Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Курсова робота

освітній ступінь – бакалавр

на тему: «**Дослідження методів розробки чат-ботів**»

Виконав: студент 3-го року навчання,

Спеціальності

121 «Інженерія Програмного Забезпечення»

Студента Ковриженка Владислава Євгенійовича

Керівник Глибовець А.М.

Доктор технічних наук, декан

«01» червня 2022 р.

Київ – 2022

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Освітній ступінь бакалавр

Спеціальність 121 «Інженерія Програмного Забезпечення»

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики

Гороховський С. С.

“10” жовтня 2021 року

ЗАВДАННЯ
ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Ковриженку Владиславу

1. Тема роботи **«Дослідження методів розробки чат-ботів»**, керівник роботи Глибовець Андрій Миколайович, доктор технічних наук, декан

2. Строк подання студентом роботи 5 червня 2022

3. План роботи

Вступ

Розділ 1. Теоретична частина

1.1 Історія появи чат-ботів
1.2 Типи чат-ботів
1.3 Сфери застосування різних типів чат ботів
1.4 Аналіз ринку фрілансу
1.5 Ознайомлення з методами розробки
1.6 Приклад порядку дій при розробленні чат-бота для месенджера

Розділ 2. Практична частина

ЗМІСТ

Вступ

Перед початком з ознайомленням методів розробки потрібно хоч приблизно зрозуміти, що таке чат-бот. Оксфордський словник англійської мови пропонує визначення - "Чат-бот - комп’ютерна програма, яка може вести розмову з людиною, як правило, через Інтернет", що в цілому описує застосування цього типу програм, але не в повному обсязі. Тому для повного розуміння як і для чого створюються чат-боти потрібно трохи заглибитись в історію.

Історія появи чат-ботів

Англійський словник для знаходження визначення був використаний неспроста, адже перша програма даного типу під назвою ELIZA, була розроблена в Америці і могла оброблювати природню англійську мову. Перший чат-бот був створений Джозефом Вайзенбаумом у 1966 році і використовував методологію "matching and substitution" для моделювання розмови. ELIZA була запрограмована на імітацію терапевтичної розмови в стилі Роджеріанської терапії, в основному це сенс коли терапевт ставить запитання пацієнту, перефразуючи його запитання. Так, наприклад, відповідь на "У мене болить голова" може бути "Чому ти кажеш, що у тебе болить голова?", а відповіддю на "Моя мати мене ненавидить" - "Хто ще у вашій родині вас ненавидить?". Наступним відомим проектом був PARRY, це чат-бот розроблений Кеннетом Колби через 6 років після створення ELIZA, котрий імітував пацієнта з шизофренією, проект використовував схожу методологію до ELIZA та зміг пройти тодішній тест Тьюрінга та тест-перевірку на параноїдальний розлад.

Наступними проектами були Dr.Sbaitso та ALICE, цікаві вони для нас насамперед через використання штучного інтелекту в чат-ботах. Dr.Sbaitso

був створений в 1992 та імітував розмову з психологом, але що більш цікаво використовував ШІ. З ботом досі можна поспілкуватись ввівши в пошуку його назву, але замість складної взаємодії з користувачем, на більшість повідомлень бот буде відповідати запитанням “Why do you feel that way?”. ALICE була зроблена в 1995, й мала більш розвинений ШІ та імітувала розмову з реальною людиною через Інтернет.

Наступною цікавою групою чат ботів є Siri, Alexa, Cortana, Google Assistant всіх цих ботів об'єднує те, що вони розроблені після 2010го року, мають схожий функціонал та ціль з якою вони були розроблені. Те що боти розроблені в цьому тисячоріччі означає, що були використані максимально нові та потужні технології для реалізацій бажаних функцій. Функції були спеціалізовані для допомоги користувачам працювати з певною платформою у самий зручний, для користувача, спосіб. Бот сприймає будь який формат запиту голосовий(аудіо), текстовий, фото тощо та відповідає користувачеві, а при потребі виконує певні дії. Наприклад користувач IPhone в котрому вбудований чат-бот Siri, може вигукнути команду "Привіт Сірі" для початку роботи, а далі "Зателефонуй "ім'я контакту"" аби зателефонувати людині з вашого списку контактів.

Типи чат-ботів

Судячи з вище наведених прикладів відомих чат-ботів можемо їх розділити на три чітко виражені типи:

1) ELIZA та PARRY - звичайні чат-боти з використанням метадології "matching and substitution" або ж скриптові чат-боти котрі працюють з певним набором команд по заданому сценарію.

