Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення інтелектуального боту для ведення особистого графіку»

Виконав: студент 4 курсу
групи 2ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Довбиш П.І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Бондаренко З.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Довбишу Павлу Івановичу

1. Тема роботи – «Розробка програмного забезпечення інтелектуального боту для ведення особистого графіку».

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 24 березня 2022 р. № 66.

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: середовище розробки Visual Studio 2019, мова розробки C# , операційна система – Windows 10.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; аналіз даних; розробка моделі системи підбору й візуалізації особистого графіку; розробка архітектури та алгоритмів програмного додатка; розробка ER моделі бази даних, розробка програмного додатку; тестування додатку; висновки; список використаних джерел; додатки, графічна частина.

5. Перелік графічного матеріалу: блок-схеми алгоритмів роботи додатку; графічний інтерфейс додатку; тестування додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Черноволик Г.О, к.т.н., доцент кафедри ПЗ	25.03.2022	10.06.2022

7. Дата видачі завдання 25 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач роботи	26.03.2022 – 30.03.2022	Вик.
2	Розробка алгоритмів програмного додатку	01.03.2022 – 01.04.2022	Вик.
3	Розробка програми для реалізації особистого графіку в меседжері Telegram	02.04.2022 – 10.05.2022	Вик.
4	Тестування додатку	11.05.2022 – 10.06.2022	Вик.

Студент _____ Довбиш П. І.

(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Черноволик Г. О.

(підпис) (прізвище та ініціали)

Анотація

Бакалаврська дипломна робота складається з 89 сторінок формату А4, на яких є 51 рисунків, 3 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз методів відбору повідомлень для візуалізації персонального розкладу в telegram-месенджері.

Запропоновано метод створення особистого графіка користувача шляхом використання telegram боту, який буде персоналізовано вибирати актуальну інформацію, із сполучених груп, та формувати на її основі щоденний графік.

Розроблено алгоритми створення персоналізованого графіка, через стрічку повідомлень яка формується, на основі взаємодії користувача з ботом. Бот використовує задані користувачем джерела, з яких і формує повідомлення.

Розроблено програмний продукт з використанням мов програмування C#. Для розробки інтерфейсу користувача були використані засоби Telegram Bot API. Робота виконувалася у середовищі Visual Studio.

Результати, отримані в бакалаврській роботі, можуть бути використані для створення автоматизованої системи персонального планування.

Ключові слова: інтелектуальний бот, особистий графік, група/агрегатор.

Abstract

The bachelor's thesis consists of 89 A4 pages, which contain 51 figures, 3 tables, the list of sources used contains 20 titles.

In the bachelor's thesis the detailed analysis of methods of selection of messages for visualization of the personal schedule in the telegram-messenger is carried out.

A method of creating a personal schedule of the user by using a telegram bot, which will be personalized to select relevant information from connected groups, and form a daily schedule based on it.

Algorithms for creating a personalized graph, through the message tape that is formed, based on user interaction with the bot. The bot uses user-defined sources, from which it generates messages.

Developed a software product using C # programming languages. Telegram Bot API tools were used to develop the user interface. The work was performed in Visual Studio.

The results obtained in the bachelor's thesis can be used to create an automated system of personal planning.

Keywords: intelligent bot, personal schedule, group / aggregator.

ЗМІСТ

ВСТУП

Ефективний розподіл часу відіграє важливу роль у житті людини. З давніх-давен люди планували своє майбутнє, щоб покращити своє життя. Особливо це актуально сьогодні в наш час. Оскільки темп життя різко прискорився, це вимагає відповідної реакції. Ключовим тут є ефективне та швидке використання інформації та її аналіз для конструктивного планування.

Крім швидкості передачі, його прийом і зручність не важливі, оскільки це безпосередньо впливає на час обробки та ефективність.

Протягом усього життя людини були винайдені нові методи для покращення передачі та аналізу. Починаючи з перших спроб поширити за допомогою вогню та диму оригінальний алфавіт, телевізійне мовлення, комп'ютери та комп'ютерні технології, відбувався процес постійного вдосконалення.

Але тепер його умови вирішені. Найпоширенішим способом доставки сьогодні є Інтернет, який має багато виділених ресурсів. Однак ця зміна може легко заплутати користувачів і значно ускладнити пошук і обробку [1].

Це питання також особливо актуальне для епідеміологічної ситуації у світі і, на жаль, нинішньої війни. Все більше сфер життя людей звертаються до Інтернету, особливо важливість освіти, багато компаній, що працюють віддалено, і навіть медицина рухається в цьому напрямку. Надає ресурси, де ви можете отримати медичну консультацію від кваліфікованих фахівців, не виходячи з дому. Всі ці елементи сприяють ефективній автоматизації вашого дня, коли розклад можна коригувати індивідуально.

Не менш важливими були бойова особиста інформація та планування, які були особливо помітними під час воєн у країні [2].

Не можна забувати, що люди повинні розуміти себе, свою країну і світ. Навіть сакральне спілкування з близькими, що, безсумнівно, є однією з потреб сучасної людини, стало дуже важким.

Занадто багато вузькоспеціалізованих ресурсів лише погіршує ситуацію, оскільки користувачам доводиться використовувати сторонні ресурси на додаток до своїх улюблених програм або веб-сайтів, що призводить лише до незадоволеності та втрати часу. Це значно уповільнює процес планування власної роботи.

Тому розробка програмного забезпечення інтелектуального боту для ведення особистого графіку є актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконується згідно з планом наукових досліджень відділу програмного забезпечення

Мета та завдання дослідження. Метою бакалаврської роботи є підвищення ефективності та швидкості процесу пошуку та підвищення ефективності та економії часу шляхом створення персонального розкладу за допомогою спеціалізованих алгоритмів для вибору відповідного контенту для обраних користувачів у зручний час.

Основними завданнями роботи є:

- Інформаційне забезпечення аналітичних систем;
- Розробити методику підбору та створення персональних розкладів;
- розробляти системні моделі;
- здійснювати програмну реалізацію системи;
- Перевірте програму.

Предметом дослідження є процес розробки програмного забезпечення для реалізації єдиного графіка в Telegram Messenger.

Тема дослідження - Засоби реалізації програмних продуктів.

Методи дослідження. Методи дослідження, що використовуються в процесі дослідження:

- планування методів пошуку;
- Методи обміну повідомленнями для реалізації процесу взаємодії з ботами Telegram;

- методи моделювання UML для розробки моделі програмного продукту;
- Метод об'єктно-орієнтованого програмування для написання програм відбору контенту.

Наукова новизна отриманих результатів:

Удосконалено метод пошуку та вибору інформації, для створення персонального розкладу з урахуванням потреб конкретного користувача, шляхом визначення його персональної адреси, що підвищує надійність системи.

Практична цінність отриманих результатів. Досягнуті реальні результати у використанні алгоритмів і можливості розробки програмного забезпечення, здатного полегшити та прискорити пошук і обробку інформації користувачем для створення персональних графіків.

Особистий внесок здобувача. Усі наукові результати, представлені в бакалаврській дипломній роботі, отримані від самого автора, зокрема: алгоритми відбору інформації та візуалізації на їх основі особистого графіку у Telegram, модулі створення груп та розсилки щоденних розкладів.

Бакалаврська робота складається зі вступу, 4 розділів, висновку, списку використаних джерел в якості 20 літературних джерел та додатку, який включає технічне завдання, лістинг програми та ілюстративний матеріал до захисту.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз стану питання

За останні роки популярність Telegram, популярність різко зросла: у липні 2021 року, коли зареєструвалося аз квартал понад 400 мільйонів користувачів. Про це повідомляє американське видання Telegram. WhatsApp та Facebook зіткнувся з проблемами конфіденційності, коли користувачі обмінялися особистою інформацією, а Messenger було закрито. В результаті Telegram став найпопулярнішим додатком для смартфонів і планшетів і стрімко наздоганяє інші платформи соціальних мереж.

