Chatbot based on ChatGPT

Chatbot Id：6266019716

Ye Hanyi  
22414495

JI Shaohua22441557

Li Xin 22427600

*Abstract*—We used the Python programming language and OpenAI's GPT-3.5 model as the core of Chatbot to implement a variety of features required by the project, including Reading/writing a TV show reviews, sharing a cooking video, etc.

Keywords—python, ChatGPT, redis, Docker, Azure

# Introduction

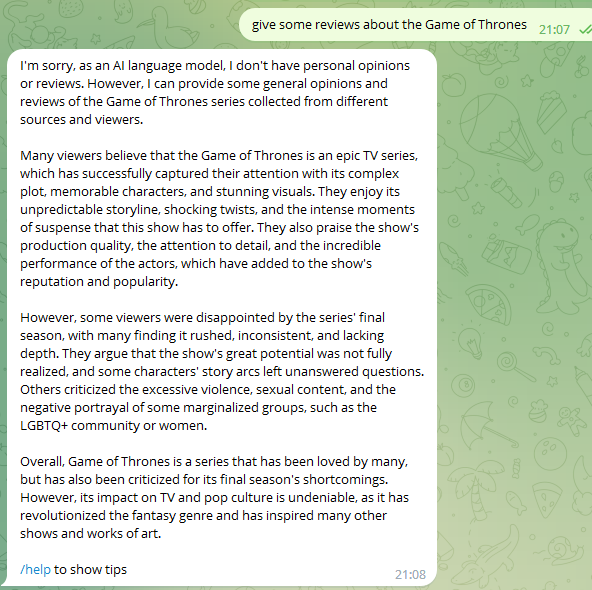
Chatbot is an artificial intelligence application that can mimic human conversation. With the development of natural language processing technology, Chatbot has become an increasingly popular form of interaction. What we have done is a Telegram Chatbot based on ChatGPT, which is able to answer basic questions, small talk, questions and answers, suggestions and knowledge quizzes from users, enabling natural language interaction. We also used the technologies required by the project. The project was managed using git throughout, we packaged the project as an image file by writing a Dockerfile, deployed the image file to Azure using Github Action automated deployment, and used the Redis service provided by Azure as our database. We have successfully blended multiple technologies together in this project, using what we have learnt in class and meeting all the requirements of the project.

# Features

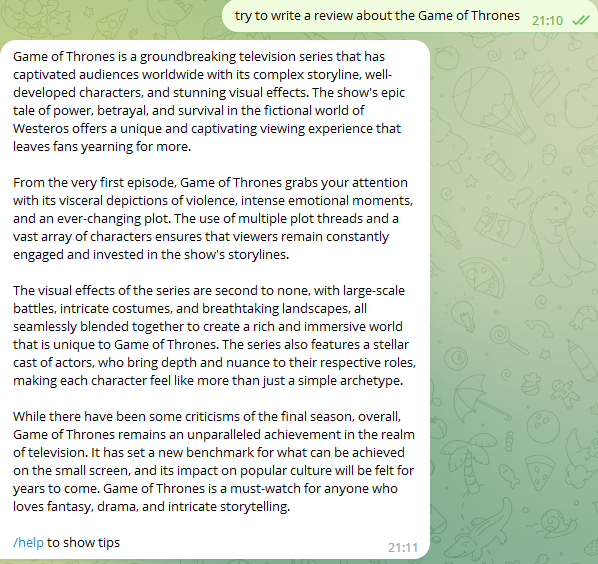
Most of the functions have been implemented and passed the tests. Able to meet requirements very well.

## Reading/writing a TV show review

We can use Chatbot to Read or write a TV show review through dialogue.



1. Chatbot shares Game of Thrones reviews



1. Chatbot write a review about the Game of Thrones

## Sharing a cooking video

We can ask Chatbot to share a cooking video with us directly through the conversation.



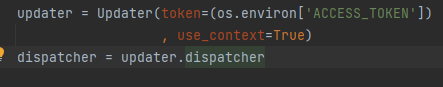
1. Chatbot Sharing a cooking video

# Technology used

In this project, we have fully met the technical requirements and successfully implemented a variety of technologies. Next, I will describe the use of these technologies in detail.

## Telegram chatbot

Access\_TOKEN is the access token for the Telegram Bot, which we obtained by creating a Bot on Telegram. The Updater object is one of the core components of the Telegram Bot, which is responsible for connecting to the Telegram Bot API and receiving and processing messages and commands from the user. The Dispatcher object is a sub-component of the Updater object and is responsible for distributing messages and commands to the corresponding handler functions for processing. [1] The Updater and Dispatcher objects allow us to control and manage the Telegram Bot and enable it to respond quickly to user messages and commands.



