**Інформаційні мережі та телекомінікації**

**Лабораторна робота №7.**

**Тема:** Адміністрування Telegram Bot: testing and monitoring issues.

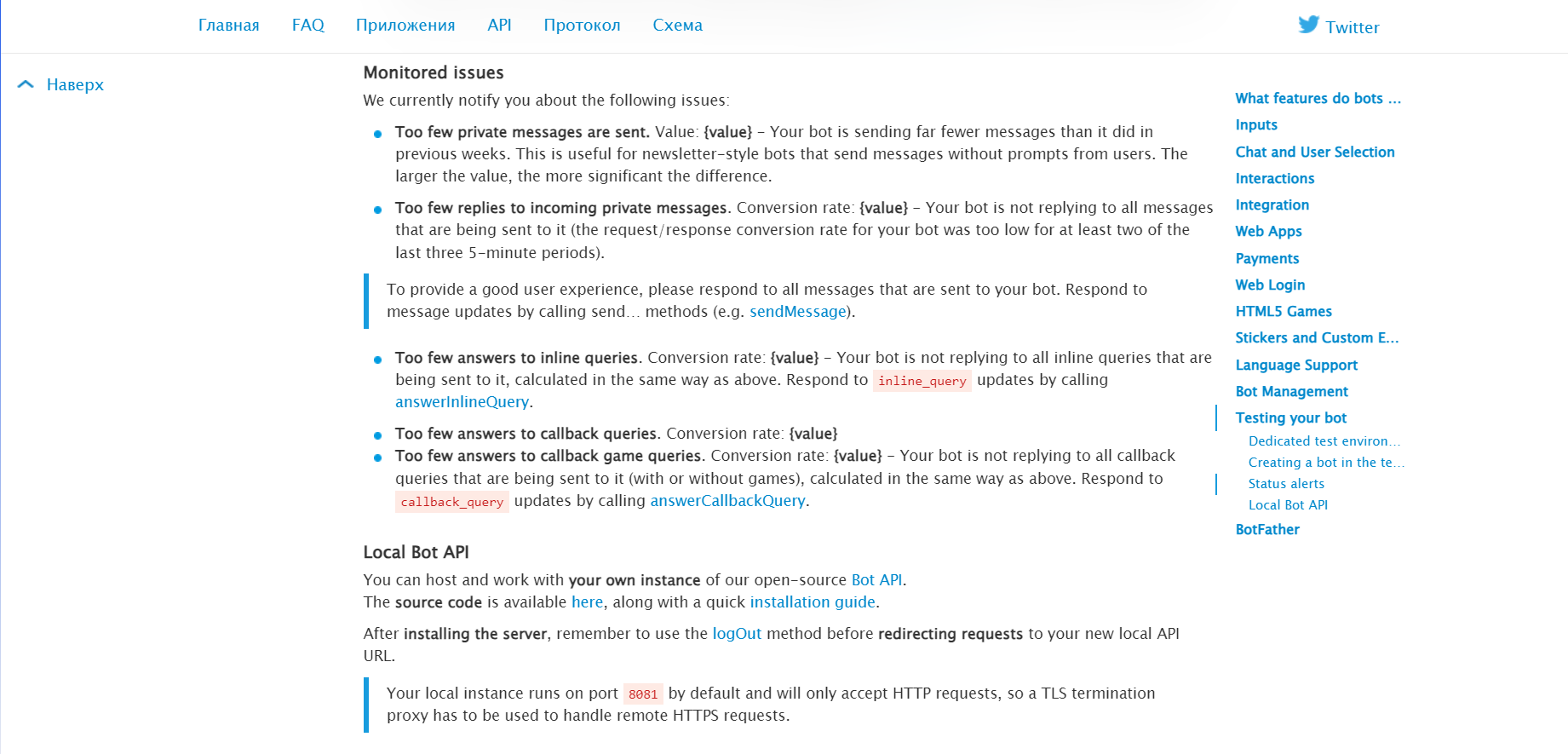
**Мета:** Навчитися адмініструвати телеграм бот.

