# ДОДАТОК А

Програмна реалізація чат-бота

## CONFIGURATION FILE

# Bot session name
session_name_bot = "sessions/bot"

# Follow this guide to obtain your API credentials:
# https://core.telegram.org/api/obtaining_api_id
API_ID = "21550467"
API_HASH = "9d5259ace5ea1cc2a4c1e992439e0539"

# Write to @BotFather in Telegram to create a bot token
BOT_TOKEN = "6779129584:AAH-ts1btK5RpPqLle_m3HiL85OMWaUO1HM"

# The ID of the OpenAI model to use for generating responses.
model_engine = "gpt-3.5-turbo"

# Get your API_KEY by creating an OpenAi account and the go here:
# https://platform.openai.com/account/api-keys
openai_key = "sk-proj-5LzR0mjTdBItK2CpMYr0T3BlbkFJeSAklKrvHHjQE6zLDi0s"

# Importing Libraries
import telethon # Library to interact with Telegram's API as a user or through a bot account
from telethon.tl.custom import Button
from telethon import TelegramClient, events

import asyncio # Provides infrastructure for writing asynchronous code using coroutines.

import Config # Custom file containing configuration settings for the bot.

import openai # Python module that provides an interface to the OpenAI API.

# Configure OpenAI API key

openai.api_key = Config.openai_key

# Configure Telegram client

client = TelegramClient(Config.session_name_bot, Config.API_ID, Config.API_HASH).start(bot_token=Config.BOT_TOKEN)

# Define button templates

keyboard_stop = [[Button.inline("Завершити розмову", b"stop")]]

MAIN_MENU = [

[Button.inline("Вітаміни", b"vitamins"), Button.inline("Продукти", b"products")],

[Button.inline("Раціон на тиждень", b"weekly_diet")],

[Button.inline("Поставити своє запитання", b"ask_question")]

]

ROOT_BUTTON = Button.inline("Головне меню", b"root")

# Define helper function to retrieve a message from a conversation and handle button clicks

async def send_question_and_retrieve_result(prompt, conv, keyboard=None):

"""

Sends a question to the user and retrieves their response.

Args:

prompt (str): The question to ask the user.

conv (telethon.client.conversations.Conversation): The conversation object to use for sending the message.

keyboard (list): The keyboard to send with the message.

Returns:

Tuple[Union[events.callbackquery.CallbackQuery.Event, str], telethon.types.Message]: A tuple containing the user's response and the message object.

"""

# Send the prompt with the keyboard to the user and store the sent message object

message = await conv.send_message(prompt, buttons=keyboard)

loop = asyncio.get_event_loop()

task1 = loop.create_task(

conv.wait_event(events.CallbackQuery())

)

task2 = loop.create_task(

conv.get_response()

)

