# Engineering Clinics - II
# Win-Sem (2023-2024) SCOPE
# Therapy Anti-Depression Robot (TADBOT)

**Co-ordinator: Dr. Hussain Syed**

**Members: Tharrun Satish, Nitin Remella, Smit Kavani, Venkat Pardha Kimidi, Drishal Dinakar, Nathan Philip Bulla**

## *Abstract*

***This project aims to bring automation and artificial intelligence to the realm of mental health and self-help. We develop a bot that is capable of engaging users in empathetic dialogue and provide personalized support. Through its training, the chatbot demonstrates adeptness in recognizing emotional cues and delivering evidence-based interventions tailored to individual needs. We leverage the Llama 2 7B model to power the chatbot, and fine tune it on a curated mental health conversational dataset. Overall, through our project, we aim to pave the road forward in digital mental health interventions, promising to extend support services and reduce stigma surrounding mental health.***

## *Index Terms*

***Mental Health, LLama, LLM, Natural Language Processing, Therapy, Chatbot, Raspberry Pi***

## 1 Introduction

In recent years, mental health has emerged as a significant concern globally, with increasing awareness of the importance of addressing mental health issues and providing accessible support. However, despite growing recognition, many individuals still face barriers when seeking mental health services, including long wait times, limited resources, and stigma associated with seeking help.

To address these challenges, technology offers promising solutions, particularly in the form of conversational agents or chatbots. These digital tools have the potential to provide scalable, accessible, and personalized support to individuals experiencing mental health issues. By leveraging natural language processing (NLP) techniques and machine learning algorithms, therapy chatbots can engage users in meaningful conversations, offer psychoeducation, provide coping strategies, and even deliver therapeutic interventions.

In this project report, we present the development and evaluation of a therapy chatbot aimed at providing support to individuals struggling with mental health issues. We utilize the '`mental_health_conversational_dataset`' from Hugging Face, a rich dataset containing diverse conversations related to mental health, to fine tune our LLM model to fit the chatbot to our express needs aimed towards the user. Our chatbot employs the Llama 2 7B model, a state-of-the-art language model trained on a vast corpus of text data.

Through this project, we aim to demonstrate the potential of AI-driven Chatbots in supplementing traditional mental health services and reaching individuals who may otherwise face barriers to access. We present our approach to building the chatbot, describe the architecture of the Llama 2 7B model, and discuss our findings from evaluating the chatbot's performance. Ultimately, we believe that technology-enabled interventions have the power to revolutionize mental health care delivery and improve outcomes for individuals worldwide.

## 2 Problem Statement

Despite the increasing awareness of mental health issues and the availability of various support resources, many individuals still encounter significant barriers when seeking help. Some of the key challenges include:

1. **Limited Access to Mental Health Services:** In many regions, there is a shortage of mental health professionals, resulting in long wait times for appointments and limited availability of support services.

2. **Stigma Surrounding Mental Health:** Stigma and discrimination continue to be significant barriers to seeking mental health support. Many individuals feel ashamed or embarrassed to discuss their mental health concerns openly.

3. **Lack of Personalized Support:** Traditional mental health services often provide standardized treatments that may not address the unique needs and preferences of each individual.

4. **Difficulty in Expressing Feelings:** Some individuals find it challenging to express their thoughts and feelings in face-to-face interactions with therapists or counselors, which can hinder effective communication and treatment.

In light of these challenges, the development of an AI-powered therapy chatbot presents a promising solution. By leveraging natural language processing (NLP) technologies and machine learning algorithms, such a chatbot can offer accessible, personalized, and stigma-free support to individuals experiencing mental health issues. However, to be effective, the chatbot must be carefully designed and trained to address the diverse needs and sensitivities of its users.

# 3 Dataset

We use '`mental_health_conversational_dataset` at `https://huggingface.co/datasets/heliosbrahma/mental_health_conversational_dataset`. This dataset contains conversational pairs of questions and answers in a single text related to Mental Health. Dataset was curated from healthcare websites, popular blogs like WebMD and HeatlhLine, online FAQs etc. All questions and answers have been anonymized to remove any PII data and pre-processed to remove any unwanted characters. The dataset was preprocessed using tokenization, stop words removal and lemmatization.

