НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«____» _____2021 р.

Дипломний проект

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»

на тему: «Бот для контролю стану здоров'я»

Виконав: студент 4 курсу, групи ІВ-72
Андрюшечкіна Діана Дмитрівна _____

Керівник:

Асистент Калюжний О.О. _____

Консультант з нормоконтролю:

Професор кафедри ОТ, д.т.н.

Сімоненко Валерій Павлович _____

Рецензент _____ _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

«Комп'ютерні системи та мережі»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Сергій СТІРЕНКО
«__» _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проект студента

Андрюшечкіної Діани Дмитрівни

1. Тема проекту _«*Бот для контролю стану здоров'я»*__

керівник проекту __*асистент Калюжний О.О.*__

Затверджені наказом по університету від «11» травня 2021 року №1139-с

2. Термін здачі студентом закінченого проекту (роботи) 03 червня 2021 року

3. Вихідні дані до проекту *технічна документація, теоретичні та статистичні дані;*

4. Зміст розрахунково-пояснювальної записки

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): *приниципова схема, функціональна схема, структурна схема*

6. Консультанти проекту, з вказівкою розділів проекту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проекту	Терміни виконання етапів проекту	Примітки
1.	*Затвердження теми проекту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021- 25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021- 5.04.2021*	
5.	*Програмна реалізація системи*	*5.04.2021- 15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021- 20.05.2021*	
7.	*Захист програмного продукту*	*25.04.2021*	
8.	*Передзахист*	*23.05.2021*	
9.	*Захист*	*14.06.2021*	

Студентка Андрюшечкіна Д.Д.

Керівник проекту Калюжний О.О.

Анотація

Дипломна робота полягала в створенні чат-бота на платформі Telegram, який буде збирати, зберігати і обробляти дані користувачів. Метою проекту було полегшити збір інформації про можливі симптоми. Розроблено функціонал опитування користувачів і оповіщення адміністратора про його результати.

Бот був реалізований на мові програмування Python і за допомогою бази даних SQLite.

Annotation

The thesis was to create a chatbot on the Telegram platform, which will collect, store and process user data. The aim of the project was to facilitate the collection of information on possible symptoms. The functionality of polling users and notifying the administrator of its results has been developed.

The bot was implemented in the Python programming language and using the SQLite database.

Опис альбому

до дипломного проекту

на тему: «Бот для контролю стану здоров’я»

Київ – 2021

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ екземпляру	Примітки
	А4		Завдання на дипломний	2		
			проэкт			
	А4	***ІАЛЦ.467200.001 ОА***	Бот для контролю	1		
			стану здоров’я			
			Опис альбому			
	А4	***ІАЛЦ.467200.001 ТЗ***	Бот для контролю	3		
			стану здоров’я			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Бот для контролю	61		
			стану здоров’я			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Додаток 1	1		
			Схема структурна			
	А4	***ІАЛЦ.467200.004 Д2***	Додаток 2	1		
			Схема функціональна			
	А4	***ІАЛЦ.467200.004 Д3***	Додаток 3	1		
			Схема принципова			
	А4	***ІАЛЦ.467200.004 Д4***	Додаток 4	6		
			Лістинг програми			

Зм	***Арк.***	***№ докум***	***Підпис***	***Дата***	***ІАЛЦ.467200.001 ОА***		
Розроб.		*Андрюшечкіна Д.Д.*			*Бот для контролю стану здоров’я* *Опис альбому*	***Літ.***	***Арк.*** / ***Аркушів***
Первір.		*Калюжний О.О.*				Т	1 / 1
Н.Контр.		*Сімоненко В.П.*				***КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72***	
Затверд.		*Калюжний О.О.*					

Технічне завдання

до дипломного проекту

на тему: «Бот для контролю стану здоров’я»

Київ – 2021

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ	
Зм.	***Арк.***	***№ докум.***	***Підпис***	***Дата***		
Розроб.		*Андрюшечкіна Д.Д.*			*Бот для контролю стану здоров'я* *Технічне завдання*	***Лит.*** | ***Арк.*** *1* | ***Аркушів*** *3*
Перевір.		*Калюжний О.О.*				
						КПІ ім. Ігоря Сікорського ФІОТ, ІВ-72
Н. Контр.		*Сімоненко В.П.*				
Затверд.						

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: Бот для контролю стану здоров’я

Область застосування: гуртожитки

2. ПІДСТАВИ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи

кваліфікаційно-освітнього рівня «бакалавр комп’ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка Телеграм боту, що буде проводити щоденне опитування та сповіщати адміністрацію про захворівших.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в мережі Інтернет з даних питань, довідники з програмування.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до розробленого продукту

- Проведення опитування
- Зберігання інформації
- Обробка інформації
- Сповіщення адміністратора

5.2 Вимоги до програмного забезпечення

- Windows/Linux/macOS
- СУБД SQLite
- Мова програмування Python

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	30.03.2021
Складання і узгодження технічного завдання	01.04.2021
Розробка структур окремих інтерфейсів програми	01.04.2021 -15.04.2021
Програмна реалізація	16.04.2021 -12.05.2021
Тестування окремих модулів системи	13.05.2021
Доопрацювання, налагодження і виправлення помилок	14.05.2021
Оформлення документації дипломної роботи	20.05.2021

Пояснювальна записка
до дипломного проекту

на тему: «Бот для контролю стану здоров’я»

Київ – 2021

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API	Application Programming Interface Прикладний програмний інтерфейс
HTTP	HyperText Transfer Protocol Протокол передачі даних
JSON	JavaScript Object Notation Текстовий формат обміну даними
СУБД	Система управління базами даних
ОС	Операційна система
URL	Uniform Resource Locator система уніфікованих адрес електронних ресурсів

ВСТУП

З розвитком технологій в наш час чат-боти стають все більш популярними і використовуються в багатьох сферах, вони імітують розмову з реальною людиною, допомагаючи їй досягти певних цілей.

З чат-ботом немає необхідності завантажувати додаткові програми, вони надають конкретну інформацію, що цікавить користувача, немає потреби розбиратися з інтерфейсом, так як боти реалізовані на основі чату. На відміну від розмови з реальною людиною, боти відповідають миттєво і доступні цілодобово, що спрощує взаємодію з великою кількістю користувачів. Це допомагає компаніям економити час і людські ресурси.

AI бере на себе задачу полегшити рутинну роботу і підвищити якість послуг, що надаються. Голосове управління і текстове спілкування за допомогою бота можна назвати основними напрямками розвитку AI. Як приклад, можна навести тривіальні завдання, як пошук і бронювання квитків, заявка на замовлення їжі, пошук вакансій і багато іншого.

На даний момент найбільш актуальною платформою для спілкування є Telegram, встановлений у більшості людей, тому це буде вдалим вибором для реалізації бота.

У Telegram реалізовано хмарне зберігання всіх даних листувань, що допоможе користувачам зберегти важливу інформацію. Відмінною рисою є власне зашифроване хмарне сховище, за допомогою якого гарантується приватність даних, а отже чат-боти можуть використовуватися в корпоративних цілях.

Перевагою є і наявність каналів, групових чатів з необмеженою кількістю користувачів і можливості додавання туди бота. Так само у телеграм доступне API, що дозволяє безкоштовно створити бота будь-якому користувачеві.

Тому Telegram найбільш перспективна платформа для створення, розвитку та просування чат-ботів.

З виникненням пандемії все більше людей занепокоєні своїм здоров'ям і своїх близьких. З кожним днем все більше жителів планети захворює covid-19,

який може бути причиною летального наслідку. Багато хворих можуть перебувати в зоні ризику, наприклад мати діабет, слабке серце і перелік інших захворювань, що робить їх ще більш вразливою ланкою. На даний момент вакцинація не була здійснена більшої частини населення, тому вірус все ще є великою загрозою. Стан здоров'я відіграє велику роль в житті кожної людини, і безумовно необхідно за ним стежити і вживати необхідних заходів, при виникненні тривожних симптомів. Набагато легше уникнути хвороби, ніж пізніше лікувати її наслідки. Тому було вирішено створити бота для контролю здоров'я, який також буде сповіщати про можливе захворювання, щоб запобігти поширення вірусу і вберегти інших жителів.

Головною метою бота буде збір і збереження інформації про здоров'я кожного користувача - жителя гуртожитку і повідомлення адміністрації, про що виникли занепокоєння і ознаки можливої хвороби.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Додаток Body Temperature Recorder

Body Temperature Recorder - це мобільний застосунок для запису виміряної температури тіла. Розроблений для платформ android та ios.

Щоб додати результати необхідно натиснути на кнопку «Add» та ввести значення. Є можливість створювати і вести записи для кількох людей. Крім самої температури, користувач може вказати найпоширеніші симптоми з вбудованого списку: нежить, закладеність, головний біль та інші, це облегшує задачу повідомлення лікаря про симптоми. А в замітці можливо додати будь-який коментар.

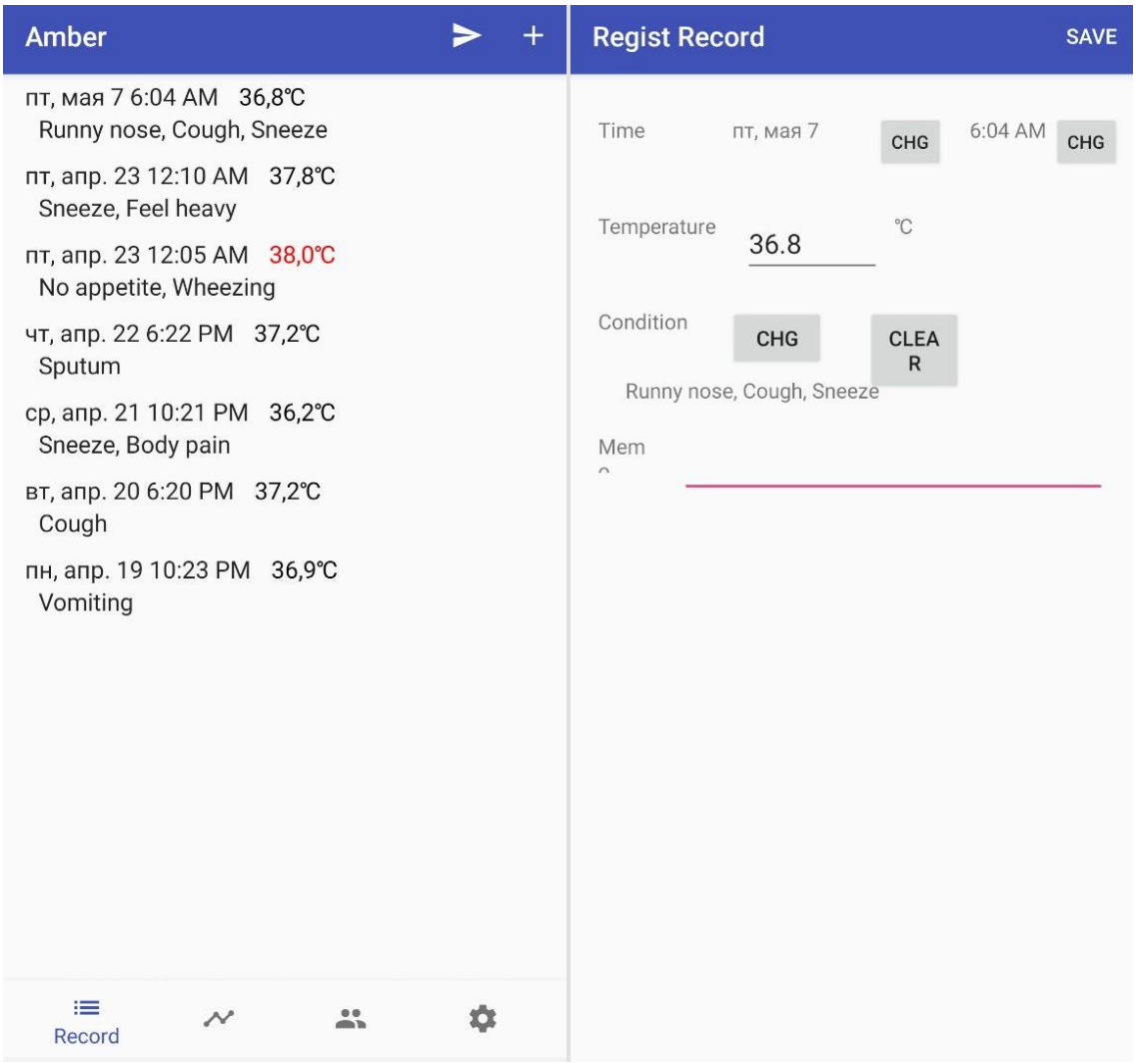


Рис. 1.1 - Body Temperature Recorder

Запис автоматично графікується, що надає змогу користувачу швидко побачити перехід температури тіла. В окремій вкладці на графіку присутня можливість відстежити зміни температури протягом 3, 7, 13 і 30 днів. Результати можливо надіслати лікарю поштою чи у зручній соцмережі. [1, 20]

1.2 Додаток Body Temperature App

У Body Temperature App потрібно самостійно записувати результати вимірювань. При необхідності, у додатку можливо зберігати та редагувати записи для декількох людей. Ця програма надає користувачеві можливість зберігати дані, а згодом переглядати історію симптомів, користувач може переглядати дані у вигляді діаграм. Так є змога перевірити діапазон температур вашого тіла на графіку на основі збережених даних. Можна отримати доступ до збережених записів температури будь-коли.

Додаток може генерувати щомісячні, щотижневі та річні звіти користувача. Окремо вказуються максимальні та мінімальні значення, щоб була змога легко відстежувати свій стан. При цьому додаток прагне підказувати (значення підсвічуються різними кольорами), коли цифри перевищують нормальні показники. Ця програма допоможе користувачеві з'ясувати його або її інтенсивність лихоманки.

Нормальна температура тіла залежить від таких факторів, як вік, стать та кліматичні умови, що також потрібно вказати у додатку. Оскільки нормальна температура тіла становить 36,5 градусів до 37,5 градусів за шкалою Цельсія та 97,7 градусів до 99,5 градусів за шкалою Фаренгейта.

