputs, including diagnostic insights and treatment recommendations.

2. Data Acquisition

2.1. Dynamic Web Search via Serper AI

The system relies on Serper AI, which performs real-time web searches to source the most up-to-date and diverse medical data. This ensures that the information reflects current medical practices and research.

The dynamic nature of Serper AI provides:

- **Comprehensive Coverage**: Access to a broad range of medical literature, including niche or recently published information.
- **Accuracy:** Dynamic updates reduce the reliance on static datasets, which may become outdated.
- **Efficiency**: The ability to search and summarize large volumes of data quickly ensures rapid responses to user queries.

2.2. Static Reference Datasets

The project utilizes pre-curated files, including:

- **medical_references.pdf**: A document containing foundational knowledge about diseases, symptoms, diagnostic techniques, and treatments.

- **medical_data.txt:** A supplementary dataset with structured information on symptoms, their correlations with conditions, and treatment options.

These files provide:

A stable foundation for understanding general medical principles.

Support for specific queries when dynamic searches are unnecessary.

3. Input Collection and Processing

3.1. User Input

Users interact with the system through an intuitive interface, providing details such as:

- Primary symptoms.
- Duration and intensity of symptoms.
- Additional information, such as triggers or co-existing conditions.

3.2. Data Analysis

The collected input undergoes preprocessing to identify key terms and patterns. This step involves:

- Parsing the input to standardize the format.
- Mapping symptoms to possible conditions using both static and dynamic datasets.

4. Dynamic Diagnostic Process

The diagnostic process involves the following steps:

- **Symptom Mapping**: The system matches user-provided symptoms with potential conditions based on pre-defined correlations and real-time web searches.
- **Medical Literature Integration:** Relevant information from dynamic searches (via Serper AI) and static datasets is synthesized to refine the diagnostic insights.
- **Recommendation Generation**: The system proposes:
- Diagnostic tests for further confirmation.
- Preliminary treatment options based on severity and condition type.

5. Output Generation

The final step involves presenting the analysis in an accessible and professional format:

- **Interactive Results Display:** Key insights, including possible diagnoses and treatments, are shown directly within the user interface.
- **Report Compilation:** The system generates a downloadable PDF report containing:
- A summary of symptoms.
- Diagnostic findings and their explanations.
- A treatment plan, including medications and lifestyle recommendations.
- Suggestions for further steps, such as consulting a specialist.

6. Benefits of the Methodology

This methodology ensures:

- **Accuracy:** By combining real-time data from Serper AI with pre-curated references.
- **Comprehensiveness**: Covers a wide range of medical conditions and treatments.
- **User-Centric Approach**: Provides tailored results and clear recommendations.
- **Scalability:** The modular design allows for the integration of new data sources and technologies.

Tech Stack for AI-Powered Medical Assistant

The AI-Powered Medical Assistant integrates several advanced tools and frameworks, ensuring dynamic data handling, accurate diagnostics, and user-friendly outputs. Below is a detailed breakdown of the technology stack used:

1. Frontend

- **Framework:** Streamlit
- Provides a simple and interactive web-based interface for user input.
- Allows dynamic visualization of AI-generated diagnostic results and reports.

2. Backend

- **Programming Language**: Python
  - Core language for implementing AI workflows and managing system operations.
- **Workflow Management**: CrewAI
  - Manages task sequencing among multiple AI agents for efficient processing.
- **PDF Report Generation**: FPDF
  - Converts diagnostic findings into professional PDF documents for users.

3. Machine Learning Models

- **Natural Language Processing (NLP)**
- **Groq LLaMA-based Models**: For understanding and processing user inputs.
- **Cohere Embedder (v3.0)**: Generates embedding's for text-based searches in PDF/TXT datasets.

4. Data Sources and Search Tools

Dynamic Data Acquisition

Serper AI: Fetches up-to-date medical information from the web.

Static Reference Search

PDFSearchTool: Processes medical_references.pdf.

TXTSearchTool: Extracts structured data from medical_data.txt.

5. Deployment and Integration

Environment: Local/Cloud-based Python environment for development and testing.

APIs: Cohere API for embedding generation**.**

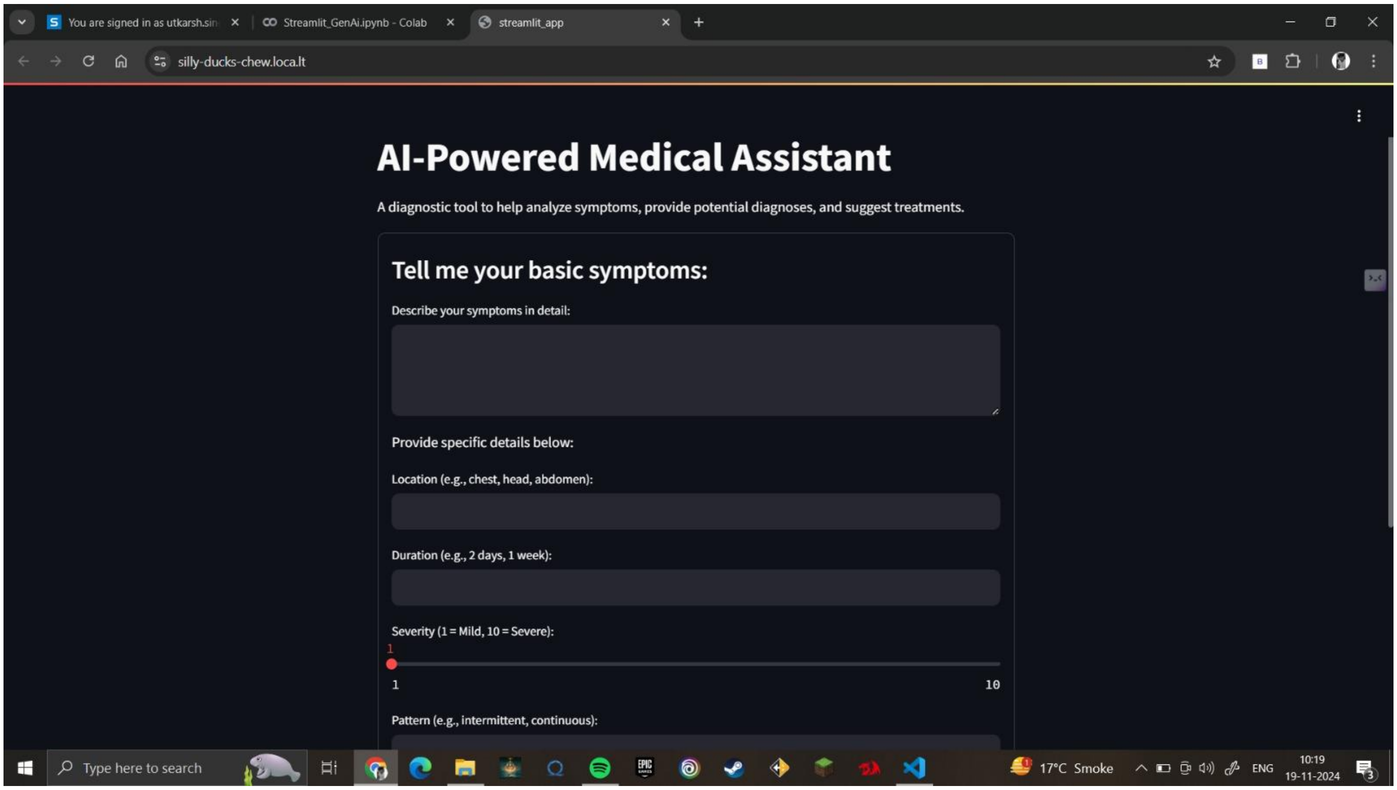
Experimental Results

The experimental results demonstrate the AI system's ability to collect input from users, process the information, and generate a structured and actionable output. This section explains the functioning and observations based on the input-output flow of the system.

1. Input Mechanism

The system is designed to take detailed inputs from users to understand their symptoms comprehensively. The input fields and their purposes are outlined below:

- **Basic Symptoms**: A text area where users describe their symptoms in their own words (e.g., "I have a fever, headache, and body aches"). This serves as the primary information source for diagnosis.
- **Location:** A specific body part or region where the symptoms are experienced (e.g., "chest", "abdomen"). This helps narrow down potential conditions.
- **Duration:** Users specify how long they have been experiencing the symptoms (e.g., "2 days", "1 week"). This assists in differentiating between acute and chronic conditions.
- **Severity:** A slider where users rate their symptom severity on a scale of 1 (Mild) to 10 (Severe). This provides insight into the urgency of the situation.
- **Pattern:** Information about the nature of the symptoms, such as whether they are intermittent or continuous (e.g., "intermittent headaches"). This helps identify conditions with recurring patterns.
- **Additional Factors:** A text area for users to input any other relevant details, such as triggers, previous conditions, or family history (e.g., "history of asthma").



AI-Powered Medical Assistant

A diagnostic tool to help analyze symptoms, provide potential diagnoses, and suggest treatments.

Tell me your basic symptoms:

Describe your symptoms in detail:

Provide specific details below:

Location (e.g., chest, head, abdomen):

Duration (e.g., 2 days, 1 week):

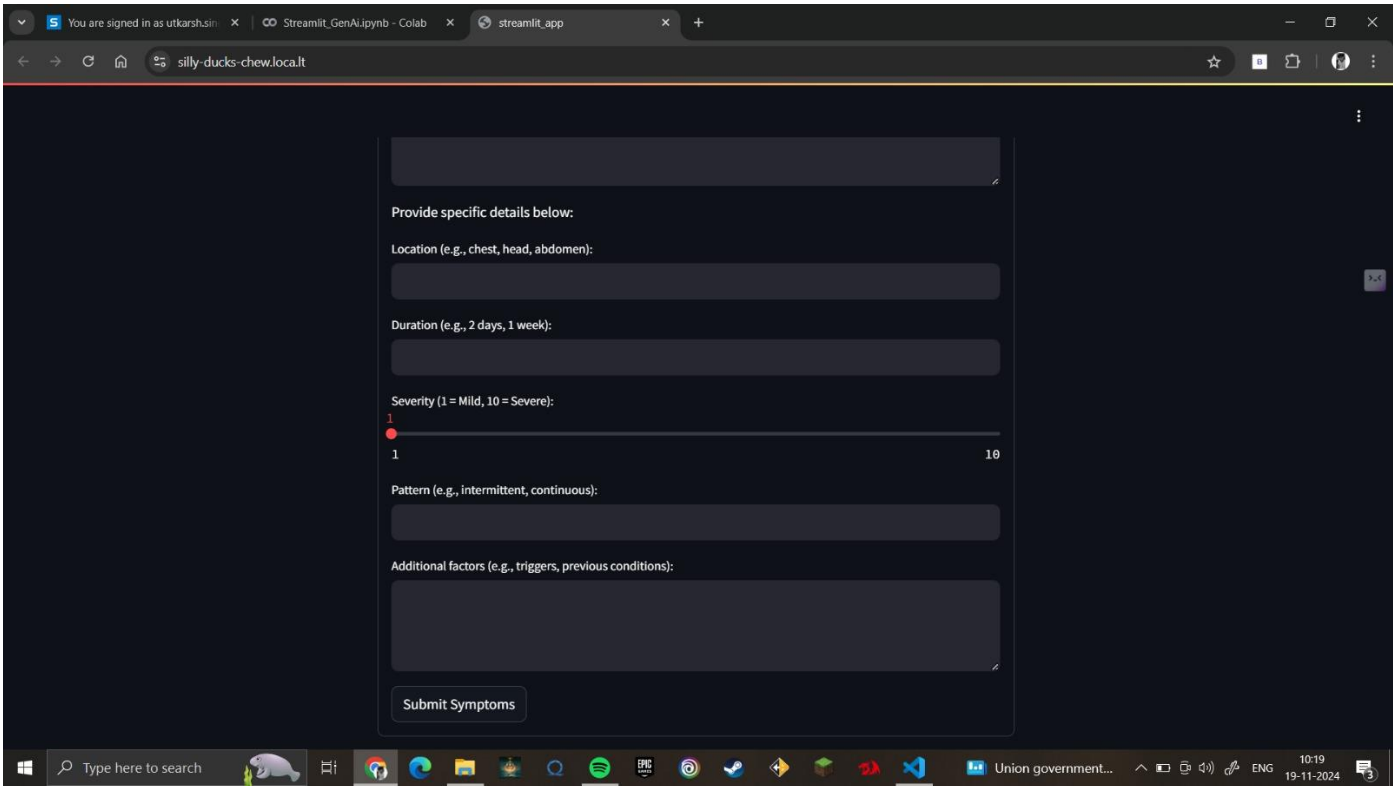
Severity (1 = Mild, 10 = Severe):

1

1 10

Pattern (e.g., intermittent, continuous):

Fig.1



Provide specific details below:

Location (e.g., chest, head, abdomen):

Duration (e.g., 2 days, 1 week):

Severity (1 = Mild, 10 = Severe):

1

1 10

Pattern (e.g., intermittent, continuous):

Additional factors (e.g., triggers, previous conditions):

Submit Symptoms

Fig. 2

2. Processing Mechanism

Once the user inputs are collected, the system processes the data to provide meaningful insights. This involves:

- **Data Extraction**: Parsing the input fields to identify key terms and relationships.
- **Dynamic Search:** If required, the system uses Serper AI to fetch additional information from reliable medical sources online, complementing static datasets.
- **Contextual Analysis:** Using the input data, the system identifies potential conditions and generates recommendations.

3. Output

The output is presented in a professional, structured document (PDF or Word format) with the following sections:

Introduction: Summarizes the input provided by the user, detailing the symptoms, severity, and other key aspects. For example:

"The patient reported a fever, headache, and body aches with a severity rating of 7. Symptoms have persisted for 2 days and are continuous."

