Overview of the project

FITNESS AI CHAT BOT

the fitness chatbot is the AI bot which is designed for providing virtual assistance to the people who want to get fit and healthy. Whether you are in the process of losing weight, building muscle or have decided to implement a healthy lifestyle, our chatbot helps you through the every step of the way provided you with personalized advice, support and motivation.

Motivation

I have created this fitness chatbot to specifically deal with the factors and obstacles people encounter when trying to pursue a healthier life. People often have full schedules, contradictory data and absence of providing personalized training that leads to demotivation and not reaching the goal in terms of fitness.

Audience

Our chatbot's target audience is everybody with any fitness level and fitness background seeking to gain more health and better wellness. Whether you are a novice just embarking on your fitness journey or a pro looking for a new level of challenges, our chatbot will be customized to your needs and provide personalized assistance and guidance that will help you reach your goals. Whether your goal is to have workout regiments, nutritional instructions, or accountability plus motivation, our chatbot is here to help you through with your journey to a healthier, happier self.

Summary

An apllication powered by streamlit, langchain and openAI API.

1)Workout plan generator:
Continue providing details and click on the Submit button to get your workout plan.

2) Answer fitness questions:
Choose a frequent fitness and health related questions and get similar answers. answer.

3) Diet Plan generator: Customize diet plan

Installation and Setup

python 3.11.5 python -m venv venv cd venv/scripts activate

```
pip install -r requirements.txt
streamlit run 1_Workout_Plan_Generator.py
```

API key management: OPEN AI API KEY

There are two ways for the app to be used.
1 - Utilize the public streamlit cloud version.
2 - Clone the repository, and specify the key name and the value in the secrets.toml file.

Architecture

1.Streamlit Frontend:
The interface development is based on Streamlit, a Python library which allows to easily create web applications.
It gives such elements as buttons, text inputs, number inputs, and form submission functionality to make up an interactive user interface.

2.Backend Interaction
The backend ensures processing of user inputs, validation of API keys, and creation of workout plans.
It communicates with OpenAI's GPT-3.5 model through the `langchain` package, mainly by using the `ChatOpenAI` and `LLMChain` class.

3.OpenAI GPT-3.5 Model
The `ChatOpenAI` class defines interactions with the OpenAI GPT-3.5 model for exemplary response generation of text to the prompts provided by user.
It requires an API key to authenticate the connection with the OpenAI service.

4.Prompt Template
The `PromptTemplate` object defines the form of prompts that are passed to the GPT-3.5 model.
It includes placeholders for variables which will be replaced by the data provided by the user.

5. LLMChain
This `LLMChain` is responsible for stitching prompts and responses together in order to create outputs that are cohesive and context-rich.

-It takes the `ChatOpenAI` object (the instantiated one) and the prompt template as the inputs.

6. User Input Handling
User parameters as workouts (e.g. program time duration, training days per week, maximum pick-up weight, etc.) are collected through Streamlit elements.

These shared inputs are subsequently forwarded to the GPT-3.5 model for the creation of tailored nutrition and work out plans.

7. Output Presentation
In the end, user designed workout plans are presented to the user by means of the Streamlit interface.
Output will be in a comprehensible format, maybe as text, with weekly workout plans as the data.

9. State Management
The session state management of Streamlit is utilized for handling and managing the state of the program, including whether the point generation mechanism has been started or not.

Finally, the architecture provides users with a text-input preference, customized GPT-3.5 model workout plans generated that is connected to the Streamlit-powered web interface.

AI and NLP Techniques

1.Prompt-based Generation
The machine intelligence of the chatbot is constituted of its ability to give responses after prompts. Prompts, which stands for predefined text templates, are used to direct the AI model so as to understand user inputs and generate responses that are related to the topic at hand. The `PromptTemplate` class is where the templates for the prompts are defined.

2.Language Model (GPT-3.5)
The chatbot uses a pre-trained language model (LLM) provided by the open-source GPT-3.5 architecture engine of OpenAI. This model, in turn, serves as an interpreter of the given prompts and develops necessary workout plans by means of the provided inputs.

3.Entity Recognition
The chat bot brings out the required information from the user input like workout period (weeks) number of sessions per week- 1-rep max values for different workouts like squat, bench press, and deadlift- and the number of accessory exercises required for each session. These entities that were pulled are utilized to make the generated workout plan personal.