2) Dr. Sbaitso та ALICE - AI chatbots(з англ. "Чат-боти з використанням ШІ"), користування цим типом ботів більш схоже до спілкування з

реальною людиною, аніж з програмою. Також варто зазначити, що можливості глибокого навчання чат-ботів з штучним інтелектом, дозволяють взаємодіям з часом, ставати більш точнішими, створюючи мережу відповідних відповідей через їхню взаємодію з людьми.

3) Siri, Alexa, Cortana, Google Assistant - virtual agents(з англ. "Віртуальний помічник"), в більшості випадках є поєднанням перших двох пунктів, чат-бот котрий має вбудований штучний інтелект та певний список команд з заготовленими відповідями/діями на них.

Сфери застосування різних типів чат-ботів

Звичайні чат-боти в сьогоденні є найбільш розповсюдженими, використовуються в усіх відомих галузях замінюючи або доповнюючи певний сервіс. Наприклад ви можете придбати квитки на потяг в касі на вокзалі, але щоб придбати квиток вам потрібно вже знати який саме потяг, а для цього підлаштовувати свій графік під звичайну купівлю квитка, інший варіант придбати його онлайн на сайті УкрЗалізниці, що є більш зручним способом, але має свої недоліки. Наприклад ви маєте кожного дня перевіряти, чи не з'явився потяг котрий підходить під ваші забаганки, тоді як в чат боті RailwayBot ви можете запустити моніторинг потрібного маршруту, та при появі подібного за параметрами квитку на потягу придбати його в один клік. Отже звичайні боти використовуються для створення сервісів або доповнення вже існуючих.

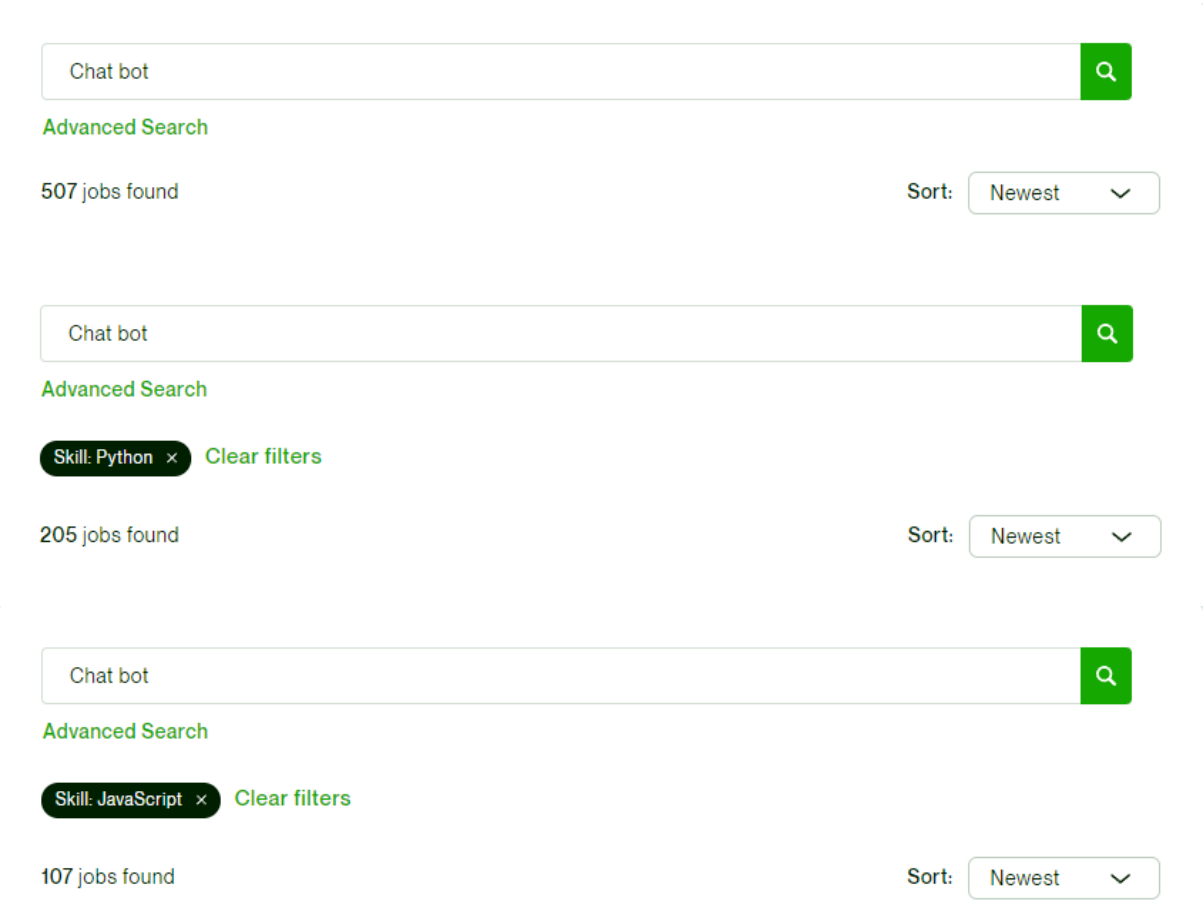
Чат-боти з ШІ використовують розуміння природної мови, щоб розпізнати потреби користувача. Вони використовують передові інструменти штучного інтелекту, щоб визначити, чого користувач намагається досягти. За допомоги машинного та глибинного навчання розроблюється все більш