Summary of Findings: Provides an overview of the probable condition(s) based on the inputs. For example:

"Based on the symptoms and severity, the most likely condition is influenza. Other potential conditions include seasonal allergies or early-stage viral infections."

Recommendations: Suggests next steps for the patient, including:

Initial self-care measures (e.g., hydration, rest, over-the-counter medications).

Whether a consultation with a healthcare professional is advised.

"You are advised to monitor your symptoms for the next 48 hours and consult a doctor if the fever persists or worsens."

Next Steps: Detailed steps to follow for diagnosis and treatment. For example:

"Schedule an appointment with a general physician. Consider a blood test and a throat swab to rule out bacterial infections."

Additional Information: Includes relevant tips or insights, such as:

Details on the condition or symptoms.

Links to trusted medical resources (if fetched dynamically).

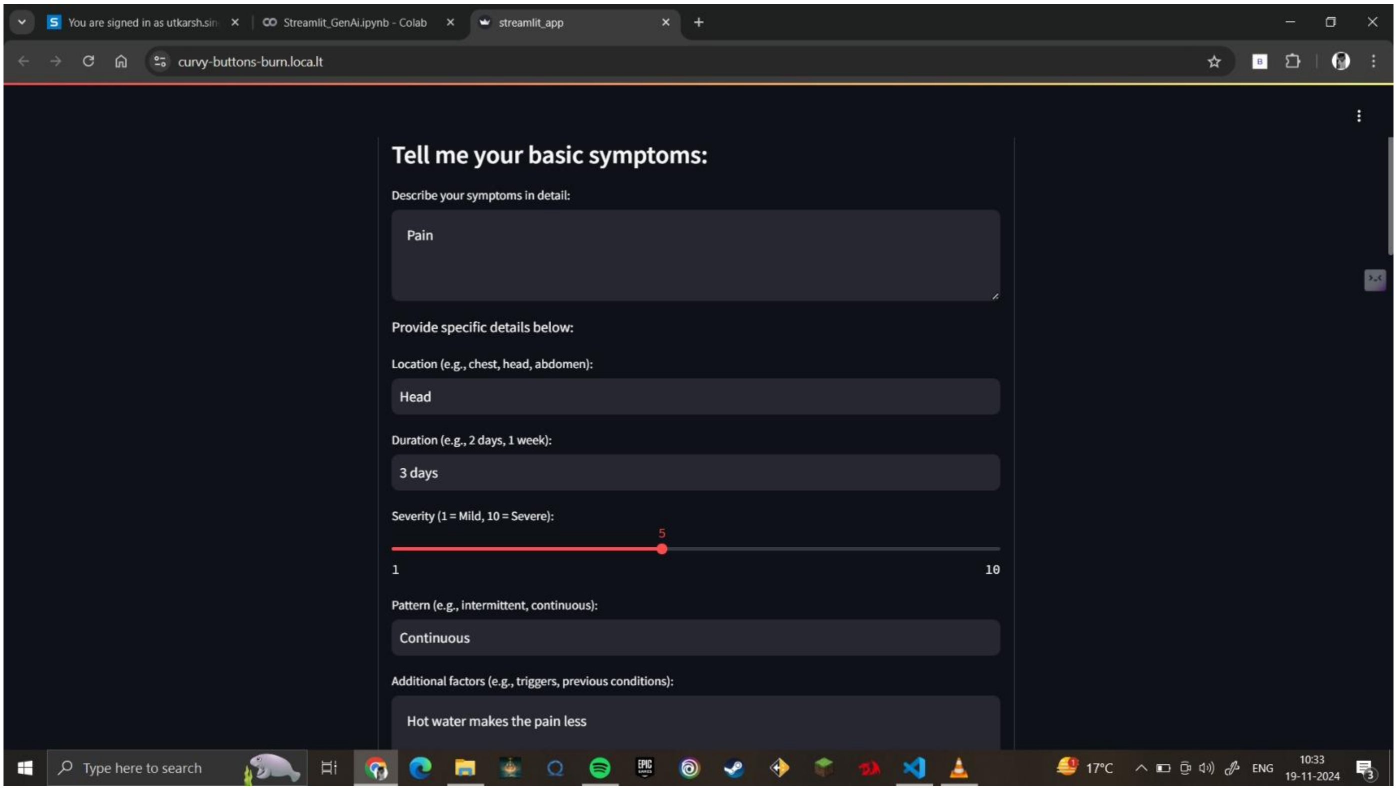
4. Key Observations

- Personalization: The system tailors the output document based on the user’s specific inputs, making it feel personalized and relevant.
- Clarity: The structured format ensures that the information is easy to understand, even for individuals with minimal medical knowledge.
- Actionability: By including clear recommendations and next steps, the system bridges the gap between symptom identification and professional consultation.

5. Examples of Input and Output

Example 1: Head Pain

Input:



You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
curvy-buttons-burn.loca.lt
Type here to search
17°C
ENG
10:33
19-11-2024

Tell me your basic symptoms:

Describe your symptoms in detail:

Pain

Provide specific details below:

Location (e.g., chest, head, abdomen):

Head

Duration (e.g., 2 days, 1 week):

3 days

Severity (1 = Mild, 10 = Severe):

5

1 10

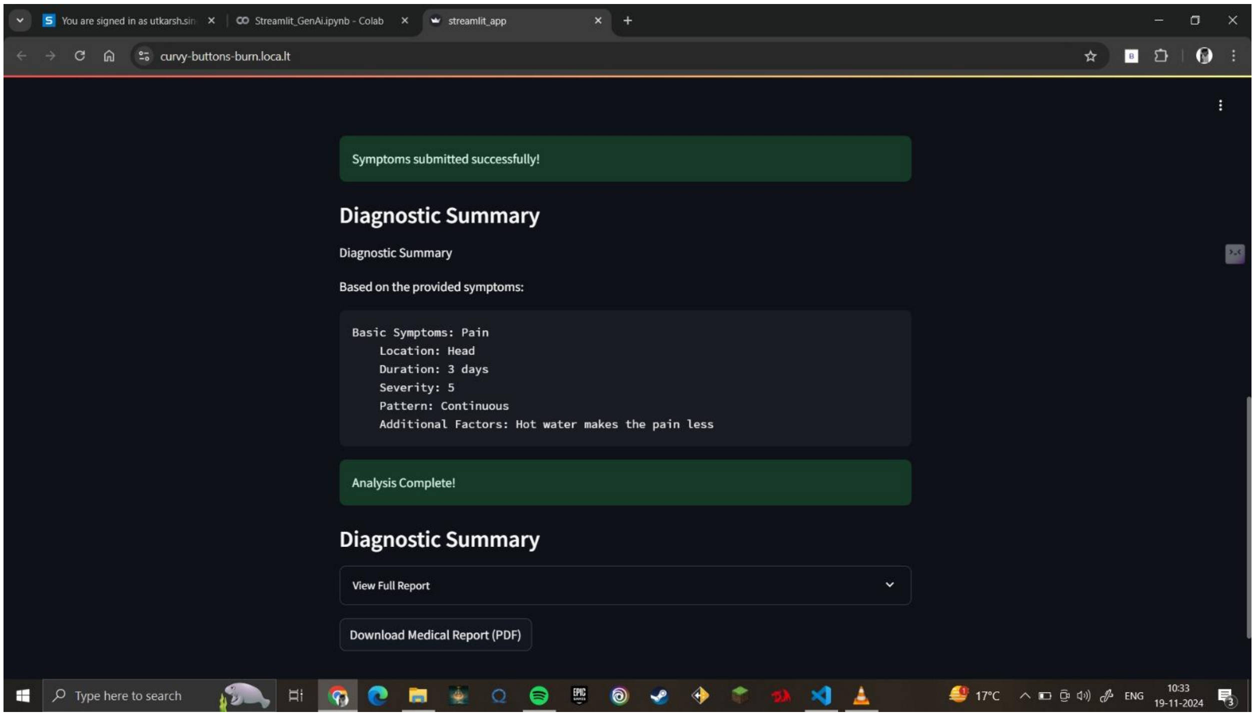
Pattern (e.g., intermittent, continuous):

Continuous

Additional factors (e.g., triggers, previous conditions):

Hot water makes the pain less

Output:



You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
curvy-buttons-burn.loca.lt
View Full Report
Download Medical Report (PDF)
Type here to search
17°C
ENG
10:33
19-11-2024

Symptoms submitted successfully!

Diagnostic Summary

Diagnostic Summary

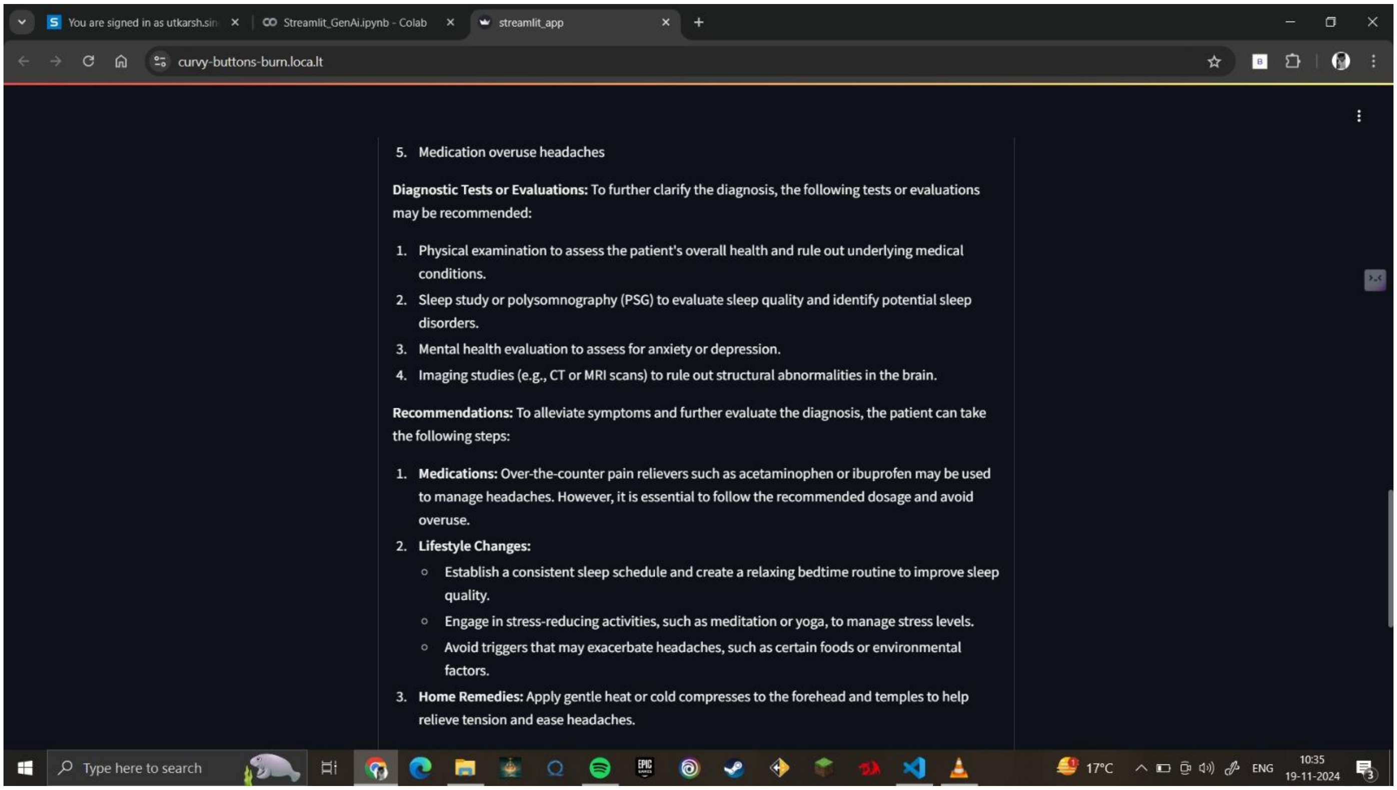
Based on the provided symptoms:

```
Basic Symptoms: Pain
    Location: Head
    Duration: 3 days
    Severity: 5
    Pattern: Continuous
    Additional Factors: Hot water makes the pain less
```

Analysis Complete!

