4) Set up RKLLM-Server-Flask: Using the complete class functions encapsulated in step 3, build the Flask server, including loading the RKLLM model based on user-specified parameters at Flask startup, defining the format of user input and the method for parsing the input, differentiating between streaming/non-streaming inference call methods, defining the return format for inference output and the specific implementation of callback functions, and handling RKLLM release.

The specific implementations of the above modules form the main body of the code in rkllm_server/flask_server.py, thereby completing the deployment of the RKLLM-Server-Flask example. Users can modify the initialization definitions for the RKLLM model to implement different custom models. Additionally, users can refer to the RKLLM-Server-Flask deployment example for implementing their own custom server deployment.

3.4.1.3 Client-side: API Access Example

In rkllm_server_demo/chat_api_flask.py, an API access example for the aforementioned RKLLM-Server-Flask server is demonstrated. When developing custom functionalities, users only need to refer to this API access example and utilize corresponding send-receive structures for data wrapping and parsing. Since the send-receive data structures in this example follow the design of the OpenAI-API, users who have previously developed using the OpenAI-API only need to replace the corresponding network interfaces. This section will provide explanations for the important code segments:

1) Define the network address of RKLLM-Server-Flask. Users need to set the target address based on the specific IP of the Linux development board, the port number set by the Flask framework, and the function name for access.

```
server_url = 'http://172.16.10.166:8080/rkllm_chat'
```

2) Define the form of API access, with options for non-streaming transmission and streaming transmission, defaulting to non-streaming transmission.

```
is_streaming = True
```

3) Define the session object, using requests.Session() to configure the communication process between the API access interface and the server. Users can customize modifications according to their actual development needs.