Автоматично можна створювати діаграми зі звітами. Існує можливість конвертації значень з Цельсія в Фаренгейт. [1,3]

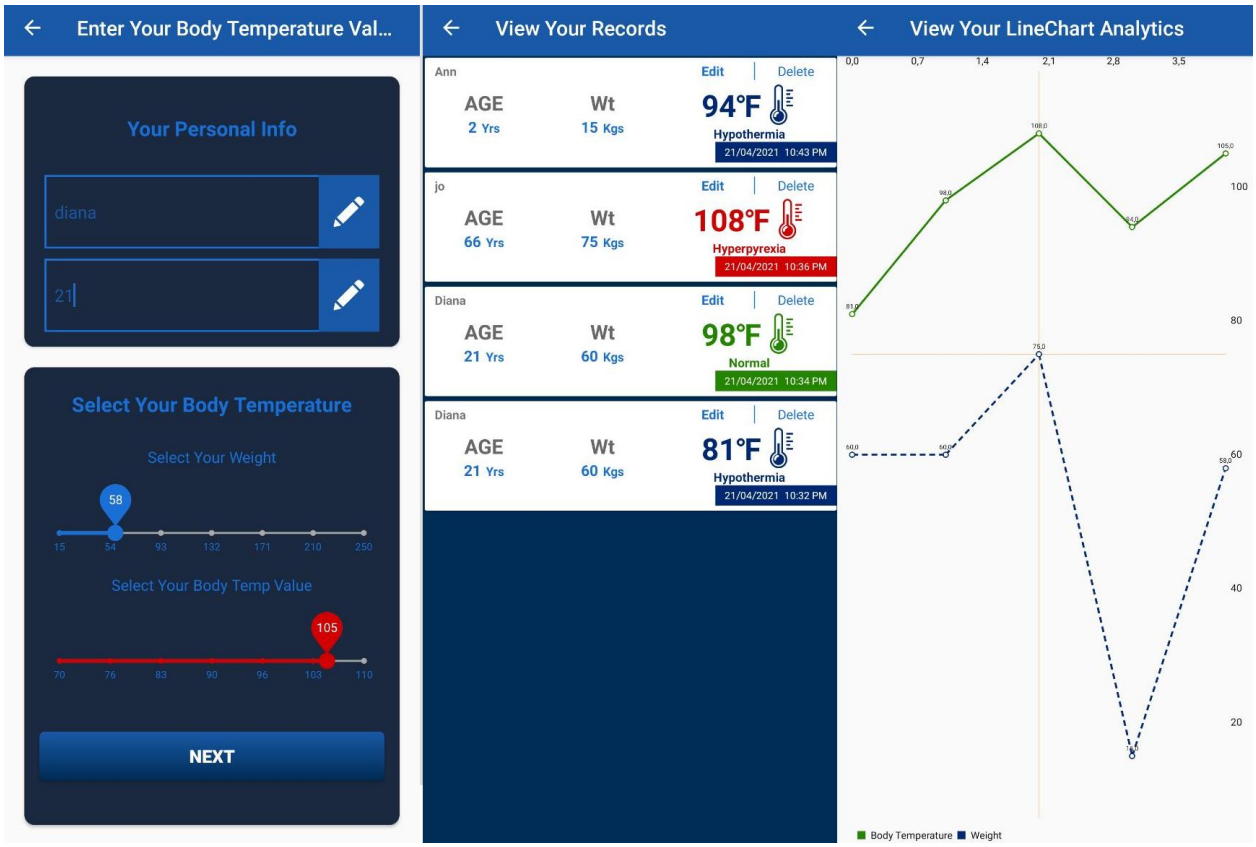


Рис.1.2 - Body Temperature App

1.3 Додаток Thermometer free

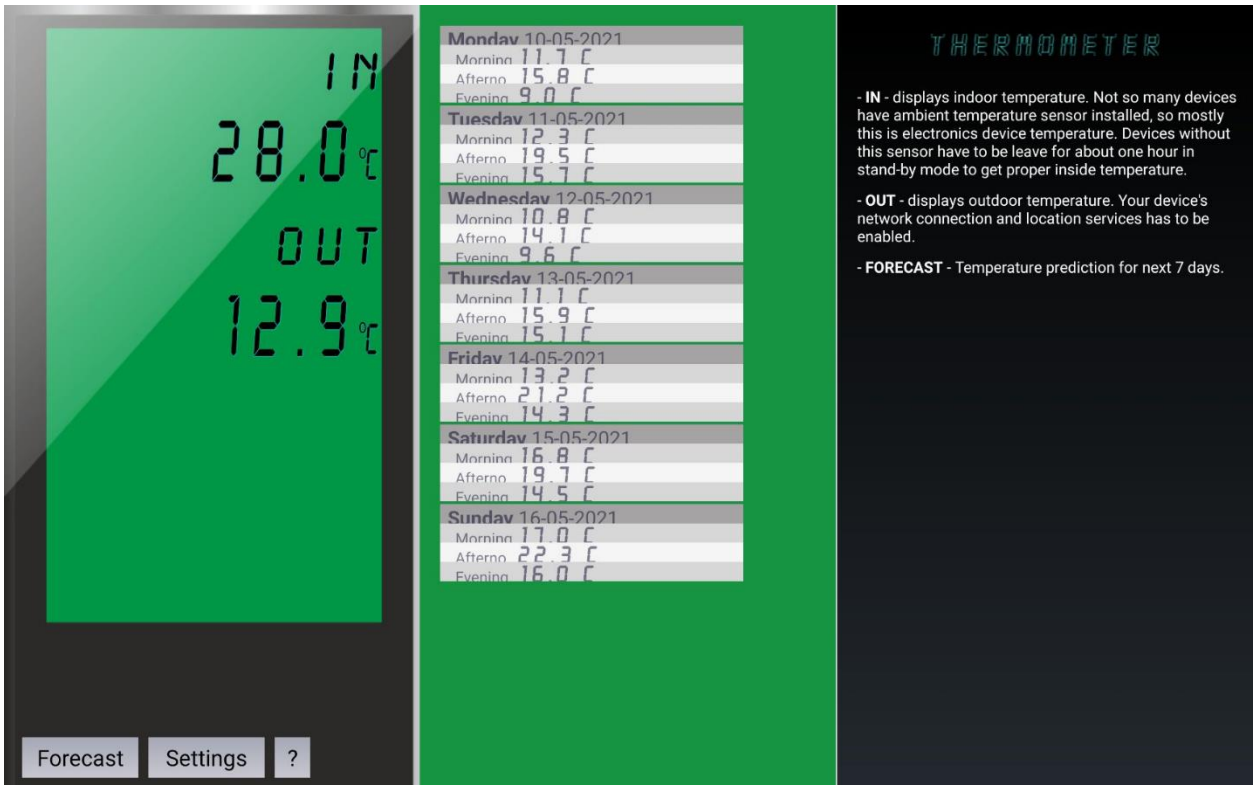


Рис. 1.3 - Thermometer free

Thermometer free - додаток термометра для користувачів Android та iOS, який показує температуру вашого поточного місцезнаходження. Є можливість дізнатися вимірювання внутрішньої та зовнішньої температури за допомогою цього додатка.

Щоб користуватись цією програмою користувачу необхідно увімкнути розташування на своєму смартфоні. Для кращого використання даної програми слід мати підключення для Інтернету, а також дозволити отримувати доступ до місцезнаходження.

Не так багато пристроїв мають датчик зовнішньої температури, тому показані дані температури повітря в приміщенні - це температура електроніки пристрою. Пристрої, що не мають цього датчика, мають бути залишені приблизно на годину в режимі очікування, щоб отримати актуальні дані.

Щоб скористатися функцією прогнозу (огляд на наступні 7 днів), необхідно мати підключення до мережі.

За замовчуванням дані представлені в одиницях Цельсія, в опціях є можливість змінити їх на градуси Фаренгейта. [4]

1.4 Ada Health

Ada Health – додаток, що допомагає користувачеві проаналізувати симптоми та виявити, що їх може викликати - вдень чи вночі без домовленості. Безкоштовна програма перевірки симптомів Ади має можливість допомогти знайти відповіді та повідомити, чи варто звертатися до лікаря.

За допомогою програми користувач може шукати симптоми, пов’язані зі здоров’ям, отримувати уявлення про потенційні причини, а також отримувати чітку, стислу та корисну інформацію про охорону здоров’я для ведення життя без хвороб.

Потрібно надавати відповіді на прості запитання, щодо свого чи чужого здоров'я та симптомів. На основі медичного словника про тисячі розладів та захворювань, Штучний Інтилект додатка оцінює отримані відповіді.

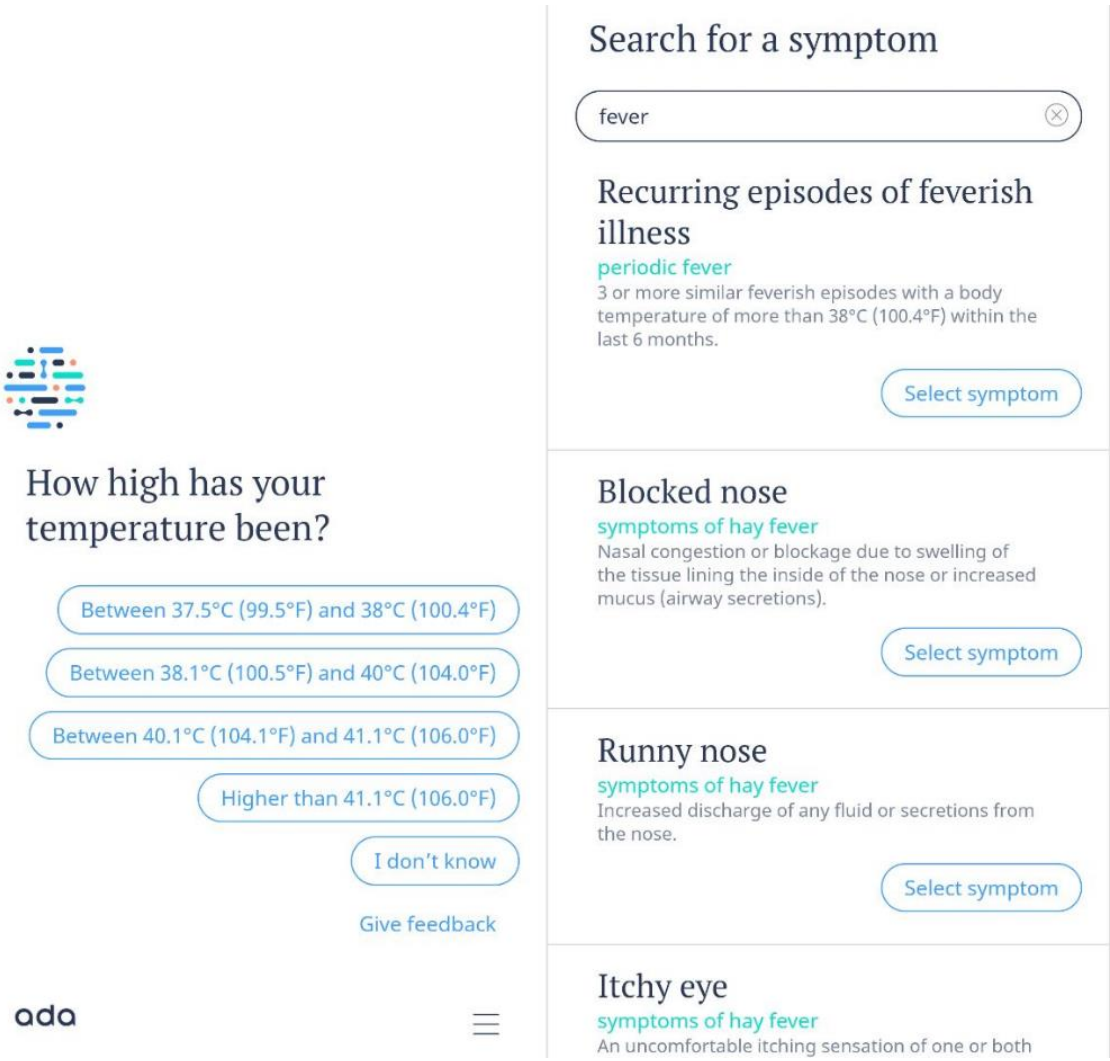


Рис. 1.4 - Збір анамнезу у Ada Health

Користувач отримуєте персональний звіт про оцінку, який повідомляє, що може бути не так і що робити далі.

Індивідуальні рекомендації відповідають унікальному профілю здоров'я. Якщо оцінюється стан іншої людини, їх інформація залишається окремо від вашої власної.

Оцінки включають інформацію, яка може бути доречною та корисною для сімейного лікаря. Ада відповідає на будь-які медичні питання, наприклад, такі поширені симптоми, як лихоманка, алергічний риніт, втрата апетиту, головний біль, втома, запоморечення та інші.

Додаток не може поставити точний медичний діагноз та не замінює зустріч з лікарем, але надасть можливі варіанти хвороби. [5]

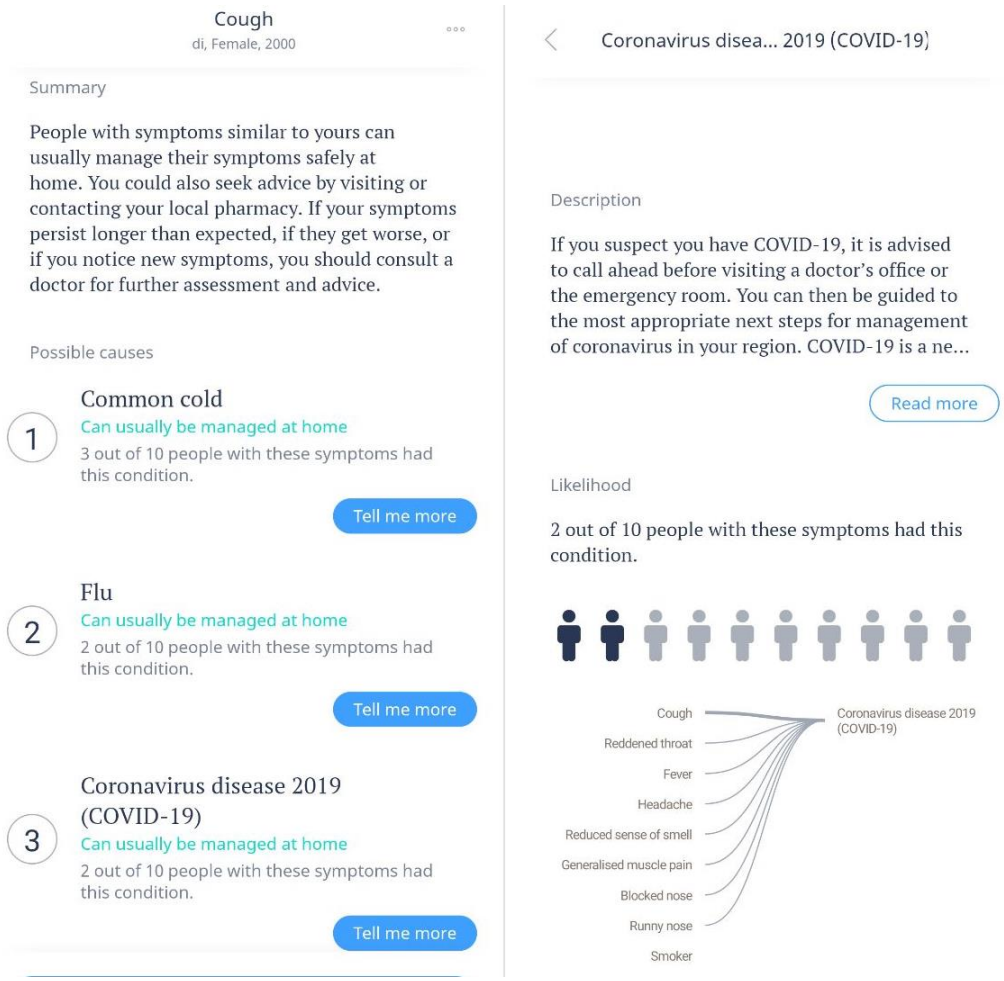


Рис. 1.5 - Пояснення симптомів у Ada Health

1.5 Temperature bot

«Temperature bot» – це чат-бот, реалізований на платформі Facebook Messenger. Система закликає користувачів стежити за станом свого здоров'я і щодня проходити опитування від бота. Існує можливість спостерігати за динамікою показників вказаних вимірів. Кожен день від бота приходить автоматичне повідомлення, з проханням про оновлення зазначених даних. Опитування відбувається в простій і розважальній формі, що полегшує використання бота. На базі отриманих даних від користувачів проводитися деперсоналізована аналітика симптоматики, це є додатковим джерелом для оцінки епідеміологічної ситуації в регіонах України і країні в цілому.

