Projects\S3LR_Bot.py

```
import telegram
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters, PicklePersistence, CallbackQueryHandler, ConversationHandler
import openai
import pytube
import os
# Set up Telegram bot
bot = telegram.Bot(token='5642356671:AAH1UpD-kTBE-xPXVwA5VxZZ-FSdcc2-gbI')

# Set up OpenAI API
openai.api_key = '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'

# Handle incoming messages
def handle_message1(update, context):
    message = update.message.text

    # Send message to OpenAI API
    response = openai.Completion.create(
        engine='text-davinci-003',
        prompt=message,
        max_tokens=1000,
        temperature=0.7
    )

    # Retrieve OpenAI response
    generated_text = response.choices[0].text.strip()

    # Send response back to user
    context.bot.send_message(chat_id=update.effective_chat.id, text=generated_text)

# Set up message handler
def start(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text="""I'm a ChatGPT bot. My Fuctionalities are
    I work effectively like ChatGpt where you can ask me any question and it will answer for you.
    I can also help you make your todo list.
    You can send me the Youtube link and i will download the video for you.
    you can find below commands to have more information about creators.
    /contributors - To get the information of contributors for this Bot
    /JSPM - To know about JSPM college.
```

```
    /help - To use the todo list.
    Type /video and provide the link further to download it.
    /bmi - the use same command and use space and use arguments to get bmi""")
# To use the todo list
def help(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text="""
    /create - to create a todo (use /create "and your todo for today")
    /view - to view the todo
    /clear - to clear the todo
    """)

def JSPM(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text="""JSPM's Tathawde Branch Link =>\
    https://www.jspmrscoe.edu.in/""")

def contributors(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text="""Available Commands :-
    /Sahil - to access linkdin profile of Sahil
    /Loukik - to access linkdin profile of Loukik
    /Shounak - to access linkdin profile of Shounak
    /Siddharth - to access linkdin profile of Siddharth
    /Riya - to access linkdin profile of Riya""")


def Sahil(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text=
        "LinkedIn URL => \
        https://www.linkedin.com/in/sahilarankalle/")

def Loukik(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text=
        "LinkedIn URL => \
        https://www.linkedin.com/in/loukik-sancheti-b3a43125b/")

def Shounak(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text=
        "LinkedIn URL => \
        https://www.linkedin.com/in/shounak-sanpurkar-159832254/")

def Siddharth(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text=
        "LinkedIn URL => \
```

```
        https://www.linkedin.com/in/siddharth-surana-97383a259/")

def Riya(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text=
        "LinkedIn URL => \
        https://www.linkedin.com/in/riya-gharat-31a984259/")




# Define conversation states
WEIGHT, HEIGHT = range(2)

# Handle incoming messages
def handle_message2(update, context):
    message = update.message.text

    # Check if the message is a command
    if message.startswith('/'):
        command, *args = message[1:].lower().split()

        if command == 'bmi':
            if len(args) < 2:
                context.bot.send_message(chat_id=update.effective_chat.id, text="Please provide both weight(kg) and height(m) as arguments.")
                return

            try:
                weight = float(args[0])
                height = float(args[1])
            except ValueError:
                context.bot.send_message(chat_id=update.effective_chat.id, text="Invalid weight or height. Please provide valid numbers.")
                return

            bmi = calculate_bmi(weight, height)

            context.bot.send_message(chat_id=update.effective_chat.id, text=f"Your BMI is: {bmi}")
            context.bot.send_message(chat_id=update.effective_chat.id, text="Thank you for using the BMI Calculator Bot!")

        else:
            context.bot.send_message(chat_id=update.effective_chat.id, text="Invalid command. Please use /help for available commands.")
```

```
    else:
        handle_message1(update, context)  # Forward non-command messages to the handle_message1 function for OpenAI API processing


def calculate_bmi(weight, height):
    bmi = weight / (height ** 2)
    return round(bmi, 2)

def handle_weight(update, context):
    message = update.message.text

    # Parse the weight entered by the user
    try:
        weight = float(message)
    except ValueError:
        context.bot.send_message(chat_id=update.effective_chat.id, text="Invalid weight. Please enter a valid number.")