5.Dialogue Management
The chatbot directs the dialogue flow by offering scheduled workout plans for every week, based upon the user's inputs. It keeps track of current week and when at the user's request, adjustment of the program's difficulty level will be made.

Data sources and APIs

1. OpenAI API (GPT-3.5)
This is the main API for accessing natural language processing and conversion.
The `openai` Python package is used to interface with the OpenAI API.
The API key is given by the user and acquired via the Streamlit interface.
We will validate the API key by running a request to the OpenAI API using the key.

2. Streamlit
Streamlit is the framework that we use to build the interative web application
It contains different user interface components such as buttons, text inputs, number inputs, and forms that make it easy to develop applications.
Streamlit session state is used to keep information including user input and API key to be reused in the different sessions.

3. External Data Sources
External data sources are not referenced here. The chatbot output workout plans solely on the basis of inputs from the user and the features of the GPT-3.5 model.

4. Data Fishing, Processing, and Utilization
User input is gathered through the Streamlit inputs, such as text, number, and select boxes.
The gathered information parameters are, after that, utilized to build up prompts/templates for the GPT-3.5 model.
The GPT-3.5 model informs the user on the workout plans via a given prompt and input parameters. The exercise routine is easily seen by the user through the Streamlit interface.

CI/CD pipeline for an LLM (large language model) model

1. Version Control: Use version control systems like Git to manage your LLM code, datasets and configurations. Have your repository hosted on platforms like GitHub, GitLab or Bitbucket.

2. Continuous Integration (CI):

Get a CI server i.e Jenkins, Travis CI, CircleCI or GitHub Actions which can be used to automatically initiate builds when changes are pushed into the repository.

Define CI jobs that run tests, code quality checks, and any preprocessing steps required before training the model. We can use a sonarqube

CI pipeline should have data quality and consistency checks, as well as checks on performance metrics for the models.

3. Continuous Deployment (CD):

Deploy the trained LLM model after successful CI builds.

use tools like Docker for containerization so that development environment does not differ much from production environment.

Automate deployment to staging or production environments using deployment scripts or tools such as Kubernetes for container orchestration.

Deployment on Cloud service

We can use AWS EC2 is ISAS

Using Amazon Elastic Compute Cloud (EC2) within which you can provision a virtual server (instance) in AWS and then deploy your application on it. Here are the steps at a glance:

1. Launch an EC2 Instance

Open the AWS Management Console.

Select EC2 from the list of services.

Select an Amazon Machine Image(AMI) to launch an EC2 instance, choose an instance type, configure its details e.g.networking and storage, and create security groups(firewall rules).

2. Connect to EC2 Instance

After launching your EC2 instances, you will need to connect to them using SSH (Secure Shell) protocol from either a local machine or another computer.

To authenticate with AWS, use the specified key pair during instance launch.

Example of SSH command is as follows:

```bash

ssh ec2-user@your-ec2-instance-ip -i /path/to/your/key.pem

```

3.Prepare Environment:

On your EC2 instance install any software packages on which your application depends, for example Python, Docker or any other tool that might be needed for your service.

Now copy your code from Git repository into the instances created in step 1 above.

4. Configure Application

Customize the settings of your application based on its environment by altering configuration files and supplying desired environment variables such as secret keys etc

5.RUN APPLICATION

Switch on the EC2 instance for your application. It can be a Python script, web server or Docker container that is running.

6.NETWORKING SETUP

The security group of the EC2 instance requires to be modified in such a way as to enable communication between your application and its users.

Your EC2 instance could have an Elastic IP address associated with it for public access.

7.TEST DEPLOYMENT

Open the app by typing either the domain name or the public IP address on your browser's URL bar and click enter. This should direct you to the webpage displaying "Hello World" message. If it does not, then there were problems during the set up process that need to be checked.

Try out different functionalities of your app just to make sure everything works well as expected.

8.MONITORING AND MAINTENANCE

Set up monitoring and logging system in AWS CloudWatch or any other tool that will do it on behalf of you for both you EC2 instance and your app.

Check often how healthy and efficient is working your app also don't forget about patches updates etc.,

9.MAXIMUM EXPANSION AND DISTRIBUTED LOADS (Optional)

When growing, you may consider scaling horizontally by adding more instances behind a load balancer so as to evenly distribute traffic among them.

10. BACKUP & DISASTER RECOVERY:

Implement plans for backup and recovery which will ensure integrity and availability regarding application data.