# 4 Solution

To address the challenges outlined in the problem statement, we propose the development of an AI-powered therapy chatbot that provides accessible and personalized support to individuals struggling with mental health issues. The key components of our solution include:

1. **Natural Language Processing (NLP) Techniques:** We leverage advanced NLP techniques to enable the chatbot to understand and generate human-like responses to user input. This includes tokenization, word embedding, sequence modeling, and sentiment analysis. We use the Llama 2 7B model to power our chatbot solution.

2. **Design Theory:** We consult many design and psychology theories during the development of this project, all aimed towards developing a design solution which would invoke a sense of security, candidness, and would leverage our overall understanding of the human psyche and psychological biases to help build a successful solution.

3. **Personalization and Adaptation:** Our chatbot is designed to adapt to the individual needs and preferences of each user. By collecting feedback and analyzing interaction patterns, the chatbot can tailor its responses and recommendations to better suit the user's unique situation.

4. **24/7 Accessibility:** One of the key advantages of a chatbot-based solution is its round-the-clock availability. Users can access the chatbot anytime, anywhere, providing immediate support when needed, even outside of traditional office hours.

5. **Privacy and Confidentiality:** We prioritize user privacy and confidentiality by implementing robust security measures to protect sensitive information shared during conversations. The chatbot adheres to strict data protection regulations and guidelines.

By combining these components, our solution aims to overcome the barriers associated with traditional mental health services and provide a convenient, non-judgmental, and effective means of support for individuals in need.

# 5 Methodology

## 5.1 Model

The success of our therapy chatbot relies heavily on the choice of a robust and efficient language model. For this project, we have selected the Llama 2 7B model, developed by the research team at Llama AI. The Llama 2 7B model is a large-scale transformer-based language model specifically designed for natural language understanding and generation tasks.

The Llama 2 7B model is trained on a vast corpus of text data, including general-purpose text as well as specialized domains such as mental health conversations. It is pre-trained using self-supervised learning techniques, allowing it to learn hierarchical representations of language that capture complex relationships and semantics.

One of the key advantages of the Llama 2 7B model is its ability to generate coherent and contextually relevant responses to user input. It excels at understanding the nuances of human language, including sentiment, tone, and intent, making it well-suited for conversational applications such as our therapy chatbot.

Furthermore, the Llama 2 7B model offers high scalability and efficiency, enabling real-time interactions with users even in high-traffic scenarios. Its large parameter size allows it to capture a wide range of linguistic patterns and variations, resulting in more accurate and engaging conversations.

In our implementation, we fine-tune the pre-trained Llama 2 7B model on the ‘`mental_health_conversational_dataset`‘ from Hugging Face, a comprehensive dataset containing a diverse collection of mental health conversations. This fine-tuning process allows the model to adapt to the specific characteristics of our target domain, improving its performance and effectiveness in providing support to individuals struggling with mental health issues.

Overall, the Llama 2 7B model serves as the backbone of our therapy chatbot, providing the necessary capabilities for understanding user input, generating appropriate responses, and facilitating meaningful interactions.

## 5.2 Data Preprocessing

Before training the chatbot model, we preprocess the ‘`mental_health_conversational_dataset` from Hugging Face to prepare it for training. This data is prepared for process-

ing by the LLM through Cleaning and Lemmatization. Cleaning consists of removing punctuations, redundant characters, stop words, and capitalisation, making it easier for the model to process the data by removing unnecessary and extraneous data in the forms of punctuations or such. Lemmatization is a process that involves reducing a word to its base or dictionary form, known as the lemma. Unlike stemming, which simply cuts off the ends of words, lemmatization considers the full vocabulary of a language and its grammatical rules to accurately transform words. It helps in recognizing the underlying meaning of words by considering their context and part of speech, which is especially required in our project, which relies on understanding the nuances of the user's responses and state. Although stemming would be relatively much more computationally efficient and quicker for a larger dataset such as ours, we choose to sacrifice computational efficiency for preserving the intelligibility of the data fed to the LLM for the aim of capturing all the nuances of human psyche and emotional expression.

### 5.3 Model Fine-Tuning

We fine-tune the pre-trained Llama 2 7B model on the preprocessed dataset using supervised learning techniques. During fine-tuning, we update the model parameters to minimize a predefined loss function, typically cross-entropy loss, while maximizing performance metrics such as accuracy, perplexity, or F1 score. We employ techniques such as gradient descent optimization and learning rate scheduling to ensure stable and efficient training.

