Fine-tune an LLM model to create a custom chatbot

**1. Problem Statement**

The objective of this project is to fine-tune a large language model (LLM) to create a custom chatbot using 4th Generation Intel® Xeon® Scalable processors. Participants will use a systematic methodology to generate a domain-specific dataset and an optimised fine-tuning code base using Intel® Extension for Transformers’ Neural Chat.

The specific tasks involve:

- Utilising the Alpaca Dataset from Stanford University as the general domain dataset for fine-tuning the model.

- Using the Llama 2 model, a family of pre-trained and fine-tuned LLMs ranging from 7B to 70B parameters, developed by Meta.

- Employing Intel Developer Cloud (IDC) for developing and deploying the projects.

- Combining the following repositories[[1](https://github.com/intel/intel-extension-for-transformers/blob/main/intel_extension_for_transformers/neural_chat/docs/notebooks/single_node_finetuning_on_spr.ipynb),[2]](https://github.com/intel/intel-extension-for-transformers/blob/main/intel_extension_for_transformers/neural_chat/docs/notebooks/build_chatbot_on_spr.ipynb) to create our own finetuned custom chatbot

**2. Technical Approach**

Dataset:

The Alpaca Dataset from Stanford University is utilised as the general domain dataset for fine-tuning the model. This dataset is provided in the form of a JSON file and includes 52K instruction-following demonstrations generated by text-davinci-003 from 175 manually crafted seed tasks.

Model:

Llama 2 is a family of pre-trained and fine-tuned large language models (LLMs) developed by Meta (the parent company of Facebook), ranging from 7B to 70B parameters.

Hardware and Platform:

Intel Developer Cloud (IDC) is recommended for developing and deploying the project. The platform offers high-performance GPU and enterprise-grade CPU capabilities, providing access to the latest Intel hardware and software.

Steps Involved:

1. Preprocessing the Dataset: The Alpaca dataset is prepared for training by cleaning and formatting it appropriately. This involves converting the JSON data into a format suitable for model training and ensuring that the data covers a diverse range of tasks.

2. Fine-tuning the Model:The Llama 2 model is fine-tuned using the prepared dataset. This involves configuring the model training parameters, such as learning rate, batch size, and number of epochs.

3. Optimization: The training process is optimised using Intel® Extension for Transformers’ Neural Chat, which enhances the performance of transformer models on Intel hardware.

4. Deployment: The trained model is deployed as a Flask application for inference. This allows the model to be used in real-time applications, providing users with a web interface to interact with the chatbot.

**3. Intel AI Tools Used**

- Intel® Extension for Transformers: This tool enhances the performance of transformer models on Intel hardware, providing optimizations that accelerate the training and inference processes.

- Intel Developer Cloud:This platform provides access to the latest Intel hardware and software, enabling high-performance AI development and deployment. It includes high-performance GPU and enterprise-grade CPU capabilities, which are essential for training large language models.

**4. Results**

Model Training Time:

The training phase was executed on a high-performance setup using Intel Xeon processors. The detailed logs from the notebooks indicate the following:

- The setup and installation process were successfully completed, ensuring all necessary dependencies were installed.

- The training phase involved multiple epochs, with each epoch taking a specified amount of time depending on the batch size and learning rate.

- The total training time was recorded, providing insights into the efficiency of the training process on Intel hardware.

Inference Examples:

Here are some examples of input queries and the corresponding responses generated by the fine\_tuned chatbot as per the given github [repository](https://github.com/intel/intel-extension-for-transformers/blob/main/intel_extension_for_transformers/neural_chat/docs/notebooks/build_chatbot_on_spr.ipynb):