# Wait for the user to respond or tap a button using asyncio.wait()

done, _ = await asyncio.wait({task1, task2}, return_when=asyncio.FIRST_COMPLETED)

```
        # Retrieve the result of the completed coroutine and delete the sent message
        result = done.pop().result()
        await message.delete()

        # Return the user's response or None if they tapped a button
        if isinstance(result, events.CallbackQuery.Event):
            return None
        else:
            return result.message.strip()

    # Define the main chatbot handler
    @client.on(events.NewMessage(pattern="(?i)/start"))
    async def handle_start_command(event):
        SENDER = event.sender_id
        prompt = "Вітаю 🤖! Я NUTRIHELPER – твій помічник у справах здорового харчування."
        try:
            # Greet the user
            await client.send_message(SENDER, prompt, buttons=MAIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці команди /start.</b>", parse_mode='html')

    # Vitamins
```

```
# Define vitamins subcategories keyboard
VITAMIN_MENU = [
    [
        Button.inline("Жиророзчинні", b"fat_soluble"),
        Button.inline("Водорозчинні", b"water_soluble")
    ],
    [ROOT_BUTTON]
]

@client.on(events.CallbackQuery(data=b'vitamins'))
async def handle_vitamins_category(event):
    SENDER = event.sender_id
    try:
        # Send a message with the new menu options for vitamins category
        await client.send_message(SENDER, "Оберіть тип вітаміну та отримайте список із 10 продуктів, в яких він міститься:", buttons=VITAMIN_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER, "<b>Помилка при обробці вибору категорії вітамінів.</b>", parse_mode='html')

# Define event handler for fat-soluble vitamins category
# Define fat-soluble vitamins subcategories keyboard
fat_soluble_vitamins_keyboard = [
    [Button.inline("A", b"a"), Button.inline("E", b"e")],
    [Button.inline("K", b"k"), Button.inline("D", b"d")],
    [ROOT_BUTTON]
```

```
]

@client.on(events.CallbackQuery(data=b'fat_soluble'))
async def handle_fat_soluble_vitamins(event):
    SENDER = event.sender_id
    try:
        # Send a message with the options for fat-soluble vitamins
        await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=fat_soluble_vitamins_keyboard)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER, "<b>Помилка при обробці вибору категорії 'Жиророзчинні'.</b>", parse_mode='html')

# Define event handler for button A click
@client.on(events.CallbackQuery(data=b'a'))
async def handle_button_a(event):
    SENDER = event.sender_id
    # Ask the question about foods containing fat-soluble vitamin A
    question = "В яких продуктах міститься жиророзчинний вітамін А. Наведи список від 10 продуктів"

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=fat_soluble_vitamins_keyboard)

    except Exception as e:
```

```
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Define event handler for button E click
    @client.on(events.CallbackQuery(data=b'e'))
    async def handle_button_e(event):
        SENDER = event.sender_id
        # Ask the question about foods containing fat-soluble vitamin E
        question = "В яких продуктах міститься жиророзчинний вітамін Е. Наведи список від 10 продуктів"

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=fat_soluble_vitamins_keyboard)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return
```

```
# Define event handler for button K click
@client.on(events.CallbackQuery(data=b'k'))
async def handle_button_k(event):
    SENDER = event.sender_id
    # Ask the question about foods containing fat-soluble vitamin K
    question = "В яких продуктах міститься жиророзчинний вітамін K. Наведи список від 10 продуктів"

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=fat_soluble_vitamins_keyboard)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return

# Define event handler for button D click
@client.on(events.CallbackQuery(data=b'd'))
async def handle_button_d(event):
    SENDER = event.sender_id
    # Ask the question about foods containing fat-soluble vitamin D
    question = "В яких продуктах міститься жиророзчинний вітамін D. Наведи список від 10 продуктів"
```

```
        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=fat_soluble_vitamins_keyboard)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔ </b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return



    # Define fat-soluble vitamins subcategories keyboard
    WATER_SOLUBE_VITAMINS_MENU = [
        [Button.inline("B1", b"b1"), Button.inline("B2", b"b2")],
        [Button.inline("B5", b"b5"), Button.inline("C", b"c")],
        [Button.inline("B6", b"b6"), Button.inline("B9", b"b9")],
        [ROOT_BUTTON]
    ]
    @client.on(events.CallbackQuery(data=b'water_soluble'))
    async def handle_water_soluble_vitamins(event):
        SENDER = event.sender_id
        try:
            # Send a message with the options for water-soluble vitamins