Потужний інструмент чату Telegram дозволяє користувачам вести групові бесіди та протести без лідера. Ви також можете додати будь-яку кількість каналів і повідомлень до папки. Крім того, ви можете організувати їхні чати в різні категорії. Таким чином ви можете організувати свій чат і насолоджуватися ним, не відволікаючись на кілька програм і вікон. У цьому посібнику ви дізнаєтеся про функції та переваги Telegram Messenger.

Конфіденційність особистих розмов має першорядне значення, а Telegram має багато функцій, які забезпечують безпеку ваших розмов. Ви можете встановити термін дії для всіх повідомлень та інших медіа. Ви також можете вибрати таймер для віддалених повідомлень. Нарешті, ви також можете синхронізувати свої дані, якщо хочете, а Telegram — це безкоштовна програма для обміну повідомленнями, яка ніколи не стягує з вас плату.

Безпека Telegram вже давно широко прийнята, але головна мета Telegram — надавати послуги безпечного обміну повідомленнями, а також підтримувати конфіденційність користувачів. Telegram ділиться інформацією про користувачів зі своєю материнською компанією та членами спільноти, але залишає за собою право надавати IP-адреси або телефони користувачів. цифри до правоохоронних органів

Завантаженість Telegram Messenger – не єдиний фактор, який впливає на вартість програми обміну повідомленнями. Попри величезну кількість

користувачів можна все одно не побачити рекламу в приватних групових чатах. Це зробить Telegram більш привабливим для рекламодавців і ЗМІ.

Тепер багато людей, які згадали Telegram, можуть відразу подумати про конфіденційність, але розробники відомі своїми функціями шифрування та конфіденційності. Крім того, на одному пристрої може бути до десяти облікових записів. Отже, якщо ви шукаєте нову програму обміну повідомленнями, перегляньте останню версію Telegram Messenger.

Ще однією перевагою Telegram Messenger є його безпека та зручність. І ми абсолютно безкоштовні, ми не продаємо ваші дані і не показуємо рекламу на нашому сайті. Він також не підтримується сторонніми рекламними платформами, а це означає, що ви можете використовувати його функції, не турбуючись про конфіденційність. Крім того, його легко налаштувати та використовувати, а кількість користувачів постійно збільшується. Зараз найкращий час для завантаження Telegram.

Кількість користувачів Telegram в Україні, наведених на рисунку 1.1, становить понад 4,7 мільйона, з них понад 55% – громадяни віком від 16 до 24 років, 24% – студенти і майже 43% – працівники. Усі вони навчаються або працюють у малих групах[1].

Примітно, що Telegram зараз мають високий рівень конфіденційності серед інших повідомлень. У налаштуваннях конфіденційності ви можете обробляти будь-кого, хто переглядає вашу інформацію, що міститься в Telegram, підключати вас до груп і зв’язуватися з вами.

Ви можете переглянути всю інформацію про себе, зробити так, щоб могли зв’язатися лише з вами чи з кимось іншим. Рекомендації, та додавання контактів. Перше Telegram вікно показано на рисунку 1.1.

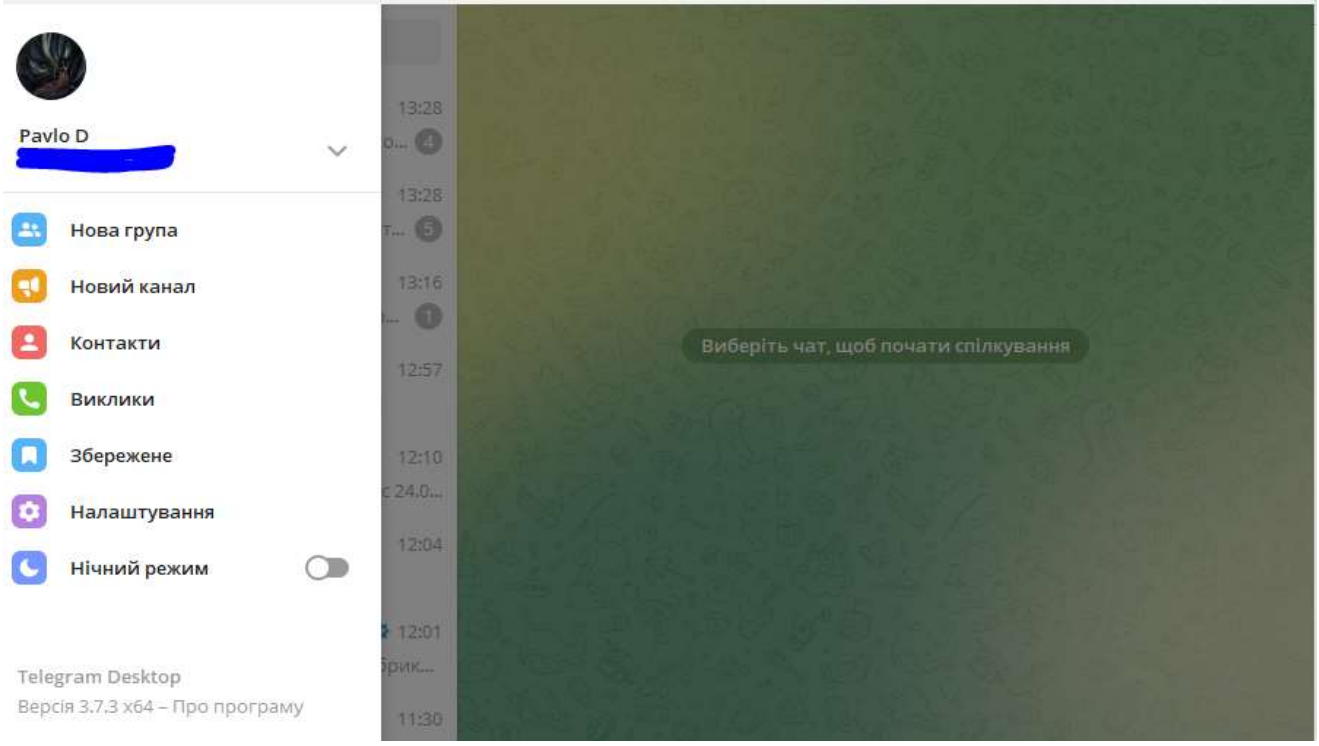

Рисунок 1.1 – Telegram

Існує кілька варіантів відправки повідомлень маленьким групам користувачів, кожен з яких має свої плюси і мінуси.

Один із них — створити групу, де кожен зможе щось написати, показано на рисунку 1.2. Цей варіант має переваги і недоліки. Незважаючи на простоту використання, великим недоліком є те, що часто важлива інформація відсутня від адміністраторів, в повідомленнях інших користувачів.

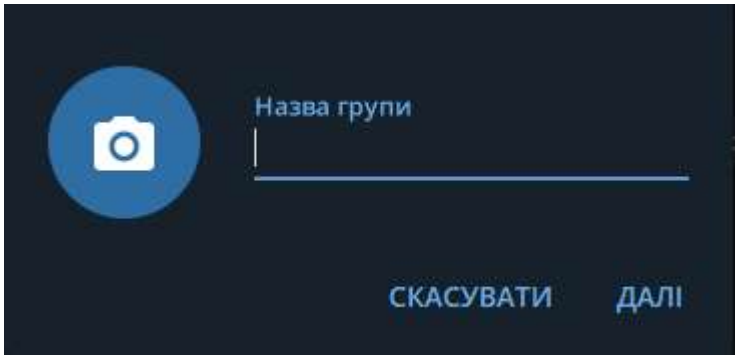

Рисунок 1.2 – Процес створення нової групи

Інший варіант — створити канал (рисунок 1.3), де ви можете написати щось лише в якості автора, це спосіб який дозволить вам легко знайти інформацію на будь-яку тему, але змусить вас увійти, використовуючи кілька каналів та імен користувачів. Він їх завжди перевіряє.