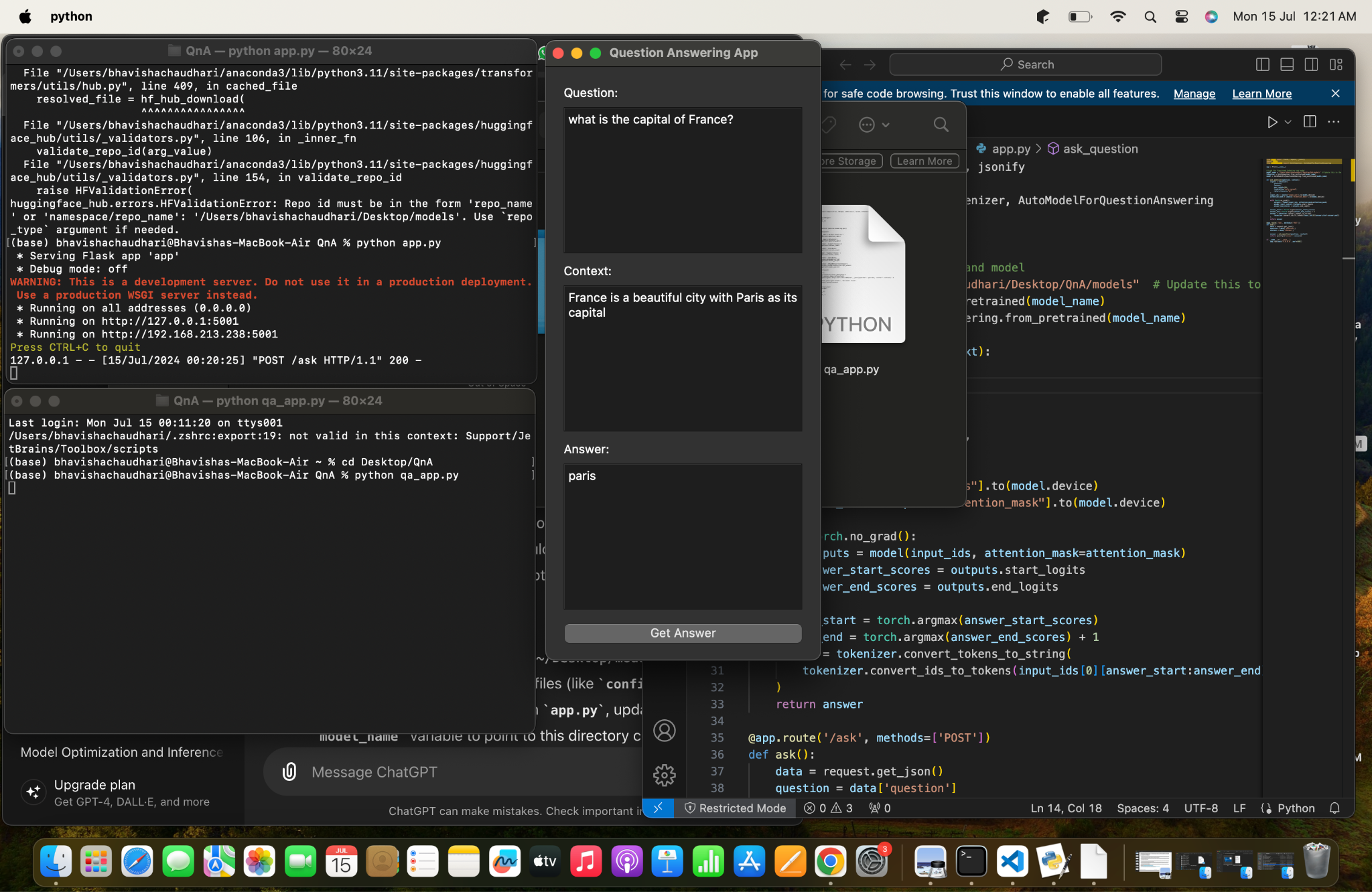
1. Query: "How do you optimise transformer models?"