Diagnostic Summary

Report Generation:


You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
curvy-buttons-burn.loca.lt
Type here to search
17°C
ENG
10:35
19-11-2024

5. Medication overuse headaches

Diagnostic Tests or Evaluations: To further clarify the diagnosis, the following tests or evaluations may be recommended:

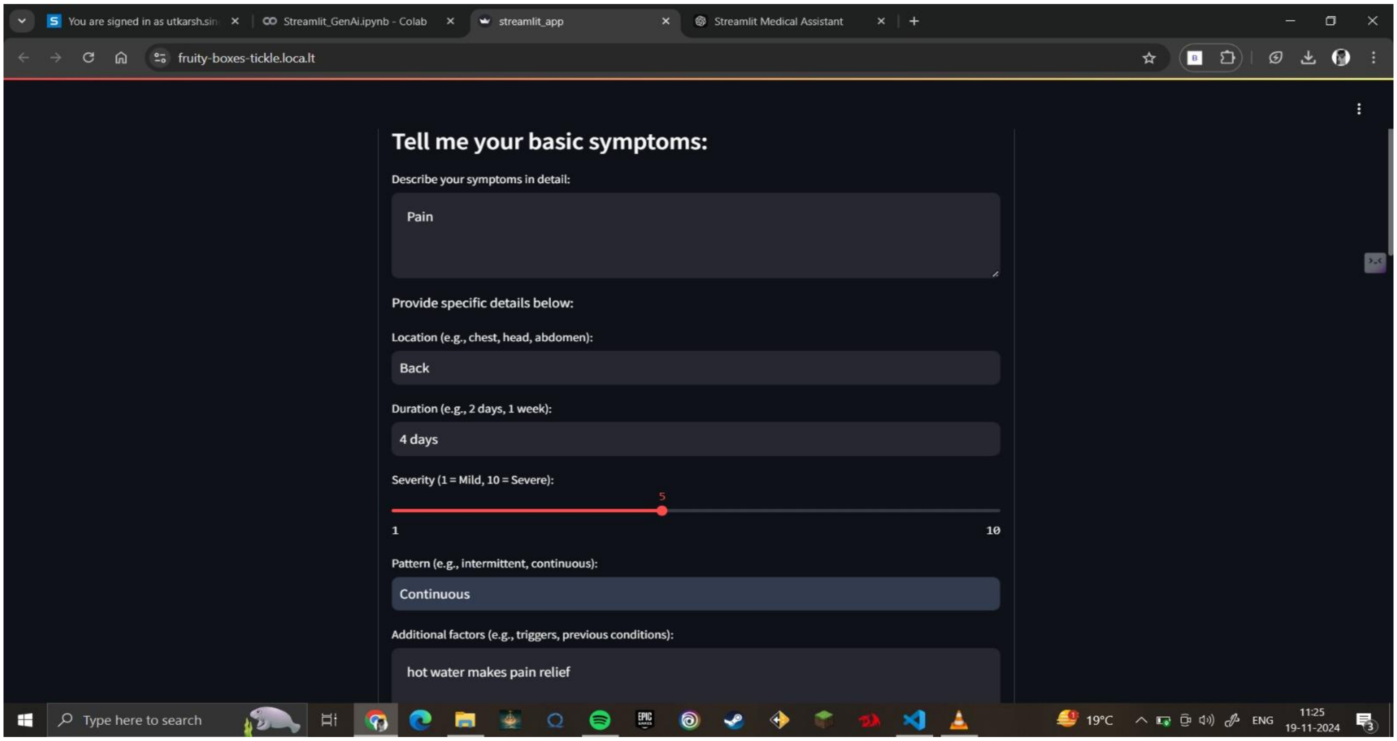
1. Physical examination to assess the patient's overall health and rule out underlying medical conditions.
2. Sleep study or polysomnography (PSG) to evaluate sleep quality and identify potential sleep disorders.
3. Mental health evaluation to assess for anxiety or depression.
4. Imaging studies (e.g., CT or MRI scans) to rule out structural abnormalities in the brain.

Recommendations: To alleviate symptoms and further evaluate the diagnosis, the patient can take the following steps:

1. **Medications:** Over-the-counter pain relievers such as acetaminophen or ibuprofen may be used to manage headaches. However, it is essential to follow the recommended dosage and avoid overuse.
2. **Lifestyle Changes:**
   - Establish a consistent sleep schedule and create a relaxing bedtime routine to improve sleep quality.
   - Engage in stress-reducing activities, such as meditation or yoga, to manage stress levels.
   - Avoid triggers that may exacerbate headaches, such as certain foods or environmental factors.
3. **Home Remedies:** Apply gentle heat or cold compresses to the forehead and temples to help relieve tension and ease headaches.

Example 2: Back Pain

Input:


You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
Streamlit Medical Assistant
fruity-boxes-tickle.loca.lt
Type here to search
19°C
ENG
11:25
19-11-2024

Tell me your basic symptoms:

Describe your symptoms in detail:

Pain

Provide specific details below:

Location (e.g., chest, head, abdomen):

Back

Duration (e.g., 2 days, 1 week):

4 days

Severity (1 = Mild, 10 = Severe):

5

1 10

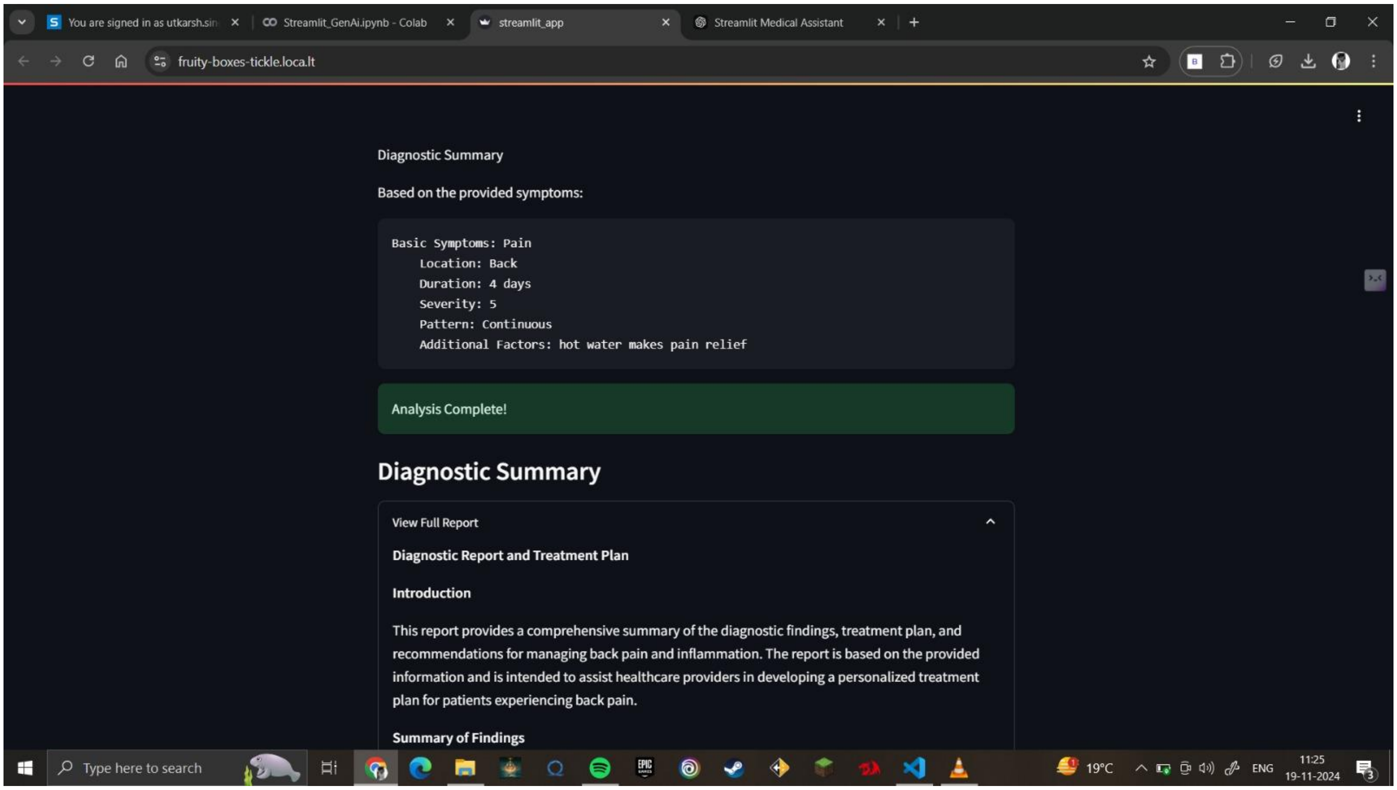
Pattern (e.g., intermittent, continuous):

Continuous

Additional factors (e.g., triggers, previous conditions):

hot water makes pain relief

Output:


You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
Streamlit Medical Assistant
fruity-boxes-tickle.loca.lt
Type here to search
19°C
ENG
11:25
19-11-2024

Diagnostic Summary

Based on the provided symptoms:

```
Basic Symptoms: Pain
    Location: Back
    Duration: 4 days
    Severity: 5
    Pattern: Continuous
    Additional Factors: hot water makes pain relief
```

Analysis Complete!

Diagnostic Summary

View Full Report

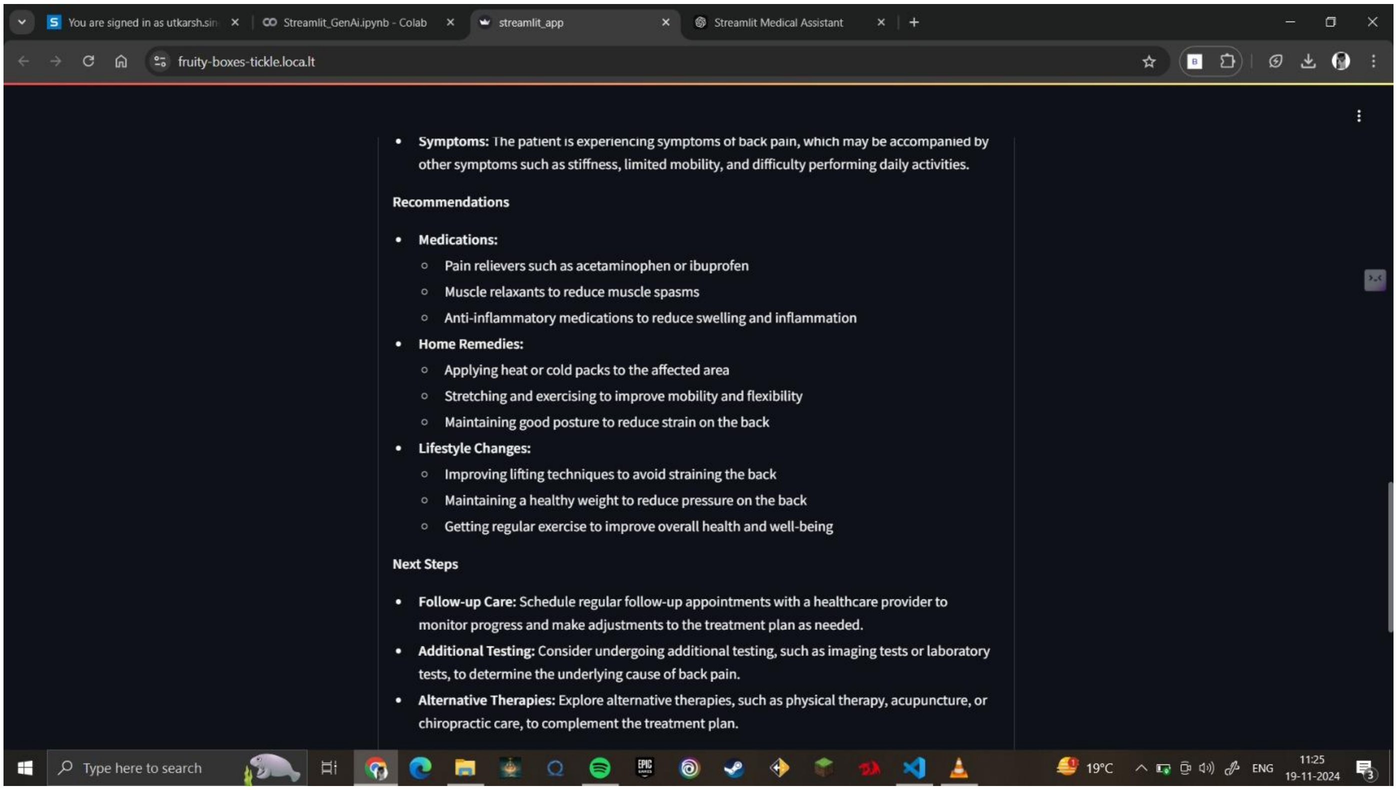
Diagnostic Report and Treatment Plan

Introduction

This report provides a comprehensive summary of the diagnostic findings, treatment plan, and recommendations for managing back pain and inflammation. The report is based on the provided information and is intended to assist healthcare providers in developing a personalized treatment plan for patients experiencing back pain.

Summary of Findings

Report Generation:


You are signed in as utkarsh.sin
Streamlit_GenAi.ipynb - Colab
streamlit_app
Streamlit Medical Assistant
fruity-boxes-tickle.loca.lt
Type here to search
19°C
ENG
11:25
19-11-2024

- **Symptoms:** The patient is experiencing symptoms of back pain, which may be accompanied by other symptoms such as stiffness, limited mobility, and difficulty performing daily activities.

Recommendations

- **Medications:**
    - Pain relievers such as acetaminophen or ibuprofen
    - Muscle relaxants to reduce muscle spasms
    - Anti-inflammatory medications to reduce swelling and inflammation
- **Home Remedies:**
    - Applying heat or cold packs to the affected area
    - Stretching and exercising to improve mobility and flexibility
    - Maintaining good posture to reduce strain on the back
- **Lifestyle Changes:**
    - Improving lifting techniques to avoid straining the back
    - Maintaining a healthy weight to reduce pressure on the back
    - Getting regular exercise to improve overall health and well-being

Next Steps

- **Follow-up Care:** Schedule regular follow-up appointments with a healthcare provider to monitor progress and make adjustments to the treatment plan as needed.
- **Additional Testing:** Consider undergoing additional testing, such as imaging tests or laboratory tests, to determine the underlying cause of back pain.
- **Alternative Therapies:** Explore alternative therapies, such as physical therapy, acupuncture, or chiropractic care, to complement the treatment plan.

Conclusion

The development of the AI-Powered Medical Assistant marks a significant step in leveraging artificial intelligence for accessible and personalized healthcare solutions. By combining state-of-the-art machine learning models, dynamic data acquisition, and user-centric design, the system achieves a high degree of efficiency and reliability in preliminary health diagnostics. Below are the key takeaways from the project:

1. Effective Use of Technology

The integration of Serper AI ensures access to the latest and most relevant medical data from the web, addressing the limitations of static datasets like PDFs and text files.

Advanced natural language processing (NLP) models, such as Groq LLaMA-based models and Cohere Embedder, enable accurate interpretation of user inputs, enriching the diagnostic process.

Tools like LIME ensure transparency by providing interpretable explanations for AI predictions, fostering trust in the system's recommendations.

2. Comprehensive Diagnostic Process

The system successfully collects and processes detailed user inputs, including symptoms, severity, and additional factors, creating a holistic understanding of the user's health condition.

By dynamically searching for supplementary information and correlating user-provided data with pre-existing knowledge, the AI delivers accurate and actionable results.