**Хiд роботи**

**І. Виконайте наступні завдання** (у файлі звіту опишіть відповідні процеси):

Результат виконання цієї лабораторної закомітьте в гілку lab7 вашого проекту.

**Завдання 1.** Ознайомтеся з документацію по privacy mode налаштуваннях бота <https://core.telegram.org/bots/features#monitored-issues>



**Завдання 2.** Додати нову команду /deleteProfile яка б дозволяла користувачу видалити його профіль з системи. Використовуючи можливість відправки ботом у відповіді кнопок через reply\_markup і buttons

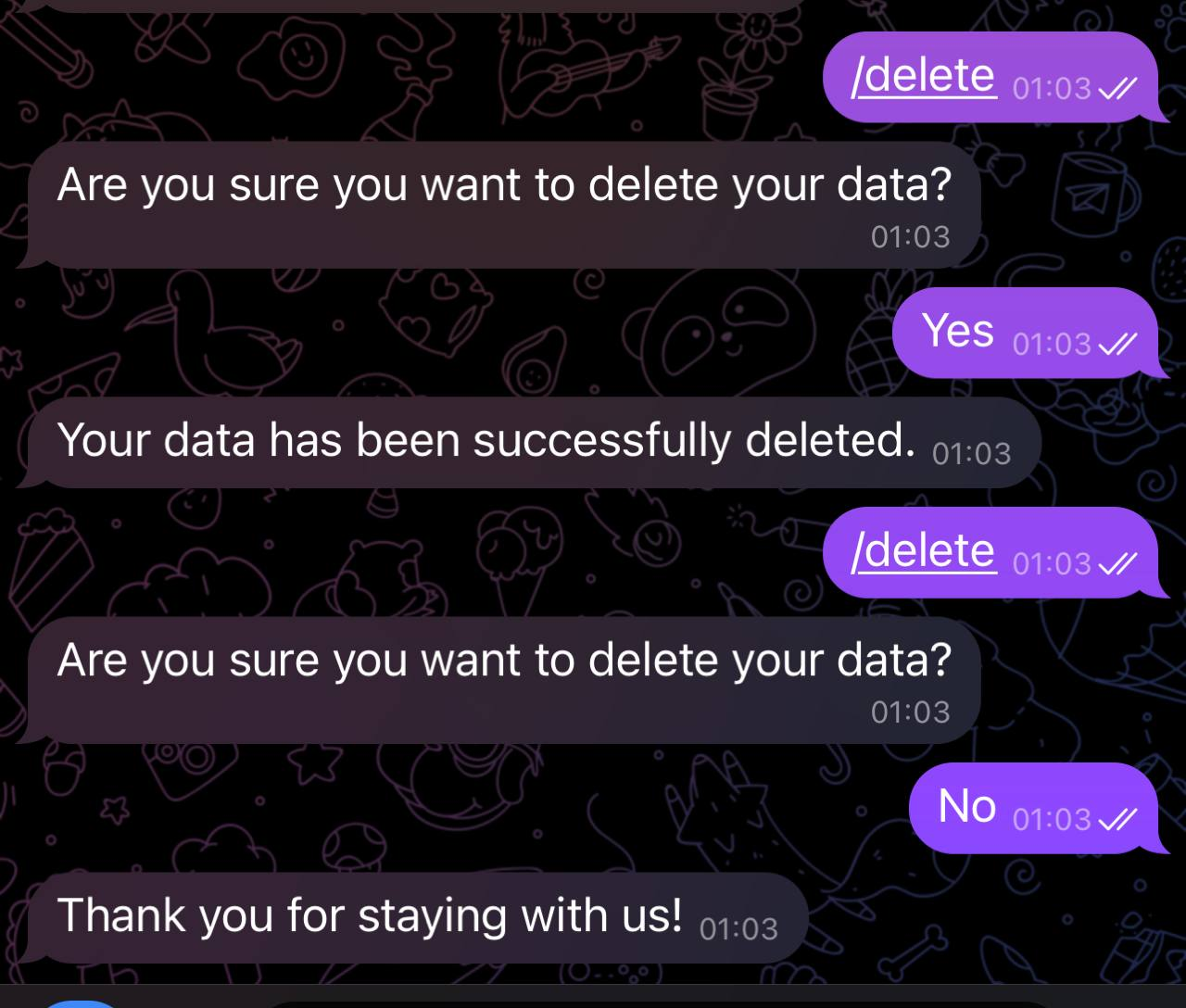




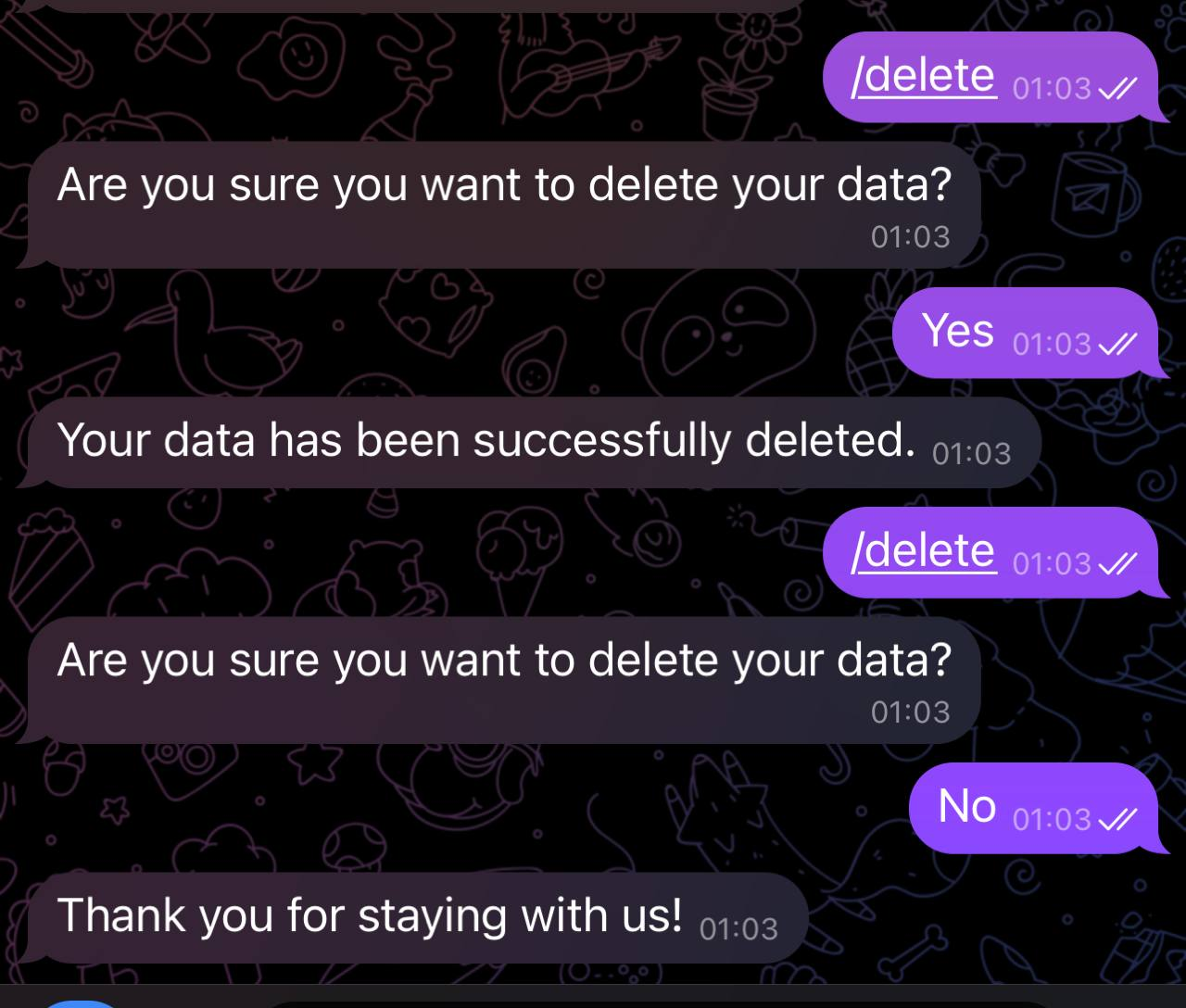
<https://core.telegram.org/bots/api#inlinekeyboardmarkup>