Опитування складається з питання про наявність сухого кашлю, задухи, слабкості, відчуття запаху/смаку, так само запитує значення температури, і наявність поїздок закордон і по Україні в останні кілька тижнів. За схожими

симптомами, які дають привід про занепокоєння, виникає чітке поняття того, де краще зосередитися на тестуванні на COVID-19, і в якому саме місці знаходиться вогнище хвороби. [2]

1.6 CoronaCovid_19_bot

Це чат-бот, який відповідає на всі ключові питання про новий коронавірус SARS-CoV-2 з точки зору останніх наукових даних і доказової медицини.

Функціонал бота спрямований на те, щоб проінформувати користувача про можливі симптоми при захворюванні ковід, так само може надати інформацію про шляхи передачі, профілактики та лікуванні, розвіє міфи про хвороби, котрі можуть ввести в оману, підкаже в виборі аналізу і проконсультує у вакцинації.

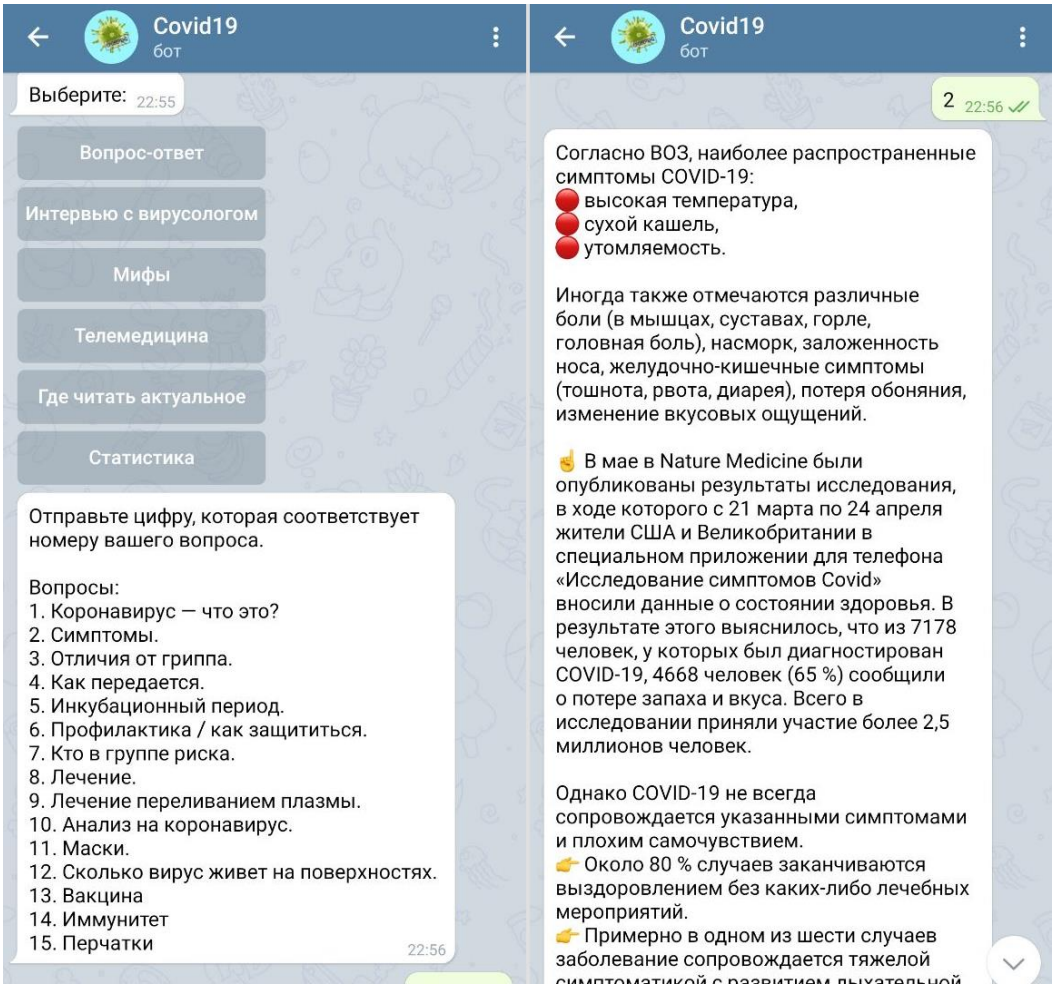


Рис. 1.6 - CoronaCovid_19_bot

1.7 Додаток Thermometer For Fever

Thermometer For Fever - мобільний додаток, щоденник для відстеження температури тіла, яка вимірюється термометром. З переваг програми можна виділити можливість переглянути середню темпаратури тіла за вказаний проміжок часу, подивитися всю інформацію про стан здоров'я за зазначену дату, налаштувати автоматичне оповіщення на різні дати.

Додаток може будувати лінійні діаграми з двома осями, підтримує декількох користувачів, температура відображається і в Цельсіях, і в Фаренгейтах. Додатку не потрібен доступ до Інтернет з'єднання, дані зберігаються локально на пристрої.

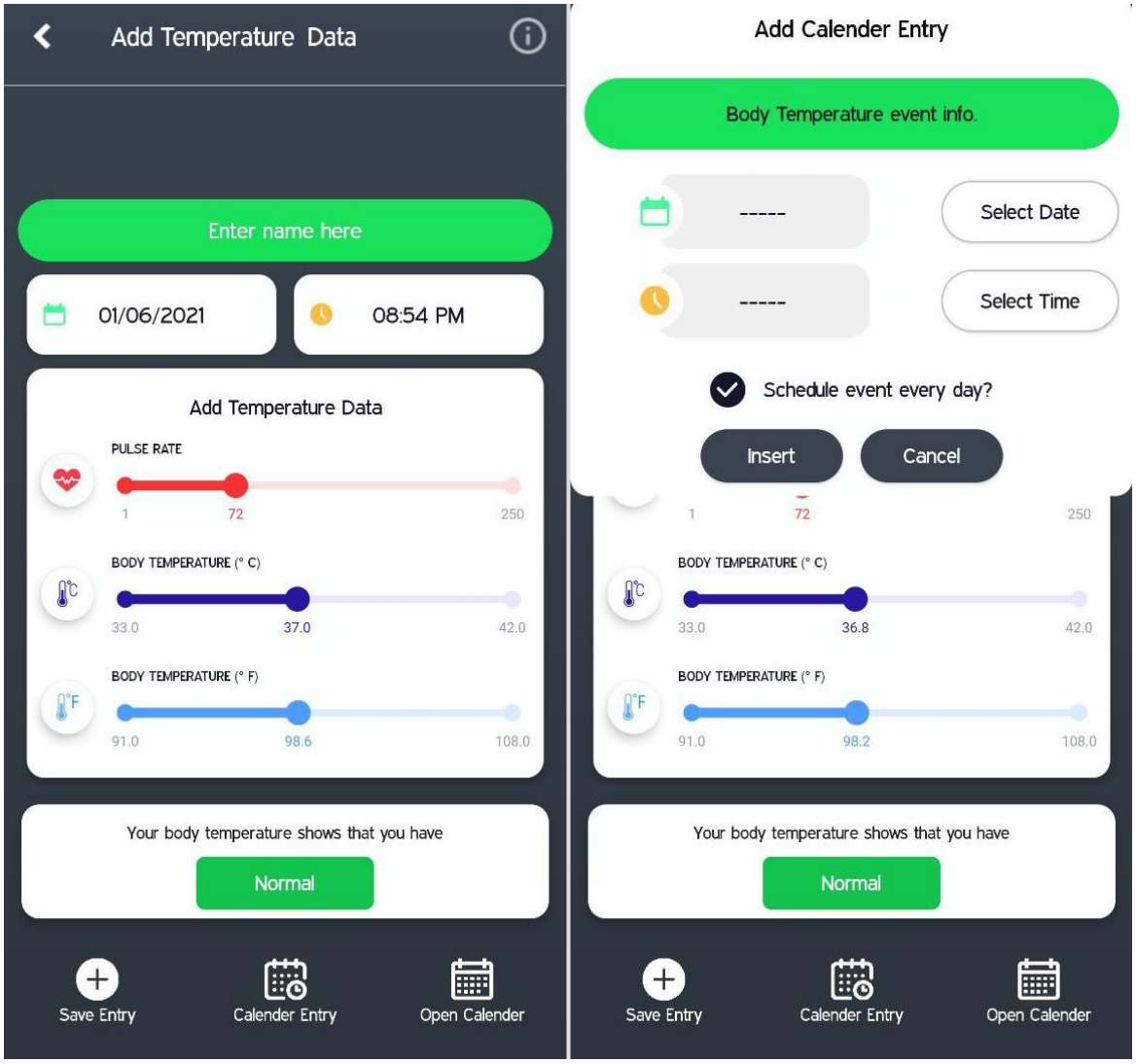


Рис. 1.7 -Thermometer For Fever

Також крім щоденника вимірів температури є можливість вказати і відстежувати артеріальний тиск, прийом медикаментів та інше. Додаток також показує температуру зовні і всередині (температура пристрою). [6]

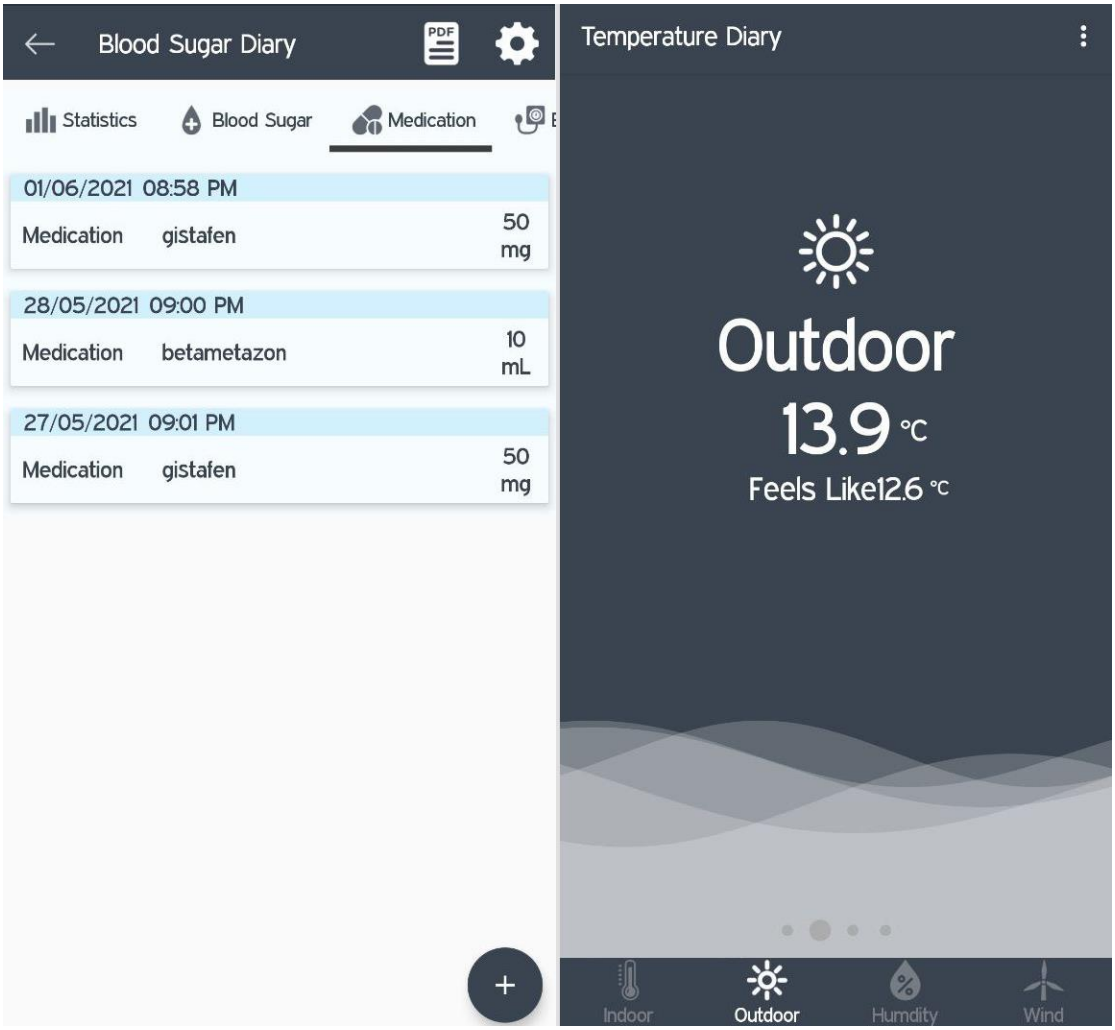


Рис. 1.8 - Додаткові можливості Thermometer For Fever

1.8 Додаток Thermometer

Thermometer - мобільний додаток отримує дані з найближчих метеорологічних станцій і відображає дані користувачеві у Цельсіях чи Фаренгейтах, також показує вологість і тиск на вулиці. Для розташування необхідне підключення до мережі. [7]

Рис. 1.9 - Thermometer

1.9 Аналіз оглянутих рішень

Створимо таблицю і порівняємо розглянуті вище додатки див. таблицю 1.1

Табл. 1.1 Порівнянь додатків

Додаток/ Можливості	Підтримка кількох користувачів	Побудова графіку	Можливість вказати свої симптоми	Можливість добавляти свої коментарі до запису
Body Temperature Recorder	+	+	+	+
Temperature bot	-	-	+	-

Продовження таблиці 1.1

Додаток/ Можливості	Підтримка кількох користувачів	Побудова графіку	Можливість вказати свої симптоми	Можливість добавляти свої коментарі до запису
Body Temperature App	+	+	+	+
Thermometer free	-	-	-	-
Ada Health	+	-	+	-
CoronaCovid_19_ bot	-	-	-	-
Thermometer For Fever	+	+	-	+
Thermometer	-	-	-	-

Продовження таблиці 1.1

Додаток/ Можливості	Можливість Відправити свої результати	Аналізує вказані симптоми	Надає уявлення про потенційну хворобу	Самостійно вимірює температуру
Body Temperature Recorder	+	+	-	-
Temperature bot	-	-	-	-

Продовження таблиці 1.1

Додаток/ Можливості	Можливість Відправити свої результати	Аналізує вказані симптоми	Надає уявлення про потенційну хворобу	Самостійно вимірює температуру
Body Temperature App	+	+	-	-
Thermometer free	-	-	-	+
Ada Health	-	+	+	-
CoronaCovid_19_bot	-	-	+	-
Thermometer For Fever	-	-	-	+
Thermometer	-	-	-	+

ВИСНОВОК ДО РОЗДІЛУ 1

Одним з найбільш корисних і функціональних додатків є Thermometer For Fever. Проект містить в собі можливості кількох інших додатків (що замінить кілька завантажень на одне), таких як ведення щоденника стану здоров'я, можливість конвертувати результати в файл, перегляд графіків, відстеження прийому ліків, значення температур на вулиці і в кімнаті.

Також багатим функціоналом відрізняється додаток Ada Health, котре не тільки зберігає вказану інформацію, але й може надати детальну консультацію за симптомами і причини їх виникнення.

З даного аналізу можна підкреслити корисні функції для реалізації у задуманому боті.