детальна база знань із запитань і відповідей, яка ґрунтуються на взаємодії з користувачами. Це покращує їх здатність точно передбачати потреби користувачів і правильно реагувати з часом. Ця технологія сьогодні є невід'ємною частиною всіх онлайн-консультантів.

Віртуальні помічники завжди були приємним доповненням до нового гаджету, а тепер компанії та їх розробники змагаються в створенні максимально швидкого, зручного та корисного віртуального асистента аби за рахунок нього зробити товар більш бажаним для кінцевого користувача.

Аналіз ринку чат-ботів на фрілансі

Переглянувши замовлення на одній з найвідоміших бірж фрілансу upwork, помітно що найбільший попит мають звичайні чат-боти написані на JS або Python.

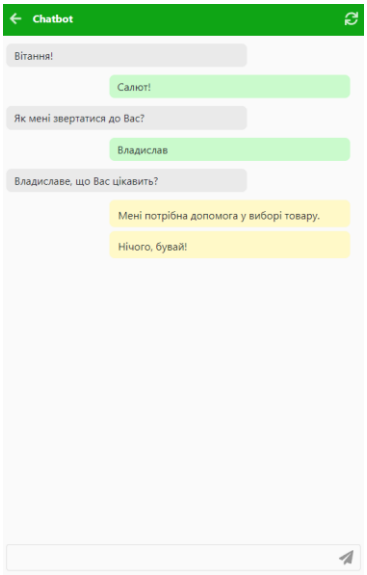


Ознайомлення з методами розробки

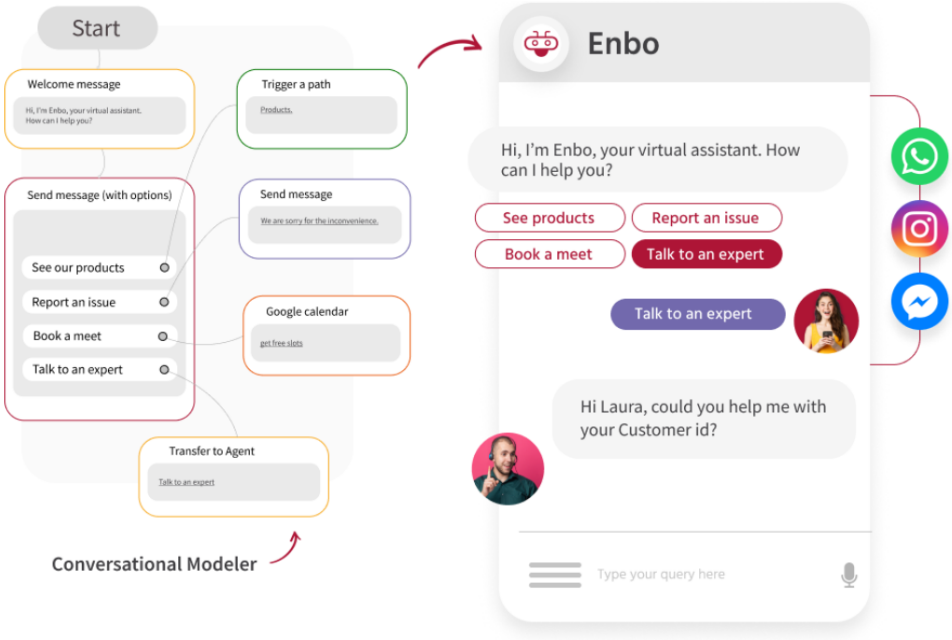
Чат-бот повинен бути інтегрованим в певний застосунок, котрий використовує користувач, існує два популярних рішення - чат-бот в месенджері, чат-бот в якості консультанта на сайті.

В першому варіанті, розробники месенджеру мають готові API для роботи з ботами, на базі яких написані бібліотеки.

В другому, є декілька варіантів створення чат ботів:
- Якщо сценарій розмови людини з чат-ботом невеликий то написати чат-бота власноруч, для серверної частини вистачить JS (прописуєтсья сценарій з заготовленими запитаннями користувача та відповідями бота на них), а для клієнтської частини PHP(відмальовуємо елемент схожий на діалогове вікно).