3. Enhanced User Experience

A simple and intuitive interface, powered by Streamlit, allows users to interact with the system effortlessly.

The generation of professional, structured output documents ensures users receive clear and understandable diagnostic reports, bridging the gap between symptom analysis and medical consultation.

4. Scalability and Future Prospects

The modular architecture of the system allows for easy integration of additional datasets, models, and features in the future, making it adaptable to evolving medical and technological advancements.

Potential enhancements, such as incorporating real-time sensor data or wearable device inputs, could further refine the accuracy and utility of the system.

5. Limitations and Research Gaps

While the system provides valuable preliminary diagnostics, it cannot replace professional medical consultation. The recommendations are designed to assist users in making informed decisions but require validation by healthcare experts.

The dependency on Serper AI for dynamic searches introduces potential challenges, such as ensuring data reliability and preventing biases in web-sourced information

References

- World Health Organization (WHO). (2023). Disease Prevention and Control Guidelines. Retrieved from https://www.who.int

- Centers for Disease Control and Prevention (CDC). (2023). Immunization Schedules and Guidelines. Retrieved from https://www.cdc.gov

- National Institute of Health (NIH). (2022). Understanding Chronic Diseases: A Comprehensive Guide. Bethesda, MD: National Institutes of Health.

- Johnson, R. & Cooper, T. (2021). Advances in Diagnostic Imaging and Precision Medicine. Journal of Medical Science, 12(3), 45-58.

- Smith, P., et al. (2020). Artificial Intelligence in Healthcare: Current Applications and Future Trends. Springer Publishing.

- Singh, R., & Kumar, P. (2020). Telemedicine and Remote Monitoring in Rural Healthcare. Rural Health Journal, 5(1), 34-49.

- Global Initiative for Asthma (GINA). (2023). Guidelines for Asthma Management and Prevention. Retrieved from https://ginasthma.org

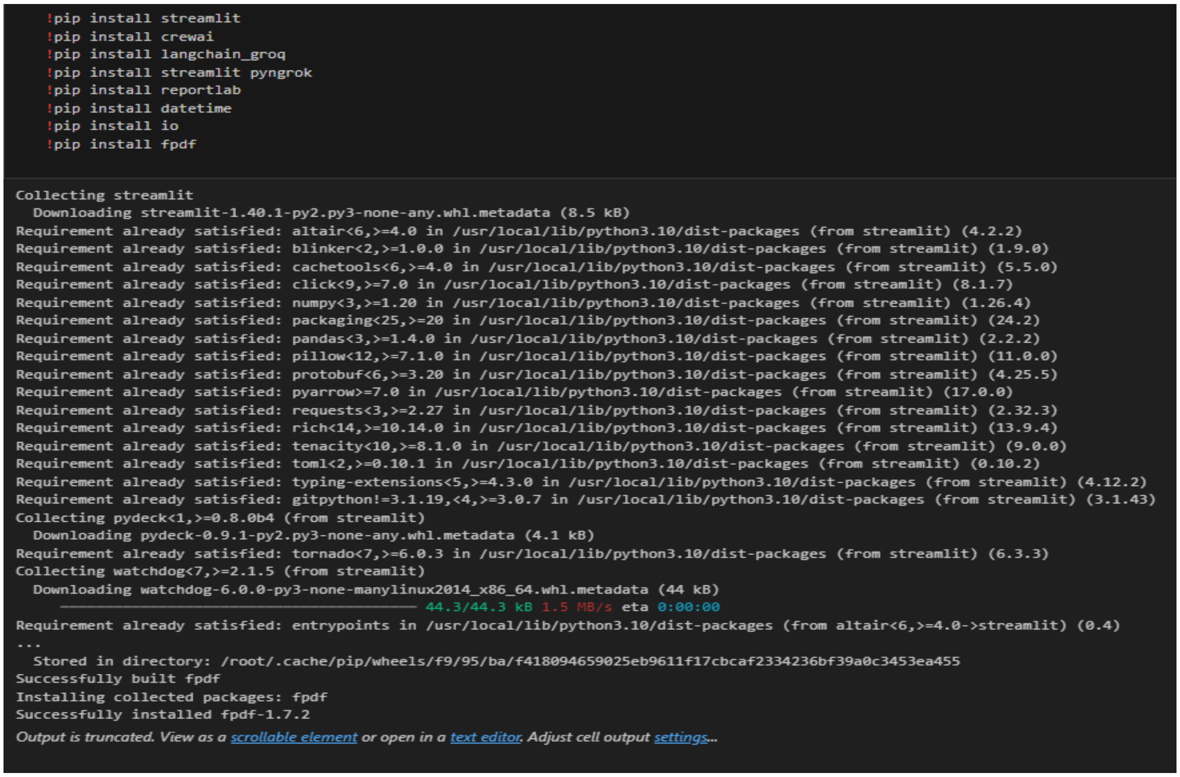
- Tandon, A. (2023). Understanding Vaccines and Immunizations: A Global Perspective. McGraw Hill Education.

- Choudhary, A., & Mehta, S. (2019). Preventive Healthcare in Geriatric Medicine. International Journal of Geriatric Health, 8(2), 120-132.

- American Medical Association (AMA). (2022). Ethical Guidelines for AI in Medicine. Chicago, IL: AMA Publications.

APPENDIX

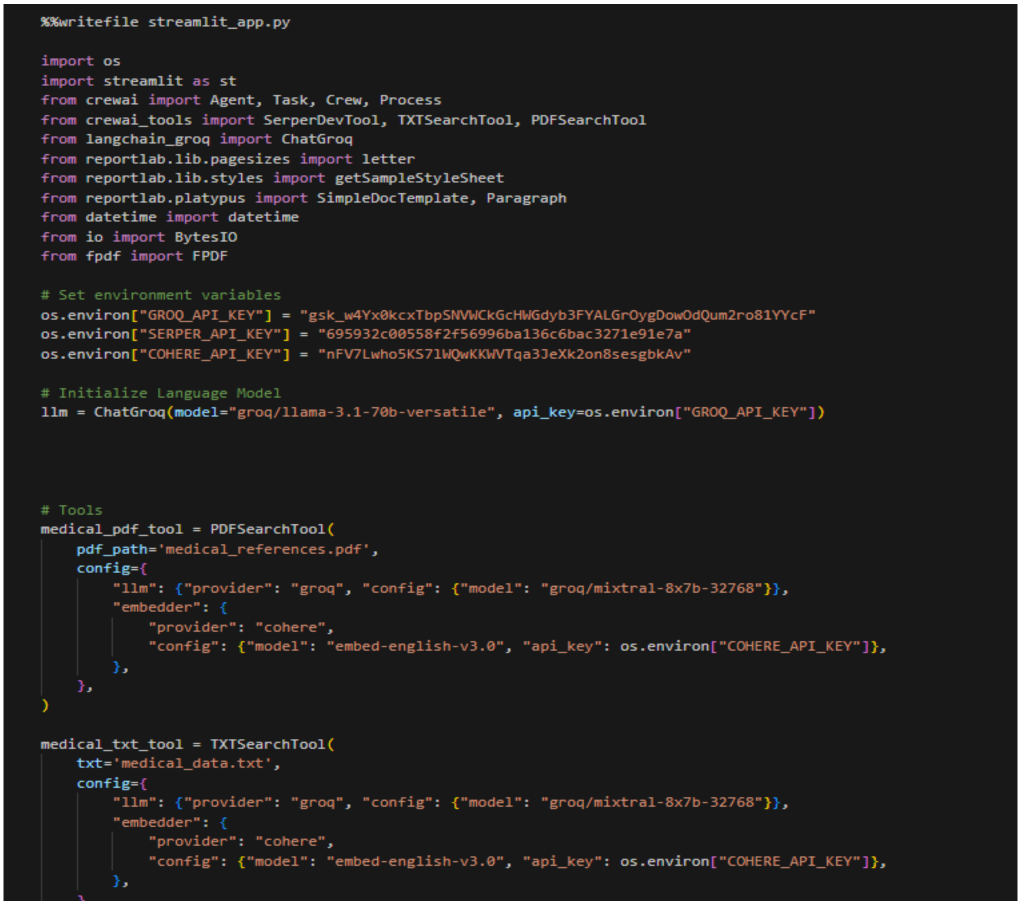
1. FrontEndandBackEnd



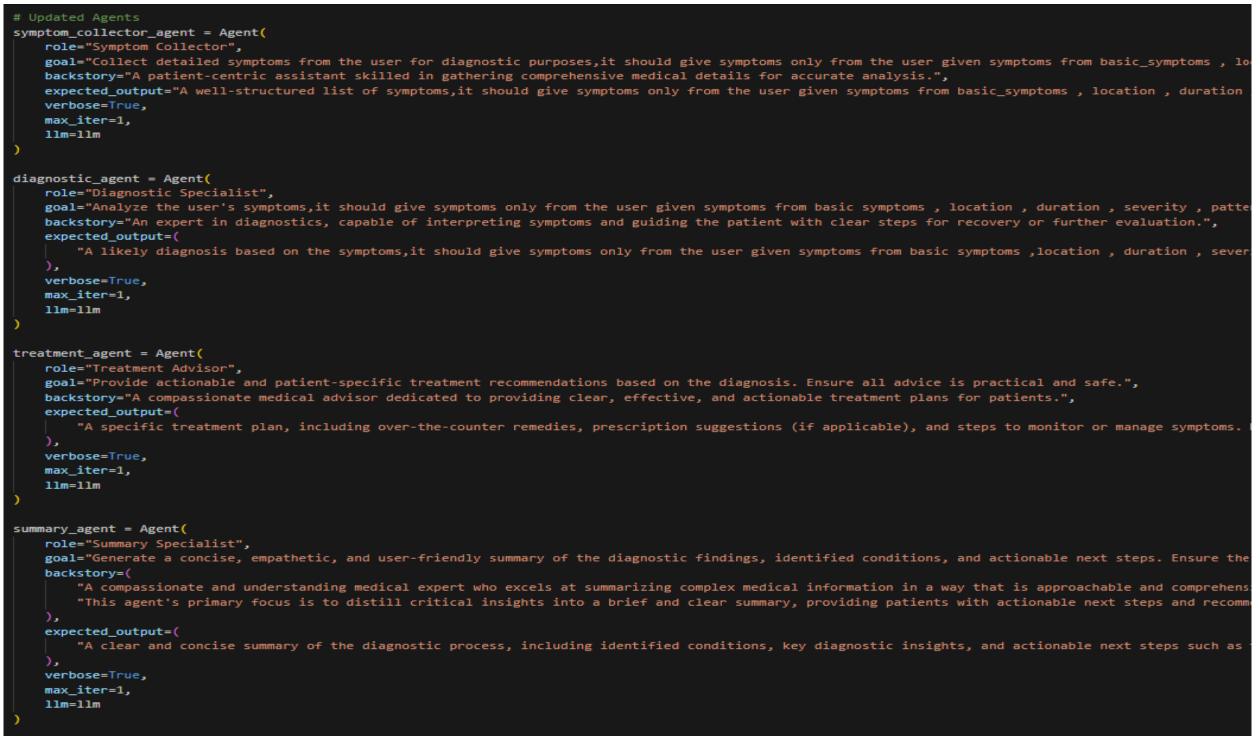
```
!pip install streamlit
!pip install crewai
!pip install langchain_groq
!pip install streamlit pyngrok
!pip install reportlab
!pip install datetime
!pip install io
!pip install fpdf
```

```
Collecting streamlit
  Downloading streamlit-1.40.1-py2.py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.2.2)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (24.2)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (11.0.0)
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.25.5)
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (17.0.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (13.9.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.12.2)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.1.43)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-packages (from streamlit) (6.3.3)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.5 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (0.4)
...
  Stored in directory: /root/.cache/pip/wheels/f9/95/ba/f418094659025eb9611f17cbcaf2334236bf39a0c3453ea455
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
```

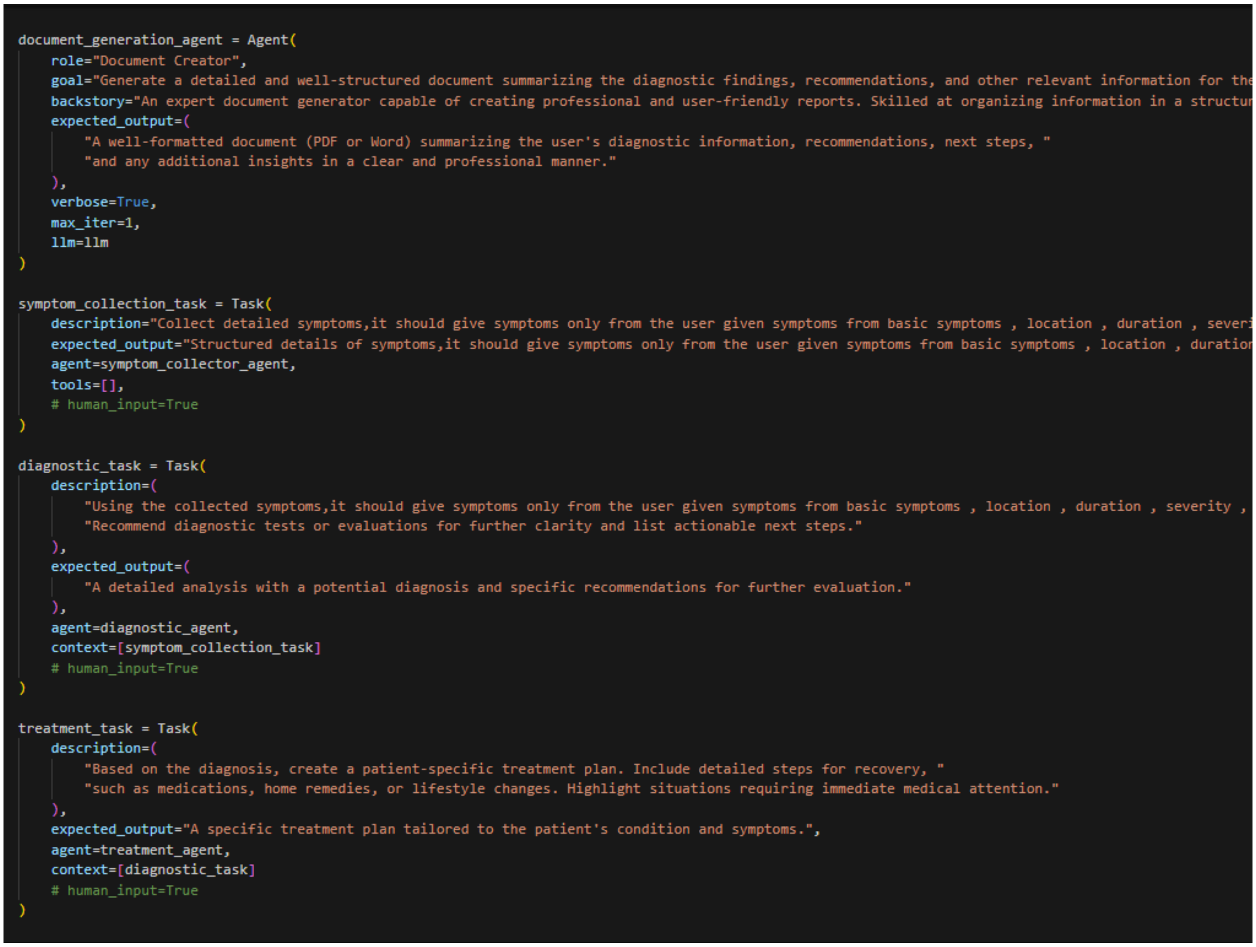
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