реалізуйте перевірку, чи дійсно користувач хоче видалити профіль, дайте йому дві кнопки на вибір: Так і Ні.

**Завдання 3.** При натиску на Так, видаліть інформацію про користувача з бази даних за його номером телефону чи телеграм ід використовуючи запит SQL DELETE і повідомте йому про результат виконання.



**Завдання 4.** При натиску на Ні, повідомте користувачу про скасування операції видалення користувача.



import json

import requests

import pymysql

import qrcode

from django.http import HttpResponse, HttpResponseBadRequest

from django.views.decorators.csrf import csrf\_exempt

from myapp.credentials import TELEGRAM\_API\_URL, URL, HOSTDB, DBNAME, PORTDB, USERDB, PASSDB, TIMEOUT

timeout = TIMEOUT

# Встановлення з'єднання з базою даних

connection = pymysql.connect(

    charset="utf8mb4",

    connect\_timeout=timeout,

    cursorclass=pymysql.cursors.DictCursor,

    db=DBNAME,

    host=HOSTDB,

    password=PASSDB,

    read\_timeout=timeout,

    port=PORTDB,

    user=USERDB,

    write\_timeout=timeout,

)

@csrf\_exempt

def setwebhook(request):

    response = requests.post(TELEGRAM\_API\_URL + "setWebhook?url=" + URL).json()

    return HttpResponse(f"{response}")

@csrf\_exempt

def telegram\_bot(request):

    if request.method == 'POST':

        update = json.loads(request.body.decode('utf-8'))

        handle\_update(update)

        return HttpResponse('ok')

    else:

        return HttpResponseBadRequest('Bad Request')

def handle\_update(update):

    try:

        chat\_id = update['message']['chat']['id']

        telegram\_id = update['message']['from']['id']

        text = update['message'].get('text', '')

        if text == '/register':

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': 'Please send your contact:',

                'reply\_markup': {

                    'keyboard': [

                        [

                            {

                                'text': 'My phone',

                                'request\_contact': True

                            }

                        ]

                    ],

                    'resize\_keyboard': True,

                    'one\_time\_keyboard': True,

                }

            })

        elif text == '/delete':

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': 'Are you sure you want to delete your data?',

                'reply\_markup': {

                    'keyboard': [

                        [

                            {

                                'text': 'Yes'

                            },

                            {

                                'text': 'No'

                            }

                        ]

                    ],

                    'resize\_keyboard': True,

                    'one\_time\_keyboard': True,

                }

            })

        elif text == 'Yes':

            success = delete\_user\_data(telegram\_id)

            if success:

                send\_message("sendMessage", {

                    'chat\_id': chat\_id,

                    'text': 'Your data has been successfully deleted.',

                    'reply\_markup': {

                        'remove\_keyboard': True,

                    }

                })

            else:

                send\_message("sendMessage", {

                    'chat\_id': chat\_id,

                    'text': 'Failed to delete your data. Please try again later.',

                    'reply\_markup': {

                        'remove\_keyboard': True,

                    }

                })

        elif text == 'No':

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': 'Thank you for staying with us!',

                'reply\_markup': {

                    'remove\_keyboard': True,

                }

            })

        elif text == '/getmyid':

            send\_user\_qr(chat\_id, telegram\_id)

        elif 'contact' in update['message']:

            contact = update['message']['contact']

            phone\_number = contact.get('phone\_number', 'Номер телефону відсутній')

            name = contact.get('first\_name', 'Ім\'я відсутнє')

            last\_name = contact.get('last\_name', 'Прізвище відсутнє')