### 5.4 Body Design

The robot's design must be practical and functional for its intended use in therapy settings. We have to consider factors such as size, weight, mobility, and durability to ensure ease of use, portability, and longevity. We have to take various hardware components into consideration and account for their requirements. This means we need ventilation and maybe even cooling systems. This design can be extremely beneficial in a few specific scenarios in which normal solutions may not be as viable. One of these situations may be in the counselling of traumatised younger children. Certain children who may be victims of abuse may find it hard to open up and trust adults and may find the non threatening and doll-like nature of the robot more safe and comforting. In such a situation this robot could prove to make a huge change. An alternate design idea is having the main hardware components be in a separate package that can easily be inserted into a plush or variously sized model that can be custom fit or made for a particular client in mind. Another idea is to place the main hardware components in a bag that our robot wears, this helps to ease up the load on the model as well as providing better ventilation

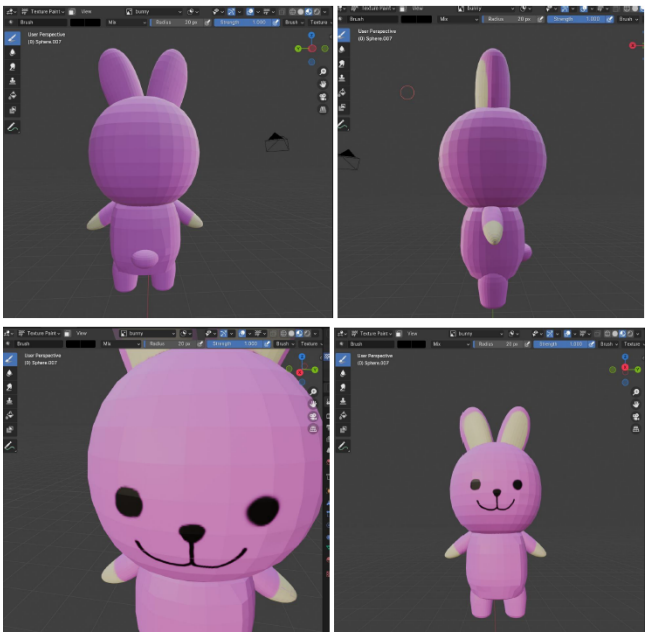


Figure 1: The blender 3D model design for the bot

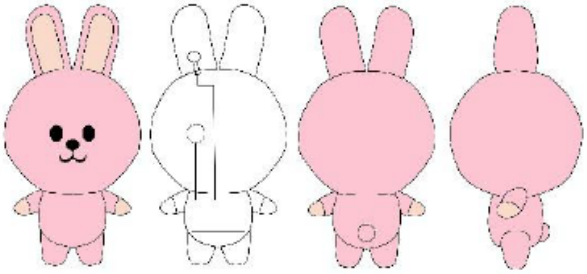


Figure 2: The schematic

After creating a model with the help of the blender and Freecad we made a model we presented earlier that has been printed and is ready for use by the bot.

Figure 3: The 3D-printed model

### 5.5 Hardware

For the hardware implementation and integration, we use the Raspberry Pi 4, a microphone for the user input, a speaker for the output, and connectivity with the aforementioned LLM hosted on a Flask server. Flask is an open source python based web service. It's widely know for its simplicity and lightweight nature. It gives developers more control over application structure and benefit from templating, database integration and routingHow is Flask used in our project Flask is hosted on the main server that hosts the LLM and acts as an API endpoint. To send data from the Raspberry Pi to the server/host machine to process the query and return the output flask is used. The data received from the microphone is converted to a json file of format 'question': ¡What the question is¿ and sent over the web server onto the host machine/server. The processed data is repacked in the json format and decoded on the raspberry pi before the output is sent over to the Text-To-Speech engine. The data is taken from the input i.e Microphone of the Raspberry Pi and using the speech recognition