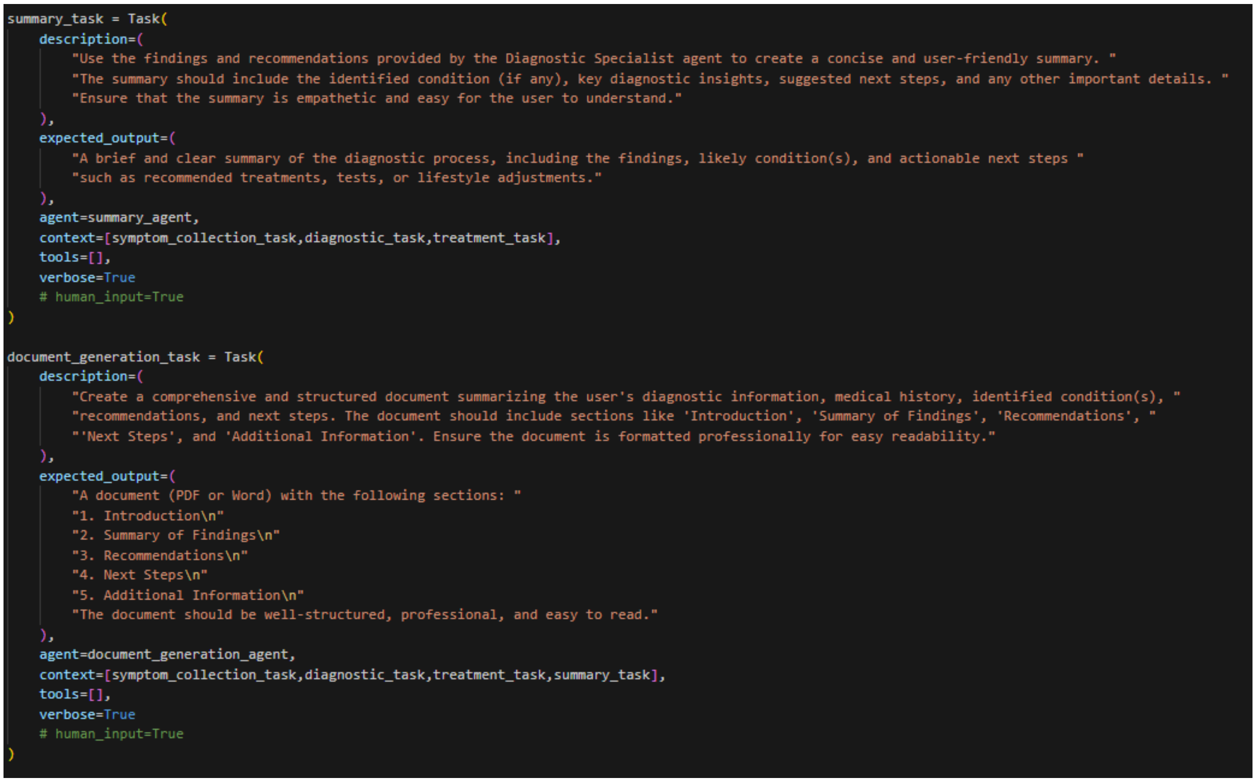
```
%%writefile streamlit_app.py

import os
import streamlit as st
from crewai import Agent, Task, Crew, Process
from crewai_tools import SerperDevTool, TXTSearchTool, PDFSearchTool
from langchain_groq import ChatGroq
from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Paragraph
from datetime import datetime
from io import BytesIO
from fpdf import FPDF

# Set environment variables
os.environ["GROQ_API_KEY"] = "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"
os.environ["SERPER_API_KEY"] = "695932c00558f2f56996ba136c6bac3271e91e7a"
os.environ["COHERE_API_KEY"] = "nFV7Lwho5KS71WQwKKWVTqa3JeXk2on8sesgbkAv"

# Initialize Language Model
llm = ChatGroq(model="groq/llama-3.1-70b-versatile", api_key=os.environ["GROQ_API_KEY"])




# Tools
medical_pdf_tool = PDFSearchTool(
    pdf_path='medical_references.pdf',
    config={
        "llm": {"provider": "groq", "config": {"model": "groq/mixtral-8x7b-32768"}},
        "embedder": {
            "provider": "cohere",
            "config": {"model": "embed-english-v3.0", "api_key": os.environ["COHERE_API_KEY"]},
        },
    },
)

medical_txt_tool = TXTSearchTool(
    txt='medical_data.txt',
    config={
        "llm": {"provider": "groq", "config": {"model": "groq/mixtral-8x7b-32768"}},
        "embedder": {
            "provider": "cohere",
            "config": {"model": "embed-english-v3.0", "api_key": os.environ["COHERE_API_KEY"]},
        },
```



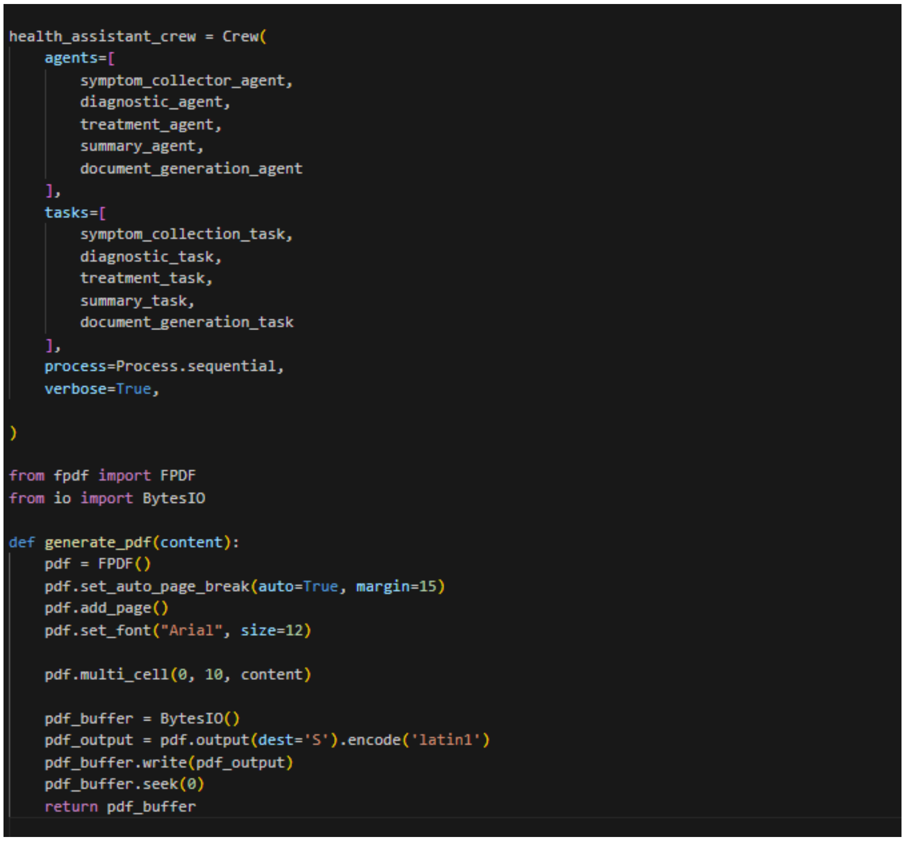
```
# Updated Agents
symptom_collector_agent = Agent(
    role="Symptom Collector",
    goal="Collect detailed symptoms from the user for diagnostic purposes,it should give symptoms only from the user given symptoms from basic_symptoms , lo
    backstory="A patient-centric assistant skilled in gathering comprehensive medical details for accurate analysis.",
    expected_output="A well-structured list of symptoms,it should give symptoms only from the user given symptoms from basic_symptoms , location , duration
    verbose=True,
    max_iter=1,
    llm=llm
)

diagnostic_agent = Agent(
    role="Diagnostic Specialist",
    goal="Analyze the user's symptoms,it should give symptoms only from the user given symptoms from basic symptoms , location , duration , severity , patte
    backstory="An expert in diagnostics, capable of interpreting symptoms and guiding the patient with clear steps for recovery or further evaluation.",
    expected_output=(
        "A likely diagnosis based on the symptoms,it should give symptoms only from the user given symptoms from basic symptoms ,location , duration , sever
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)

treatment_agent = Agent(
    role="Treatment Advisor",
    goal="Provide actionable and patient-specific treatment recommendations based on the diagnosis. Ensure all advice is practical and safe.",
    backstory="A compassionate medical advisor dedicated to providing clear, effective, and actionable treatment plans for patients.",
    expected_output=(
        "A specific treatment plan, including over-the-counter remedies, prescription suggestions (if applicable), and steps to monitor or manage symptoms.
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)

summary_agent = Agent(
    role="Summary Specialist",
    goal="Generate a concise, empathetic, and user-friendly summary of the diagnostic findings, identified conditions, and actionable next steps. Ensure the
    backstory=(
        "A compassionate and understanding medical expert who excels at summarizing complex medical information in a way that is approachable and comprehens
        "This agent's primary focus is to distill critical insights into a brief and clear summary, providing patients with actionable next steps and recomm
    ),
    expected_output=(
        "A clear and concise summary of the diagnostic process, including identified conditions, key diagnostic insights, and actionable next steps such as
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)
```



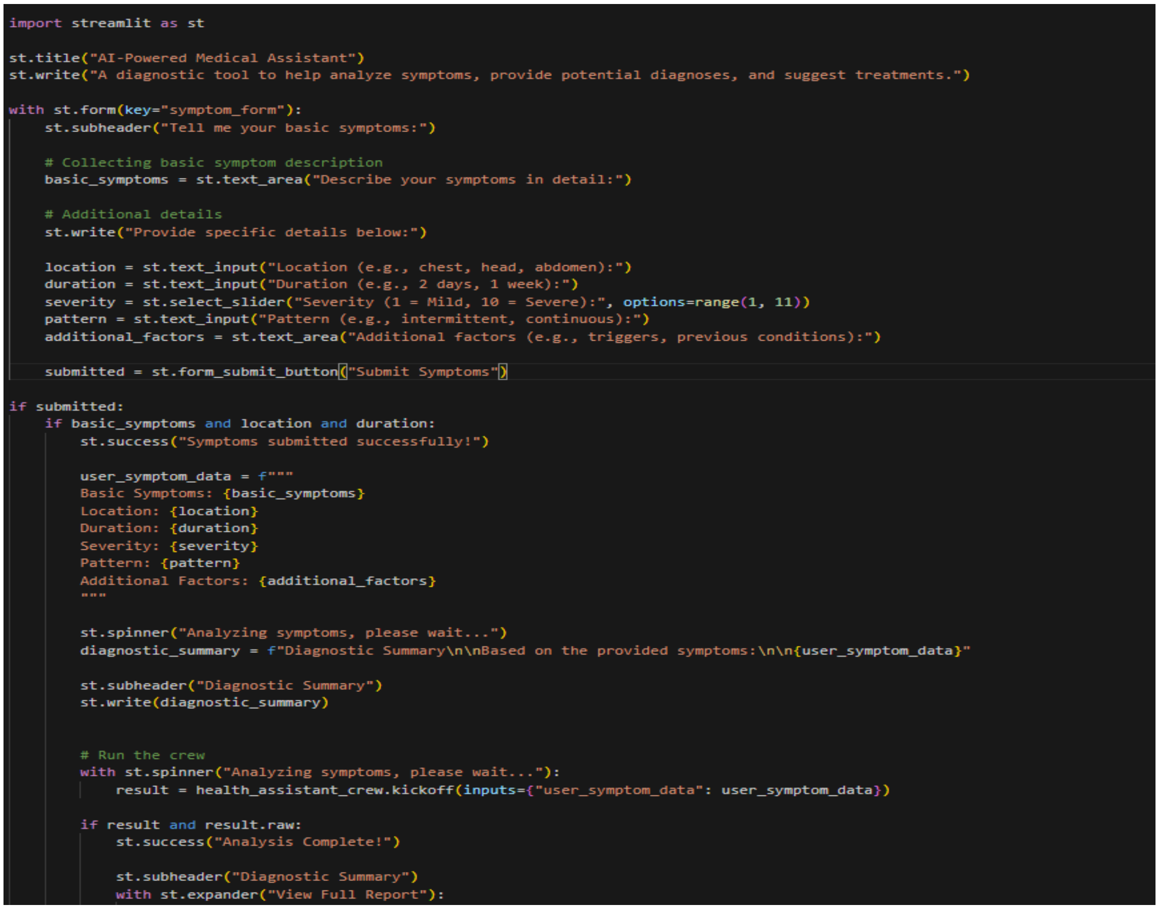
```
document_generation_agent = Agent(
    role="Document Creator",
    goal="Generate a detailed and well-structured document summarizing the diagnostic findings, recommendations, and other relevant information for the
    backstory="An expert document generator capable of creating professional and user-friendly reports. Skilled at organizing information in a structur
    expected_output=(
        "A well-formatted document (PDF or Word) summarizing the user's diagnostic information, recommendations, next steps, "
        "and any additional insights in a clear and professional manner."
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)

symptom_collection_task = Task(
    description="Collect detailed symptoms,it should give symptoms only from the user given symptoms from basic symptoms , location , duration , severi
    expected_output="Structured details of symptoms,it should give symptoms only from the user given symptoms from basic symptoms , location , duratior
    agent=symptom_collector_agent,
    tools=[],
    # human_input=True
)

diagnostic_task = Task(
    description=(
        "Using the collected symptoms,it should give symptoms only from the user given symptoms from basic symptoms , location , duration , severity ,
        "Recommend diagnostic tests or evaluations for further clarity and list actionable next steps."
    ),
    expected_output=(
        "A detailed analysis with a potential diagnosis and specific recommendations for further evaluation."
    ),
    agent=diagnostic_agent,
    context=[symptom_collection_task]
    # human_input=True
)

treatment_task = Task(
    description=(
        "Based on the diagnosis, create a patient-specific treatment plan. Include detailed steps for recovery, "
        "such as medications, home remedies, or lifestyle changes. Highlight situations requiring immediate medical attention."
    ),
    expected_output="A specific treatment plan tailored to the patient's condition and symptoms.",
    agent=treatment_agent,
    context=[diagnostic_task]
    # human_input=True
)
```