1. Codes using python-telegram-bot library

图形用户界面, 文本, 应用程序, 聊天或短信

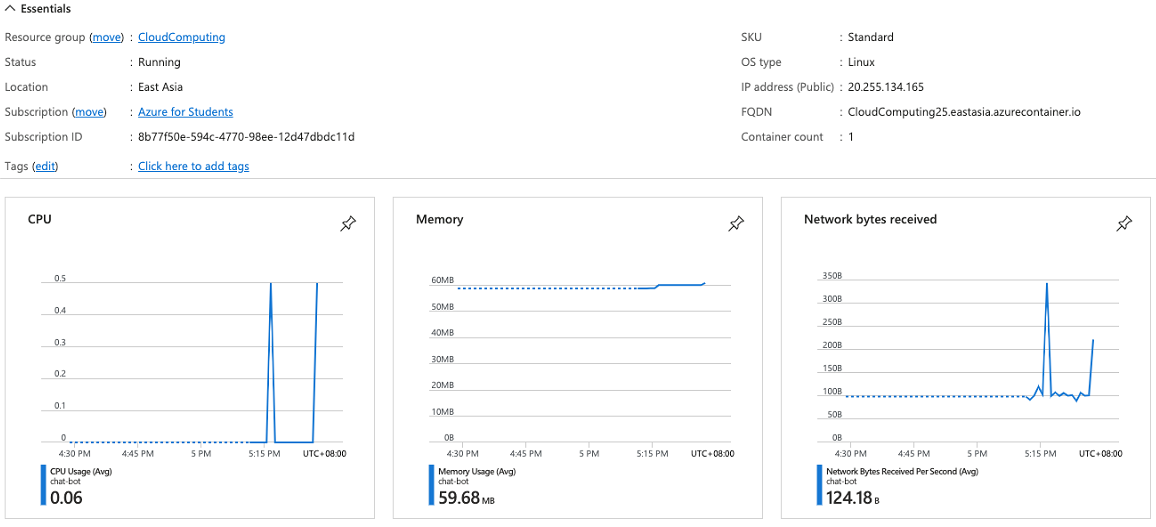
描述已自动生成图形用户界面, 文本

描述已自动生成

1. Interacting with chatbot

## Azure

We use azure to deploy our chatbot. Azure is a cloud computing platform from Microsoft that includes a range of cloud services such as computing, storage, databases, networking, artificial intelligence and a wide range of support and development tools. azure can help users achieve digital transformation and innovation, increase efficiency and flexibility.[2]

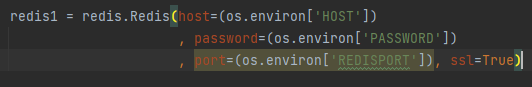


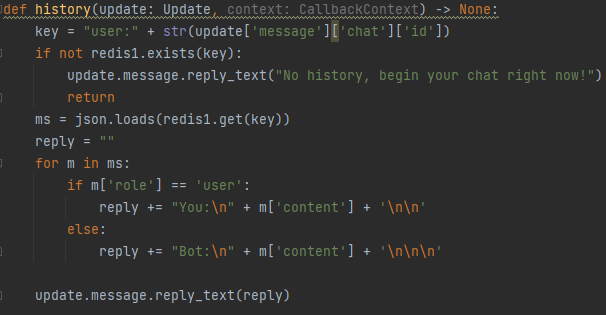
1. Chatbot running on Azure

## Redis

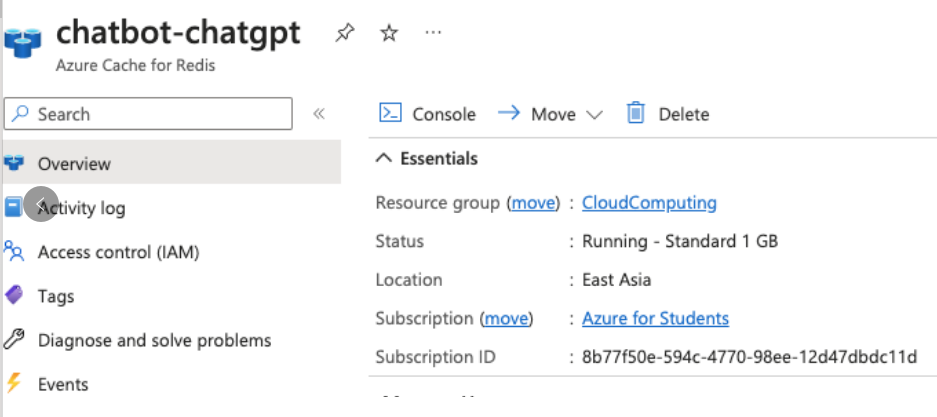
We have implemented the ability to view the history by using Azure's Redis service.