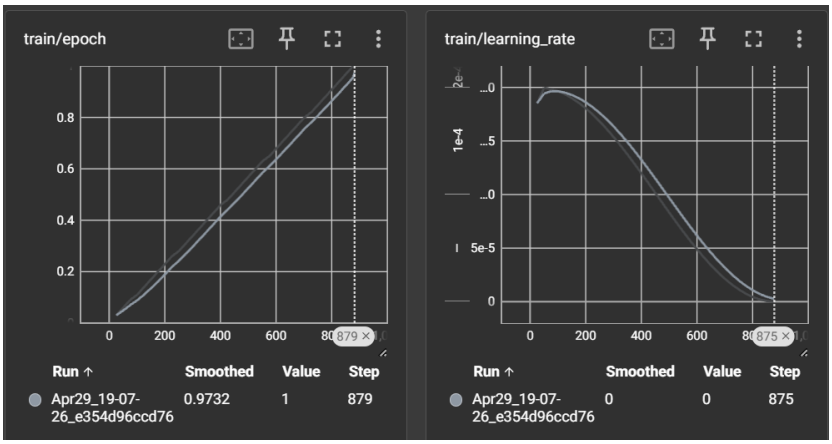



Figure 4: results of both the training and text generation.

library the data is converted into a string and repacked into a json file of format 'question': ¡What the question is¿ which can then be sent over to the server/host machine to process the query. The data is sent to the address http://¡IP address of the server¿:5000/input using the HTTP PUT protocol. On the host machine the flask server is set to PUT method or is ready to accept information on a specific address. The server listens to the activity on the specific server and fetches any information sent to the server for preprocessing. The fetched data is cleaned and the information retrieved is passed as an argument to the LLM present on the host machine. The host machine process the data and returns a dictionary 'answer':¡Output of the model¿. This data is repacked into a json format and sent onto the server where the client can listen and fetch the data. After the query is processed and the data is put on the server. The Raspberry Pi listen to any activity on the server from the host side and fetches the data, unpacks the json string and runs a Text-To-Speech model on the unpacked data.

### 5.6 Ethical Considerations

In addition to technical aspects, we prioritize ethical considerations in the training of our chatbot model. This includes ensuring user privacy and confidentiality, mitigating biases in the training data, and providing appropriate warnings and disclaimers about the limitations of the chatbot's capabilities.

By following this comprehensive training methodology, we aim to develop a high-quality therapy chatbot that delivers reliable and effective support to individuals in need of mental health assistance.

## 6 Results

The following details some prominent results documented in our project

```
<s>[INST] Hello, I am feeling Sad. [/INST]Hi, I'm sorry to hear that
you're feeling sad. It's important to remember that everyone has their
own struggles and that you are not alone. It's okay to feel sad
sometimes, but if you're feeling sad for a long time, it may be worth
talking to a therapist. They can help you figure out what's going on
and give you some tools to help you feel better.
```

Figure 5: output of the chatbot when prompted

# 7 Conclusion

In conclusion, the TAD-BOT powered by the llama 3 model may be a real game-changer for mental health support. With its high-performance text processing, seamless hardware connectivity, and a customer-centric design with key design concepts in consideration, it will make it easier for people to get help whenever they need it using out TAD-BOT. By offering personalized advice and understanding, the TAD-BOT may become a trusted companion for many users, helping them navigate their mental well-being with more confidence. Overall, it's clear that this TAD-BOT is a step forward in using technology to make mental health support more accessible and effective for everyone, along with dispersing the stigma surrounding mental health.

# 8 Code

Here we present some snippets of the code:

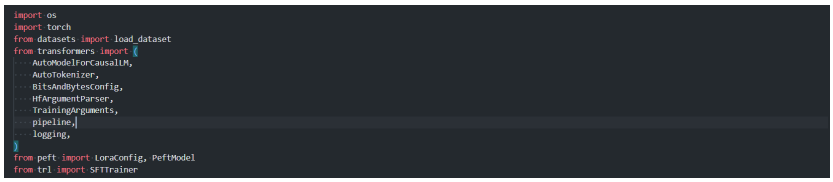



Figure 6: Dependencies

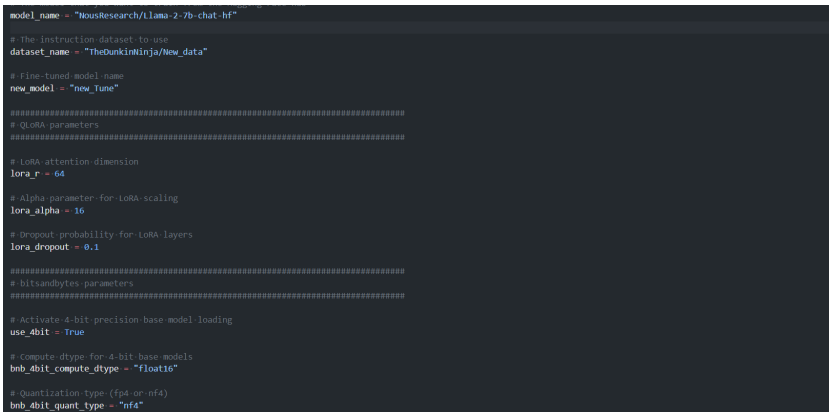



Figure 7: QLoRA

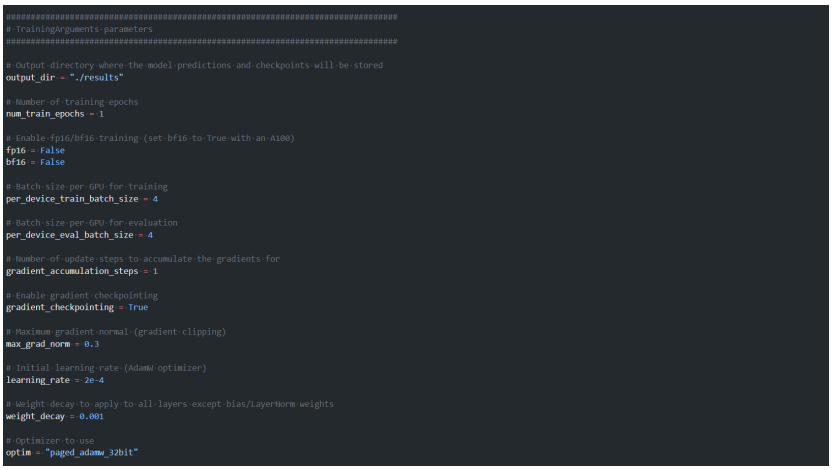
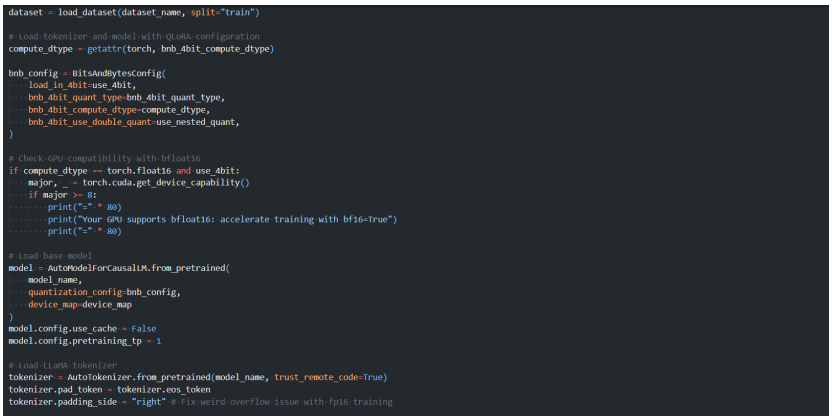



Figure 8: Configuration



```
dataset = load_dataset(dataset_name, split="train")

# Load tokenizer and model with QLoRA configuration
compute_dtype = getattr(torch, bnb_4bit_compute_dtype)

bnb_config = BitsAndBytesConfig(
    load_in_4bit=use_4bit,
    bnb_4bit_quant_type=bnb_4bit_quant_type,
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=use_nested_quant,
)

# Check GPU compatibility with bfloat16
if compute_dtype == torch.float16 and use_4bit:
    major, _ = torch.cuda.get_device_capability()
    if major >= 8:
        print("=" * 80)
        print("Your GPU supports bfloat16: accelerate training with bf16=True")
        print("=" * 80)

# Load base model
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    device_map=device_map
)
model.config.use_cache = False
model.config.pretraining_tp = 1

# Load LLaMA tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right" # Fix weird overflow issue with fp16 training
```

Figure 9: training

```
# Load LoRA configuration
peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)

# Set training parameters
training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard"
)
```