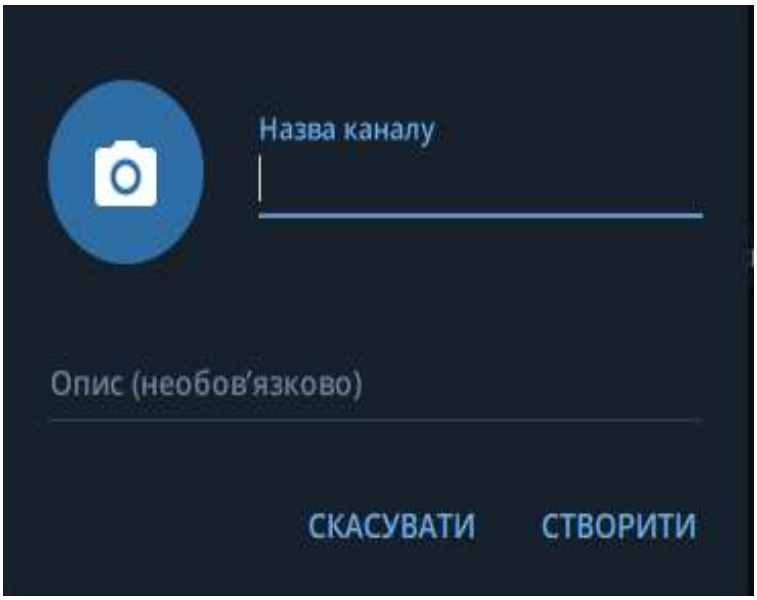

Рисунок 1.3 – Процес створення нового каналу

Тому телеграм-боти стають все більш популярними. Це програми, які використовують API Telegram для збору, передачі та обробки , довіреної клієнту. Як правило це профілі користувачів, які керуються алгоритмами. Ці профілі (оснащені базовими формами штучного інтелекту) здатні виконувати ряд завдань і призначені для різних дій.

Є боти, які можуть розважати та взаємодіяти, а є інші, які досліджують конкретні теми в Інтернеті. Деякі боти можуть витягувати всі види файлів (музику, книги, фільми) з Інтернету, а інші займаються торгівлею або дослідженням ринку.

Детальніше бот – це програма, яка містить алгоритми штучного інтелекту, які дозволяють йому взаємодіяти з користувачем. Боти в Telegram можуть

автономно взаємодіяти з користувачами і діяти як реальні користувачі. Насправді у них є персональні дані, але за ними стоять не реальні люди, а алгоритми.

Основною функцією робота є надання послуги користувачеві, і кожен робот запрограмований на виконання певного завдання. Насправді є боти, які можуть шукати пісні чи відео в Інтернеті, боти, які повідомляють користувачів, коли певний продукт доступний у певному магазині, і боти, які можуть виконувати прості замовлення.

Деякі боти створені Telegram, хоча більшість створюються та керуються за межами. Бот не побачить ваш номер телефону (якщо ви не згодні), але все одно рекомендується уникати обміну конфіденційними даними з ботами. Крім того, зверніть увагу на файли, які вони вам надають, і переконайтеся, що боти захищені, перш ніж відкривати їх.

Якщо ви хочете спілкуватися з роботом, вам спочатку потрібно його знайти. Щоб знайти бота, ви можете скористатися рядком пошуку у верхньому правому куті (для звичайних користувачів) і ввести ім’я бота, якого ви шукаєте.

Знайшовши бота, просто натисніть його, щоб почати взаємодію зі своїм звичайним обліковим записом.

У Telegram немає спеціальних розділів для ботів, потрібно шукати в Інтернеті або запитувати поради у інших користувачів, щоб їх знайти.

Якщо бот не відповідає вашим очікуванням, ви також можете скористатися блокуванням, натиснувши піктограму з трьома крапками у верхньому правому куті, а потім клацнувши заблокувати. На цьому етапі натисніть OK, щоб завершити завдання.

У деяких випадках Telegram дозволяє блокувати ботів прямо зі свого профілю за допомогою кнопки «Стоп» (програма дозволяє деактивувати та повторно активувати, якщо потрібно).

Telegram також дозволяє додавати ботів до груп, але в цьому випадку я рекомендую повідомити всіх учасників групи про існування бота, щоб уникнути непорозумінь або неприємних незручностей.

Серед переваг ботів – їх гнучкість для користувача, оскільки всі їхні внутрішні компоненти, які відповідають за обробку , тобто за логікою реалізації, повністю підконтрольні творцю бота.

Також менш важливою є конфіденційність , яка буде вищою в цій техніці, оскільки вся обробка є відповідальністю розробника, а не програми та бота.

Не дивно, що кількість ботів з кожним роком збільшується. Серед них ви знайдете як вузькоспеціалізовані, так і широкий спектр функцій.

В результаті стає складно зрозуміти, які з них насправді відповідають вимогам замовника, а які є просто марною тратою часу.

1.2 Порівняльний аналіз аналогів

Список телеграм-ботів для поширення інформації досить великий. Але багато з них дуже складні у використанні, містять масу невідомих параметрів і функцій, які просто лякають користувача. Найпопулярніші з них:

- Gmail Bot;
- Ukrposhta chatbot;
- Tgrec bot;
- Russian war tribunal bot.

Створений Google, бот Telegram, також відомим як бот Gmail, показаний на рис. 1.4. Він спрямований на швидку взаємодію з Gmail.com, Gmail — це безкоштовна служба електронної пошти, що надається Google. Надає доступ до поштових скриньок через веб-інтерфейс.

У жовтні 2012 року Gmail став найпопулярнішим сервісом у світі, випередивши Microsoft Hotmail за кількістю унікальних користувачів. Gmail має понад 420 мільйонів користувачів. особливо на можливість швидкого перегляду електронної пошти та керування нею. На жаль, як видно з опису, він вимагає реєстрації на сторонньому веб-сайті і працює лише з інформацією поштової скриньки.

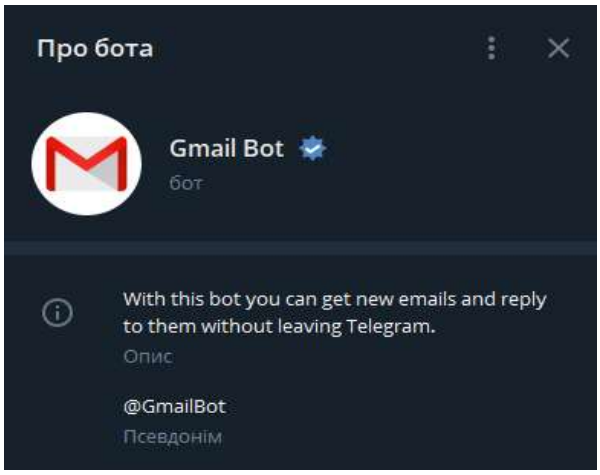

Рисунок 1.4 – Gmail Bot

Ще одним телеграм-ботом для відправки повідомлень є чат-бот Укрпошти, показаний на рисунку 1.5.

Має власний сайт, де міститься вся інформація про розробника, механізм передачі даних та тарифи.

Функції чат-бота включають відстеження та відстеження номерів, сповіщення про зміни статусу посилки, пошук відділення та його геолокацію за індексом, а також ознайомлення користувачів з основними тарифами компанії.

Надалі функціонал чат-бота буде розширено відповідно до потреб користувача. Зокрема, будуть додані мовні модулі та тарифний калькулятор, щоб бот міг розрахувати вартість відправлення клієнта[5].