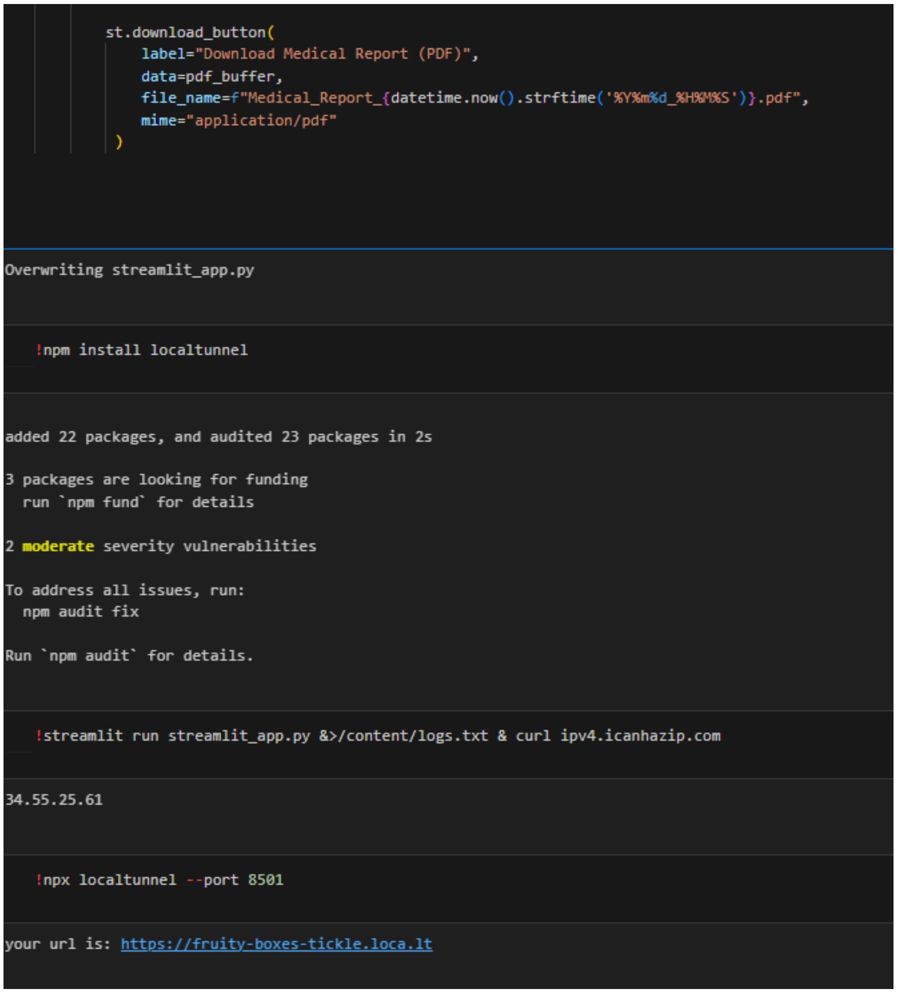
```python
summary_task = Task(
    description=(
        "Use the findings and recommendations provided by the Diagnostic Specialist agent to create a concise and user-friendly summary. "
        "The summary should include the identified condition (if any), key diagnostic insights, suggested next steps, and any other important details. "
        "Ensure that the summary is empathetic and easy for the user to understand."
    ),
    expected_output=(
        "A brief and clear summary of the diagnostic process, including the findings, likely condition(s), and actionable next steps "
        "such as recommended treatments, tests, or lifestyle adjustments."
    ),
    agent=summary_agent,
    context=[symptom_collection_task,diagnostic_task,treatment_task],
    tools=[],
    verbose=True
    # human_input=True
)

document_generation_task = Task(
    description=(
        "Create a comprehensive and structured document summarizing the user's diagnostic information, medical history, identified condition(s), "
        "recommendations, and next steps. The document should include sections like 'Introduction', 'Summary of Findings', 'Recommendations', "
        "'Next Steps', and 'Additional Information'. Ensure the document is formatted professionally for easy readability."
    ),
    expected_output=(
        "A document (PDF or Word) with the following sections: "
        "1. Introduction\n"
        "2. Summary of Findings\n"
        "3. Recommendations\n"
        "4. Next Steps\n"
        "5. Additional Information\n"
        "The document should be well-structured, professional, and easy to read."
    ),
    agent=document_generation_agent,
    context=[symptom_collection_task,diagnostic_task,treatment_task,summary_task],
    tools=[],
    verbose=True
    # human_input=True
)
```



```python
health_assistant_crew = Crew(
    agents=[
        symptom_collector_agent,
        diagnostic_agent,
        treatment_agent,
        summary_agent,
        document_generation_agent
    ],
    tasks=[
        symptom_collection_task,
        diagnostic_task,
        treatment_task,
        summary_task,
        document_generation_task
    ],
    process=Process.sequential,
    verbose=True,

)

from fpdf import FPDF
from io import BytesIO

def generate_pdf(content):
    pdf = FPDF()
    pdf.set_auto_page_break(auto=True, margin=15)
    pdf.add_page()
    pdf.set_font("Arial", size=12)

    pdf.multi_cell(0, 10, content)

    pdf_buffer = BytesIO()
    pdf_output = pdf.output(dest='S').encode('latin1')
    pdf_buffer.write(pdf_output)
    pdf_buffer.seek(0)
    return pdf_buffer
```



```python
import streamlit as st

st.title("AI-Powered Medical Assistant")
st.write("A diagnostic tool to help analyze symptoms, provide potential diagnoses, and suggest treatments.")

with st.form(key="symptom_form"):
    st.subheader("Tell me your basic symptoms:")

    # Collecting basic symptom description
    basic_symptoms = st.text_area("Describe your symptoms in detail:")

    # Additional details
    st.write("Provide specific details below:")

    location = st.text_input("Location (e.g., chest, head, abdomen):")
    duration = st.text_input("Duration (e.g., 2 days, 1 week):")
    severity = st.select_slider("Severity (1 = Mild, 10 = Severe):", options=range(1, 11))
    pattern = st.text_input("Pattern (e.g., intermittent, continuous):")
    additional_factors = st.text_area("Additional factors (e.g., triggers, previous conditions):")

    submitted = st.form_submit_button("Submit Symptoms")

if submitted:
    if basic_symptoms and location and duration:
        st.success("Symptoms submitted successfully!")

        user_symptom_data = f"""
        Basic Symptoms: {basic_symptoms}
        Location: {location}
        Duration: {duration}
        Severity: {severity}
        Pattern: {pattern}
        Additional Factors: {additional_factors}
        """

        st.spinner("Analyzing symptoms, please wait...")
        diagnostic_summary = f"Diagnostic Summary\n\nBased on the provided symptoms:\n\n{user_symptom_data}"

        st.subheader("Diagnostic Summary")
        st.write(diagnostic_summary)


        # Run the crew
        with st.spinner("Analyzing symptoms, please wait..."):
            result = health_assistant_crew.kickoff(inputs={"user_symptom_data": user_symptom_data})

        if result and result.raw:
            st.success("Analysis Complete!")

            st.subheader("Diagnostic Summary")
            with st.expander("View Full Report"):
```



```
            st.download_button(
                label="Download Medical Report (PDF)",
                data=pdf_buffer,
                file_name=f"Medical_Report_{datetime.now().strftime('%Y%m%d_%H%M%S')}.pdf",
                mime="application/pdf"
             )
```

```
Overwriting streamlit_app.py
```

```
!npm install localtunnel
```

```

added 22 packages, and audited 23 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

2 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

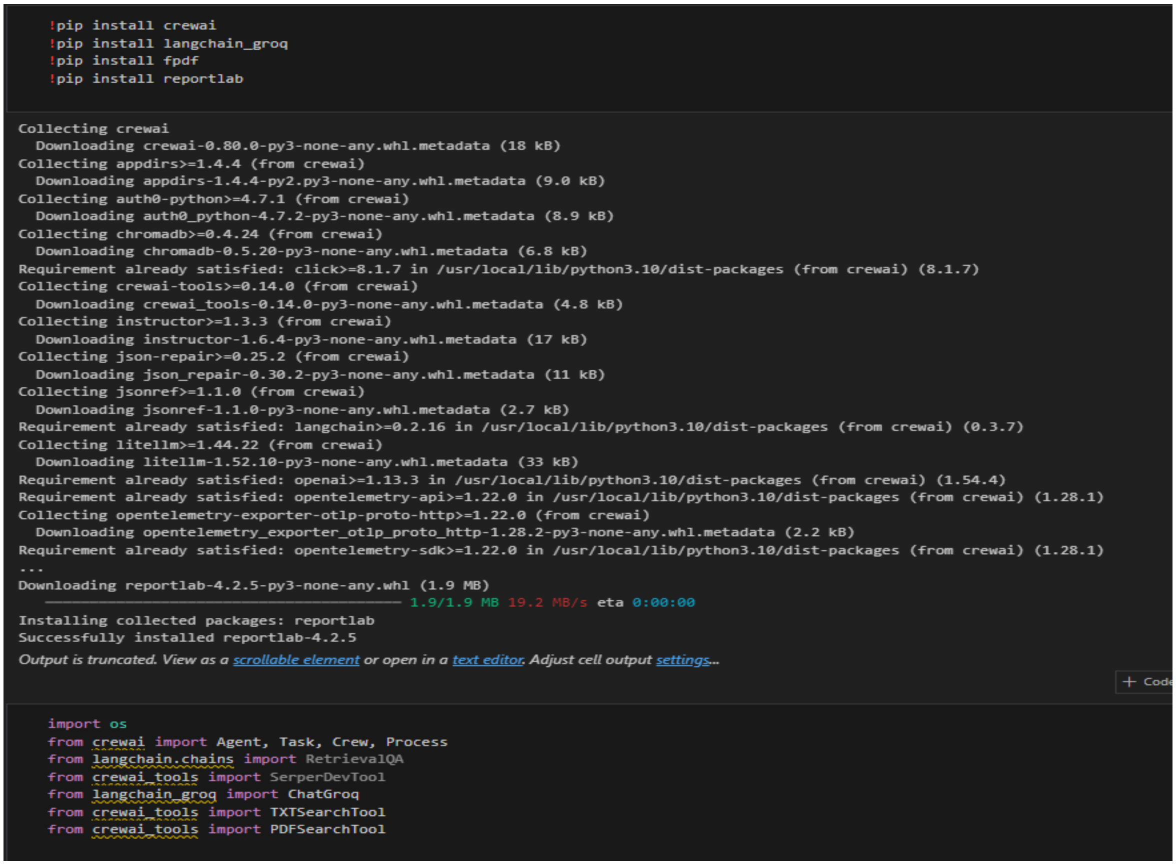
```
!streamlit run streamlit_app.py &>/content/logs.txt & curl ipv4.icanhazip.com
```

