ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

AI-АГЕНТ ДЛЯ ПЕРСОНАЛІЗОВАНОГО КОНСУЛЬТУВАННЯ ПРЕДСТАВНИКІВ ВЕЛОСИПЕДНОЇ СПІЛЬНОТИ

Кваліфікаційна робота
Перший(бакалаврський рівень)

Автор:
Нишкур Д.С.
ст.гр. КІУКІ-21-1

Керівник:
Корнієнко Є.Д.
ас. каф. ЕОМ

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

AI-АГЕНТ ДЛЯ ПЕРСОНАЛІЗОВАНОГО КОНСУЛЬТУВАННЯ ПРЕДСТАВНИКІВ ВЕЛОСИПЕДНОЇ СПІЛЬНОТИ

Кваліфікаційна робота
Перший(бакалаврський рівень)

Автор:
Нишкур Д.С.
ст.гр. КІУКІ-21-1

Керівник:
Корнієнко Є.Д.
ас. каф. ЕОМ

Мета і задача роботи

- Мета: Розробка AI-агента для персоналізованих консультацій представників велосипедної спільноти у Telegram. Завдання – подолати інформаційну перевантаженість та забезпечити швидкий, достовірний доступ до знань у зручному діалоговому форматі.
- Завдання до роботи:
- Аналіз предметної області та ШІ-технологій (LLM, ASR).
- Проектування архітектури та логіки агента.
- Реалізація ключових функціональних модулів (профіль, консультування, голосове введення).

2

Актуальність та аналіз предметної області

- Робота є актуальною через гостру потребу у персоналізованих експертних знаннях для нішевих спільнот,що стикаються з інформаційною перевантаженістю та складнощами у пошуку достовірних даних. У технологічному контексті, розвиток Штучного Інтелекту (AI), Великих Мовних Моделей (LLM) для розуміння та генерації мови, а також систем Автоматичного Розпізнавання Мовлення (ASR), дозволяє створити зручні діалогові системи. Telegram обрано завдяки його популярності та зручності для інтеграції.

3

Вибір месенджера

Характеристика	Telegram	Viber	WhatsApp
Відкритість API	Відкрите	Відкрите (реєстрація)	Обмежене
Модерація бота	Ні	Так	Так
Комунікаційні формати	Текст, медіа, кнопки	Текст, кнопки	Текст
Гнучкість у інтеграції	Висока	Середня	Низька
Використання в спільноті	Висока	Низька	Низька
Підтримка ШІ	Так	Обмежено	Через платні шлюзи

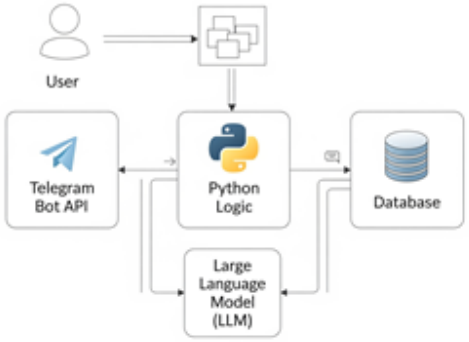
Технологічний Стек: Python та Ключові Бібліотеки

Чому Python?
Простота та читабельність коду
Велика екосистема для AI/ML/Data Science
Широкі можливості інтеграції (API)

Ключові Бібліотеки:
python-telegram-bot (Взаємодія з Telegram API)
SpeechRecognition (Розпізнавання мови)
pydub (Обробка аудіофайлів)
Request(для прямої взаємодії з OpenRouter API)
Sqlite3(для роботи з БД)

Загальна Архітектура Рішення

Architecture of AI Agent



- **Користувач ↔ Telegram Bot API:** Взаємодія через месенджер.
- **Telegram Bot API ↔ Python Logic:** Центральний модуль обробки запитів.
- **Python Logic ↔ Database:** Зберігання та керування профілями користувачів та їх даними.
- **Python Logic ↔ Large Language Model (LLM):** Обробка запитів та генерація інтелектуальних відповідей.