        return

    # Request the user to enter their height
    context.bot.send_message(chat_id=update.effective_chat.id, text="Please enter your height (in meters):")
    context.user_data['weight'] = weight
    return HEIGHT

def handle_height(update, context):
    message = update.message.text

    # Parse the height entered by the user
    try:
        height = float(message)
    except ValueError:
        context.bot.send_message(chat_id=update.effective_chat.id, text="Invalid height. Please enter a valid number.")
        return

    # Retrieve weight from user data
    weight = context.user_data.get('weight')

    # Calculate BMI
    bmi = calculate_bmi(weight, height)

    # Send BMI result to user
    context.bot.send_message(chat_id=update.effective_chat.id, text=f"Your BMI is: {bmi}")
```

```
    context.bot.send_message(chat_id=update.effective_chat.id, text="Thank you for using the BMI Calculator Bot!")

    # End the conversation
    return ConversationHandler.END

# Todo list functionality

def create_todo(update, context):
    todo_text = ' '.join(context.args)
    context.user_data.setdefault('todos', []).append(todo_text)
    context.bot.send_message(chat_id=update.effective_chat.id, text='Todo created successfully.')

def view_todos(update, context):
    todos = context.user_data.get('todos', [])
    if todos:
        todos_str = '\n'.join(todos)
        context.bot.send_message(chat_id=update.effective_chat.id, text=f'Your Todos:\n{todos_str}')
    else:
        context.bot.send_message(chat_id=update.effective_chat.id, text='You have no todos.')

def clear_todos(update, context):
    context.user_data['todos'] = []
    context.bot.send_message(chat_id=update.effective_chat.id, text='Todos cleared successfully.')

 #YOUTUBE music downloader

#YOUTUBE music downloader

# Telegram Bot Token
TOKEN = '5642356671:AAH1UpD-kTBE-xPXVwA5VxZZ-FSdcc2-gbI'
bot = telegram.Bot(token=TOKEN)


def download_video(update, context):
    video_url = update.message.text
    try:
        youtube = pytube.YouTube(video_url)
        video = youtube.streams.first()
        #video.download('./downloads')
        import os#used for saving video in downloads of pc(152-158)

        # Get the path to the user's "Downloads" folder
```

```
        downloads_dir = os.path.join(os.path.expanduser("~"), "Downloads")

        # Download the video to the "Downloads" folder
        video.download(downloads_dir)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Video downloaded successfully!")
    except Exception as e:
        context.bot.send_message(chat_id=update.effective_chat.id, text=f"Error: {str(e)}")

def main():
    updater = Updater(TOKEN, use_context=True)
    updater = Updater(token='5642356671:AAH1UpD-kTBE-xPXVwA5VxZZ-FSdcc2-gbI',
persistence=PicklePersistence(filename='persistence.pkl'), use_context=True)
    dispatcher = updater.dispatcher
      # Register command handlers
    dispatcher.add_handler(CommandHandler("start", start))
    dispatcher.add_handler(CommandHandler("help", help))
    dispatcher.add_handler(CommandHandler("create", create_todo))
    dispatcher.add_handler(CommandHandler("view", view_todos))
    dispatcher.add_handler(CommandHandler("clear", clear_todos))
    dispatcher.add_handler(CommandHandler('Sahil', Sahil))
    dispatcher.add_handler(CommandHandler('Loukik', Loukik))
    dispatcher.add_handler(CommandHandler('Shounak', Shounak))
    dispatcher.add_handler(CommandHandler('Siddharth', Siddharth))
    dispatcher.add_handler(CommandHandler('Riya', Riya))
    dispatcher.add_handler(CommandHandler('JSPM', JSPM))
    dispatcher.add_handler(CommandHandler('contributors', contributors))
    dispatcher.add_handler(MessageHandler(Filters.text & ~Filters.command, handle_message1))
    dispatcher.add_handler(CommandHandler('video', download_video))
        # Define conversation handler
    conversation_handler = ConversationHandler(
        entry_points=[CommandHandler('BMI', handle_message2)],
        states={
            WEIGHT: [MessageHandler(Filters.text & ~Filters.command, handle_weight)],
            HEIGHT: [MessageHandler(Filters.text & ~Filters.command, handle_height)],
        },
        fallbacks=[CommandHandler('BMI', handle_message2)]
    )

    # Add conversation handler to dispatcher
    dispatcher.add_handler(conversation_handler)
    updater.start_polling()
    updater.idle()
```

```
245
246 if __name__ == '__main__':
247     main()
```