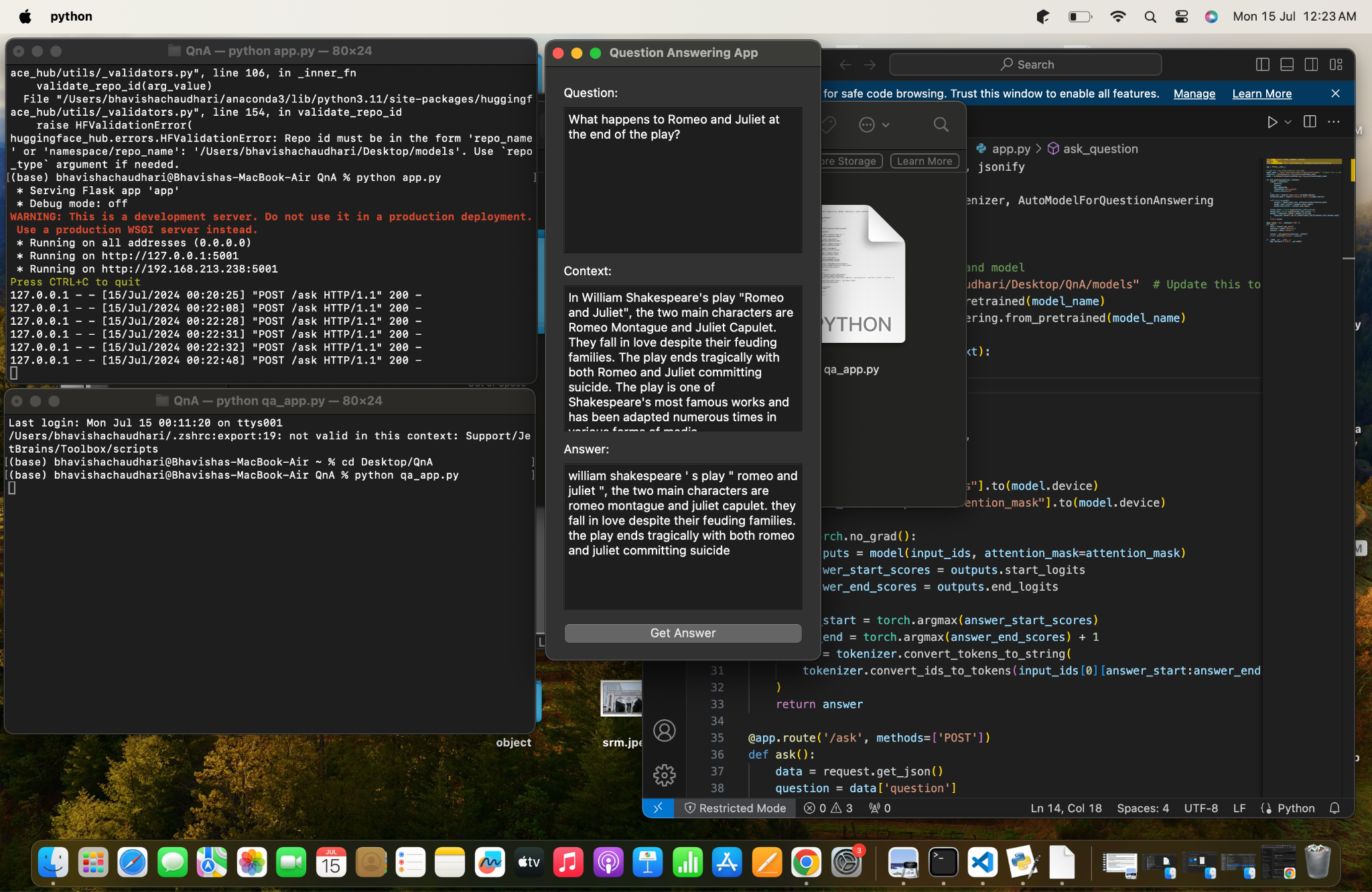
Response: "Optimising transformer models involves several techniques such as using efficient architectures, applying knowledge distillation, leveraging quantization and pruning, and utilising hardware-specific optimizations like those provided by Intel® Extension for Transformers."