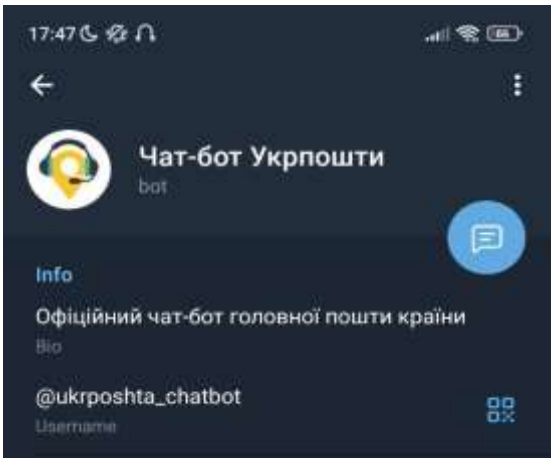

Рисунок 1.5 – Ukrposhta chatbot

Існує також популярний сервіс масового чату Telegram Tgrec bot, показаний на малюнку 1.6.

Дозволяє користувачеві надсилати повідомлення з готового списку або з власного файлу. Також є можливість дублювати повідомлення, тобто надсилати повідомлення в чат через певний проміжок часу. До повідомлення можна додати посилання, зображення або відеовміст [6].

До переваг можна віднести простоту експлуатації, недоліки повної оплати ліцензії на використання і, відповідно, повідомлень.

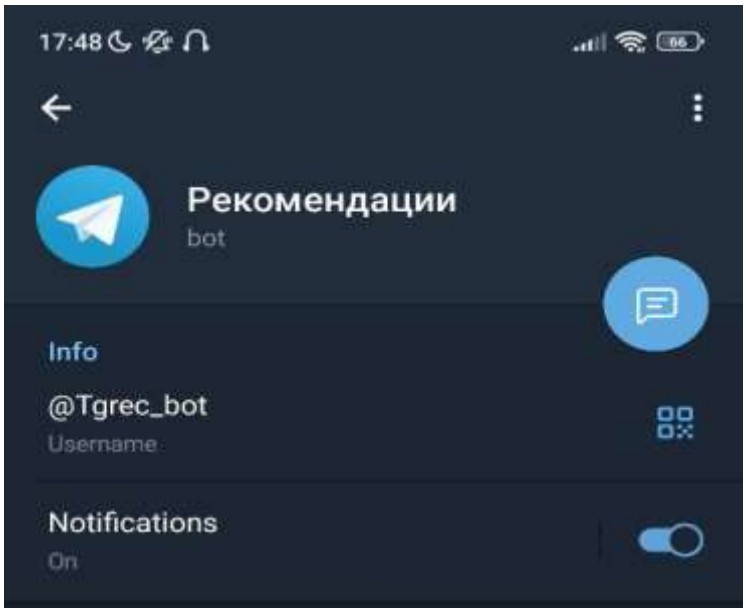

Рисунок 1.6 – Tgrec bot

Серед аналогів слід також назвати бота російського військового трибуналу, показаного на рисунку 1.7.

Це сервіс електронної пошти зворотного зв’язку, а також функція співпраці з месенджерами, зокрема, Telegram [7].

Важливою перевагою є збір повної статистики. Він відіграє важливу роль у війні, оскільки дозволяє анонімно повідомляти про злочини, скоєні окупантом. З перших же днів вторгнення ми отримали тисячі повідомлень про позиції та стратегії противника. І завдяки цьому наші військові завдали великих втрат

російським окупантам. Ви можете надіслати бота про розташування російської техніки, солдатів та небезпечних об’єктів в Україні. Це легко зробити без Інтернету.

Потрібно розуміти що цей бот є частиною сімї інших ботів, запущених українською IT армією.

Для ознайомлення навиду список який складав власноруч нижче, з дописом рекомендації дій в теперішній воєнній ситуації:

- Щоб повідомити про переміщення російських військ або диверсантів, скористайтеся цим ботом: @stop_russian_war_bot;
- Щоб повідомити про чуток і шпигунів, які розкрили місцезнаходження української армії, напишіть, будь ласка, цьому боту: @stopdrugsbot;
- Щоб Повідомити про пограбування та випадки НВН: @ukraine_avanger_bot;
- Щоб боротися з дезінформацією, зв’яжіться з цим ботом: @stoprussiachannel

Підсумовуючи до недоліків – складність роботи з ботом.

Рисунок 1.7 – Russian war tribunal bot

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	Gmail Bot	Ukrposhta chatbot	Tgrec bot	Russian war tribunal bot	Власний додаток
Зручність інтерфейсу	+	-	+	-	+
Відсутність необхідності реєстрації в інших сервісах	-	-	-	+	+
Безкоштовна розсилка повідомлень	+	+	-	+	+
Збір статистики	+	+	-	-	-
Можливість надсилати посилання, фото, відео	-	-	+	-	+
Сума	**3**	**2**	**3**	**3**	**4**

Таблиця порівняння характеристик показала, що створення нового бота Telegram є логічним рішенням. В результаті виходить продукт, який перекриває недоліки сучасних рішень і забезпечує кращу ефективність.

1.3 Аналіз методів підбору і формування особистого графіку

Щоб вирішити поставлену задачу, потрібно побудувати систему, яка буде включати робочу частину сервера, інтерфейсу та бази даних.

Перший крок – впровадження механізму створення груп і підписки на нових користувачів. Дані, введені користувачем, будуть відправлені на сторону сервера і записані в базу даних.

Наступний крок – запровадження механізму відбору важливої. Адміністратор групи надішле повідомлення боту. На сервер надсилається лише унікальний код повідомлення, що забезпечить конфіденційність та продуктивність системи.

Також треба розробити код що буде перевіряти інформацію на адекватність.

Наступним кроком є впровадження системи розсилки кінцевому користувачеві та розробка персонального графіка. З сервера буде надіслано лише код повідомлення, який буде перетворено на фактичні інформацію за допомогою методів телеграм.

1.4 Висновки

У першому розділі було розглянуто стан поширення в Telegram Messenger.

Розроблений додаток оцінює та порівнює поточні бот інструменти, такі як Gmail Bot, чат-бот Укрпошти, бот Tgrec і бот російського військового трибуналу.

Основним недоліком чат-бота Укрпошти, бота Tgrec і бота російського військового трибуналу, є необхідність витрачати багато часу на відправку кожного повідомлення. Для використання бота Gmail необхідно зареєструватися на сторонньому ресурсі.

Порівняння доводить ефективність розробки власного програмного продукту, який перекриває всі недоліки існуючих рішень.

Проведено аналіз існуючих підходів до вирішення поставленого завдання. В результаті аналізу були обрані методи виконання необхідної функції.

Виведено основні завдання, які необхідно виконати для створення програмного продукту.

2 РОЗРОБКА АЛГОРИТМІВ ТА МОДЕЛІ СИСТЕМИ

2.1 Принципи функціонування ботів

Будь-який бот, як і звичайний додаток працює з даними, отримує їх, обробляє і повертає клієнту. Тому можна розповсюджувати на роботу з ботом принципи звичайної роботи над програмними додатками. Далі в розділі йтиметься про функціонування ботів, в якості додатка. Даними у розробленому додатку є інформація у вигляді повідомлень, яка використовується при створенні графіку. Тобто він керує повідомленнями та вбудованими посиланнями, фотографіями та відео.

Оновлення розміщуються в групах, створених користувачами. Коли ви створюєте групу, користувач стає її адміністратором і може не тільки переглядати інформацію, але й надсилати їх.

Інші користувачі можуть підписатися, використовуючи унікальний код групи та пароль, надані адміністратором. Це створить джерело інформації, на яке користувач підпишеться.

Для забезпечення механізмів аутентифікації програма зберігає інформацію про користувача разом з його унікальним кодом.