            # Перевірка наявності користувача за номером телефону в базі даних

            user\_id = check\_user\_existence(phone\_number)

            if user\_id:

                user\_info = f'Користувач з номером телефону {phone\_number} вже існує.\nID користувача: {user\_id}'

            else:

                user\_id = save\_user\_data(telegram\_id, phone\_number, name, last\_name)

                user\_info = f'Ви успішно зареєструвалися.\nID користувача: {user\_id}\nТелефон: {phone\_number}\nІм\'я: {name}\nПрізвище: {last\_name}'

                send\_user\_qr(chat\_id, telegram\_id)

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': user\_info,

                'reply\_markup': {

                    'remove\_keyboard': True,

                }

            })

        else:

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': f'Ти сказав: {text}'

            })

    except Exception as e:

        send\_message("sendMessage", {

            'chat\_id': chat\_id,

            'text': 'Щось пішло не так. Спробуйте ще раз.'

        })

def send\_message(method, data):

    return requests.post(TELEGRAM\_API\_URL + method, json=data)

def check\_user\_existence(phone\_number):

    try:

        with connection.cursor() as cursor:

            sql = "SELECT id FROM customers WHERE phone\_number = %s"

            cursor.execute(sql, (phone\_number,))

            result = cursor.fetchone()

            if result:

                return result['id']

            else:

                return None

    except Exception as e:

        print(f"Помилка при перевірці наявності користувача: {str(e)}")

        return None

def save\_user\_data(telegram\_id, phone\_number, name, last\_name):

    try:

        with connection.cursor() as cursor:

            sql = "INSERT INTO customers (telegram\_id, phone\_number, name, last\_name) VALUES (%s, %s, %s, %s)"

            cursor.execute(sql, (telegram\_id, phone\_number, name, last\_name))

            connection.commit()

            return cursor.lastrowid

    except Exception as e:

        print(f"Помилка при збереженні даних користувача: {str(e)}")

        connection.rollback()

        return None

def delete\_user\_data(telegram\_id):

    try:

        with connection.cursor() as cursor:

            sql = "DELETE FROM customers WHERE telegram\_id = %s"

            cursor.execute(sql, (telegram\_id,))

            connection.commit()

            return True

    except Exception as e:

        print(f"Помилка при видаленні даних користувача: {str(e)}")

        connection.rollback()

        return False

def send\_user\_qr(chat\_id, telegram\_id):

    try:

        with connection.cursor() as cursor:

            sql = "SELECT id FROM customers WHERE telegram\_id = %s"

            cursor.execute(sql, (telegram\_id,))

            result = cursor.fetchone()

            if result:

                user\_id = result['id']

                qr\_data = f"User ID: {user\_id}"

                qr\_filename = f"user\_{user\_id}\_qr.png"

                generate\_qr\_code(qr\_data, qr\_filename)

                send\_document(chat\_id, qr\_filename)

            else:

                send\_message("sendMessage", {

                    'chat\_id': chat\_id,

                    'text': 'Користувач не зареєстрований. Будь ласка, зареєструйтеся за допомогою команди /register.'

                })

    except Exception as e:

        print(f"Помилка при відправленні QR-коду користувачу: {str(e)}")

def generate\_qr\_code(data, filename):

    qr = qrcode.QRCode(

        version=1,

        error\_correction=qrcode.constants.ERROR\_CORRECT\_L,

        box\_size=10,

        border=4,

    )

    qr.add\_data(data)

    qr.make(fit=True)

    img = qr.make\_image(fill\_color="black", back\_color="white")

    img.save(filename)

def send\_document(chat\_id, filename):

    url = f"{TELEGRAM\_API\_URL}sendDocument"

    files = {'document': open(filename, 'rb')}

    data = {'chat\_id': chat\_id}

    response = requests.post(url, files=files, data=data)

    if response.status\_code == 200:

        print("Document sent successfully")

    else:

        print("Failed to send document")