6

BMX Bot

Демонстрація Роботи / Інтерфейс Користувача

Початок взаємодії з ботом

Створення профілю

Редагування профілю

Інтелектуальне консультування (запит/відповідь)

7

Голосова взаємодія

ДОДАТОК Б
ЛІСТИНГ ТЕЛЕГРАМ-БОТА

Лістинг Б.1 Конфігурація API та ініціалізація бота

```
import telebot
import sqlite3
import requests
from telebot import types
import speech_recognition as sr
from pydub import AudioSegment
import os
import time

#Токени
TELEGRAM_TOKEN = "7347861811:AAEqMwHHxjql4nx-
9EZ4gKkJqEIufa0G_M8"
OPENROUTER_API_KEY = "sk-or-v1-
27942ba90265c1ab38ef2c9b21b23a95d52acf999fd5a5edbeed1c0bba05ebdb
"
#OpenRouter
OPENROUTER_URL = "https://openrouter.ai/api/v1/chat/completions"

MODEL_NAME = "google/gemini-2.0-flash-exp:free"

bot = telebot.TeleBot(TELEGRAM_TOKEN)
```

Лістинг Б.2 Створення, редагування та перегляд профілю

```
#Створення профілю
@bot.message_handler(func=lambda message: message.text == "💼
Створити профіль")
def create_profile(message):
    bot.send_message(message.chat.id, "Напиши своє ім'я:")
    bot.register_next_step_handler(message, set_name)

def set_name(message):
    name = message.text
    bot.send_message(message.chat.id, "Введи свій рівень:")
    bot.register_next_step_handler(message, set_level, name)

def set_level(message, name):
    level = message.text
    bot.send_message(message.chat.id, "Інтереси (наприклад,
стріт, парк, дьорт):")
    bot.register_next_step_handler(message, set_interests, name,
level)
```

```
def set_interests(message, name, level):
    interests = message.text
    bot.send_message(message.chat.id, "Стиль катання (наприклад,
фрістайл, рейсинг):")
    bot.register_next_step_handler(message, save_profile, name,
level, interests)

def save_profile(message, name, level, interests):
    style = message.text
    user_id = message.from_user.id
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()

    cursor.execute('INSERT OR REPLACE INTO profiles (user_id,
name, level, interests, style) VALUES (?, ?, ?, ?, ?)',
                   (user_id, name, level, interests, style))
    conn.commit()
    conn.close()
    bot.send_message(message.chat.id, "✅ Профіль створено!")

#Перегляд профілю
@bot.message_handler(func=lambda message: message.text == "📋
Переглянути профіль")
def view_profile(message):
    user_id = message.from_user.id
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('SELECT name, level, interests, style FROM
profiles WHERE user_id = ?', (user_id,))
    profile = cursor.fetchone()
    conn.close()
    if profile:
        bot.send_message(
            message.chat.id,
            f"👤 Ім'я: {profile[0]}\n🏆 Рівень: {profile[1]}\n🔥
Інтереси: {profile[2]}\n🛹 Стиль: {profile[3]}"
        )
    else:
        bot.send_message(message.chat.id, "☹ Профіль не
знайдено! Будь ласка, створіть його за допомогою кнопки '💼
Створити профіль'.")


#Редагування профілю
@bot.message_handler(func=lambda message: message.text == "✏
Редагувати профіль")
def edit_profile(message):
    user_id = message.from_user.id
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('SELECT 1 FROM profiles WHERE user_id = ?',
(user_id,))
```

```
    profile_exists = cursor.fetchone()
    conn.close()

    if profile_exists:
        bot.send_message(message.chat.id, "Введіть нове ім'я
(або поточне, якщо не змінюєте):")
        bot.register_next_step_handler(message,
set_name_for_edit)
    else:
        bot.send_message(message.chat.id, "☹ Профіль не знайдено
для редагування. Будь ласка, створіть його спочатку.")


def set_name_for_edit(message):
    name = message.text
    bot.send_message(message.chat.id, "Введіть новий рівень (або
поточний):")
    bot.register_next_step_handler(message, set_level_for_edit,
name)

def set_level_for_edit(message, name):
    level = message.text
    bot.send_message(message.chat.id, "Введіть нові інтереси
(або поточні):")
    bot.register_next_step_handler(message,
set_interests_for_edit, name, level)

def set_interests_for_edit(message, name, level):
    interests = message.text
    bot.send_message(message.chat.id, "Введіть новий стиль
катання (або поточний):")
    bot.register_next_step_handler(message, save_profile_edit,
name, level, interests)

def save_profile_edit(message, name, level, interests):
    style = message.text
    user_id = message.from_user.id
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('INSERT OR REPLACE INTO profiles (user_id,
name, level, interests, style) VALUES (?, ?, ?, ?, ?)',
                   (user_id, name, level, interests, style))
    conn.commit()
    conn.close()
    bot.send_message(message.chat.id, "✅ Профіль оновлено!")
```