Взаємодія з програмою здійснюється за допомогою команд. Приклад використання команд наведено на рисунку 2.1. Після введення певної команди інформація про сповіщення обробляється певним чином залежно від того, яку команду введено.

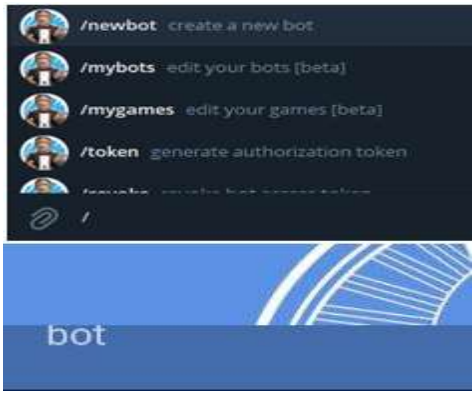

Рисунок 2.1 - Приклад використання команд Telegram боту

Повідомлення містить інформацію, додану абонентом або адміністратором групи. Залежно від команди, для правильної роботи програма може зберігати унікальний код повідомлення, що містить інформацію, код групи, до якої ви хочете надіслати, створити або підписатися на повідомлення, код користувача. Потрібно розуміти основні принципи взаємодії користувачів з додатками на телефоні.

Коли користувачі торкаються спільної цілі, пов'язаної з певною дією, вони можуть переглянути та відредагувати вміст перед його використанням. Це особливо важливо для текстових даних.

Клацнувши на будь-якій цілі прямого спільного доступу, користувач повинен перейти до інтерфейсу, де можна виконувати дії безпосередньо з цілью. Уникайте показу користувачеві екрана підтвердження або надсилання користувача до інтерфейсу, не пов'язаного з вибраним пунктом призначення.

Зокрема, не надсилайте користувача на екран підтвердження контакту, щоб підтвердити або повторно вибрати контакт для спільного доступу, оскільки користувач уже зробив це, торкнувшись цілі в таблиці спільного доступу Android. Клавіатура має бути видимою, а повідомлення має бути попередньо заповнене загальною інформацією.

Таким чином, введення різних команд відрізняється, але завжди містить інформацію про те, яку команду використовує користувач, і код користувача. Ви також можете використовувати групові коди та паролі під час створення та підписки, коди повідомлень та коди груп під час надсилання .

Дані, введені користувачем, переміщуються з інтерфейсу в бекенд. Вони використовують HTTP, Web Socket та інші протоколи даних для обміну інформацією між собою. Найпопулярнішими форматами передачі даних в Інтернеті є JSON і XML.

JSON – це поширений формат для представлення значень та об'єктів. Його опис задокументовано в RFC 4627. Спочатку він був створений для JavaScript, але багато інших мов також мають бібліотеки, які можуть працювати з ним. Таким

чином, JSON легко використовується для обміну даними, коли клієнт використовує JavaScript, а сервер написаний на .Net / Java або іншій мові [9].

XML означає розширену мову. XML — це мова розмітки, яка визначає набір правил для кодування документів у форматі людино-машинного читання [10]. Порівняння JSON і XML. Можна сказати, що читабельність JSON і XML порівнянна, з одного боку, синтаксис простий і форма мітки стандартизована.

XML в основному розширюється, як і JSON, звичайно. Однак JSON конкурує з власним JavaScript і може зберігати складені об'єкти JavaScript, що має безпрецедентні переваги перед xml.

XML має багаті інструменти кодування, такі як Dom4j і JDom, а JSON також надає інструменти. Я думаю, що досвідчені розробники без інструментів можуть швидко написати правильні документи XML і рядки JSON, але документи XML вимагають більш структурованих символів.

Існує два способи розбору XML: Один з них аналізує модель документа, тобто індексує набір тегів за їхніми батьківськими тегами. Наприклад: xmlData.getElementsByTagName ("tagName"), але його слід використовувати, коли структура документа вже відома і звичайна інкапсуляція не може бути виконана

Інший підхід — пошук вузлів (документів і дітей). Це можна зробити ітеративно, але аналізовані дані все одно відрізняються і часто не відповідають передумовам.

Аналіз будь-яких таких масштабованих структурованих даних може бути складним. Те ж саме стосується JSON. Якщо ви вже знаєте структуру JSON, просто використовуйте JSON для передачі даних, ви можете написати дуже функціональний, красивий і читабельний код. Якщо ви розробник зовнішнього інтерфейсу, вам сподобається JSON. Але якщо ви розробник додатків, вам це може не сподобатися, оскільки xml — це дійсно структурована мова розмітки.

Розбирання JSON, не знаючи його структури, є кошмаром. Не кажучи вже про складність і час, код буде багатослівним, а результати не будуть задовільними. Однак це не впливає на вибір JSON для багатьох розробників інтерфейсу. Оскільки toJSONString () у json.js дозволяє побачити структуру рядка

JSON. Звичайно, без використання цього дроту це все одно кошмар. З огляду на цей рядок, люди, які використовують JSON, зазвичай мають чітке уявлення про структуру JSON, і їм легко маніпулювати.

Зрештою, основним полем у сфері JavaScript є JSON, і його переваги, звичайно, набагато перевершують переваги xml. Якщо JSON зберігається в складеному об’єкті JavaScript, я не знаю його структури, і я думаю, що багато програмістів будуть плакати й аналізувати JSON.

Для порівняння було обрано формат JSON для обміну інформацією між сервером і клієнтом. Тому що це зрозуміліше, простіше і краще.

2.2 Розробка моделі роботи системи

Кожен користувач при реєстрації в Telegram отримує унікальний код облікового запису, який являє собою набір чисел і його унікальний ідентифікатор [11].

Оскільки він унікальний для кожного користувача і не може бути скопійований, було вирішено використовувати його для ідентифікації користувача.

До його переваг можна віднести конфіденційність, оскільки навіть якщо користувач знає код, отримати іншу інформацію про нього неможливо.

При створенні групи, крім її імені користувача, пароля та пароля, буде збережений код користувача, який буде її адміністратором. Це дозволяє йому не тільки переглядати інформацію з цієї групи, а й надсилати її.

Для інших користувачів запроваджено систему групової підписки. Коли ви отримуєте груповий код і пароль за допомогою команди від адміністратора, користувач вводить ці дані в систему і створює блок підписки, який містить код користувача, якому він призначений, код групи, на який вам потрібно підписатися, і власний код. Його визнання.

Пошук і вибір здійснюється на основі внутрішніх підписок користувачів.

При створенні особистого розкладу система перевіряє список підписок для групи, в якій була розміщена важлива інформація, і надсилає його до вашого розкладу, використовуючи збережені коди абонентів як сповіщення.

Таким чином, при спілкуванні з ботом Telegram створюється щоденний графік, що містить важливу інформацію, яка являє собою постійно обновлюваний набір інформації з груп агрегаторів, на які ви є абонентом.

Модель відбору та візуалізації інформації складається з послідовності дій:

1. Авторизація користувача.
2. Вибір теми розкладу.
3. Підписка на групу/агрегатом.
4. Запуск завдання телеграм-бота на формування персонального розкладу, на основі поданої у групі інформації .