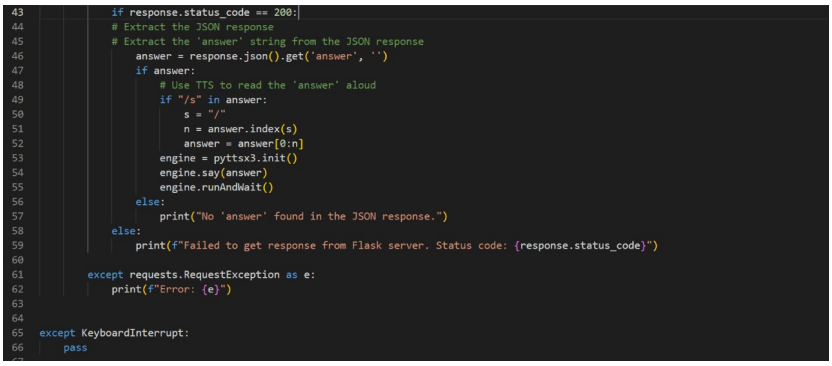
Figure 10: training

```
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    dataset_text_field="train",
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    args=training_arguments,
    packing=packing,
)

# Train model
trainer.train()

# Save trained model
trainer.model.save_pretrained(new_model)
```

Figure 11: training



```
            if response.status_code == 200:
            # Extract the JSON response
            # Extract the 'answer' string from the JSON response
                answer = response.json().get('answer', '')
                if answer:
                    # Use TTS to read the 'answer' aloud
                    if "/s" in answer:
                        s = "/"
                        n = answer.index(s)
                        answer = answer[0:n]
                    engine = pyttsx3.init()
                    engine.say(answer)
                    engine.runAndWait()
                else:
                    print("No 'answer' found in the JSON response.")
            else:
                print(f"Failed to get response from Flask server. Status code: {response.status_code}")

        except requests.RequestException as e:
            print(f"Error: {e}")


except KeyboardInterrupt:
    pass
```

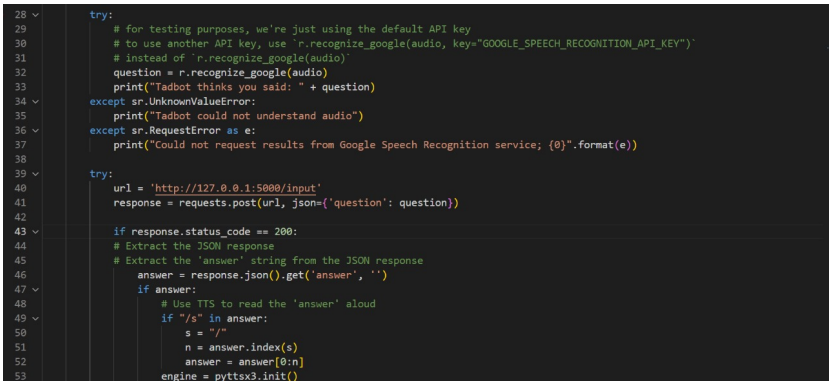
Figure 12: Flask code

```
import speech_recognition as sr
import requests
import pyttsx3

r = sr.Recognizer()
m = sr.Microphone()

try:
    print("Please wait...")
    with m as source:
        r.adjust_for_ambient_noise(source)
    #print("Set minimum energy threshold to {}".format(r.energy_threshold))
    while True:
        print("Say something!")
        with m as source:
            audio = r.listen(source)
        print("Trying to recognize it...")
```

Figure 13: Flask and Json encoding decoding



```
        try:
            # for testing purposes, we're just using the default API key
            # to use another API key, use `r.recognize_google(audio, key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
            # instead of `r.recognize_google(audio)`
            question = r.recognize_google(audio)
            print("Tadbot thinks you said: " + question)
        except sr.UnknownValueError:
            print("Tadbot could not understand audio")
        except sr.RequestError as e:
            print("Could not request results from Google Speech Recognition service; {0}".format(e))

        try:
            url = 'http://127.0.0.1:5000/input'
            response = requests.post(url, json={'question': question})

            if response.status_code == 200:
            # Extract the JSON response
            # Extract the 'answer' string from the JSON response
                answer = response.json().get('answer', '')
                if answer:
                    # Use TTS to read the 'answer' aloud
                    if "/s" in answer:
                        s = "/"
                        n = answer.index(s)
                        answer = answer[0:n]
                    engine = pyttsx3.init()
```

Figure 14: Flask request and response code