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)
```

```
        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці вибору категорії 'Водорозчинні'.</b>", parse_mode='html')

    # Define event handler for button b1 click
    @client.on(events.CallbackQuery(data=b'b1'))
    async def handle_button_b1(event):
        SENDER = event.sender_id
        question = "В яких продуктах міститься водорозчинний вітамін B1 (тіамін). Наведи список від 10 продуктів"

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Define event handler for button b2 click
    @client.on(events.CallbackQuery(data=b'b2'))
    async def handle_button_b2(event):
```

```
        SENDER = event.sender_id
        question = "В яких продуктах міститься водорозчинний вітамін B2 (рибофлавін). Наведи список від 10 продуктів"

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Define event handler for button b5 click
    @client.on(events.CallbackQuery(data=b'b5'))
    async def handle_button_b5(event):
        SENDER = event.sender_id
        question = "В яких продуктах міститься водорозчинний вітамін B5 (пантотенова кислота). Наведи список від 10 продуктів"

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)
```

```
        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Define event handler for button C click
    @client.on(events.CallbackQuery(data=b'c'))
    async def handle_button_c(event):
        SENDER = event.sender_id
        # Ask the question about foods containing fat-soluble vitamin C
        question = "В яких продуктах міститься водорозчинний вітамін С (аскорбінова кислота). Наведи список від 10 продуктів"

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return
```

```
# Define event handler for button B6 click
@client.on(events.CallbackQuery(data=b'b6'))
async def handle_button_b6(event):
    SENDER = event.sender_id
    question = "В яких продуктах міститься водорозчинний вітамін B6 (піридоксин). Наведи список від 10 продуктів"

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return

# Define event handler for button B9 click
@client.on(events.CallbackQuery(data=b'b9'))
async def handle_button_b9(event):
    SENDER = event.sender_id
    question = "В яких продуктах міститься водорозчинний вітамін B9 (фолієва кислота). Наведи список від 10 продуктів"

    try:
```

```
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип вітаміну:", buttons=WATER_SOLUBE_VITAMINS_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Products
    # Define products subcategories keyboard
    PRODUCTS_MENU = [
        [Button.inline("Рослинного походження", b"plant_origin")],
        [Button.inline("Тваринного походження", b"animal_origin")],
        [ROOT_BUTTON]
    ]

    @client.on(events.CallbackQuery(data=b'products'))
    async def handle_products_category(event):
        SENDER = event.sender_id
        try:

            await client.send_message(SENDER, "Класифікація продуктів. Оберіть тип походження продукту:", buttons=PRODUCTS_MENU)

        except Exception as e:
```

```
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці вибору виду походження продукту.</b>", parse_mode='html')

# Plant Origin
PLANT_ORIGIN_MENU = [
    [
        Button.inline("Овочі", b"vegetables"),
        Button.inline("Фрукти", b"fruits"),
        Button.inline("Зернові", b"crops")
    ],
    [
        Button.inline("Бобові", b"beans"),
        Button.inline("Горіхи ", b"nuts"),
        Button.inline("Гриби", b"mushrooms")
    ],
    [
        Button.inline("Ягоди", b"berries"),
        Button.inline("Насіння ", b"seeds")
    ],
    [ROOT_BUTTON]
]

@client.on(events.CallbackQuery(data=b'plant_origin'))
async def handle_plant_origin(event):
    SENDER = event.sender_id
    try:
```

```
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці вибору типу продукту.</b>", parse_mode='html')



VEGETABLES_MENU = [
    [
        Button.inline("Листові", b"leaves_vegetables"),
        Button.inline("Кореневі", b"root_vegetables"),
        Button.inline("Плоди", b"yield_vegetables"),
    ],
    [ROOT_BUTTON]
]
@client.on(events.CallbackQuery(data=b'vegetables'))
async def handle_plant_origin(event):
    SENDER = event.sender_id
    try:

        await client.send_message(SENDER, "Оберіть тип походження овочей:", buttons=VEGETABLES_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
```

```
            await client.send_message(SENDER, "<b>Помилка при обробці вибору.</b>", parse_mode='html')

    @client.on(events.CallbackQuery(data=b'leaves_vegetables'))
    async def handle_button_leaves_vegetables(event):
        SENDER = event.sender_id
        question = "Надай список до 10 листяних овочів (салат, шпинат, рукола, капуста, базилік, петрушка, окріп). І в кожному пункті поясни чим корисний даний продукт."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть тип походження овочей:", buttons=VEGETABLES_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'root_vegetables'))
    async def handle_button_root_vegetables(event):