Метод пошуку та вибору інформації для створення особистого графіку зображено на рисунку 2.2

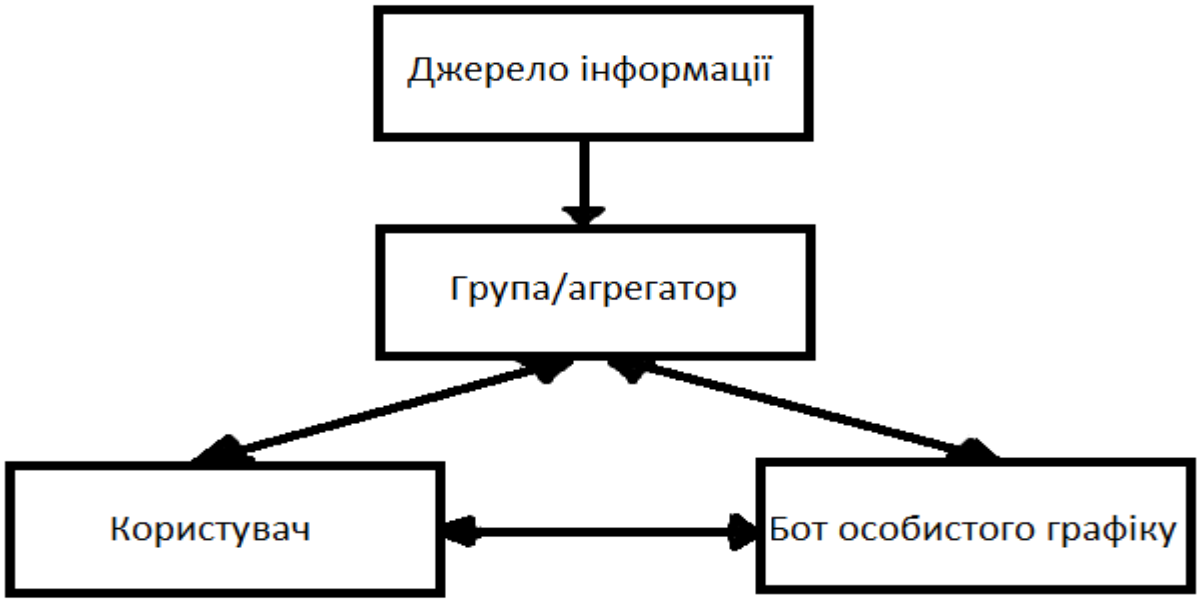

Рисунок 2.2 – Метод пошуку та вибору інформації для створення особистого графіку

2.3 Розробка моделі системи підбору й візуалізації особистого графіку

Програму можна розділити на кілька модулів: Модуль обробки даних, Модуль керування, Модуль зберігання даних та Модуль персонального графічного надсилання [12].

Після обробки інформація надходить в модуль управління, який відповідає за подальшу передачу даних у структуру програми, визначаючи, куди вони будуть відправлені. Ця модель є класичною і її можна побачити в більшості сучасних ботів.

Загальна модель додатку зображена на рисунку 2.2

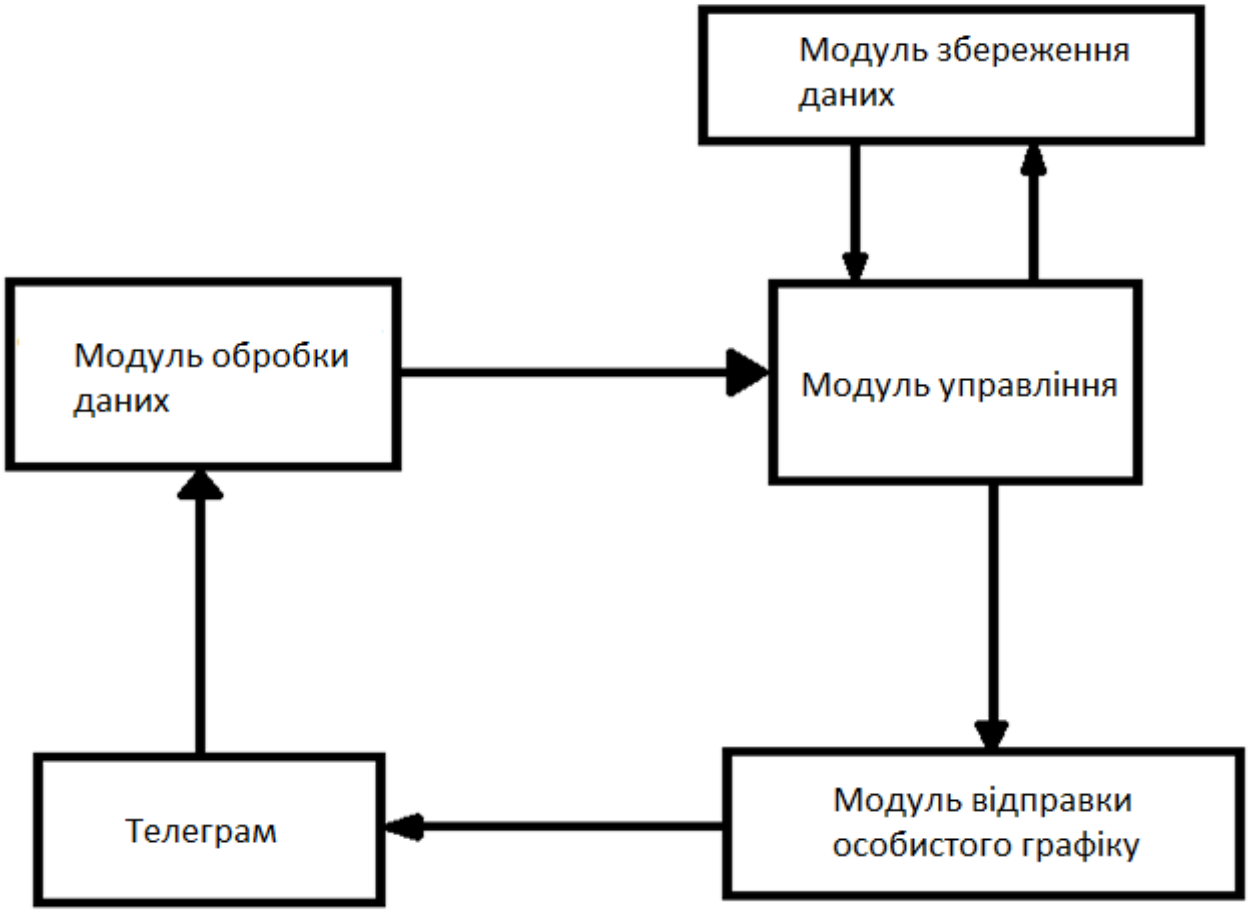

Рисунок 2.2 - Загальна модель додатку

При необхідності деталь зберігається завдяки модулю зберігання. Для коректної роботи програми потрібні деякі дані, особливо в цьому модулі є записи про нові групи та підписки [13].

Потім оброблені дані включаються в персональний модуль передачі розкладу, який відповідає за передачу інформації користувачам Telegram.

2.4 Розробка інтерфейсу користувача

Для ефективного використання програмного продукту в складі клієнта був обраний Telegram Messenger. Структура інтерфейсу програми показана на рис. 2.3.

У центрі екрана блок з повідомленнями. Саме тут відбувається основна взаємодія користувача з програмою. Ви можете надсилати текст, посилання, фотографії та відео.

Контактний блок розташований ліворуч. Доступний фільтр з різними варіантами моніторингу.