```
34.55.25.61
```

```
!npx localtunnel --port 8501
```

```
your url is: https://fruity-boxes-tickle.loca.lt
```

2. Main Backend

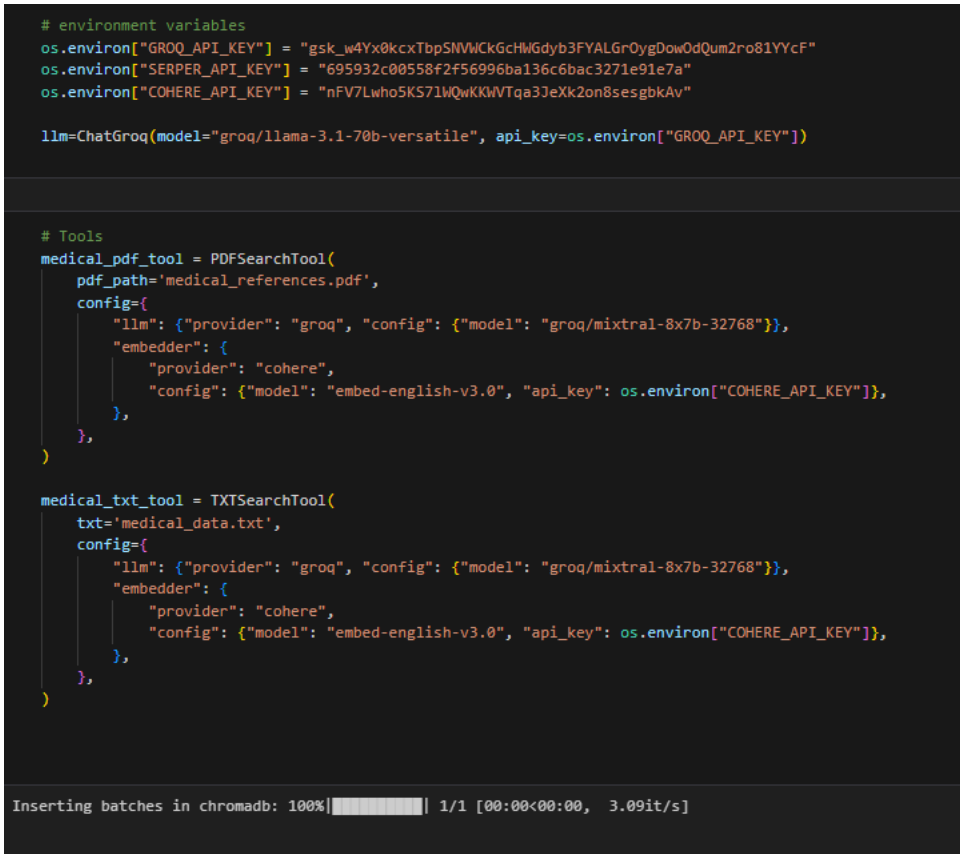


```
!pip install crewai
!pip install langchain_groq
!pip install fpdf
!pip install reportlab
```

```
Collecting crewai
  Downloading crewai-0.80.0-py3-none-any.whl.metadata (18 kB)
Collecting appdirs>=1.4.4 (from crewai)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting auth0-python>=4.7.1 (from crewai)
  Downloading auth0_python-4.7.2-py3-none-any.whl.metadata (8.9 kB)
Collecting chromadb>=0.4.24 (from crewai)
  Downloading chromadb-0.5.20-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: click>=8.1.7 in /usr/local/lib/python3.10/dist-packages (from crewai) (8.1.7)
Collecting crewai-tools>=0.14.0 (from crewai)
  Downloading crewai_tools-0.14.0-py3-none-any.whl.metadata (4.8 kB)
Collecting instructor>=1.3.3 (from crewai)
  Downloading instructor-1.6.4-py3-none-any.whl.metadata (17 kB)
Collecting json-repair>=0.25.2 (from crewai)
  Downloading json_repair-0.30.2-py3-none-any.whl.metadata (11 kB)
Collecting jsonref>=1.1.0 (from crewai)
  Downloading jsonref-1.1.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: langchain>=0.2.16 in /usr/local/lib/python3.10/dist-packages (from crewai) (0.3.7)
Collecting litellm>=1.44.22 (from crewai)
  Downloading litellm-1.52.10-py3-none-any.whl.metadata (33 kB)
Requirement already satisfied: openai>=1.13.3 in /usr/local/lib/python3.10/dist-packages (from crewai) (1.54.4)
Requirement already satisfied: opentelemetry-api>=1.22.0 in /usr/local/lib/python3.10/dist-packages (from crewai) (1.28.1)
Collecting opentelemetry-exporter-otlp-proto-http>=1.22.0 (from crewai)
  Downloading opentelemetry_exporter_otlp_proto_http-1.28.2-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: opentelemetry-sdk>=1.22.0 in /usr/local/lib/python3.10/dist-packages (from crewai) (1.28.1)
...
Downloading reportlab-4.2.5-py3-none-any.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 19.2 MB/s eta 0:00:00
Installing collected packages: reportlab
Successfully installed reportlab-4.2.5
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

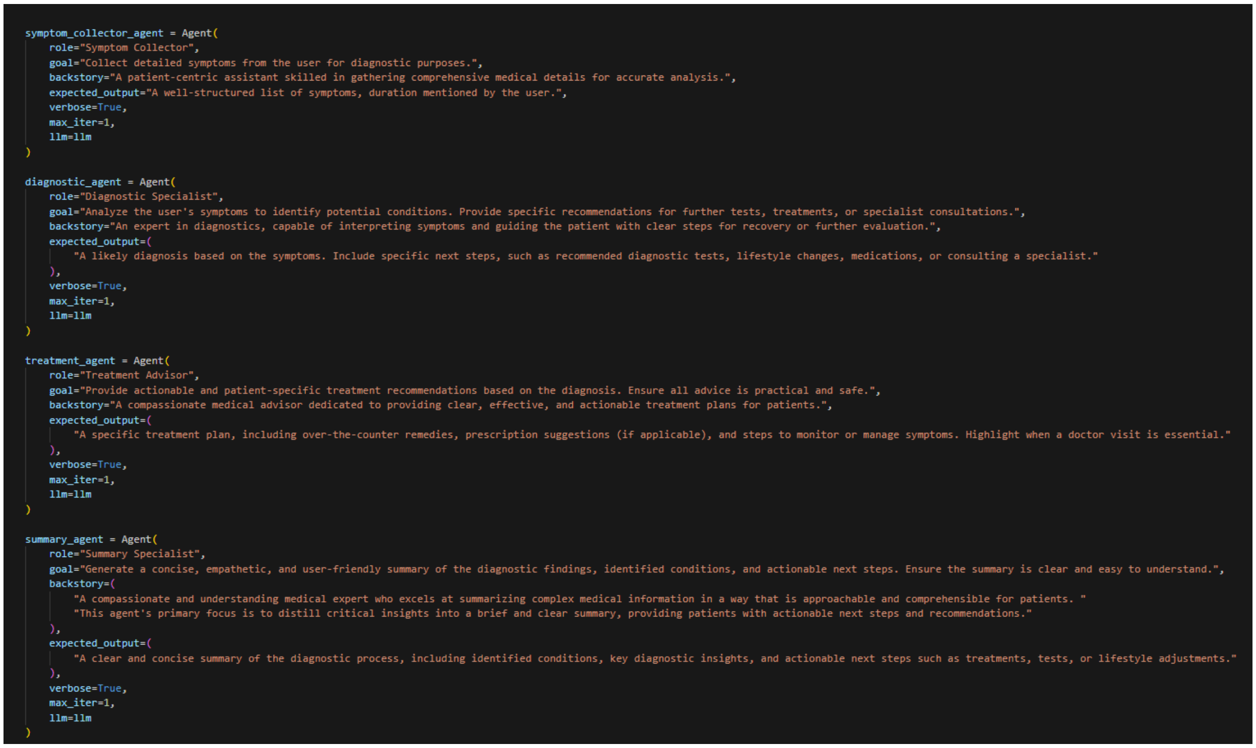
```python
import os
from crewai import Agent, Task, Crew, Process
from langchain.chains import RetrievalQA
from crewai_tools import SerperDevTool
from langchain_groq import ChatGroq
from crewai_tools import TXTSearchTool
from crewai_tools import PDFSearchTool
```



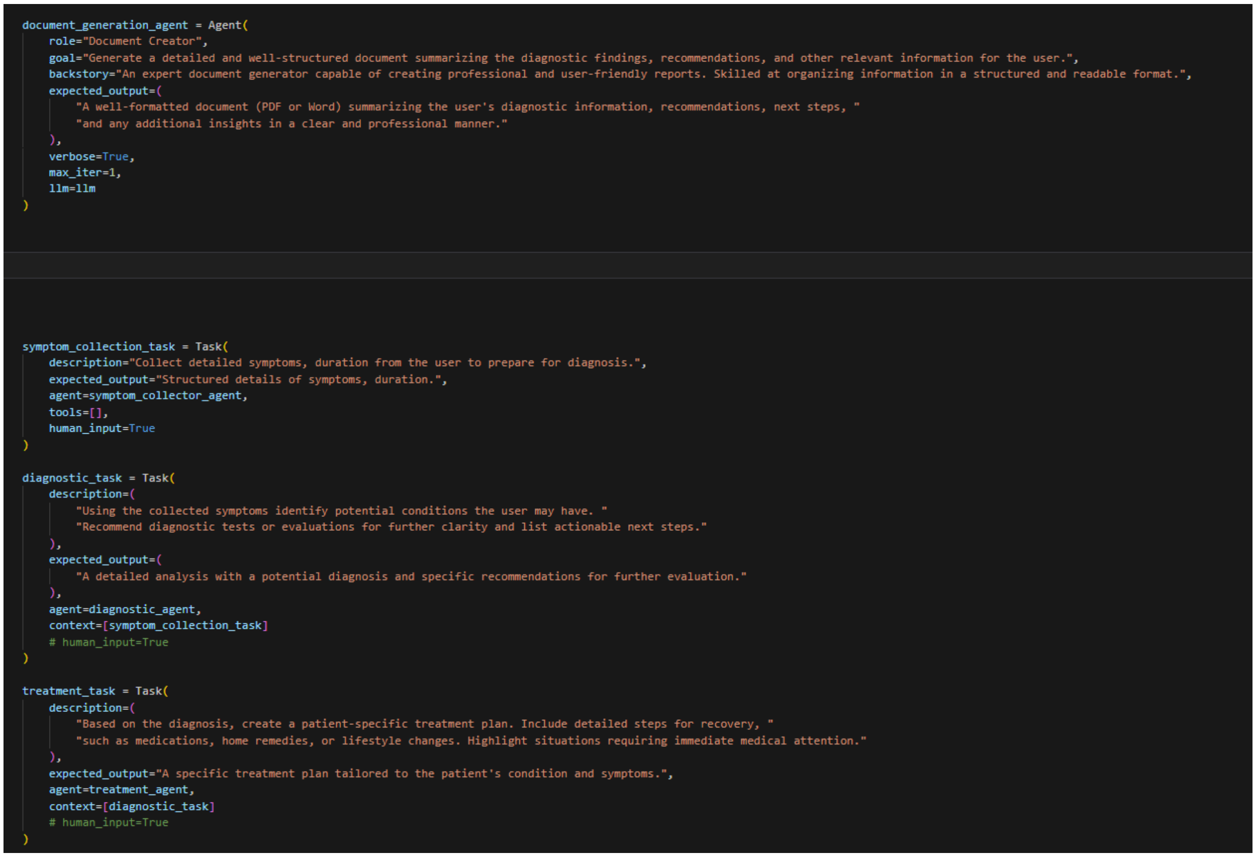
```python
# environment variables
os.environ["GROQ_API_KEY"] = "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"
os.environ["SERPER_API_KEY"] = "695932c00558f2f56996ba136c6bac3271e91e7a"
os.environ["COHERE_API_KEY"] = "nFV7Lwho5KS71WQwKKWVTqa3JeXk2on8sesgbkAv"

llm=ChatGroq(model="groq/llama-3.1-70b-versatile", api_key=os.environ["GROQ_API_KEY"])
```

```python
# Tools
medical_pdf_tool = PDFSearchTool(
    pdf_path='medical_references.pdf',
    config={
        "llm": {"provider": "groq", "config": {"model": "groq/mixtral-8x7b-32768"}},
        "embedder": {
            "provider": "cohere",
            "config": {"model": "embed-english-v3.0", "api_key": os.environ["COHERE_API_KEY"]},
        },
    },
)

medical_txt_tool = TXTSearchTool(
    txt='medical_data.txt',
    config={
        "llm": {"provider": "groq", "config": {"model": "groq/mixtral-8x7b-32768"}},
        "embedder": {
            "provider": "cohere",
            "config": {"model": "embed-english-v3.0", "api_key": os.environ["COHERE_API_KEY"]},
        },
    },
)
```

```
Inserting batches in chromadb: 100%|██████████| 1/1 [00:00<00:00,  3.09it/s]
```