- Для більш великих сценаріїв є можливість використання конструкторів чат-ботів. (наприклад engati де в зручному інтерактивному інтерфейсі є можливість створити досить обширну базу запитання-відповідей для генерації чат-бота для сайту)



- Останній варіант, підходить для складних чат-ботів з величезними сценаріями та великим функціоналом - це використання сторонніх фреймворків (наприклад talkjs), який дає змогу розробляти ботів за схожим принципом до розробки ботів для месенджерів.

Приклад порядку дій при розробленні чат-бота для месенджера

1. Перед написанням логіки чат-бота в будь якому месенджері, для початку потрібно його зареєструвати аби отримати токен котрий зв'яже написаний код з певним ботом в месенджері. В кожному месенджері це робиться в різний спосіб, але процедура не займає більше 5ти хвилин.

Так для Viber потрібно перейти за посиланням https://partners.viber.com/login та пройти реєстрацію,
в WhatsApp за посиланням https://business.whatsapp.com ,
а в Telegram потрібно зареєструвати бота в офіційному боті телеграму @BotFather.

2. Наступним кроком, обираємо бібліотеку для взаємодії з ботом, встановлюємо її, додаємо залежності, вказуємо токен отриманий з попереднього кроку, та пишемо простенький обробник повідомлень з виводом даних про повідомлення в консоль та надсиланням ідентичного повідомлення у відповідь.

```
const TelegramAPI = require('node-telegram-bot-api')

const token = '5479628261:AAHVfpqzGLi8n_5zns0OyEHoanM0MWU9IjA'

const bot = new TelegramAPI(token, options: {polling: true})



bot.on( event: 'message', listener: async msg => {
    console.log(msg);
    const text = msg.text;
    const chatId = msg.chat.id;

    await bot.sendMessage(chatId, text);
```

Це проста перевірка підключення до боту, яка також демонструє структуру JSON який ми отримуємо від користувача.

```
{
  message_id: 254,
  from: {
    id: 691323674,
    is_bot: false,
    first_name: '♂Vladdik',
    last_name: 'Kiddalv♂',
    username: 'kovrud',
    language_code: 'uk'
  },
  chat: {
    id: 691323674,
    first_name: '♂Vladdik',
    last_name: 'Kiddalv♂',
    username: 'kovrud',
    type: 'private'
  },
  date: 1654348279,
  text: '/start',
  entities: [ { offset: 0, length: 6, type: 'bot_command' } ]
}
```

Принцип написання логіки самого бота в залежності від обраного месенджера, мови програмування та бібліотеки майже незмінний. Єдина різниця в синтаксисі окремої мови та бібліотеки.

Приклад обробника повідомлень в JS для Telegram:

```
bot.on('message',  async msg => {
    const text_received = msg.text;
    const chatId = msg.chat.id;
    let message;
    if(text_received === 'text') {
        // TextMessage object
        message = new TextMessage('hello world');
    }
    else if(text_received === 'url') {
        // Url Message object
        let url = "https://google.com"
        message = new UrlMessage(url);
    }
    else {
        message = new TextMessage("Hi!" + sender_name + " (" + sender_id + ")");
    }
    console.log(message);
    bot.sendMessage(chatId,message);
})
```

Приклад обробника повідомлень в JS для Viber:

```
function checkUrlAvailability(botResponse, text_received) {
    let sender_name = botResponse.userProfile.name;
    let sender_id = botResponse.userProfile.id;

    if (text_received === '') {
        say(botResponse, 'I need a Text to check');
        return;
    }
    let message;
    if(text_received === 'text') {
        // TextMessage object
        message = new TextMessage('hello world');
    }
    else if(text_received === 'url') {
        // Url Message object
        let url = "https://google.com"
        message = new UrlMessage(url);
    }
    else {
        message = new TextMessage("Hi!" + sender_name + " (" + sender_id + ")");
    }
    console.log(message);
    botResponse.send(message);
}
```

Приклад обробника повідомлень в Python для Telegram:

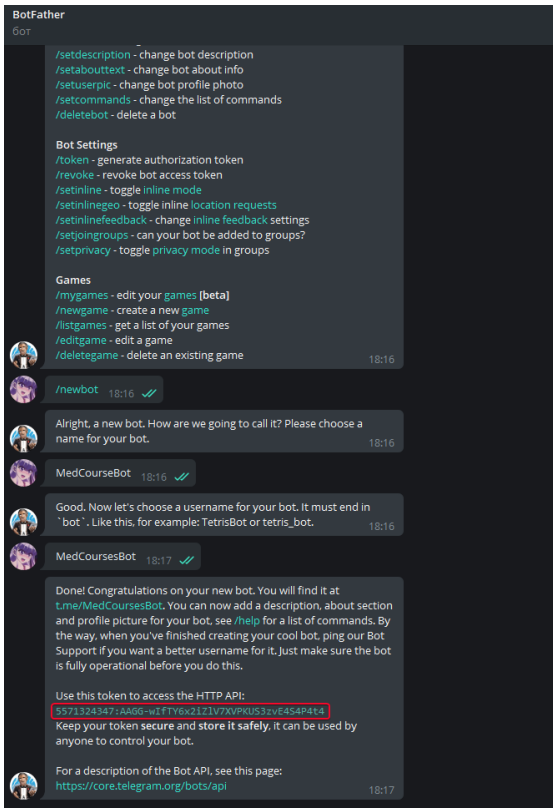
```python
import os
import telebot
import yfinance as yf

API_KEY = os.getenv('API_KEY')
bot = telebot.TeleBot(API_KEY)

@bot.message_handler(commands=['Greet'])
def greet(message):
  bot.reply_to(message, "Hey! Hows it going?")

@bot.message_handler(commands=['hello'])
def hello(message):
  bot.send_message(message.chat.id, "Hello!")
```

Практична частина курсової роботи

Опис проекту

Ідея боту: дати людям зручну можливість записуватись на курси медичної допомоги в телеграм-боті. З обов'язкових функцій додати обмеження кількості людей на курсі та можливість виписуватись з нього. Також створити інтерфейс адміністратора для зручного додавання курсів та взяття списку людей котрі відвідують курс. Пишеться бот на платформі Node.JS, з використанням СКБД MySQL, та бібліотеки node-telegram-bot-api.

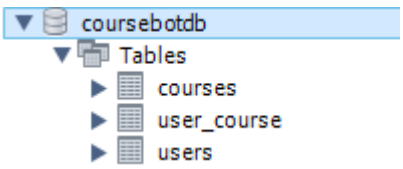
Реєстрація бота



Створюємо бота в @BotFather та зберігаємо токен.

Створення БД

Для проекту обрано СКБД MySQL, створимо таблиці та зв’язки між ними в графічному інтерфейсі MySQL Workbench:



Users - відповідає за зберігання інформації про користувача

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
user_id	INT	☑	☑	☑	☐	☐	☐	☑	☐
chat_id	VARCHAR(200)	☐	☑	☑	☐	☐	☐	☐	☐
username	VARCHAR(45)	☐	☐	☑	☐	☐	☐	☐	☐
		☐	☐	☐	☐	☐	☐	☐	☐

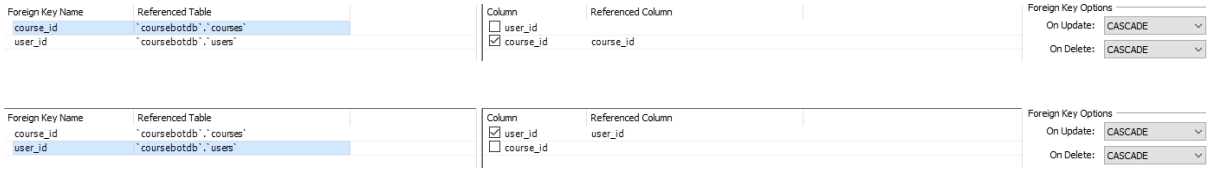
Courses – відповідає за зберігання інформації про курси.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
course_id	INT	☑	☑	☑	☐	☐	☐	☑	☐
course_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
course_info	TINYTEXT	☐	☑	☐	☐	☐	☐	☐	☐
course_date	DATE	☐	☑	☐	☐	☐	☐	☐	☐
course_capacity	INT	☐	☑	☐	☐	☐	☐	☐	☐
		☐	☐	☐	☐	☐	☐	☐	☐

User_course – відповідає за зберігання курсів обраних певними користувачами.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
user_id	INT	☑	☑	☐	☐	☐	☐	☐	☐
course_id	INT	☑	☑	☐	☐	☐	☐	☐	☐
		☐	☐	☐	☐	☐	☐	☐	☐

Та зв’язки між User_course – Users та User_course – Courses



Конфігураційний файл з підключенням до MySQL:

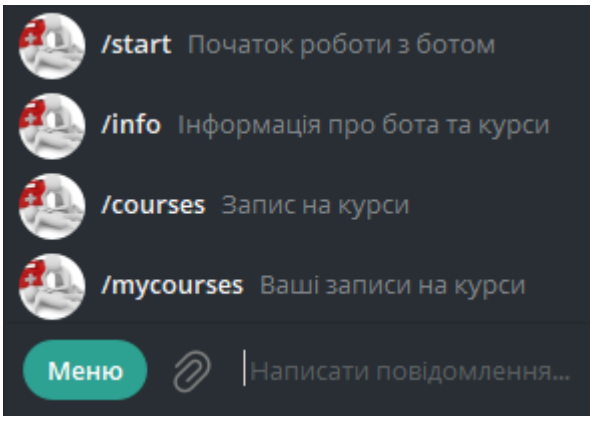
```
const mysql = require("mysql");

var mysqlConnection = mysql.createConnection({
    host : "localhost",
    user : "root",
    password : "password",
    database : "coursebotdb",
    multipleStatements : true
});

mysqlConnection.connect((err)=>{
    if(!err)
    {
        console.log("Connected");
    }
    else
    {
        console.log("Connection failed");
    }
});

module.exports = mysqlConnection;
```

Написання логіки бота

Почнемо з визначення команд:

```
bot.setMyCommands([
    {command: '/start', description: 'Початок роботи з ботом'},
    {command: '/info', description: 'Інформація про бота та курси'},
    {command: '/courses', description: 'Запис на курси'},
    {command: '/mycourses', description: 'Ваші записи на курси'}
])
```

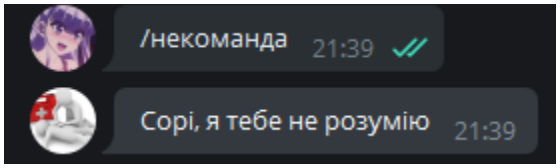
Ця частина коду додає панель МЕНЮ до лівого нижнього краю месенджера та дає змогу швидко запускати команди.



Далі йде перевірка через конструкцію if else, яка саме команда прийшла на вхід. (в середині лісенеру ми одразу дістаємо з JSON ідентифікатор чату з якого надіслано повідомлення та текст повідомлення)

```
bot.on( event: 'message',  listener: async msg => {
    const text = msg.text;
    const chatId = msg.chat.id;

    if (text === '/start'){
```

Якщо це не команда то в програмі спрацьовує останній else і бот відповідає:



Якщо повідомлення від користувача є командою із списку то виконується певні послідовні дії. Розглянемо найбільший за розміром код аби зрозуміти як працюють всі інші команди:

```
else if (text === '/mycourses'){
    mysqlConnection.query("SELECT * FROM users WHERE chat_id = ?",[chatId],(err,rows,fields)=> {
        if (!err) {
            userid = rows[0].user_id;
            mysqlConnection.query("SELECT * FROM user_course WHERE user_id = ?",[userid],(err,rows1,fields)=> {
                if (!err) {
                    if(rows1.length === 0)bot.sendMessage(chatId, text: `Ти ще не записаний до жодного курсу`);
                    else{
                    rows1.forEach(element => {
                        mysqlConnection.query("SELECT * FROM courses WHERE course_id = ?",[element.course_id],(err,rows2,fields)=> {
                            if(!err) {
                                bot.sendMessage(chatId, text: `Назва: ${rows2[0].course_name}
                                \n\nТрішки про курс: ${rows2[0].course_info}
                                \n\nДата проведення: ${formatDate(rows2[0].course_date)}` , form: {
                                    reply_markup: {
                                        resize_keyboard: true,
                                        inline_keyboard: [
                                            [{ text: `Виписатись з курсу`, callback_data: `leave${element.course_id}`}]
                                        ],
                                    },
                                })}
                        })
                    })};
                }
            })
        }
    })
}
```

Червоним виділено SQL запити до СКБД які мають повертати масив JSONів. Перший запит знаходить ідентифікатор користувача за ідентифікатором чату в таблиці Users. Другий є вкладеним в перший та знаходить ідентифікатори курсів за ідентифікатором користувача в таблиці user_course. Третій вкладений в другий та виконується рівно стільки разів скільки знайдено рядочків в другому запиті. В середині 3тього запиту викликається лиш один метод sendMessage, параметрами якого є обов’язково ідентифікатором чату та повідомлення. Третій параметр в методі необов’язковий але дає змогу додати форм(в такій реалізації – кнопку для виписування з курсу).