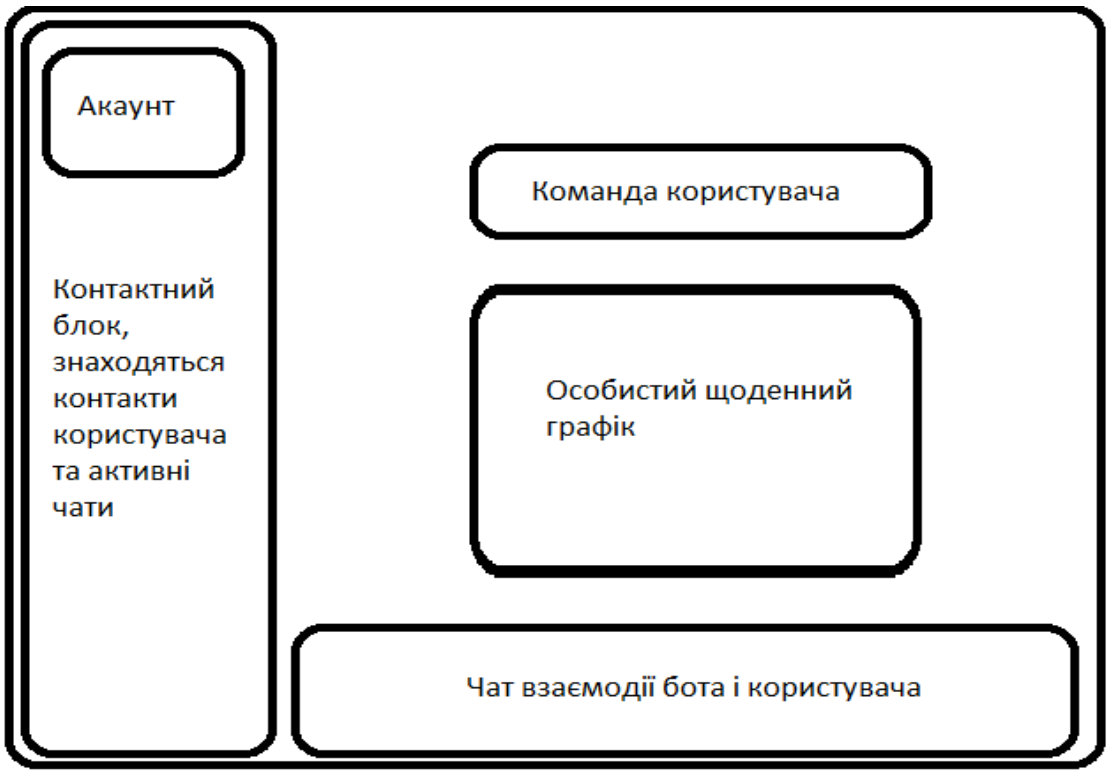

Рисунок 2.3 – Структура інтерфейсу додатку

2.5 Розробка ER-моделі бази даних для додатку

Модель ER, повна назва моделі сутність-відношення, моделі сутності-відношення або діаграми сутність-відношення (ERD), винайдена китайсько-американським вченим-комп'ютерником Ченом Піншаном, є моделлю даних або шаблоном схеми даних для концептуального високого рівня. модель даних.

Моделі ER часто використовуються при проектуванні інформаційних систем, наприклад, вони використовуються на етапі проектування концептуальної основи для опису інформаційних потреб та/або типів інформації, які будуть зберігатися в базі даних, але методи моделювання даних використовують будь-яку онтологію, які описують конкретну предметну область. Пізніше у випадку проектування інформаційної системи на основі бази даних концептуальні моделі відображаються на логічні моделі, такі як реляційні моделі, які, у свою чергу, відображаються на фізичні моделі під час фізичного проектування. Зауважте, що іноді ці дві разом фази називаються «фізичною конфігурацією».

Існують певні угоди щодо діаграм зв’язків між об’єктами (ERD). Решта цієї статті зосереджена на концептуальному моделюванні та описує класичні концепції. Найбільш часто використовуваними поняттями в логічних і фізичних базах даних є інформаційна інженерія, IDEF1x (мова ICAM DEFinition) і просторове моделювання. Модель діаграми ER складається з сутностей, атрибутів і зв’язків. Сутності — це користувачі даних, які представляють реальні речі, які об’єктивно існують у програмній системі, наприклад, люди, тварини, об’єкти, списки, відділи, проекти тощо.

Суб’єктами одного типу є багато сутностей. Інтенсивність об'єктів представлена типами об'єктів. Тип сутності — це визначення сутності в групі сутностей. Усі характеристики об’єкта називаються якостями. Наприклад, у користувача є ім’я, стать, адреса, номер телефону тощо. Ідентифікатор об’єкта — це ідентифікатор об’єкта, який може однозначно представляти ознаки об’єкта та набори ознак.

Для ідентифікації об’єкта можна використовувати лише один «ідентифікатор об’єкта». Ідентифікатор об'єкта також є первинним ключем об'єкта. На діаграмі ER функції, що відповідають об’єктам, представлені точками, а підкреслені імена – це те, що ми називаємо ідентифікаторами. У світі, в якому ми живемо, організми не існують самі по собі, вони нерозривно пов’язані з іншими організмами.

Наприклад, людина працює у відділі компанії, суб’єктами є «хтось» і «відділ компанії», і між ними існує багато відносин. Компонентами моделі ER є набір сутностей, набір ознак і зв’язків.

1. Група об'єктів представлена прямокутним полем, а ім'я сутності записується в прямокутному полі.

2. Атрибут сутності представлений еліпсом, ім'я атрибута записується в полі, а група сутностей з'єднується неголовним ребром.

3. Відношення між сутностями зображується ромбом, об'єднання називається відповідним значенням, ім'я записується ромбом, прямокутник сутностей, що беруть участь у відношенні, пов'язаний з ромбом скалярного зв'язку, а комбінація дріт 1, 1-N або M-N.

Підводячи підсумок, модель сутності-відносин — це модель даних, яка дозволяє описати роботу програми.

Концептуальні схеми з використанням загальних блокових проектів. Основними конструкціями цієї моделі є сутність і зв'язки [15].

По суті, ми розуміємо основний зміст явища, процесу або об’єкта, про який збирається інформація в базі даних. Суть визначається як те, що турбує клієнта. Атрибут (або набір атрибутів), що використовується для визначення одиниці, називається ключем одиниці.

Блоки можуть бути з'єднані за допомогою різних рівнів зв'язку. Наприклад, співвідношення 1:1 означає, що копія одного блоку може бути пов’язана не більше ніж з однією копією іншого блоку. Отже, рівень зв’язку 1:n означає, що екземпляр одиниці в деяких випадках може бути пов’язаний з іншим об’єктом, але не навпаки [16].

Модель ER зазвичай реалізується у вигляді баз даних. У випадку реляційної бази даних, яка зберігає дані в таблицях, кожен рядок у кожній таблиці є прикладом одиниці. Деякі поля даних у цій таблиці вказують на індекси в інших таблицях. Такі поля є індикаторами фізичної реалізації зв'язків між підрозділами.

Модель ER, показана в 2.4, також призначена для використання.

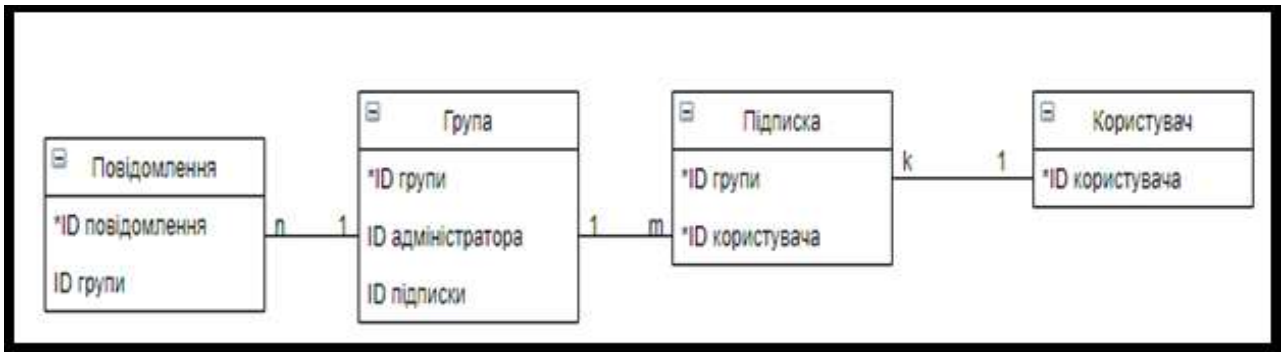

Рисунок 2.4 – ER-модель

2.6 Висновки

Другий розділ аналізує дані та вибирає між серверним і клієнтським форматами обміну даними. Розроблено програмний інтерфейс, методологію та модель системи відбору та візуалізації інформації. Для відділень розроблено модель швидкої допомоги. Де буде працювати додаток..