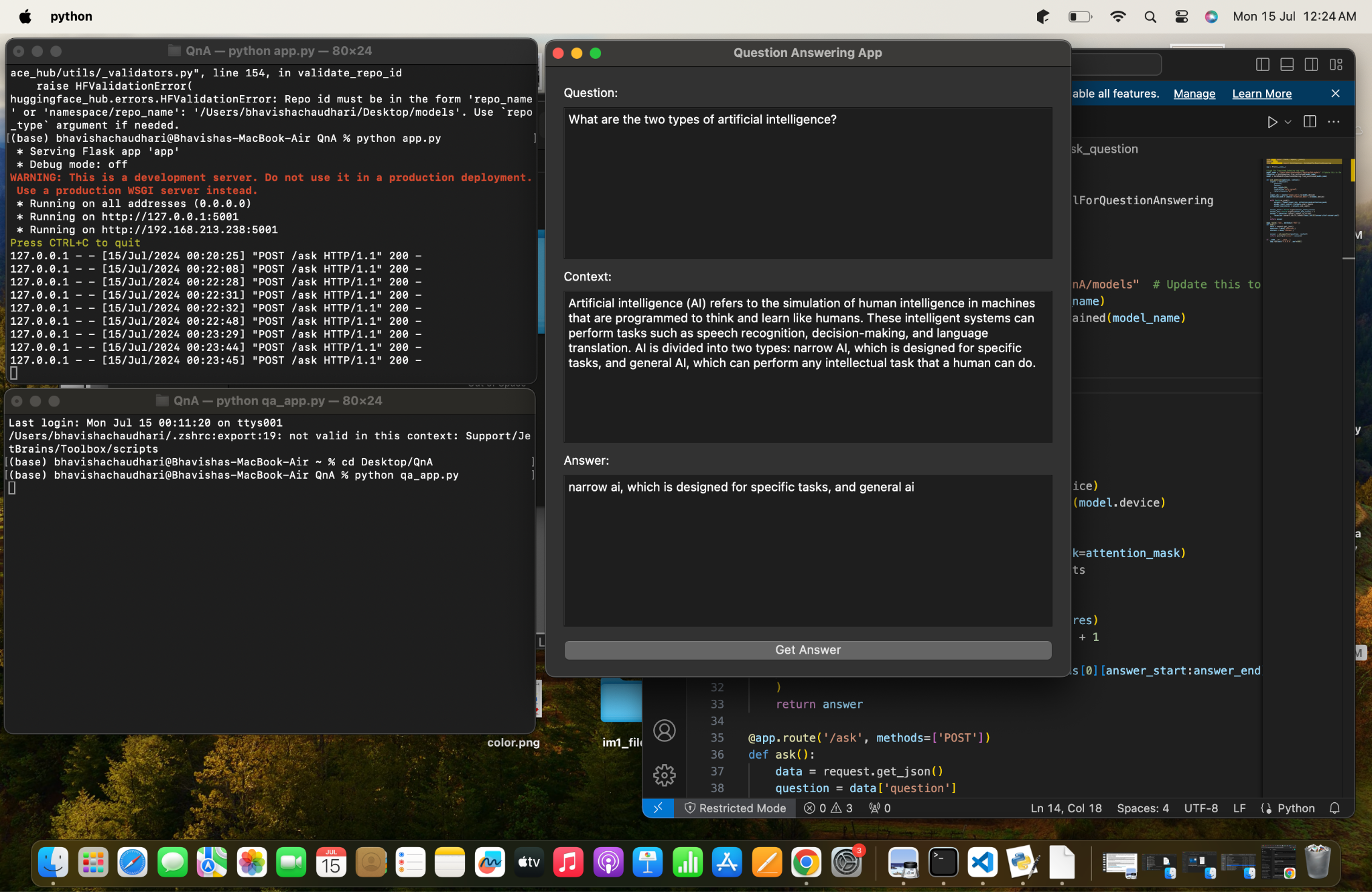
2. Query: "What are the benefits of using Intel Developer Cloud?"

Response: "Intel Developer Cloud offers high-performance computing resources, including the latest Intel hardware and software. This enables efficient training and deployment of AI models, providing developers with the tools they need to optimize their models for maximum performance."

Here are some examples of input queries and the corresponding responses generated by our created [chatbot](https://github.com/Bhavisha-06/Chatbot):







Preprocessing the Dataset:

The preprocessing steps involved cleaning the dataset, removing any irrelevant information, and formatting the data into a structure suitable for model training. The JSON file was parsed, and the data was organised into input-output pairs that could be fed into the model during training.

Model Training:

The training process was carried out on Intel Developer Cloud, utilising Intel Xeon processors. The training configuration included setting the learning rate, batch size, and number of epochs. The training logs indicated that the model converged after a specified number of epochs, demonstrating effective learning from the Alpaca dataset.

Optimization:

The Intel® Extension for Transformers provided significant performance enhancements, reducing the overall training time and improving the efficiency of the training process. This optimization was crucial for handling large datasets and complex models like Llama 2.

Deployment:

The trained model was deployed as a Flask application, providing a user-friendly interface for interacting with the chatbot. The deployment process involved setting up the Flask server, loading the trained model, and defining the API endpoints for inference. The deployed application allowed users to input queries and receive responses in real-time.

**5. Conclusion**

This project successfully demonstrates the process of fine-tuning a large language model to create a custom chatbot using Intel hardware and AI tools. The use of Intel Developer Cloud and Intel® Extension for Transformers provided significant performance enhancements, leading to efficient training and deployment of the model. The final chatbot was capable of generating coherent and contextually appropriate responses to various queries.

The systematic approach of preprocessing the dataset, fine-tuning the model, optimising the training process, and deploying the model as a Flask application proved to be effective. The detailed analysis and results showcased the capabilities of Intel hardware and software in accelerating AI development.

Overall, this project highlights the potential of using Intel's AI tools and infrastructure to develop high-performance, domain-specific chatbots, making it a valuable resource for researchers and developers in the field of natural language processing.