        SENDER = event.sender_id
        question = "Надай список 10 кореневих овочів. І в кожному пункті поясни чим корисний даний продукт."
```

```
try:
    await process_question(question, event)
    await client.send_message(SENDER, "Оберіть тип походження овочей:", buttons=VEGETABLES_MENU)

except Exception as e:
    # Something went wrong
    print("An error occurred:", e)
    await client.send_message(SENDER,
                              "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                              parse_mode='html')
    return

@client.on(events.CallbackQuery(data=b'yield_vegetables'))
async def handle_button_yield_vegetables(event):
    SENDER = event.sender_id
    question = "Надайте список до 10 овочів у вигляді плоду (таких як помідор, огірок). Не називай фрукти. І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть тип походження овочей:", buttons=VEGETABLES_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
```

```
                        "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                        parse_mode='html')
        return

@client.on(events.CallbackQuery(data=b'fruits'))
async def handle_button_fruits(event):
    SENDER = event.sender_id
    question = "Надай список 10 фруктів. І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                        "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                        parse_mode='html')
        return

@client.on(events.CallbackQuery(data=b'crops'))
async def handle_button_crops(event):
    SENDER = event.sender_id
```

```
    question = "Наведи список 10 зернових продуктів рослинного походження (рис, кукурудза). І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return

@client.on(events.CallbackQuery(data=b'beans'))
async def handle_button_beans(event):
    SENDER = event.sender_id
    question = "Наведи список 10 бобових продуктів рослинного походження. І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

    except Exception as e:
```

```
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'nuts'))
    async def handle_button_nuts(event):
        SENDER = event.sender_id
        question = "Наведи список 10 продуктів рослинного походження горіхового виду. І в кожному пункті поясни чим корисний даний продукт."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'mushrooms'))
    async def handle_button_mushrooms(event):
```

```
    SENDER = event.sender_id
    question = "Назви 10 корисних їстівних грибів для людини. І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return

@client.on(events.CallbackQuery(data=b'berries'))
async def handle_button_berries(event):
    SENDER = event.sender_id
    question = "Назви 10 корисних їстівних ягід для людини. І в кожному пункті поясни чим корисний даний продукт."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

    except Exception as e:
```

```
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'seeds'))
    async def handle_button_seeds(event):
        SENDER = event.sender_id
        question = "Назви 10 корисних їстівних насіння для людини. І в кожному пункті поясни чим корисний даний продукт."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=PLANT_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    # Animal Origin
    ANIMAL_ORIGIN_MENU = [
```

```
        [
            Button.inline("Риба", b"fish"),
            Button.inline("М’ясо", b"meat"),
            Button.inline("Яйця", b"eggs"),
            Button.inline("Молочні продукти", b"dairy")
        ],
        [ROOT_BUTTON]
    ]

    @client.on(events.CallbackQuery(data=b'animal_origin'))
    async def handle_animal_origin(event):
        SENDER = event.sender_id
        try:

            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=ANIMAL_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці вибору.</b>", parse_mode='html')

    @client.on(events.CallbackQuery(data=b'fish'))
    async def handle_button_fish(event):
        SENDER = event.sender_id
        question = "Назви 10 видів їстівної риби для людини. І в кожному пункті поясни чим корисний даний продукт"

        try:
```

```
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=ANIMAL_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'meat'))
    async def handle_button_meat(event):
        SENDER = event.sender_id
        question = "Назвіть 10 видів їстівного м'яса для людини. І в кожному пункті поясни чим корисний даний продукт."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=ANIMAL_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
```

```
                        parse_mode='html')
        return

    @client.on(events.CallbackQuery(data=b'eggs'))
    async def handle_button_eggs(event):
        SENDER = event.sender_id
        question = "Назвіть 10 видів їстівних яєць для людини. І в кожному пункті поясни чим корисний даний продукт."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=ANIMAL_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                        "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                        parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'dairy'))