РОЗДІЛ 2
РОЗРОБКА БОТА

2.1 Реєстрація Телеграм боту

BotFather- єдиний бот, для створення і управління іншими ботами. Для початку роботи з ним необхідно нажати кнопку "Старт". Бот відправить повідомлення в якому можна переглянути існуючі команди, основна з них «/new_bot» - необхідна для створення нового бота. Потім з'явиться прохання вказати ім'я та username нового бота. Ім'я буде показуватися в списку контактів, а username служить посиланням на діалог з ботом. Username повинен складатися з 5-32 символів і обов'язково закінчуватися на "_bot" або "Bot".[11]

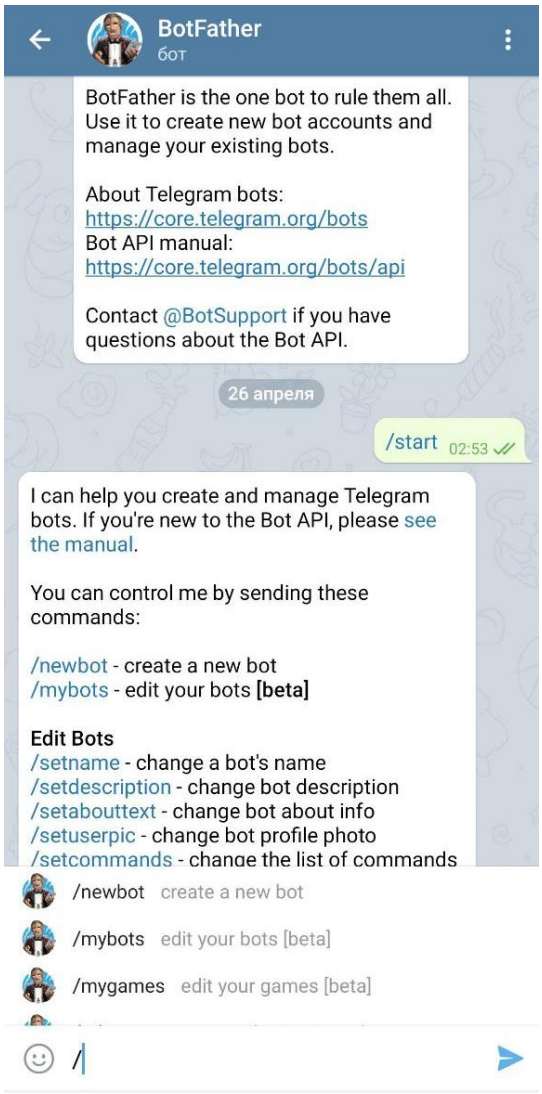


Рис.2.1 - BotFather

Після всіх виконаних кроків бот надасть унікальний токен доступу (рядок з певними символами) до HTTP API, за допомогою якого буде здійснюватися всі маніпуляції зі створеним ботом.

Таблиця 2.1 – редагування боту

Команда	Опис
/setname	Змінює встановлене ім'я бота
/setdescription	Змінює опис бота, який показується у початку діалогу з ботом
/setabouttext	Змінює основну інформацію про бота, яку видно на сторінці профілю, і також відправляється разом з посиланням на бота
/setuserpic	Встановлює нову картинку профілю
/setcommands	Змінює список команд бота. При введенням користувача "/" видно підказки за доступними командам
/deletebot	Видаляє бот і звільнює його ім'я

Через існуючі команди можливо додати опис до боту, встановити аватар, змінити ім'я, додати підказки до командам, які вводяться, видалити бота.

Так само є список команд для змінення самих налаштувань боту.

Таблиця 2.2 – Зміна налаштувань боту

Команда	Опис
/token	Створює новий токен для існуючого бота
/revoke	Анулює токен доступу боту
/setinline	Дає можливість виклику бота з будь-якого чату
/setinlinegeo	Дає можливість дозволити або заборонити запит даних про місцезнаходження
/setjoingroups	Дає можливість дозволити або заборонити приєднання бота до групового чату

Продовження таблиці 2.2

Команда	Опис
/setprivacy	Дозволяє отримувати і обробляти або всі повідомлення з групового чату, або певні. Після зміни налаштувань слід заново додати бота в чат
/setinlinefeedback	Показує інформацію про результати, обрані користувачами. [8, 9]

Після виконаних кроків і налаштувань телеграм бота можна приступати до програмування функціоналу.

2.2 PyCharm - середовище розробки

PyCharm - це кроссплатформне, інтегроване середовище розробки, яке є сумісним з Windows, macOS, Linux.

PyCharm має повноцінну підтримку різних веб-фреймворків і платформ для розробки на Python, а також різних популярних мов:

a) JavaScript
b) TypeScript
c) CoffeeScript
d) HTML / CSS
e) AngularJS
f) Node.js і багато інших.

У PyCharm є ряд функцій, Які максимально оптимізують розробку завдяки аналізу коду, автодоповненню (за першими літерами команди), швидких виправлень і миттєвого підсвічування помилок.

Зручна навігація допомагає швидко переміщатися по проекту, а автоматичний рефакторинг підвищує ефективність редагування коду. IDE має великий набір інструментів: вбудований відладчик, інструмент запуску тестів, інструменти для роботи з базами даних, інструменти для роботи з популярними системами контролю версій, вбудований повнофункціональний термінал. Так само надає можливість віддаленої розробки, підтримує інтеграцію з Docker і Vagrant. PyCharm був розроблений на основі IntelliJ IDEA компанією JetBrains. [18,19]

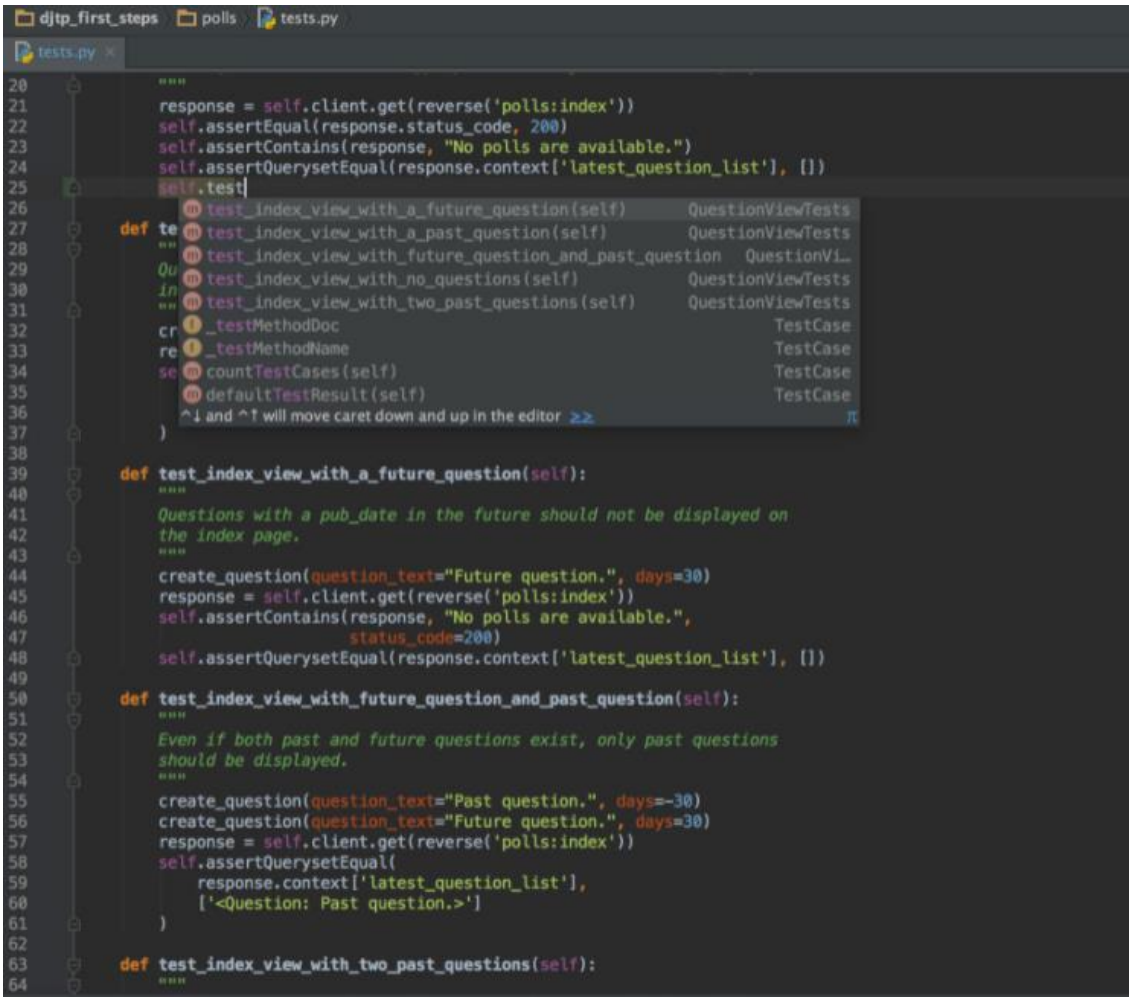


Рис.2.2 - Інтерфейс PyCharm

2.3 Python

Python - це високорівнева інтерпретована мова програмування загального призначення, орієнтований на читабельність коду і підвищення продуктивності розробника. Це універсальна мова програмування, що застосовується у всіх можливих сферах і практично не має обмежень.

Веб-додатки, ігри, машинне навчання, аналіз даних, автоматизація процесів і багато іншого можливо реалізувати за допомогою Python.

Ще один перевагою мови є його простота і компактність коду, в порівнянні з Java і C, що полегшує майбутній супровід програм. Так само плюсом Python є масштабна стандартна бібліотека з автоматичним управлінням пам'яттю, безліч парадигм програмування, можливість інтеграції.

До недоліків мови можна віднести його сповільнену швидкість виконання великих проектів, так як замість компілятора використовується інтерпретатор.

Так само Python має безліч конструктивних обмежень через динамічної типізації, тому може знадобитися більше часу на тестування. Рівні доступу до баз даних можуть показатсья трохи недопрацьованими і примітивними. Також можуть виникати складнощі при переході на інші мови програмування, з більш складним синтаксисом. [10]

2.3.1 Модуль TeleBot

Модуль Телебот є оболонкою для всіх викликів API.

Всі типи визначені в types.py. За винятком поля повідомлення from (оскільки from - це зарезервований токен Python), яке перейменовано в from_user, всі типи повністю відповідають визначенню типів в Telegram API. Тому, до таких атрибутів, як message_id, можна отримати доступ безпосередньо за допомогою message.message_id. Також важливо звернути увагу, що message.chat може бути екземпляром окремого користувача або групового чату. Об'єкт Message має атрибут content_type, що визначає тип повідомлення, і може бути текстом, аудіо, відео, документом, фото, стікером і тд. Так само є можливість використовувати кілька типів в одній функції.

Всі методи API перейменовані в наслідок загальних угод про імена Python і знаходяться в класі TeleBot. У приклад можна привести getMe, який перейменований в get_me, sendMessage перейменований в send_message. Обробники повідомлень (функція, прикрашена декоратором message_handler примірника TeleBot) можуть складатися з одного чи декількох фільтрів. Для певного повідомлення повертається True або False кожним фільтром, в разі True обробник повідомлення має право його обробляти. [8]

2.3.2 Модуль Schedule

Schedule - бібліотека для простого вирішення завдань планування. Використовується для періодичних завдань і запуску функцій в потрібний час. Не має зовнішніх залежностей і використовує простий і зручний, легкий для читання синтаксис.

Schedule не призначений для збереження завдань (розкладу між перезапусками), паралельного виконати (для цього знадобиться модуль

threading) і не враховує час, необхідний для виконання функції завдання, тому для стабільного графіка виконання планування варто перемістити тривалі завдання з основного потоку, де запускається планувальник. [12]

2.3.3 Модуль Time

Модуль time використовується при вирішенні задач, пов'язаних з часом. Семантика функцій може відрізнятися при використанні різних платформ.

Кілька найбільш часто використовуваних функцій:

Табл. 2.3 - Функції модуля Time

Функція	Опис
time.time()	показує число секунд, що пройшли з початку епохи (для Unix 1.01.1970)
time.ctime()	перетворює час, виражений в секундах з початку епохи, в рядок, що представляє місцевий час
time.sleep()	відкладає виконання даного потоку на вказану кількість секунд. [13]

2.3.4 Модуль Threading

Модуль threading полегшує роботу з потоками і дозволяє запускати кілька завдань одночасно.

Для більших проектів і кращого використання обчислювальних ресурсів, більше підійде модуль multiprocessing. Через те, що інтерпретатор Python запускає всі потоки всередині головного потоку. Тому якщо є необхідність запустити кілька інтенсивних операцій з потоками, швидкість роботи може буде невисокою.

Модуль threading також має клас Thread, необхідний для підтримки об'єктно-орієнтованій реалізації потоків.

Розглянемо методи у даному класі і для чого вони існують, у вигляді таблиці:

Табл. 2.4 Методи класа Thread

Метод	Опис
run ()	виповнюється при ініціалізації і запуску нового потоку
start ()	запускає ініціалізований об'єкт потоку, за допомогою виклику відповідного методу run ()
join ()	очікує завершення відповідного об'єкта потоку перш ніж продовжити виконання решти програми
isAlive ()	повертає значення true або false, яке вказує на виконання об'єкта потоку в даний момент
getName ()	повертає назву об'єкта потоку
setName ()	встановлює відповідну назву викликаного об'єкта потоку. [14]

2.4 СУБД SQLite

SQLite - це інтегрована бібліотека, яка реалізує самостійний, безсерверний, транзакційний механізм баз даних SQL. Цю базу даних як і інші бази даних, не потрібно налаштовувати у своїй системі. SQLite написана на Сі і має відкритий вихідний код.

Механізм SQLite не є самостійним процесом, як інші бази даних, користувач може пов’язати її із потрібним додатком, статично або динамічно відповідно до заданих вимог. SQLite отримує безпосередній доступ до своїх файлів зберігання.

Серед переваг SQLite можна виділити такі:

- не вимагає роботи окремого серверного процесу або системи (без серверів).
- налаштування та адміністрування не потрібні, бо SQLite поставляється з нульовою конфігурацією.

- Повна база даних зберігається в одному крос-платформному дисковому файлі.
- SQLite дуже мала та компактна за розміром, менше 400 КБ, коли все повністю налаштовано або менше 250 КБ, коли деякі додаткові функції опущено.
- відсутність зовнішніх залежностей, що підтверджує її автономність.
- безпечний доступ з декількох процесів або потоків, що забезпечується повною сумісністю транзакцій SQLite з ACID
- підтримується більша частина функцій мови запитів, існуючих у стандарті SQL92 (SQL2).
- написаний на ANSI-C і забезпечує простий і зручний у використанні API.
- доступна в як в Windows (Win32, WinCE, WinRT), так і в UNIX (Linux, Mac OS-X, Android, iOS). [15]

Розглянемо кілька прикладів застосування СУБД:

```
C:\>sqlite3
SQLite version 3.7.15.2 2013-01-09 11:53:05
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

Рис. 2.3 - Початок роботи з SQLite

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print "Opened database successfully";
```

Рис. 2.4 - Підключення до бази даних

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute('''CREATE TABLE COMPANY
         (ID INT PRIMARY KEY     NOT NULL,
         NAME           TEXT    NOT NULL,
         AGE            INT     NOT NULL,
         ADDRESS        CHAR(50),
         SALARY         REAL);''')
print "Table created successfully";

conn.close()
```

Рис. 2.5 - Створення таблиці

Існує 4 основних операції Insert, Select, Update, Delete.