Лістинг Б.3 Розпізнавання голосу

```
@bot.message_handler(content_types=['voice'])
def handle_voice(message):
    try:

        file_info = bot.get_file(message.voice.file_id)
        downloaded_file = bot.download_file(file_info.file_path)


        ogg_path = f"voice_{message.message_id}.ogg"
        with open(ogg_path, 'wb') as f:
            f.write(downloaded_file)


        wav_path = f"voice_{message.message_id}.wav"
        audio = AudioSegment.from_ogg(ogg_path)
        audio.export(wav_path, format="wav")


        recognizer = sr.Recognizer()
        with sr.AudioFile(wav_path) as source:
            audio = recognizer.record(source)


        text = recognizer.recognize_google(audio, language="uk-
UA")


        os.remove(ogg_path)
        os.remove(wav_path)


        message.text = text
        handle_message(message)

    except sr.UnknownValueError:
        bot.send_message(message.chat.id, "⚠️ Не вдалося розпізнати
мову. Будь ласка, спробуйте ще раз.")
    except sr.RequestError as e:
        bot.send_message(message.chat.id, f"⚠️ Помилка з сервісом
розпізнавання мови; {e}")
    except Exception as e:
        bot.send_message(message.chat.id, f"⚠️ Сталася невідома
помилка під час обробки голосу: {e}")
```

Лістинг Б.4 Основне повідомлення

```
@bot.message_handler(func=lambda message: True)
def handle_message(message):
        if message.text in ["💼 Створити профіль", "
```

```
Переглянути профіль", "✏ Редагувати профіль"]:
                return

    user_id = message.from_user.id
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('SELECT name, level, interests, style FROM
profiles WHERE user_id = ?', (user_id,))
    profile = cursor.fetchone()
    conn.close()

    if profile:
        profile_context = f"Ім'я: {profile[0]}\nРівень:
{profile[1]}\nІнтереси: {profile[2]}\nСтиль: {profile[3]}"
        user_message = message.text
        full_message = f"Профіль
користувача:\n{profile_context}\nПовідомлення: {user_message}"
        headers = {
            "Authorization": f"Bearer {OPENROUTER_API_KEY}",
            "Content-Type": "application/json",
            "HTTP-Referer": "https://t.me/velo_guide_bot",
            "X-Title": "BMX Guide Bot"
        }
        data = {
            "model": MODEL_NAME,
            "messages": [
                {"role": "system", "content": "Ви -
доброзичливий та корисний вело-гід з питань BMX. Надавайте
поради, враховуючи профіль користувача (рівень, інтереси,
стиль), щоб допомогти йому покращити навички, освоїти трюки та
розвиватися. Використовуйте нейтральний та теплий тон, уникаючи
звертань на 'ти'. Пам'ятайте про релевантність порад."},
                {"role": "user", "content": full_message}
            ]
        }
        try:
            response = requests.post(OPENROUTER_URL,
headers=headers, json=data)
            response.raise_for_status()  # Викликає HTTPError
для 4xx/5xx відповідей
            response_json = response.json()
            if 'choices' in response_json and
response_json['choices']:
                reply =
response_json['choices'][0]['message']['content']
            else:
                reply = "⚠ Вибачте, я не зміг отримати відповідь від
AI. Спробуйте ще раз пізніше."
                print(f"Помилка: Немає 'choices' у відповіді AI:
{response_json}")

        except requests.exceptions.HTTPError as e:
            if e.response.status_code == 429:
```

```
                retry_after = int(e.response.headers.get('Retry-
After', 5))
                bot.send_message(message.chat.id, f"⚠️ Сервіс
перевантажений (занадто багато запитів). Будь ласка, зачекайте
{retry_after} секунд і спробуйте ще раз.")

                print(f"Отримано 429. Затримка на {retry_after}
секунд.")
                time.sleep(retry_after) # Затримка перед
наступною можливою взаємодією
                return
            else:
                reply = f"⚠️ Сталася помилка при зверненні до AI:
{e}. Код статусу: {e.response.status_code}"
                print(f"HTTP Помилка: {e}") # Логування
        except requests.exceptions.ConnectionError as e:
            reply = f"⚠️ Помилка з'єднання з сервісом AI: {e}.
Перевірте ваше інтернет-з'єднання."
            print(f"Connection Error: {e}") # Логування
        except Exception as e:
            reply = f"⚠️ Сталася невідома помилка під час обробки запиту
AI: {e}"
            print(f"Загальна помилка при обробці AI запиту:
{e}") # Логування

        bot.send_message(message.chat.id, reply)
    else:
        bot.send_message(message.chat.id, "Будь ласка, спочатку
створіть або оновіть свій профіль через головне меню (кнопка '💼
Створити профіль').")

print("✅ Бот запущено...")
bot.polling(non_stop=True, interval=0)
```

Лістинг Б.5 База даних користувачів

```
def create_or_update_db():
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS profiles (
        user_id INTEGER PRIMARY KEY,
        name TEXT,
        level TEXT,
        interests TEXT,
        style TEXT
    )''')
    conn.commit()
    conn.close()

create_or_update_db()
```