    async def handle_button_dairy(event):
        SENDER = event.sender_id
        question = "Назвіть 10 видів молочних продуктів для людини. І в кожному пункті поясни чим корисний даний продукт."

        try:
```

```
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть вид походження продукту:", buttons=ANIMAL_ORIGIN_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

# Weekly Diet
WEEKLY_DIET_MENU = [
    [Button.inline("Баланс", b"balance")],
    [
        Button.inline("Схуднення", b"weight_loss"),
        Button.inline("Набір ваги", b"weight_gain")
    ],
    [
        Button.inline("Вегетаріанський", b"vegetarian"),
        Button.inline("Безлактозний", b"lactose_free")
    ],
    [ROOT_BUTTON]
]

@client.on(events.CallbackQuery(data=b'weekly_diet'))
async def handle_weekly_diet(event):
    SENDER = event.sender_id
```

```
        try:

            await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER, "<b>Помилка при обробці вибору.</b>", parse_mode='html')

    @client.on(events.CallbackQuery(data=b'weight_loss'))
    async def handle_button_weight_loss(event):
        SENDER = event.sender_id
        question = "Склади мені раціон харчування з трьох страв на кожен день тижня для втрати ваги."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return
```

```
@client.on(events.CallbackQuery(data=b'weight_gain'))
async def handle_button_weight_gain(event):
    SENDER = event.sender_id
    question = "Склади мені раціон харчування з трьох страв на кожен день тижня для набору ваги."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return

@client.on(events.CallbackQuery(data=b'balance'))
async def handle_button_balance(event):
    SENDER = event.sender_id
    question = "Склади мені раціон харчування з трьох страв на кожен день тижня для підтримки балансу."

    try:
        await process_question(question, event)
```

```
            await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
            return

    @client.on(events.CallbackQuery(data=b'vegetarian'))
    async def handle_button_vegetarian(event):
        SENDER = event.sender_id
        question = "Склади мені раціон харчування з трьох страв на кожен день тижня для вегетаріанця."

        try:
            await process_question(question, event)
            await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')
```

```
        return


@client.on(events.CallbackQuery(data=b'lactose_free'))
async def handle_button_lactose_free(event):
    SENDER = event.sender_id
    question = "Склади раціон харчування з трьох страв на кожен день тижня без продуктів, які містять лактозу."

    try:
        await process_question(question, event)
        await client.send_message(SENDER, "Оберіть раціон на тиждень:", buttons=WEEKLY_DIET_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER,
                                  "<b>Розмову завершено ✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
        return



# Define event handler for root button click
@client.on(events.CallbackQuery(data=b'root'))
async def handle_root_button(event):
    SENDER = event.sender_id
    try:
        # Send the previous menu back
```

```
        await client.send_message(SENDER, "Оберіть категорію:", buttons=MAIN_MENU)

    except Exception as e:
        # Something went wrong
        print("An error occurred:", e)
        await client.send_message(SENDER, "<b>Помилка при обробці вибору кнопки Головне меню.</b>", parse_mode='html')

# Ask Question
@client.on(events.CallbackQuery(data=b'ask_question'))
async def handle_ask_question(event):
    SENDER = event.sender_id
    prompt = "Будь ласка, задайте своє запитання Nutri Helper"

    try:

        # Start a conversation
        async with client.conversation(await event.get_chat(), exclusive=True, timeout=600) as conv:
            # Create an empty history to store chat history
            history = []

            # Keep asking for input and generating responses until the conversation times out or the user clicks the stop button
            while True:
                # Prompt the user for input
                prompt = "Будь ласка, задайте своє запитання Nutri Helper"
                user_input = await send_question_and_retrieve_result(prompt, conv, keyboard=keyboard_stop)
```

```
# Check if the user clicked the stop button
if user_input is None:
    # If the user clicked the stop button, send a prompt to reset the conversation
    prompt = "Розмову завершено. Натисни /start, щоб розпочати нову."
    await client.send_message(SENDER, prompt)
    break
else:
    # Send a "I'm thinking message..."
    prompt = "Дякую! Я думаю про відповідь..."
    thinking_message = await client.send_message(SENDER, prompt)

    # If the user did not click the stop button, generate a response using OpenAI API