```
#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");

conn.commit()
print "Records created successfully";
conn.close()
```

Рис. 2.6 - Операція INSERT

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

Рис. 2.7 - Операція SELECT

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")
conn.commit()
print "Total number of rows updated :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

Рис. 2.8 - Операція UPDATE

```
conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("DELETE from COMPANY where ID = 2;")
conn.commit()
print "Total number of rows deleted :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

Рис. 2.9 - Операція DELETE

Необхідно знати процедури для роботи з базою даних SQLite з Python програми користувача.

Табл. 2.5 - Методів SQLite

API	Опис
sqlite3.connect	Відкриває підключення до файлу бази даних. ":memory:" використовується для підключення до бази даних, що знаходиться в оперативній пам'яті, а не на диску. При успішному відкритті бази даних, вона повертає об'єкт з'єднання. Якщо процес модифікує базу даних, вона блокується, доки дія не буде завершена. Цей виклик створить базу даних, якщо вона не існує. Також можна вказати шлях, де користувач хоче створити базу даних.
connection.cursor	Створює курсор для програмування бази даних на Python. Має один необов'язковий параметр cursorClass.
cursor.execute	Виконує оператор SQL, який може бути параметризований. Модуль sqlite3 приймає два види заповнювачів: іменовані та знаки питання.

Продовження табл. 2.5

API	Опис
connection.execute	Створує проміжний об'єкт и викликає метод виконання курсора.
connection.commit()	Фіксує зміни у базі даних. Без цього методу змін з іншого з'єднання не видно.
connection.close()	Закриває з'єднання з базою даних. Без попереднього commit дані будуть втрачені.
cursor.fetchone()	Отримує рядок набору результатів запиту, повертає одну послідовність або None, якщо даних немає.
cursor.fetchall()	Отримує всі рядки результату запиту, що залишилися, повертаючи список. Список буде порожнім, якщо немає рядків. [16]

ВИСНОВОК ДО РОЗДІЛУ 2

Були розглянуті основні технології, що застосовуються в розробці бота і їх застосування. Показано недоліки і переваги, яких було більше. Обґрунтовано причини, за якими був обраний даний шлях створення бота. Описано основні функції бібліотек Time, Threading, Schedule. Розглянуто тонкощі роботи з СУБД SQLite.

РОЗДІЛ 3

TELEGRAM BOT API

Bot API - це HTTP-інтерфейс націлений на роботу з ботами в Telegram. Кожен бот має свій унікальний токен. В Bot Telegram API всі необхідні запити здійснюються за допомогою HTTPS у вигляді: "https://api.telegram.org/bot<token>/METHOD_NAME".

Підтримуються GET і POST запити. Існують чотири способи для передачі параметрів в Bot API:

- запит URL
- application / x-www-form-urlencoded
- application / json (крім завантаження файлів)
- multipart / form-data (для завантаження файлів)

У відповідь надсилається об'єкт JSON, який має булеве поле "ok" і "description", необов'язкове поле, що містить опис результату. Якщо значення поля "ok" відповідає true, то запит пройшов успішно і в поле "result" можна знайти результат виконання. Якщо значення поля "ok" відповідає false, то причини помилки описані в "description". Також присутнє цілочисельне поле "error_code", вміст якого може змінюватися в майбутньому.

Для отримання оновлень існують два взаємовиключних способу: getUpdates і Webhooks. Оновлення зберігаються на сервері не довше ніж 24 годин, поки не будуть оброблені. Не залежно від обраного способу відповіддю буде об'єкт Update, серіалізований в JSON.

Об'єкт Update - це вхідне оновлення (дія вчинена з ботом), в якому присутній максимум один необов'язковий параметр.

Табл. 3.1 Параметры Update

Поле	Опис
message	Не обов'язково. Нове вхідне повідомлення будь-якого типу (текст / фото / відео)

Продовження табл. 3.1

Поле	Опис
update_id	Унікальний ідентифікатор оновлення, що починається з позитивного числа і поступово збільшується. Найбільш зручний при використанні Webhooks.
edited_message	Не обов'язково. Нова відредагована версія повідомлення
channel_post	Не обов'язково. Нові повідомлення будь-якого типу на каналі
edited_channel_post	Не обов'язково. Нова відредагована версія повідомлення на каналі
inline_query	Не обов'язково. Новий вхідний вбудований запит.
chosen_inline_result	Не обов'язково. Результат вбудованого запиту, обраний користувачем і надісланий іншому користувачу чату.
callback_query	Не обов'язково. Новий вхідний запит при натисканні на кнопку.
shipping_query	Не обов'язково. Новий вхідний запит на доставку для рахунків із гнучкою ціною.
pre_checkout_query	Не обов'язково. Новий вхідний запит з повною інформацією про передоплату.
poll	Не обов'язково. Новий стан опитування, який відправив бот.
poll_answer	Не обов'язково. Зміна відповіді в неанонімному опитуванні, що відправив бот.

Продовження табл. 3.1

Поле	Опис
my_chat_member	Не обов'язково. Оновлення статусу бота в чаті.
chat_member	Не обов'язково. Оновлення статусу користувача в чаті.

Найбільш простий метод отримання оновлень - polling, полягає в періодичному опитуванні серверів телеграм на наявність оновлень. З недоліків методу можна вказати частоту опитування, яка обмежує актуальність даних про події.

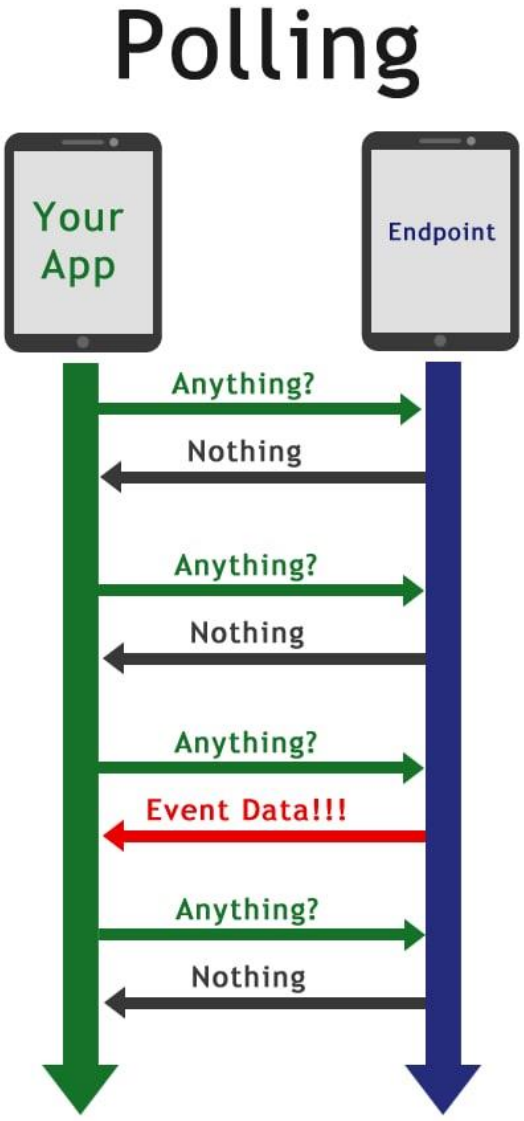


Рис. 3.1 - Принцип роботи polling

При використанні Webhook не відбувається постійного опитування на наявність оновлень, сервер сам повідомляє про наявність нової інформації, відбувається практично миттєве оновлення. Але для цього способу необхідна установка веб-сервера. Також робота веб-хуків в Телеграм здійснюється по HTTPS, тому потрібно мати власний сертифікат-SSL (Secure Sockets Layer). [17]

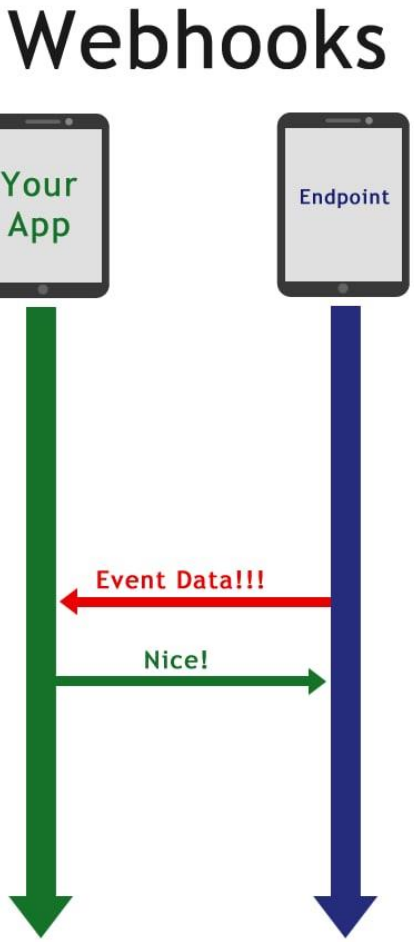


Рис. 3.2 - Принцип роботи Webhook

Розглянемо кілька доступних методів для API:

- getUpdates - метод для отримання вхідних оновлень за допомогою long polling. Повертається масив об’єктів Update.
- setWebhook – метод використовується, щоб вказати URL-адресу та отримувати оновлення через веб-хук.
- sendMessage - метод для відправки текстових повідомлень. При успіху, повертається відправлене повідомлення.

- getChat - метод отримання актуальної інформації про чат. (поточний юзернейм, ім'я користувача, групи, каналу тощо)
- answerCallbackQuery - метод використовується для відправлення відповідей на запити, що передаються при натисканні на кнопок.

getUpdates - метод отримання оновлення через спосіб long polling. Повертає масив об'єктів Update. Якщо налаштувати веб-хук, цей метод працювати не буде.

Табл. 3.2 - параметри getUpdates

Параметр	Тип	Опис
offset	Integer	Не обов'язковий. Ідентифікатор першого оновлення, на одиницю більший за ідентифікатор попереднього. Після зсуву в масиві Update, усі попередні оновлення забуваються.
limit	Integer	Не обов'язковий. Обмежує кількість отриманих оновленнь. За замовчуванням значення 100, дозволені значення від 1 до 100.
timeout	Integer	Не обов'язковий. Час очікування, вказаний у секундах для запросів long polling. За замовчуванням встановлено 0, використовується для тестування як short polling.
allowed_update	Array of String	Не обов'язковий. Серіалізований JSON список типів оновлень, які користувач отримує від бота.

Метод setWebhook використовується для отримання оновлень через веб-хук. Завжди коли наявні оновлення для бота, надсилається запит HTTPS POST на вказану URL-адресу, що містить оновлення серіалізоване в JSON. При

невдалому запиті робить декілька спроб до відмови. При успіху повертає значення "True". Рекомендується включити токен у URL адресу, наприклад: «https://www.examplewebhook.com/ <token>». Так як токен бота відомий лише користувачу, це гарантує, що запроси відсилає саме Телеграм.

Табл. 3.3 - Параметри setWebhook

Параметр	Тип	Опис
url	String	Адреса для надсилання оновлень. Порожній рядок використовується для видалення інтеграції
certificate	InputFile	Не обов'язковий. Завантаження публічного ключа сертифікату для здійснення перевірки кореневого сертифікату.
ip_address	String	Не обов'язковий. Фіксована IP-адреса для надсилання запитів веб-хуків
max_connections	Integer	Не обов'язковий. Найбільша кількість одночасних HTTPS-з'єднань з веб-хуком для отримання оновлень, від 1 до 100. Значення за замовчуванням - 40. Меньші значення призначені обмежити навантаження на сервер, а більші – збільшити пропускну здатность бота.
allowed_updates	Array of String	Не обов'язковий. Типи оновлень, які користувач хоче отримати від бота. Аналогічно до параметру у getUpdates.
drop_pending_updates	Boolean	Не обов'язковий. Скидає всі очікувані оновлення при «True».

Якщо користувач вирішив повернутися до getUpdates, існує метод deleteWebhook для скасування інтеграції веб-хука.

Щоб дізнатися актуальний стан веб-хука використовується метод getWebhookInfo, який у разі успіху повертає об'єкт WebhookInfo і не вимагає параметрів.

WebhookInfo показує дані про поточний стан веб-хука.

Для відправки текстових повідомлень використовується метод sendMessage.

Табл. 3.4 Параметри метода sendMessage

Параметр	Тип	Опис
chat_id	Integer or String	Унікальний ідентифікатор чату або username каналу
text	String	Текст повідомлення, від 1 до 4096 символів
parse_mode	String	Аналізує символи у тексті повідомлення.
entities	Array of MessageEntity	Список спеціальних символів
disable_web_page_preview	Boolean	Для конкретного повідомлення вимикає попередній перегляд посилань
disable_notification	Boolean	Беззвучне отримання повідомлень
reply_to_message_id	Integer	Повідомлення на яке надсилається відповідь

Продовження табл. 3.4

Параметр	Тип	Опис
allow_sending_without_reply	Boolean	True, якщо повідомлення необхідно відправити в будь-якому випадку, не залежно від наявності повідомлень із відповіддю.
reply_markup	InlineKeyboardMarkup	Об'єкт JSON для різних клавіатур відповідей.

Метод answerCallbackQuery використовується для надсилання відповідей після натискання на кнопки на вбудованих клавіатурах.

Табл. 3.5 - Параметри метода answerCallbackQuery

Параметр	Тип	Опис
callback_query_id	String	Ідентифікатор запиту, який потребує відповіді.
text	String	Текст повідомлення. Містить від 0 до 200 символів. При 0 користувачеві нічого не показується.
show_alert	Boolean	При True, буде надіслано попередження замість повідомлення у верхній частині екрану
url	String	URL-адреса, яка відкривається користувачем. Дає можливість відкрити гру, якщо вона вже створена через @Botfather і використовується кнопка callback_game
cache_time	Integer	найбільша кількість секунд, коли резутат запиту кешируєтся на стороні клієнта.

User - об'єкт, що описує Телеграм користувачів або ботів.

Табл. 3.6 - Поля об’єкта User та їх опис

Поле	Тип	Опис
id	Integer	Власний унікальний ідентифікатор даного користувача або бота
is_bot	Boolean	Вказує чи являється користувач ботом
first_name	String	Ім’я користувача або бота
last_name	String	Прізвище бота або користувача. Не є обов’язковим
username	String	Username бота або користувача. Не є обов’язковим
language_code	String	IETF тег мови користувача. Не є обов’язковим
can_join_groups	Boolean	Можливість додавати бота в групи. Можна перевірити в методі getMe
can_read_all_group_messages	Boolean	Якщо True, режим приватності для бота вимкнений. Можна перевірити через метод getMe.
supports_inline_queries	Boolean	Чи підтримує бот вбудовані запити. Можна перевірити через метод getMe.

Об’єкт Message описує дані повідомлення.

Табл. 3.7 - Поля об’єкта Message та їх опис

Поле	Тип	Опис
message_id	Integer	Власний унікальний ідентифікатор повідомлення у поточному чаті.