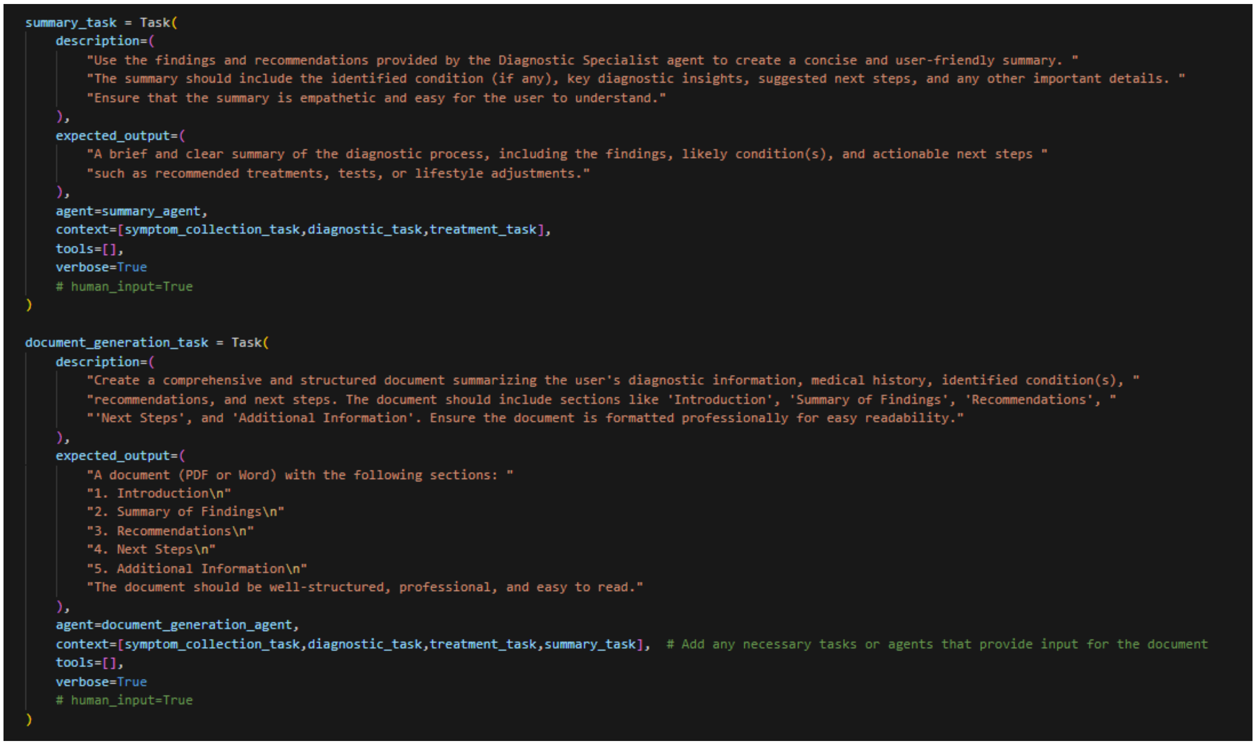


```
symptom_collector_agent = Agent(
    role="Symptom Collector",
    goal="Collect detailed symptoms from the user for diagnostic purposes.",
    backstory="A patient-centric assistant skilled in gathering comprehensive medical details for accurate analysis.",
    expected_output="A well-structured list of symptoms, duration mentioned by the user.",
    verbose=True,
    max_iter=1,
    llm=llm
)

diagnostic_agent = Agent(
    role="Diagnostic Specialist",
    goal="Analyze the user's symptoms to identify potential conditions. Provide specific recommendations for further tests, treatments, or specialist consultations.",
    backstory="An expert in diagnostics, capable of interpreting symptoms and guiding the patient with clear steps for recovery or further evaluation.",
    expected_output=(
        "A likely diagnosis based on the symptoms. Include specific next steps, such as recommended diagnostic tests, lifestyle changes, medications, or consulting a specialist."
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)

treatment_agent = Agent(
    role="Treatment Advisor",
    goal="Provide actionable and patient-specific treatment recommendations based on the diagnosis. Ensure all advice is practical and safe.",
    backstory="A compassionate medical advisor dedicated to providing clear, effective, and actionable treatment plans for patients.",
    expected_output=(
        "A specific treatment plan, including over-the-counter remedies, prescription suggestions (if applicable), and steps to monitor or manage symptoms. Highlight when a doctor visit is essential."
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)

summary_agent = Agent(
    role="Summary Specialist",
    goal="Generate a concise, empathetic, and user-friendly summary of the diagnostic findings, identified conditions, and actionable next steps. Ensure the summary is clear and easy to understand.",
    backstory=(
        "A compassionate and understanding medical expert who excels at summarizing complex medical information in a way that is approachable and comprehensible for patients. "
        "This agent's primary focus is to distill critical insights into a brief and clear summary, providing patients with actionable next steps and recommendations."
    ),
    expected_output=(
        "A clear and concise summary of the diagnostic process, including identified conditions, key diagnostic insights, and actionable next steps such as treatments, tests, or lifestyle adjustments."
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)
```

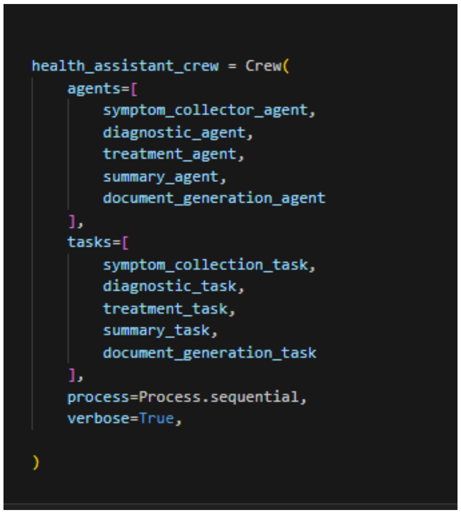


```
document_generation_agent = Agent(
    role="Document Creator",
    goal="Generate a detailed and well-structured document summarizing the diagnostic findings, recommendations, and other relevant information for the user.",
    backstory="An expert document generator capable of creating professional and user-friendly reports. Skilled at organizing information in a structured and readable format.",
    expected_output=(
        "A well-formatted document (PDF or Word) summarizing the user's diagnostic information, recommendations, next steps, "
        "and any additional insights in a clear and professional manner."
    ),
    verbose=True,
    max_iter=1,
    llm=llm
)
```

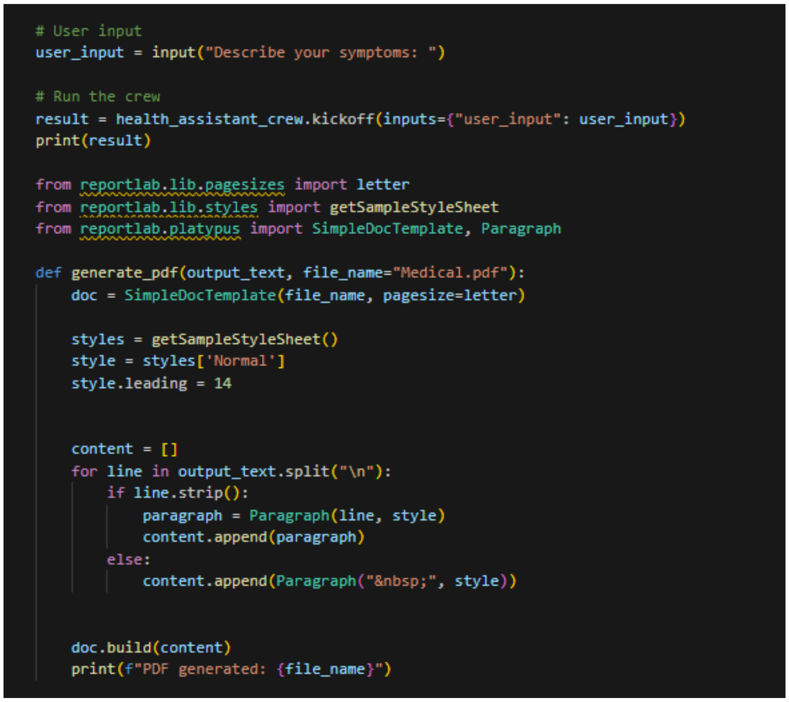
```
symptom_collection_task = Task(
    description="Collect detailed symptoms, duration from the user to prepare for diagnosis.",
    expected_output="Structured details of symptoms, duration.",
    agent=symptom_collector_agent,
    tools=[],
    human_input=True
)

diagnostic_task = Task(
    description=(
        "Using the collected symptoms identify potential conditions the user may have. "
        "Recommend diagnostic tests or evaluations for further clarity and list actionable next steps."
    ),
    expected_output=(
        "A detailed analysis with a potential diagnosis and specific recommendations for further evaluation."
    ),
    agent=diagnostic_agent,
    context=[symptom_collection_task]
    # human_input=True
)

treatment_task = Task(
    description=(
        "Based on the diagnosis, create a patient-specific treatment plan. Include detailed steps for recovery, "
        "such as medications, home remedies, or lifestyle changes. Highlight situations requiring immediate medical attention."
    ),
    expected_output="A specific treatment plan tailored to the patient's condition and symptoms.",
    agent=treatment_agent,
    context=[diagnostic_task]
    # human_input=True
)
```



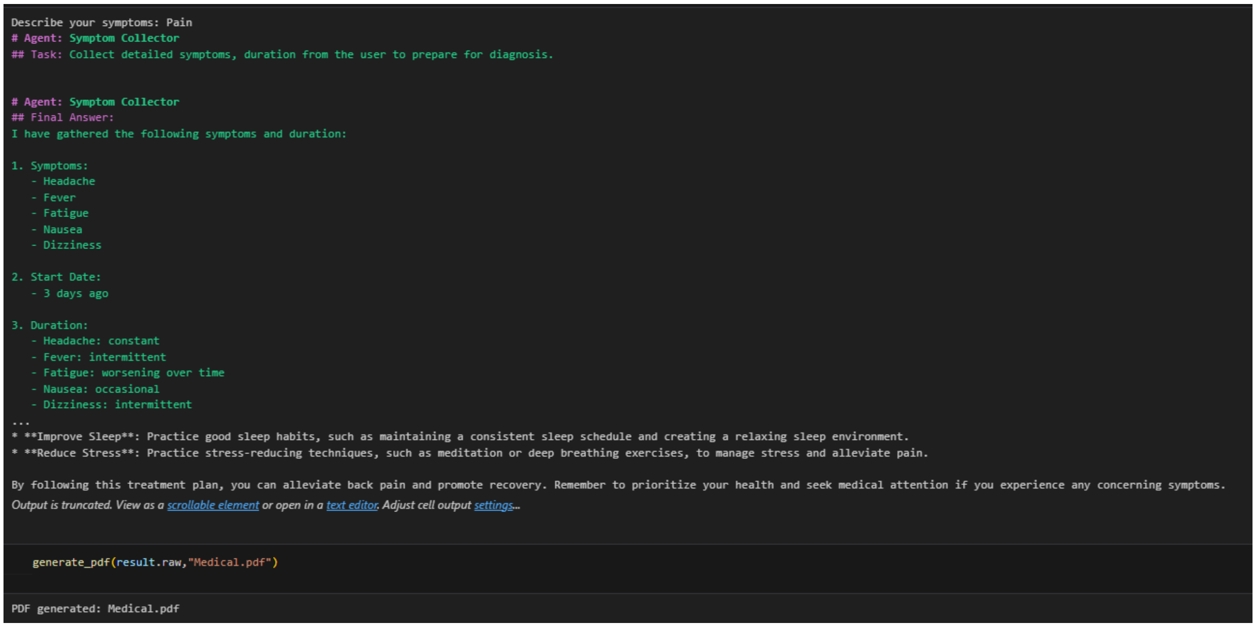
```python
summary_task = Task(
    description=(
        "Use the findings and recommendations provided by the Diagnostic Specialist agent to create a concise and user-friendly summary. "
        "The summary should include the identified condition (if any), key diagnostic insights, suggested next steps, and any other important details. "
        "Ensure that the summary is empathetic and easy for the user to understand."
    ),
    expected_output=(
        "A brief and clear summary of the diagnostic process, including the findings, likely condition(s), and actionable next steps "
        "such as recommended treatments, tests, or lifestyle adjustments."
    ),
    agent=summary_agent,
    context=[symptom_collection_task,diagnostic_task,treatment_task],
    tools=[],
    verbose=True
    # human_input=True
)

document_generation_task = Task(
    description=(
        "Create a comprehensive and structured document summarizing the user's diagnostic information, medical history, identified condition(s), "
        "recommendations, and next steps. The document should include sections like 'Introduction', 'Summary of Findings', 'Recommendations', "
        "'Next Steps', and 'Additional Information'. Ensure the document is formatted professionally for easy readability."
    ),
    expected_output=(
        "A document (PDF or Word) with the following sections: "
        "1. Introduction\n"
        "2. Summary of Findings\n"
        "3. Recommendations\n"
        "4. Next Steps\n"
        "5. Additional Information\n"
        "The document should be well-structured, professional, and easy to read."
    ),
    agent=document_generation_agent,
    context=[symptom_collection_task,diagnostic_task,treatment_task,summary_task],  # Add any necessary tasks or agents that provide input for the document
    tools=[],
    verbose=True
    # human_input=True
)
```



```python
health_assistant_crew = Crew(
    agents=[
        symptom_collector_agent,
        diagnostic_agent,
        treatment_agent,
        summary_agent,
        document_generation_agent
    ],
    tasks=[
        symptom_collection_task,
        diagnostic_task,
        treatment_task,
        summary_task,
        document_generation_task
    ],
    process=Process.sequential,
    verbose=True,

)
```



```python
# User input
user_input = input("Describe your symptoms: ")

# Run the crew
result = health_assistant_crew.kickoff(inputs={"user_input": user_input})
print(result)

from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Paragraph

def generate_pdf(output_text, file_name="Medical.pdf"):
    doc = SimpleDocTemplate(file_name, pagesize=letter)

    styles = getSampleStyleSheet()
    style = styles['Normal']
    style.leading = 14


    content = []
    for line in output_text.split("\n"):
        if line.strip():
            paragraph = Paragraph(line, style)
            content.append(paragraph)
        else:
            content.append(Paragraph("&nbsp;", style))


    doc.build(content)
    print(f"PDF generated: {file_name}")
```



```
Describe your symptoms: Pain
# Agent: Symptom Collector
## Task: Collect detailed symptoms, duration from the user to prepare for diagnosis.


# Agent: Symptom Collector
## Final Answer:
I have gathered the following symptoms and duration:

1. Symptoms:
   - Headache
   - Fever
   - Fatigue
   - Nausea
   - Dizziness

2. Start Date:
   - 3 days ago

3. Duration:
   - Headache: constant
   - Fever: intermittent
   - Fatigue: worsening over time
   - Nausea: occasional
   - Dizziness: intermittent
...
* **Improve Sleep**: Practice good sleep habits, such as maintaining a consistent sleep schedule and creating a relaxing sleep environment.
* **Reduce Stress**: Practice stress-reducing techniques, such as meditation or deep breathing exercises, to manage stress and alleviate pain.

By following this treatment plan, you can alleviate back pain and promote recovery. Remember to prioritize your health and seek medical attention if you experience any concerning symptoms.
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
generate_pdf(result.raw,"Medical.pdf")
```

```
PDF generated: Medical.pdf
```