Azure Redis Cache is a high-performance, highly available cloud caching service that uses the open source Redis database engine to support multiple caching modes, data persistence, disaster recovery and other features, and can be integrated with other Azure cloud services to improve application responsiveness and throughput, and reduce database load and costs.[3]





1. Code to call redis



1. Azure Redis

## Docker

Docker is a commonly used containerisation technology that enables the packaging and deployment of applications by defining and building Docker images using Dockerfile, a lightweight executable package that contains all the dependencies and configuration information required for an application to run and can easily be run in any Docker-enabled environment, facilitating This makes it easy to migrate and extend applications.[4]

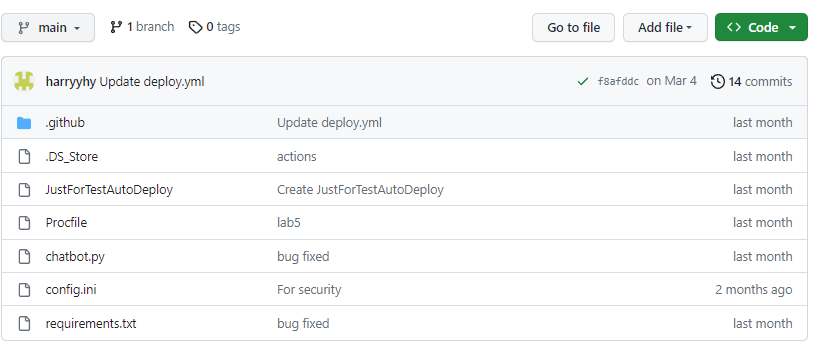
图片包含 文本

描述已自动生成

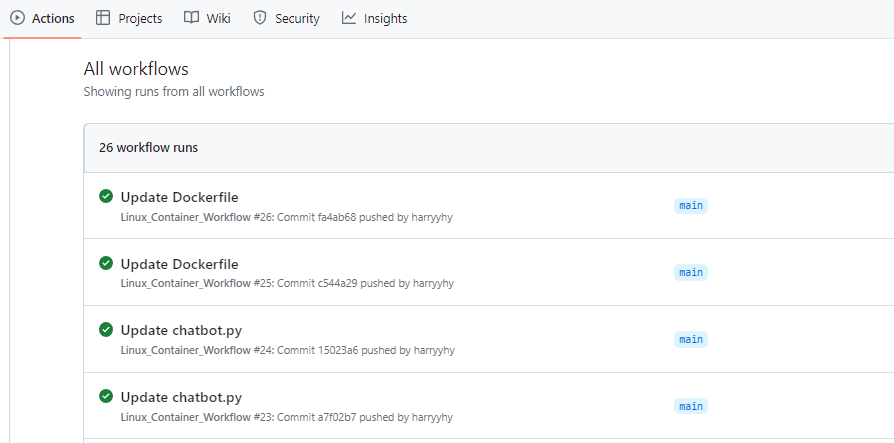
1. Dockerfile

## Git

The project was managed using git throughout. And using Github Actions, we have successfully automated the deployment.



1. Project on Github



1. Workflows

# Job division

## Li Xin - Requirements Analysis, Assistant Coder

The person responsible for the requirements analysis plays a vital role in the project, understanding the project requirements and translating them into functional and performance aspects of the system. He prepared the relevant requirements documents and communicated with other project participants to confirm that the requirements were accurate. His work ensured that the chatbot project ran smoothly and ultimately met the needs and expectations of the project. Also responsible for writing code for some related modules.

## Ye Hanyi - Lead Coder

The lead coder was a key player in the project. He designed the system architecture and modules, wrote most of the high quality code, performed code testing to ensure the reliability and stability of the system and successfully deployed the project. Through his work, the chatbot project was able to successfully implement the features and provide a quality service to the users.

## JI Shaohua - Report Writer, Assistant Coder

Mainly responsible for the preparation of project summaries and reports, the work mainly includes a comprehensive overview and summary of all aspects of the project, collecting and collating project data and information, writing logical and readable reports, helping the team to identify potential problems and propose improvement solutions to ensure the project is completed on time and meets the expected objectives. Also responsible for writing code for some related modules.

##### References

1. Python-telegram-bot, https://docs.python-telegram-bot.org/en/stable/telegram.ext.updater.html*s*
2. Azure Documentation, https://learn.microsoft.com/en-us/azure/?product=popular.
3. Azure Cache for Redis Documentation, https://learn.microsoft.com/en-us/azure/azure-cache-for-redis/.
4. Dockerfile Reference, https://docs.docker.com/engine/reference/builder/.