Продовження табл. 3.7

Поле	Тип	Опис
from	User	Відправник. Поле пусте для повідомлень з каналу. Не обов’язкове.
sender_chat	Chat	Відправник, що написав повідомлення від імені чату. Для каналів, супергруп тощо є вони самі. Не обов’язкове.
date	Integer	Дата коли було надіслане повідомлення (за часом Unix)
chat	Chat	Діалог, котрому належить повідомлення
forward_from	User	Відправник оргинального повідомлення, яке було переслано. Не обов’язкове.
reply_to_message	Message	Для повідомлень відправлених у відповідь на оригінальне. Не обов’язкове.
edit_date	Integer	Дата коли повідомлення було останній раз відредаговане(за часом Unix). Не обов’язкове.
text	String	Для текстових повідомлень від 0 до 4096 символів. Не обов’язкове.
contact	Contact	У повідомленні міститься інформація про контакт. Не обов’язкове.
pinned_message	Message	Певне повідомлення закріплюється у верхній частині екрану. Об'єкт Message не містить поле reply_to_message. Не обов’язкове.
new_chat_members	Array of User	Додані нові учасники групи або супергрупи, вказана інформація про них. Не обов’язкове.

Об’єкт Chat описує дані чату.

Табл. 3.8 - Поля об’єкта Chat та їх опис

Поле	Тип	Опис
id	Integer	Унікальний ідентифікатор конкретного чату.
type	String	Тип чату(«група», «супергруппа», «приватний», «канал» тощо)
title	String	Заголовок для групових чатів, супергруп та каналів. Не обов’язкове.
username	String	Username для чатів, супергруп, каналів, якщо можливий. Не обов’язкове.
first_name	String	Ім'я іншої сторони в приватному чаті. Не обов’язкове.
last_name	String	Прізвище іншої сторони в приватному чаті. Не обов’язкове.
photo	ChatPhoto	Фото чату. Можна перевірити тільки через метод getChat.
bio	String	Біографія співрозмовника в приватному чаті. Можна перевірити тільки через метод getChat. Не обов’язкове.

Продовження табл. 3.8

Поле	Тип	Опис
description	String	Опис каналів, груп, супергруп. Можна перевірити тільки через метод getChat. Не обов’язкове.
permissions	ChatPermissions	За замовчуванням дозволи для учасників групп, супергруп, чату. Можна перевірити тільки через метод getChat. Не обов’язкове.
message_auto_delete_time	Integer	Час в секундах, по закінченню якого всі повідомлення відправлені в чат будуть автоматично видалені. Можна перевірити тільки через метод getChat. Не обов’язкове.

Об'єкт InlineKeyboardMarkup описує вбудовану клавіатуру, яка з'являється поруч з повідомленням, до якого належить.

Табл. 3.9 - Поля об’єкта InlineKeyboardMarkup

Поле	Тип	Опис
inline_keyboard	Array of Array of InlineKeyboardButton	Масив рядків кнопок, кожен представлений масивом об’єктів InlineKeyboardButton

Об’єкт InlineKeyboardButton представляє одну кнопку вбудованої клавіатури. Можливо використовувати лише одне з необов’язкових полів.

Табл. 3.10 - Поля об’єкта InlineKeyboardButton

Поле	Тип	Опис
text	String	Текст, що відображається на кнопці.
url	String	Адреса, яка буде відкрита при натисканні на кнопку. Не обов’язкове.
login_url	LoginUrl	URL-адреса HTTP, що використовується для автоматичної авторизації користувача. Може бути заміною для Telegram Login Widget. Не обов’язкове.
callback_data	String	Дані, що надсилаються в запиті при натисканні кнопки, від 1 до 64 байта. Не обов’язкове.
switch_inline_query	String	При встановленні, натискання кнопки запропонує користувачеві вибрати один із своїх чатів, відкрити цей чат та вставити username бота та вказаний вбудований запит у поле введення. Може бути порожнім, і в цьому випадку буде вказано лише username бота. Не обов’язкове.

Продовження табл. 3.10

Поле	Тип	Опис
switch_inline_query_ current_chat	String	При встановленні, натискання кнопки вставить username бота та вказаний вбудований запит у поточне поле введення чату. При порожньому полі, т буде вказано лише username бота. Не обов’язкове.
callback_game	CallbackGame	Опис гри, яка буде запущена, при натисканні на кнопку. Цей тип кнопок завжди перша кнопка в першому рядку. Не обов’язкове.
pay	Boolean	True, щоб надіслати кнопку оплати. Цей тип кнопок завжди перша кнопка в першому рядку. Не обов’язкове.

Об'єкт CallbackQuery представляє вхідний запит від кнопки на вбудованій клавіатурі. Буде присутнє поле message, якщо кнопка, яка ініціювала запит, була прикріплена до повідомлення, надісланого ботом. Буде присутнє поле inline_message_id, якщо кнопка була прикріплена до повідомлення, надісланого через бота (у вбудованому режимі). З полів data або game_short_name буде присутнє тільке одне.

Табл. 3.11- Поля об’єкта CallbackQuery

Поле	Тип	Опис
id	String	Унікальний ідентифікатор для поточного запросу.
from	User	Відправник.
message	Message	Повідомлення після натискання на кнопку, котра відправляє запрос. Якщо повідомлення занадто старе, зміст і дані будуть недоступними. Не обов’язкове.
inline_message_id	String	Ідентифікатор повідомлення, надісланого через бота у вбудованому режимі. Не обов’язкове.
chat_instance	String	Глобальний ідентифікатор, аналогічний чату, до якого було надіслано повідомлення з кнопкою. Може бути корисним для високих балів в іграх.
data	String	Дані, пов’язані з кнопкою callback. Довільні дані можуть бути надіслані у цьому полі. Не обов’язкове.
game_short_name	String	Коротка назва гри, є унікальним ідентифікатором. Не обов’язкове.

Метод copyMessage існує для копіювання повідомлень будь-якого типу, окрім рахункових та службових. Працює аналогічно з методом forwardMessage, але не має посилання на перше повідомлення.

Табл. 3.12 - Парметри методу copyMessage

Параметр	Тип	Опис
chat_id	Integer or String	Унікальний ідентифікатор або username каналу. Обов'язковий
from_chat_id	Integer or String	Унікальний ідентифікатор або username каналу, куди було відправлене оригінальне повідомлення. Обов'язковий
message_id	Integer	Ідентифікатор повідомлення. Обов'язковий
caption	String	Новий підпис для файлів. Від 0 до 1024 сиволів. За замовчуванням зберігається оригінальний
parse_mode	String	Аналіз символів нового підпису
caption_entities	Array of MessageEntity	Список спецальних символів для нового підпису
disable_notification	Boolean	Беззвучне отримання повідомлення
reply_to_message_id	Integer	id оригінального повідомлення для повідомлень-відповідей
allow_sending_ without_reply	Boolean	Якщо True, повідомлення відправляється у будь-якому разі.
reply_markup	InlineKeyboardMarkup	Додаткові можливості інтерфейсу

Метод setMyCommands створений міняти список команд бота.

Табл. 3.13 - Параметри метода setMyCommands

Параметр	Тип	Опис
commands	Array of BotCommand	Об'єкт джсон із списком команд бота, який можна установити як список команд бота. [8,9]

ВИСНОВОК ДО РОЗДІЛУ 3

В данному розділі були розглянуті два методи отримання оновлень Webhook та polling, та принципи їх роботи. У кожного методу свої недоліки та переваги. Для простого бота найлегшим рішенням буде long polling. Для більш складної системи краще використовувати webhook.

Також були детально описані основні об'єкти та їх поля, основні методи та їх параметри для Bot API, які актуальні для створеного бота.

РОЗДІЛ 4
ОГЛЯД ВЛАСНОЇ РЕАЛІЗАЦІЇ

1. Опис власної реалізації

При розробці бота основними функціями для реалізації були:

- збір даних від багатьох користувачів
- оповіщення адміністратора, якщо у когось з користувачів виявлені симптоми
- нагадування проходити опитування кожен день

Бот був створений на мові програмування Python і за допомогою модулів TeleBot, Schedule, Threading, Time. Також для реалізації бота повинні бути продумані відповіді на коректні і некоректні вводи користувача.

Всі введені дані від користувача зберігаються в базі даних, звідти і зчитується інформація, при необхідності відправити повідомлення адміністратору.

2. Тестування створеного бота

Тестування важливий етап в розробці кожного проекту, так як від цього залежить чи буде задоволений користувач і чи працює система в цілому. Тестування бота буде проводитися на телефоні Samsung Galaxy A31, основні характеристики:

- ОС Android 10.0 (Q)
- Встановлена версія Телеграм 7.5.0 (2245)

Спочатку потрібно провести тестування позитивних сценаріїв, а потім перейти до негативних. Для перевірки повного функціоналу бота було створено два акаунта в Телеграм.

Для початку роботи з ботом натискаємо «start». У відповідь бот відправляє свій короткий опис та існуючі команди.

Рис. 4.1 - Початок роботи з ботом

При введенні першого повідомлення від користувача в таблицю бази даних автоматично вносяться значення id та імені відправника, при цьому поле room залишається порожнім. Після запиту на проходження опитування, бот відправить кнопку "Вказати дані", після цього потрібно буде ввести своє прізвище і номер кімнати, для сповіщень і подальшої роботи з ботом.

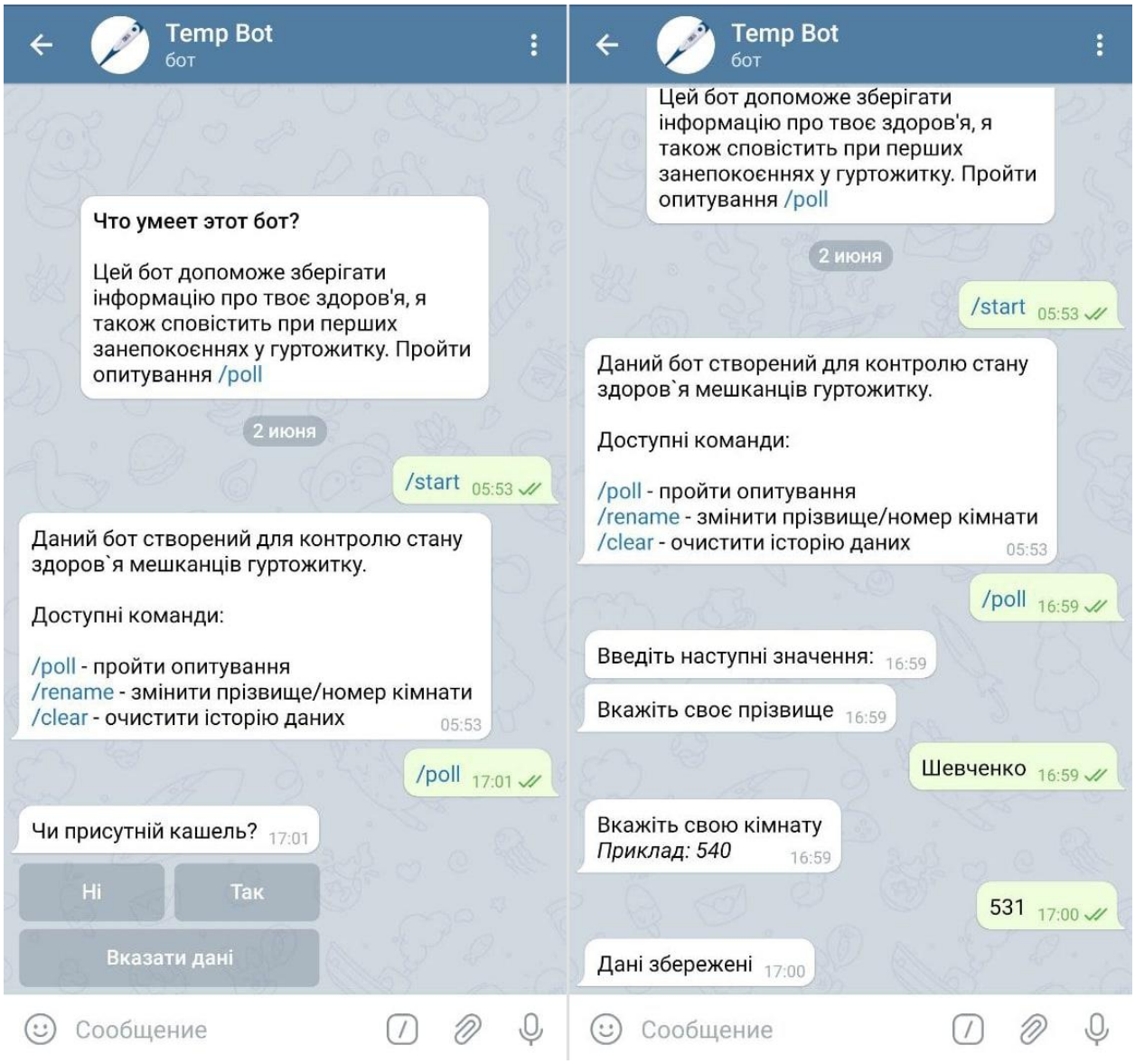


Рис. 4.2 - Вказання даних через бота

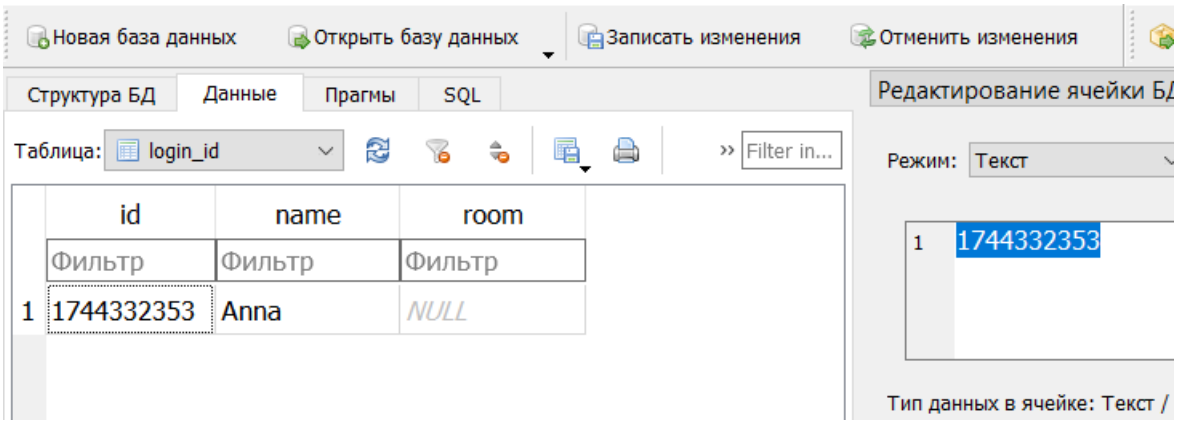


Рис. 4.3 - Таблиця після першого повідомлення користувача

Коли дані вказані користувач відповідає на питання про наявність кашлю, чи не втратив він відчуття запахів і смаку, значення температури. Після кожної відповіді на питання бот виводить відповідний текст і наступне питання, після останнього - дякує користувачу за надану інформацію.