    # Add the user input to the chat history
    history.append({"role": "user", "content": user_input})

    # Generate a chat completion using OpenAI API
    chat_completion = openai.ChatCompletion.create(
        model=Config.model_engine,  # ID of the model to use.
        messages=history,
        # The messages to generate chat completions for. This must be a list of dicts!
        max_tokens=1500,  # The maximum number of tokens to generate in the completion.
        n=1,  # How many completions to generate for each prompt.
```

```
                temperature=0.1
                # Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.
            )

            # Retrieve the response from the chat completion
            response = chat_completion.choices[0].message.content

            # Add the response to the chat history
            history.append({"role": "assistant", "content": response})

            # Delete the Thinking message
            await thinking_message.delete()

            # Send the response to the user
            await client.send_message(SENDER, response, parse_mode='Markdown')

    except asyncio.TimeoutError:
        # Conversation timed out
        await client.send_message(SENDER,
                                  "<b>Розмову завершено✔</b>\nПройшло занадто багато часу з моменту останньої відповіді. Натисни /start, щоб розпочати нову розмову.",
                                  parse_mode='html')
        return

    except telethon.errors.common.AlreadyInConversationError:
        # User already in conversation
```

```
            pass

        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(SENDER,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб розпочати нову розмову.",
                                      parse_mode='html')
            return


    async def process_question(question, event):
        try:
            async with client.conversation(await event.get_chat(), exclusive=True, timeout=600) as conv:
                # Create an empty history to store chat history
                history = []
                # Add the user input to the chat history
                history.append({"role": "user", "content": question})

                prompt = "Зачекайте на відповідь..."
                thinking_message = await client.send_message(event.sender_id, prompt)

                chat_completion = openai.ChatCompletion.create(
                    model=Config.model_engine,  # ID of the model to use.
                    messages=history,
                    # The messages to generate chat completions for. This must be a list of dicts!
```

```
                max_tokens=1500,  # The maximum number of tokens to generate in the completion.
                n=1,  # How many completions to generate for each prompt.
                temperature=0.1
                # Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.
            )

            # Retrieve the response from the chat completion
            response = chat_completion.choices[0].message.content

            # Add the response to the chat history
            history.append({"role": "assistant", "content": response})

            # Delete the Thinking message
            await thinking_message.delete()

            # Send the response to the user
            await client.send_message(event.sender_id, response, parse_mode='Markdown')
    except asyncio.TimeoutError:
        # Conversation timed out
        await client.send_message(event.sender_id,
                                  "<b>Розмову завершено✔</b>\nПройшло занадто багато часу з моменту останньої відповіді. Натисни /start, щоб повернутися у головне меню.",
                                  parse_mode='html')
    except telethon.errors.common.AlreadyInConversationError:
        # User already in conversation
        pass
```

```
        except Exception as e:
            # Something went wrong
            print("An error occurred:", e)
            await client.send_message(event.sender_id,
                                      "<b>Розмову завершено✔</b>\nВиникла помилка. Натисни /start, щоб повернутися у головне меню.",
                                      parse_mode='html')


    client.run_until_disconnected()  # Start the bot!
```

# ДОДАТОК Б

Відомість кваліфікаційної роботи

| Позначення | Найменування | Дод. відомості |
|---|---|---|
| | Текстові документи | |
| **1.** | Пояснювальна записка | 106 с. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | Інші документи | |
| | | |
| **2.** | Презентаційні матеріали | 10 плакатів |
| | | |
| | | |
| | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| . | . | Прізвище та ініціали. | Підп. | Дата | | | |
| Розробив | | Коваленко А.В. | | | Інформаційний чат-бот для здорового харчування | Шифр групи | Код напр./спец. |
| Перевірив | | Носова Я.В. | | | | СШІм-22-1 | 122 |
| Н.контр. | | Малєєва І.А. | | | | ХНУРЕ кафедра ШІ | |
| Затв. | | Філатов В.О. | | | | | |