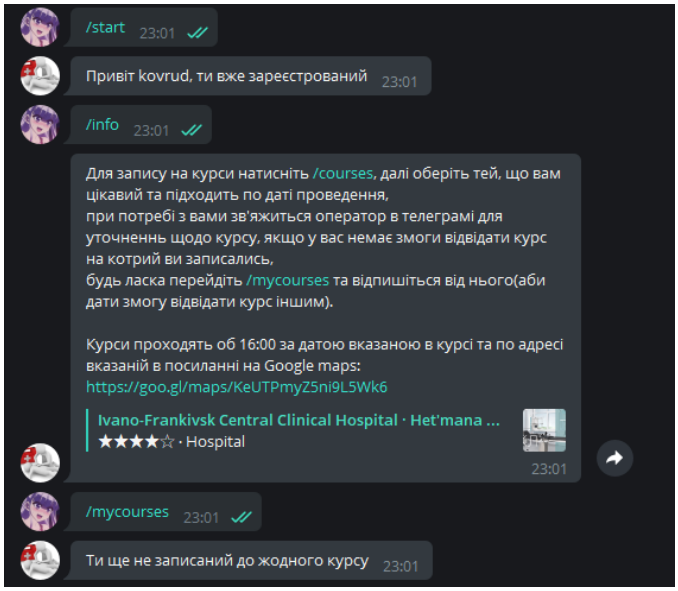
Для написання логіки відпрацювання кнопок використовується окремий лісенер, в цьому випадку використовується не === а includes по причинні того, що разом з типом команди ми передаємо і значення зазвичай ідентифікатор певного об’єкту.

```
bot.on( event: 'callback_query',  listener: msg =>{
    const data = msg.data;
    const chatId = msg.message.chat.id;

    if (data.includes( value: 'join')){
```

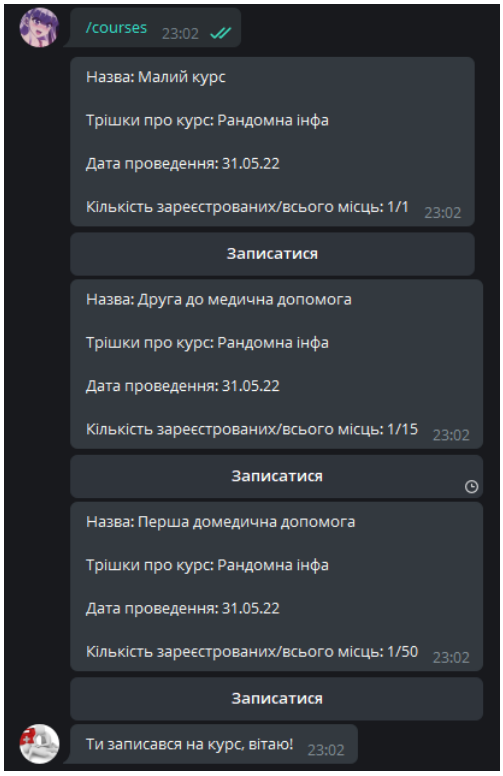
Варто наголосити на способі додання нового користувача до БД, це відбувається за командою /start, це команда котра завжди виконується коли користувач вперше користується ботом. Також важливою інформацією є те, що ідентифікатор чату є ідентичний до ідентифікатору користувача, а він є статичним.

Тестування бота

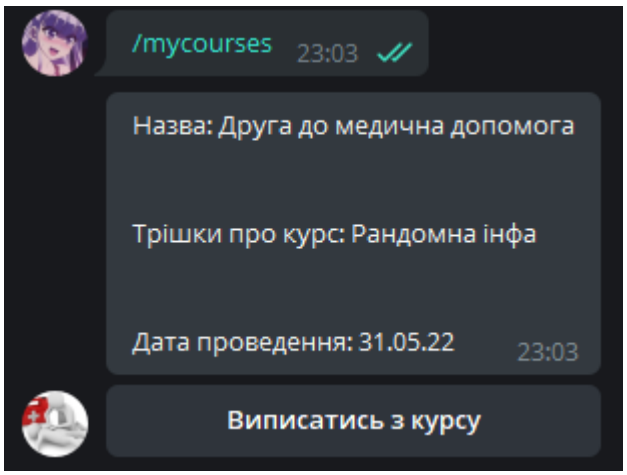
Тестування початку роботи з ботом:



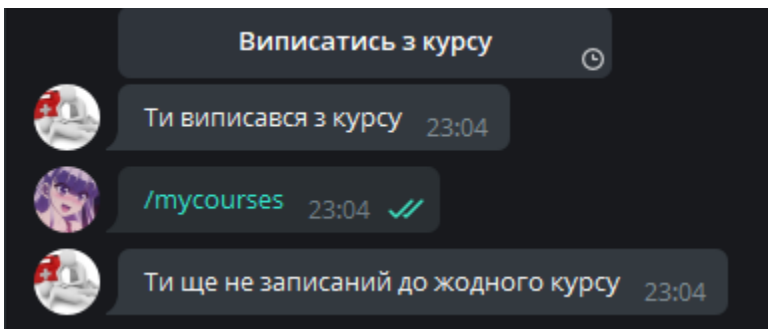
Запис на курс:



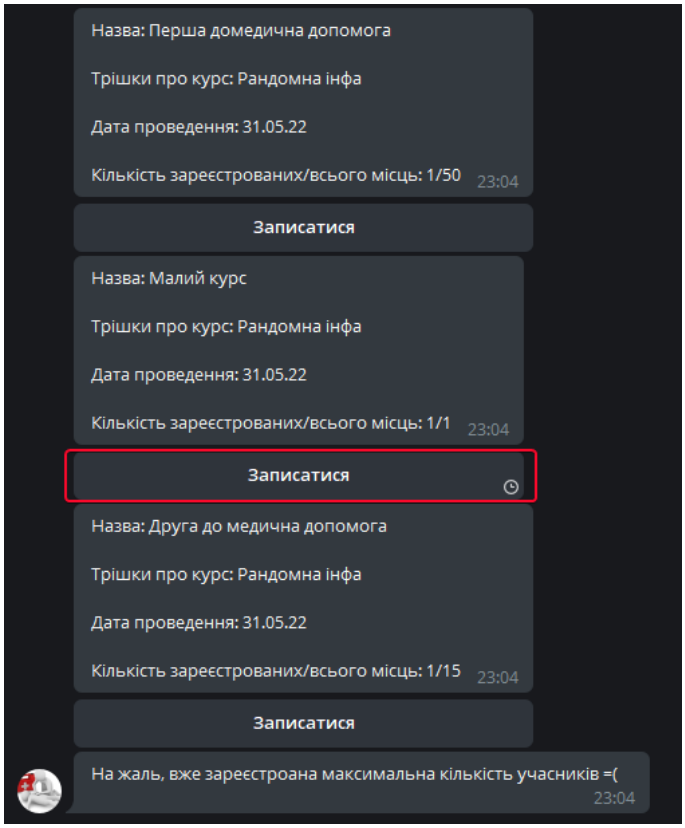
Перевірка записаних курсів:



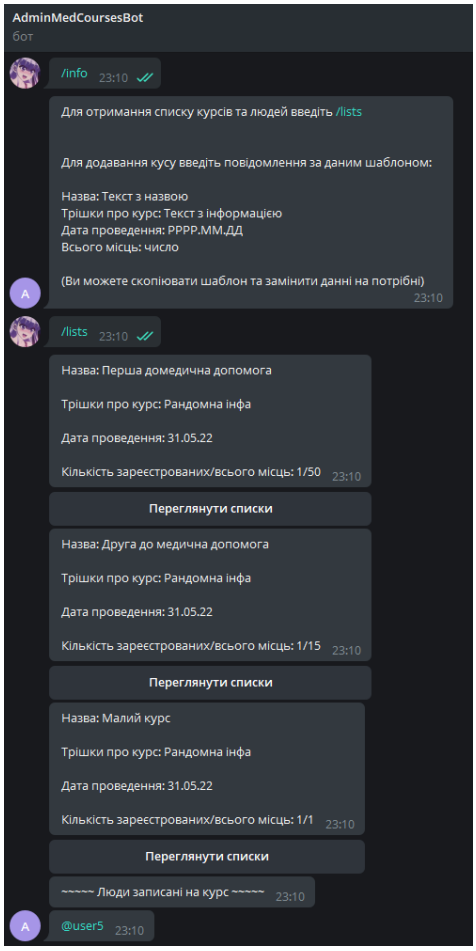
Спроба виписатись з курсу:



Спроба записатись на курс де зареєстрована максимальна кількість учасників:



Через відсутність інструментів в цій бібліотеці для створення адмін-панелі, був створений ще один бот який відповідає за функції адміна. Перевірка прав в якому виконується за перевіркою ідентифікатора чату(якщо ідентифікатор користувача збігається з ідентифікатором адміна то виконуються дії)



Висновок

Отже чат-боти є відносно простими та швидкими в розробці, вони можуть реалізувати буквально будь-який сервіс, та можуть розширюватись різноманітними сторонніми бібліотеками.

Список використаних джерел

https://www.ibm.com/cloud/learn/chatbots-explained

https://onlim.com/en/the-history-of-chatbots/

https://en.wikipedia.org/wiki/Jabberwacky

https://www.chatbots.org/chatbot/parry/

https://en.wikipedia.org/wiki/Dr._Sbaitso

https://www.chatbots.org/chatbot/eliza/

https://www.upwork.com/

https://talkjs.com/docs/

https://www.engati.com/

https://github.com/yehtutwin/viber-bot/blob/main/nodejs/index.js

Лістинг програми