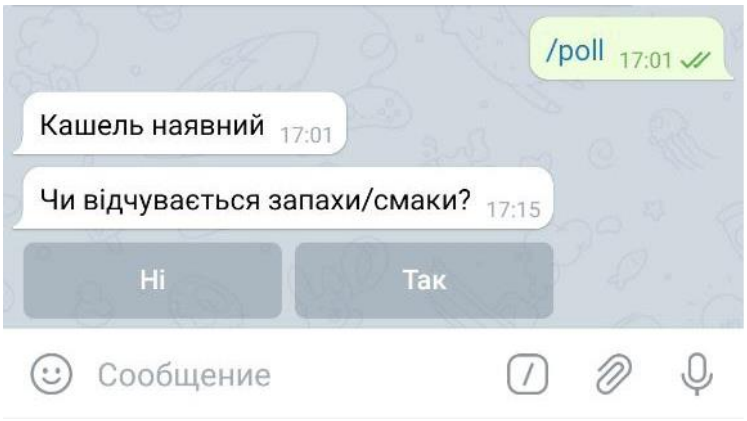


Рис. 4.4 - Приклад опитування

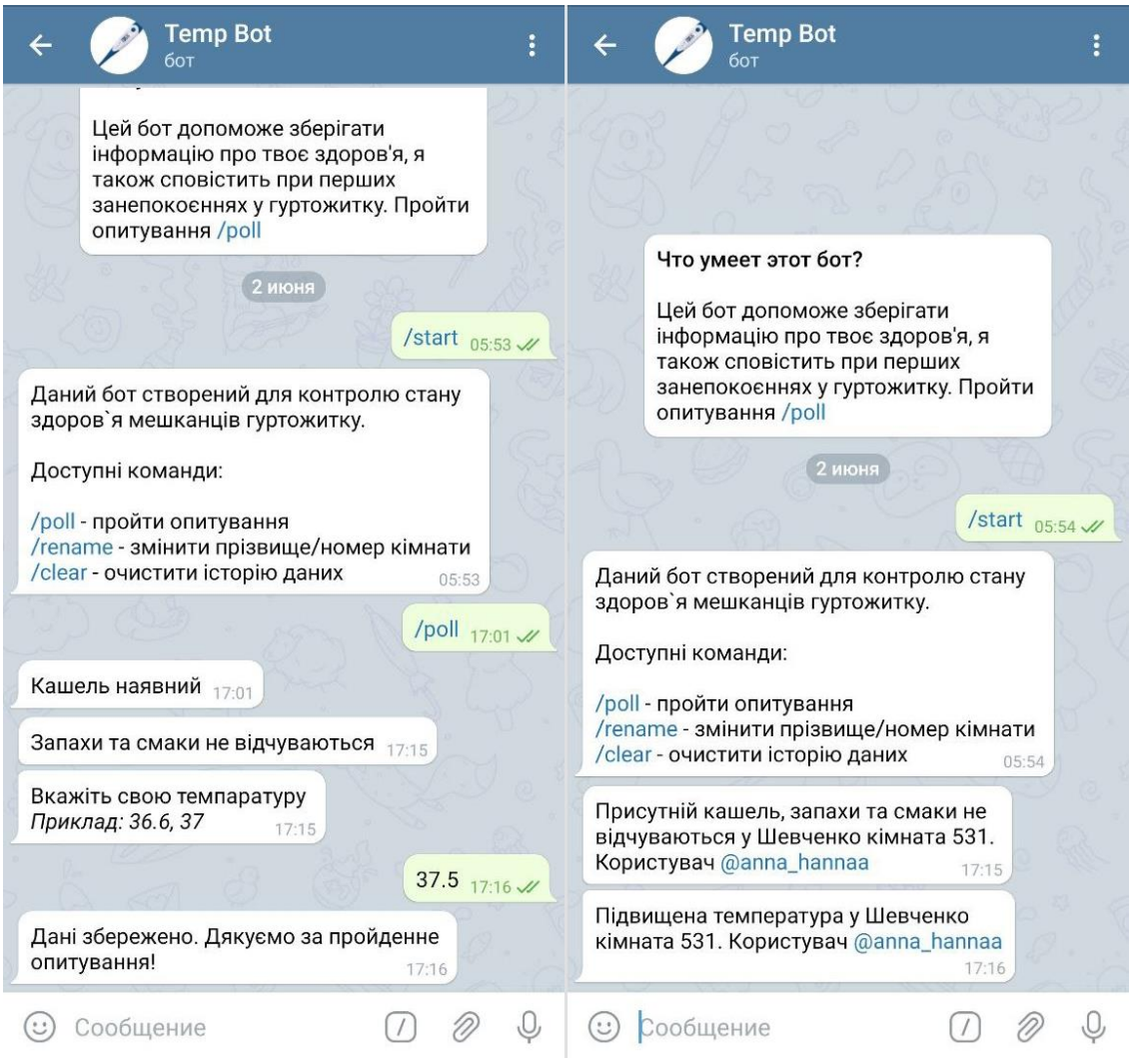


Рис. 4.5 - Оповіщення адміністратора

Якщо в ході опитування виявляється що у користувача є кашель, що не відчуваються запахи і смаки або значення температури перевищує показник 37.2, то адміністратору (для тестування вказувався другий обліковий запис в Телеграм) приходить повідомлення, в якому містяться зазначені симптоми і інформація про жителя гуртожитку: його прізвище, номер кімнати і username, через який з ним можна зв'язатися в Телеграм.

З негативних сценаріїв було протестовано введення некоректного типу значень для температури: текст, кілька точок тощо. Бот буде просити повторно ввести коректні дані температури. У таблицю невірні значення записані не будуть.

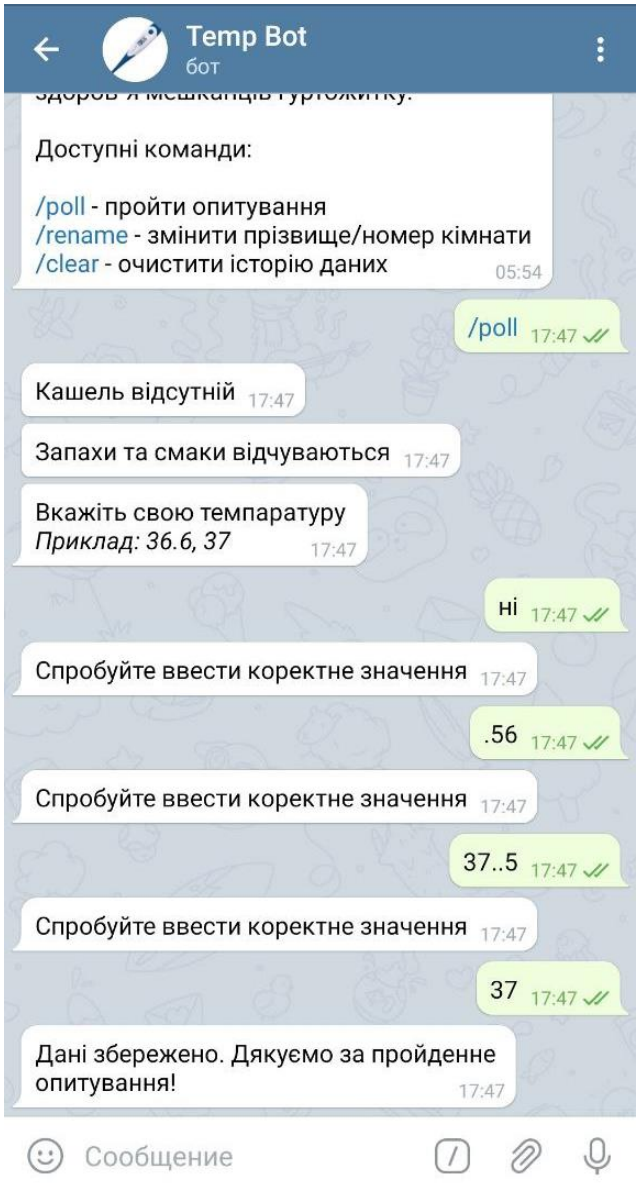


Рис. 4.6 - Тестування на введення некоректних значень

Кожен день в запланований час бот буде автоматично відправляти повідомлення-нагадування про необхідність пройти опитування заново і оновити дані на актуальні.

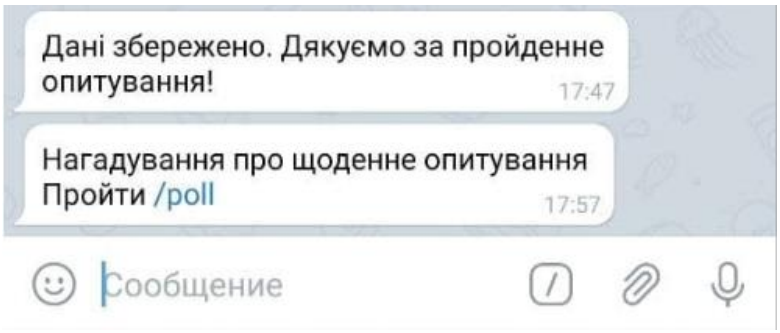


Рис. 4.7 - Нагадування від бота

База даних складається з двох таблиць, в першій міститься унікальний ідентифікатор користувача, його прізвище і номер кімнати. У другій таблиці - унікальний ідентифікатор, дата, коли було пройдено опитування і вказано значення температури (записується число, місяць, рік і час до секунд), і саме поле зі значенням температури.

Новая база данных | Открыть базу данных | Записать изменения

Структура БД | Данные | Прагмы | SQL

Таблица: temp_id

	id	date	temp
	Фильтр	Фильтр	Фильтр
1	1744332353	2021-06-01 07:16:15	37.5
2	354664477	2021-06-01 10:47:44	37
3	1744332353	2021-06-01 14:19:56	38
4	354664477	2021-06-01 16:24:08	36.6
5	354664477	2021-06-01 18:25:07	36.7
6	354664477	2021-06-02 04:20:11	36.6
7	354664477	2021-06-02 14:21:14	36.8
8	354664477	2021-06-01 15:21:25	37
9	354664477	2021-06-02 16:21:35	37.1
10	354664477	2021-06-02 17:21:52	36.9
11	354664477	2021-06-02 18:22:00	37.3
12	354664477	2021-06-02 19:22:16	37.5

Рис. 4.8 - Друга таблиця

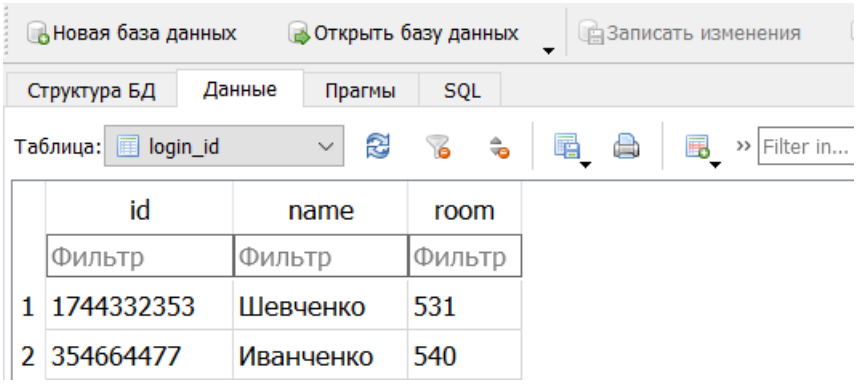


Рис. 4.9 - Перша таблиця

ВИСНОВОК ДО РОЗДІЛУ 4

Були детально розглянуті і протестовані всі можливості бота, обгрунтована мета його створення, проаналізована логіка роботи. Наочно проілюстровані фрагменти чату і відповіді бота при різних запитах. Так само протестований запис в базу даних і описані причини її створення і взаємодія з написаною програмою. Детально описана робота кожної функції і її зв'язок з іншими компонентами програми.

ВИСНОВКИ

Ця дипломна робота була націлена на створення бота для контролю стану здоров'я мешканців гуртожитку.

У першому розділі були розглянуті такі рішення, різні додатки, боти для Фейсбук і телеграм в тому числі. Було проаналізовано роботу кожного проекту і виділені корисні функції для реалізації.

Другий розділ містив в собі розбір актуальних технологій для вирішення подібних завдань. Рассмотрни переваги Python, PyCharm, SQLite. Описано взаємодію з BotFather, його можливості створення і редагування ботів.

У третьому розділі розглянуто роботу Telegram API, способи отримання оновлень і їх відмінності, вигода від застосування кожного методу в різних ситуаціях. Розказано про актуальні методи API.

У четвертому розділі був огляд власного бота, його опис і тестування, логіка роботи програми і підключених компонентів.

В кінцевому підсумку основні задумані можливості для бота були здійснені. Головні функції збору і зберігання даних були успішно реалізовані.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 5 мобильных приложений для отслеживания температуры тела [Електронний ресурс] – Режим доступу до ресурсу: https://blog.themarfa.name/5-mobilnykh-prilozhienii-dlia-otsliezhivaniia-tiempieratury-tiela/
2. Львівський IT Кластер та BotsCrew Запустили Чат-бота для Контролю Стану Здоров'я [Електронний ресурс] – Режим доступу до ресурсу: https://itcluster.lviv.ua/lvivskyj-klaster-ta-botscrew-zapustyly-chat-bota-dlya-kontrolyu-stanu-zdorovya/
3. Body Temperature App [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.brandishapps.bodytemperature
4. Thermometer Free [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=androidesko.android.electronicthermometer&showAllReviews=true
5. Ada – check your health [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.ada.app
6. Thermometer For Fever : Body Temperature Checker [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.thermometerforfever.bloodpressurechecker
7. Thermometer [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=amuseworks.thermometer
8. Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots/api
9. Справочник по Bot API [Електронний ресурс] – Режим доступу до ресурсу: https://tlgrm.ru/docs/bots/api
10. Advantages and Disadvantages of Python Programming Language [Електронний ресурс] – Режим доступу до ресурсу:

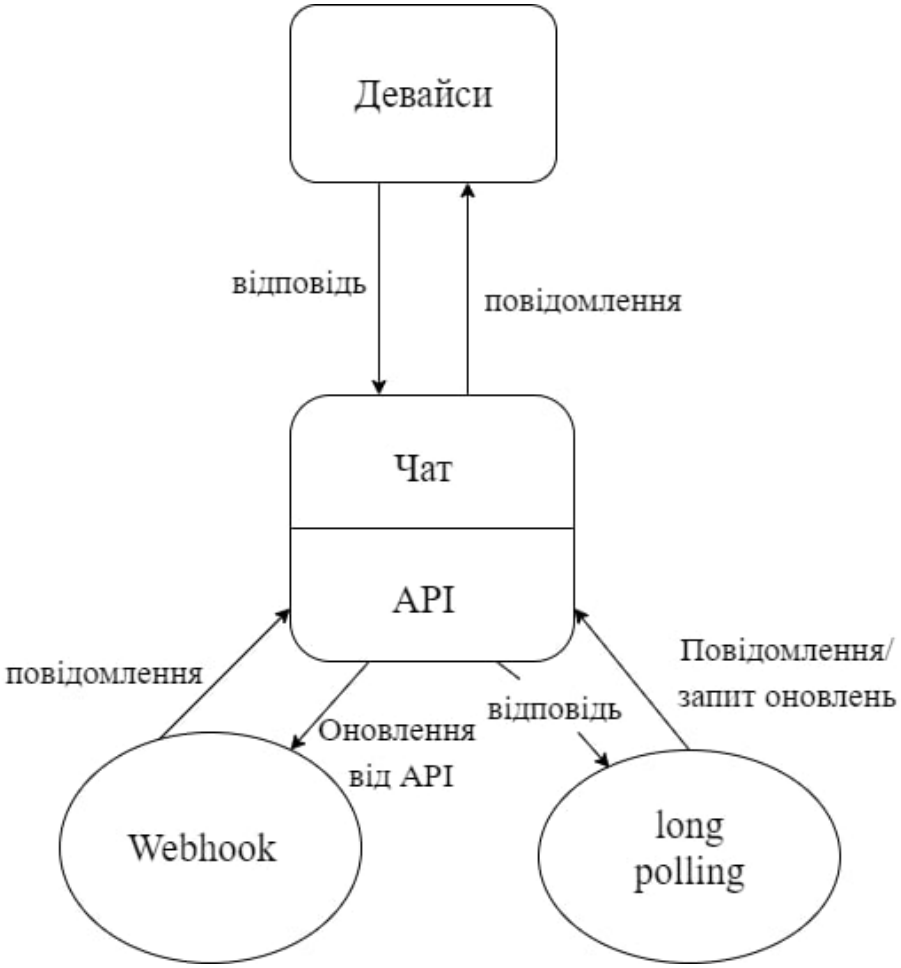
https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121#:~:text=Advantages%20and%20Disadvantages%20of%20Python%20Programming%20Language,-Mindfire%20Solutions&text=Python%20is%20a%20high%2Dlevel,to%20Java%20or%20C%2B%2B.

11. Инструкция: Как создавать ботов в Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/262247/
12. Schedule [Електронний ресурс] – Режим доступу до ресурсу: https://schedule.readthedocs.io/en/stable/
13. time — Time access and conversions [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/time.html
14. threading — Thread-based parallelism [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/threading.html
15. SQLite - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/sqlite/sqlite_overview.htm
16. SQLite - Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/sqlite/sqlite_python.htm
17. WEBHOOKS V.S. POLLING [Електронний ресурс] – Режим доступу до ресурсу: https://blog.cloud-elements.com/webhooks-vs-polling-youre-better-than-this
18. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PyCharm
19. PyCharm: IDE для профессиональной разработки на Python от JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/
20. Body Temperature Recorder - Apps on Google Play [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.tatsuo.bodytemperature

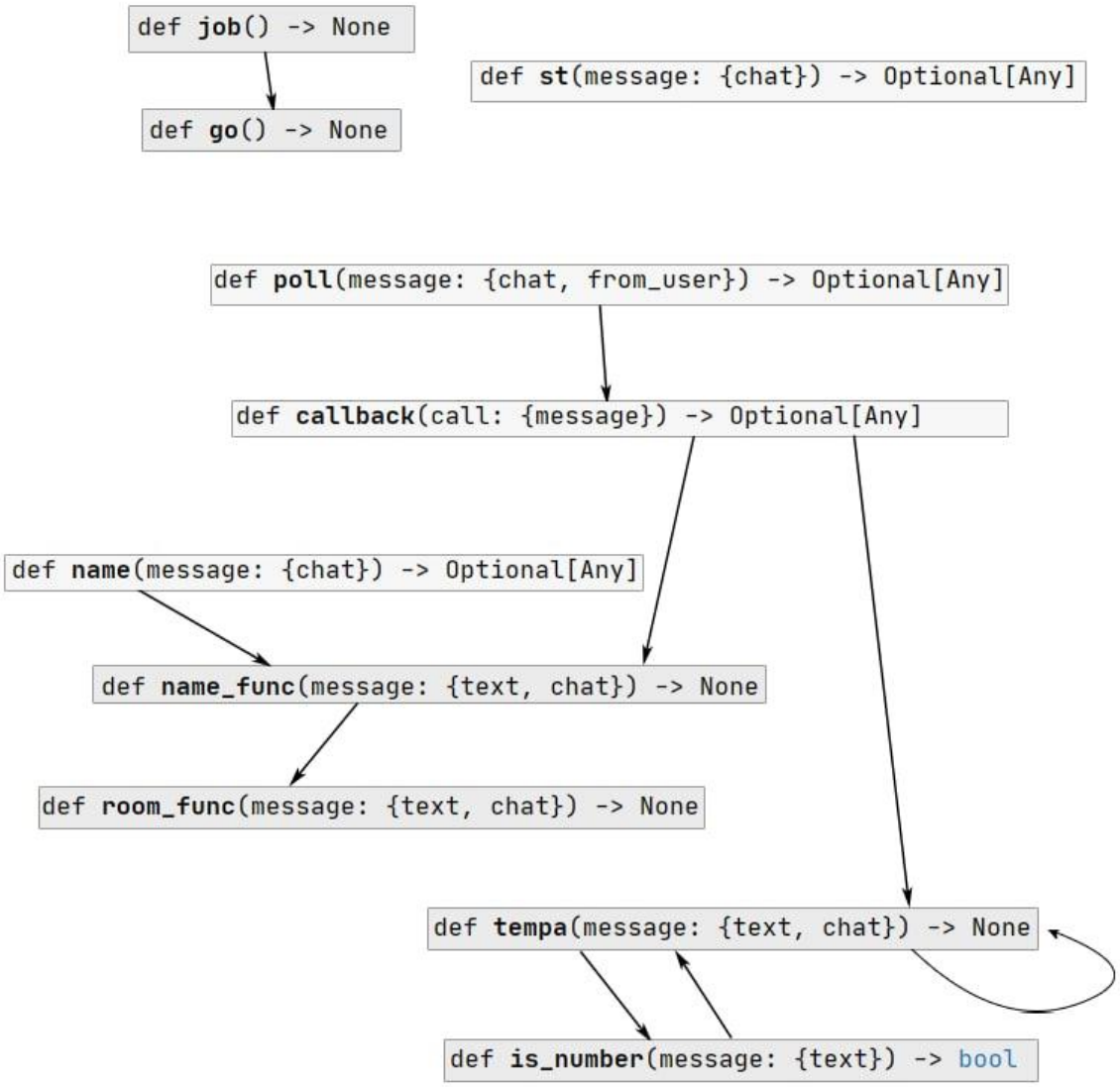
ДОДАТКИ

До дипломного проекту

на тему: «Бот для контролю стану здоров’я»



					ІАЛЦ.467200.004 Д1		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розробив		*Андрюшечкіна Д.Д.*			*Схема структурна*	*Літ.*	*Аркуш*	*Аркушів*
Перевірив		*Калюжний О.О.*					*1*	*1*
Н.контр.		*Сімоненко В.П.*			*Додаток 1*	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72*	
Затв.							



					ІАЛЦ.467200.004 Д2		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розробив		Андрюшечкіна Д.Д.			Схема функціональна	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Калюжний О.О.					
Н.контр.		Сімоненко В.П.			Додаток 2	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72	
Затв.							

					ІАЛЦ.467200.004 ДЗ		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розробив		*Андрюшечкіна Д.Д.*			*Схема принципова*	*Літ.*	*Аркуш* / *Аркушів*
Перевірив		*Калюжний О.О.*					1 / 1
Н.контр.		*Сімоненко В.П.*			*Додаток З*	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72*	
Затв.							

```
 import sqlite3
import telebot
from telebot import types
import threading
import schedule
import time
from datetime import datetime

bot = telebot.TeleBot("token")
connect = sqlite3.connect('users.db')
cursor = connect.cursor()

def job():

    connect = sqlite3.connect('users.db')
    cursor = connect.cursor()
    a = cursor.execute(
        "SELECT id FROM login_id")
    rows = cursor.fetchall()
    for row in range(0, len(rows)):

        bot.send_message(rows[row][0], "Нагадування про щоденне опитування" + "\nПройти
/poll")
        print(rows[row])

#schedule.every(15).minutes.do(job)
schedule.every().day.at("08:55").do(job)
def go():
    while True:
        schedule.run_pending()
        time.sleep(1)

t = threading.Thread(target=go, name="тест")
t.start()

@bot.message_handler(commands=['start'])
def st(message):
    bot.send_message(message.chat.id, 'Даний бот створений для контролю стану здоров`я
мешканців гуртожитку.\n\nДоступні команди:\n\n/poll - пройти опитування\n/rename -
змінити прізвище/номер кімнати\n/clear - очистити історію даних')
```

					ІАЛЦ.467200.004 Д4		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Андрюшечкіна Д..Д			Лістинг програми	Літ.	Арк. 1 / Акрушів 6
Перевір.		Калюжний О.О.					
Н. Контр.		Сімоненко В.П.			Додаток 4	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-72	
Затверд.							

```
@bot.message_handler(commands=['poll'])
def poll(message):
    connect = sqlite3.connect('users.db')
    cursor = connect.cursor()
    cursor.execute("""CREATE TABLE IF NOT EXISTS login_id(
        id,
        name,
        room
    )""")
    cursor.execute("""CREATE TABLE IF NOT EXISTS temp_id(
        id,
        date,
        temp
    )""")
    connect.commit()
    global people_id
    people_id = message.chat.id
    cursor.execute(f"SELECT id FROM login_id WHERE id = {people_id}")
    data = cursor.fetchone()
    if data is None:
        user_id = message.chat.id
        user_name = message.from_user.first_name
        user_room = None
        cursor.execute("INSERT INTO login_id VALUES(?, ?, ?);", (user_id, user_name,
user_room))
        connect.commit()

    else:
        pass
    global u
    u = message.from_user.username

    markup = types.InlineKeyboardMarkup(row_width=2)
    item = types.InlineKeyboardButton('Hi', callback_data='question_1')
    item2 = types.InlineKeyboardButton('Так', callback_data='question_2')
    item3 = types.InlineKeyboardButton('Вказати дані', callback_data='regist')
    markup.add(item, item2, item3)
    bot.send_message(message.chat.id, 'Чи присутній кашель?', reply_markup=markup)


    @bot.callback_query_handler(func=lambda call: True)
    def callback(call):
        if call.message:
            if call.data == 'question_1':
```

```
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
text='Кашель відсутній')
            markup = types.InlineKeyboardMarkup(row_width=2)
            item5 = types.InlineKeyboardButton('Ні', callback_data='question_5')
            item6 = types.InlineKeyboardButton('Так', callback_data='question_6')
            markup.add(item5, item6)
            bot.send_message(call.message.chat.id, 'Чи відчуваються запахи/смаки?',
reply_markup=markup)

        elif call.data == 'regist':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
text='Введіть наступні значення:')
            bot.send_message(call.message.chat.id, "Вкажіть своє прізвище")
            bot.register_next_step_handler(call.message, name_func)

        elif call.data == 'question_2':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
text='Кашель наявний')

            markup = types.InlineKeyboardMarkup(row_width=2)
            item3 = types.InlineKeyboardButton('Ні', callback_data='question_3')
            item4 = types.InlineKeyboardButton('Так', callback_data='question_4')
            markup.add(item3, item4)
            bot.send_message(call.message.chat.id, 'Чи відчувається запахи/смаки?',
reply_markup=markup)


        elif call.data == 'question_4':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
text='Запахи та смаки відчуваються')
            connect = sqlite3.connect('users.db')
            cursor = connect.cursor()
            people_id = call.message.chat.id
            cursor.execute(f"SELECT name FROM login_id WHERE id = {people_id}")
            rows = cursor.fetchone()
            cursor.execute(f"SELECT room FROM login_id WHERE id = {people_id}")
            ro = cursor.fetchone()
            for r in ro:
                for row in rows:
                    bot.send_message('354664477', "Присутній кашель у " + row + " кімната " + r
+ ". Користувач @"+ u )
            bot.send_message(call.message.chat.id, "Вкажіть свою темпаратуру\n" +
"_Приклад: 36\.6, 37_", parse_mode='MarkdownV2')
            bot.register_next_step_handler(call.message, tempa)
```

```
        elif call.data == 'question_3':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
text='Запахи та смаки не відчуваються')
            connect = sqlite3.connect('users.db')
            cursor = connect.cursor()
            people_id = call.message.chat.id
            cursor.execute(f"SELECT name FROM login_id WHERE id = {people_id}")
            rows = cursor.fetchone()
            cursor.execute(f"SELECT room FROM login_id WHERE id = {people_id}")
            ro = cursor.fetchone()
            for r in ro:
                for row in rows:
                    bot.send_message('354664477', "Присутній кашель, запахи та смаки не
відчуваються у " + row + " кімната " + r + ". Користувач @"+ u )

            bot.send_message(call.message.chat.id, "Вкажіть свою темпаратуру\n" +
"_Приклад: 36\.6, 37_",
                             parse_mode='MarkdownV2')
            bot.register_next_step_handler(call.message, tempa)

        elif call.data == 'question_5':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='Запахи та смаки не відчуваються')

            connect = sqlite3.connect('users.db')
            cursor = connect.cursor()
            people_id = call.message.chat.id
            cursor.execute(f"SELECT name FROM login_id WHERE id = {people_id}")
            rows = cursor.fetchone()
            cursor.execute(f"SELECT room FROM login_id WHERE id = {people_id}")
            ro = cursor.fetchone()
            for r in ro:
                for row in rows:
                    bot.send_message('354664477',
                                     "Запахи та смаки не відчуваються у " + row + " кімната " + r + ".
Користувач @" + u)




            bot.send_message(call.message.chat.id, "Вкажіть свою темпаратуру\n" +
"_Приклад: 36\.6, 37_",
                             parse_mode='MarkdownV2')
            bot.register_next_step_handler(call.message, tempa)
```

```
        elif call.data == 'question_6':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='Запахи та смаки відчуваються')

            bot.send_message(call.message.chat.id, "Вкажіть свою темпаратуру\n" +
"_Приклад: 36\.6, 37_",
                             parse_mode='MarkdownV2')
            bot.register_next_step_handler(call.message, tempa)

@bot.message_handler(commands=['rename'])
def name(message):
    bot.send_message(message.chat.id, "Вкажіть своє прізвище")
    bot.register_next_step_handler(message, name_func)

def name_func(message):
    connect = sqlite3.connect('users.db')
    cursor = connect.cursor()
    message_name = message.text
    people_id = message.chat.id
    cursor.execute("UPDATE login_id SET name = ? WHERE id = ?", (message_name,
people_id))
    connect.commit()
    bot.send_message(message.chat.id, "Вкажіть свою кімнату\n" + "_Приклад: 540_",
parse_mode='MarkdownV2')
    bot.register_next_step_handler(message, room_func)

def room_func(message):
    connect = sqlite3.connect('users.db')
    cursor = connect.cursor()
    message_room = message.text
    people_id = message.chat.id
    cursor.execute("UPDATE login_id SET room = ? WHERE id = ?", (message_room,
people_id))
    connect.commit()
    bot.send_message(message.chat.id, "Дані збережені")

def is_number(message):
    arr = message.text.split(".")
    if (len(message.text) == 0 or len(arr) > 2):
        return False
    for i in arr:
        if (not i.isnumeric()):
            return False
    return True
```

```
def tempa(message):
    connect = sqlite3.connect('users.db')
    cursor = connect.cursor()
    message_temp = message.text
    people_id = message.chat.id
    user_time = datetime.fromtimestamp(time.time())
    ust = str(user_time)
    ut = ust[:19]
    if is_number(message):
        cursor.execute("INSERT INTO temp_id VALUES(?, ?, ?);", (people_id, ut,
message_temp))
        connect.commit()
        temperaturka = float(message_temp)
        if temperaturka >= 37.2 :

            connect = sqlite3.connect('users.db')
            cursor = connect.cursor()

            cursor.execute(f"SELECT name FROM login_id WHERE id = {people_id}")
            rows = cursor.fetchone()
            cursor.execute(f"SELECT room FROM login_id WHERE id = {people_id}")
            ro = cursor.fetchone()
            for r in ro:
                for row in rows:
                    bot.send_message('354664477', "Підвищена температура у " + row + " кімната "
+ r + ". Користувач @" + u)

        bot.send_message(people_id, "Дані збережено. Дякуємо за пройденне опитування!")
    else:
        bot.send_message('354664477', "Спробуйте ввести коректне значення")
        bot.register_next_step_handler(message, tempa)
    connect.commit()

if __name__ == '__main__':
    bot.polling(none_stop=True, interval=0)
```