3 РОЗРОБКА І ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ СТВОРЕННЯ ОСОБИСТОГО ГРАФІКУ

3.1 Варіантний аналіз і обгрунтування вибору засобів реалізації програми

Поки що ви можете взаємодіяти з Telegram Messenger двома основними способами. Перший – це ручне програмування. Інший варіант — використовувати API бота Telegram.

Telegram Bot API — це інтерфейс на основі HTTP, розроблений для розробників, які хочуть створювати ботів для Telegram [17]. Це спрощує взаємодію з вебхуками, тегами повідомлень тощо. Це означає, що користувачам не потрібно виконувати функції шифрування та взаємодії на стороні сервера Telegram.

Цей варіант був обраний тому, що розробка програм за допомогою Telegram Bot API є відносно швидким і простим процесом.

На стороні сервера можна використовувати ASP.NET, Java Spring, Python.

ASP.NET Core — це безкоштовна кросплатформна платформа для створення веб-додатків з відкритим кодом. Створена Microsoft у партнерстві з громадськістю, платформа пропонує вищу продуктивність порівняно з ASP.NET. Він має модульну структуру і сумісний з такими операційними системами, як Windows, Linux і macOS.

ASP.NET Blazor вимагає використання C#, а не JavaScript для створення інтерактивних веб-інтерфейсів. Blazor дає вам справжню річ. NET працює у браузері WebAssembly.

Хоча це нова платформа, побудована на новому веб-каталозі, вона має високий ступінь концептуальної сумісності з ASP.NET. Програми ASP.NET Core підтримують паралельне керування версіями, коли різні програми, запущені на одному комп’ютері, можуть отримати доступ до різних версій ASP.NET Core. Це було неможливо в попередніх версіях ASP.NET [18].

Він також добре інтегрується з середовищем розробки Visual Studio на малюнку 3.1.

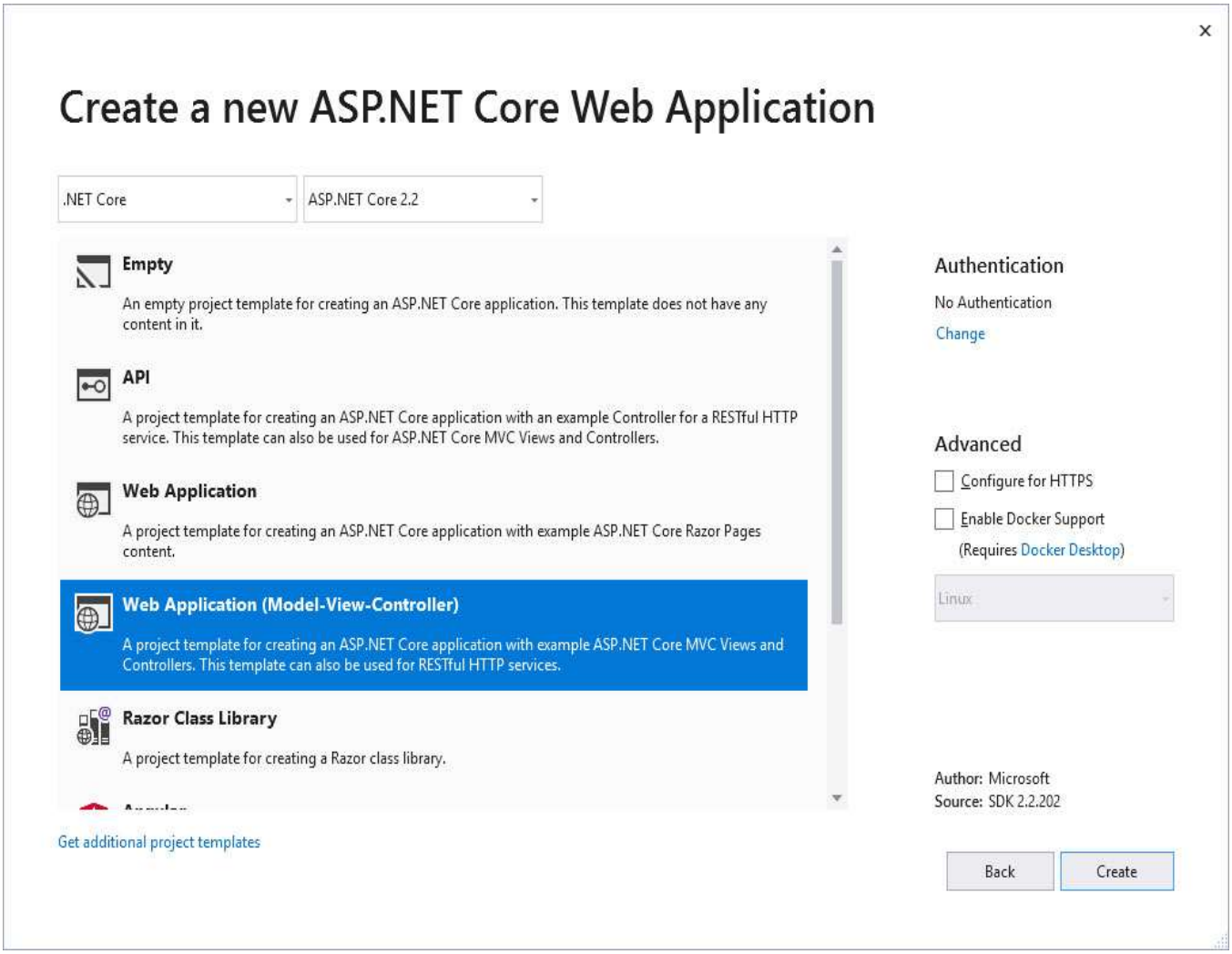

Рисунок 3.1 – ASP.NET в Visual Studio

Spring Framework — це платформа Java з відкритим вихідним кодом. Ви можете використовувати основні функції Spring Framework для створення будь-якої програми Java, але існують розширення для створення веб-програм на платформі Java EE. Платформа Spring розроблена для спрощення використання розробки J2EE та просування ефективних методів програмування з використанням моделі програмування на основі POJO.

Зауважте, що Spring Framework неодноразово критикували за надмірне прикріплення XML до контейнерів Spring. Однак, починаючи з версії 3.0.0, розробники мають можливість використовувати всі або частину анотацій у своїх програмах. Spring Boot широко використовує цей метод власних конфігурацій.

Крім того, пакет Spring Tool Suite (STS) на основі Eclipse забезпечує автоматичне заповнення коду, перевірку, контекстну інформацію та графічну візуалізацію під час редагування файлів конфігурації Spring XML.

Таблиця 3.1 – Порівняння технологій розробки

	ASP.NET Core	Java Spring	PHP
Простота конфігурації	+	-	+
Безпека	+	+	-
Продуктивність	+	+	-
Чітка структура коду	+	+	-
Простота використання	+	-	+
Сума	5	3	2

Результати аналізу привели нас до висновку, що технологія ASP.Net Core найкраще підходить для побудови серверної частини програми. Основними перевагами є продуктивність, безпека та чітка структура коду.

3.2 Аналіз середовища розробки

Було вибрано Telegram Bot API і ASP.NET Core для розробки програми. Розглянемо середовище розробки Microsoft Visual Studio, показане на малюнку 3.2. Це офіційне середовище розробки для платформи Microsoft .NET.

Microsoft Visual Studio — це сімейство продуктів Microsoft, яке включає інтегроване середовище розробки програмного забезпечення та кілька інших інструментів. Ці продукти дозволяють створювати консольні програми та ігри та програми за допомогою графічного інтерфейсу, включаючи технічну підтримку та веб-сервіси для веб-сайтів, веб-додатки, рідний код та керований код для всіх платформ.

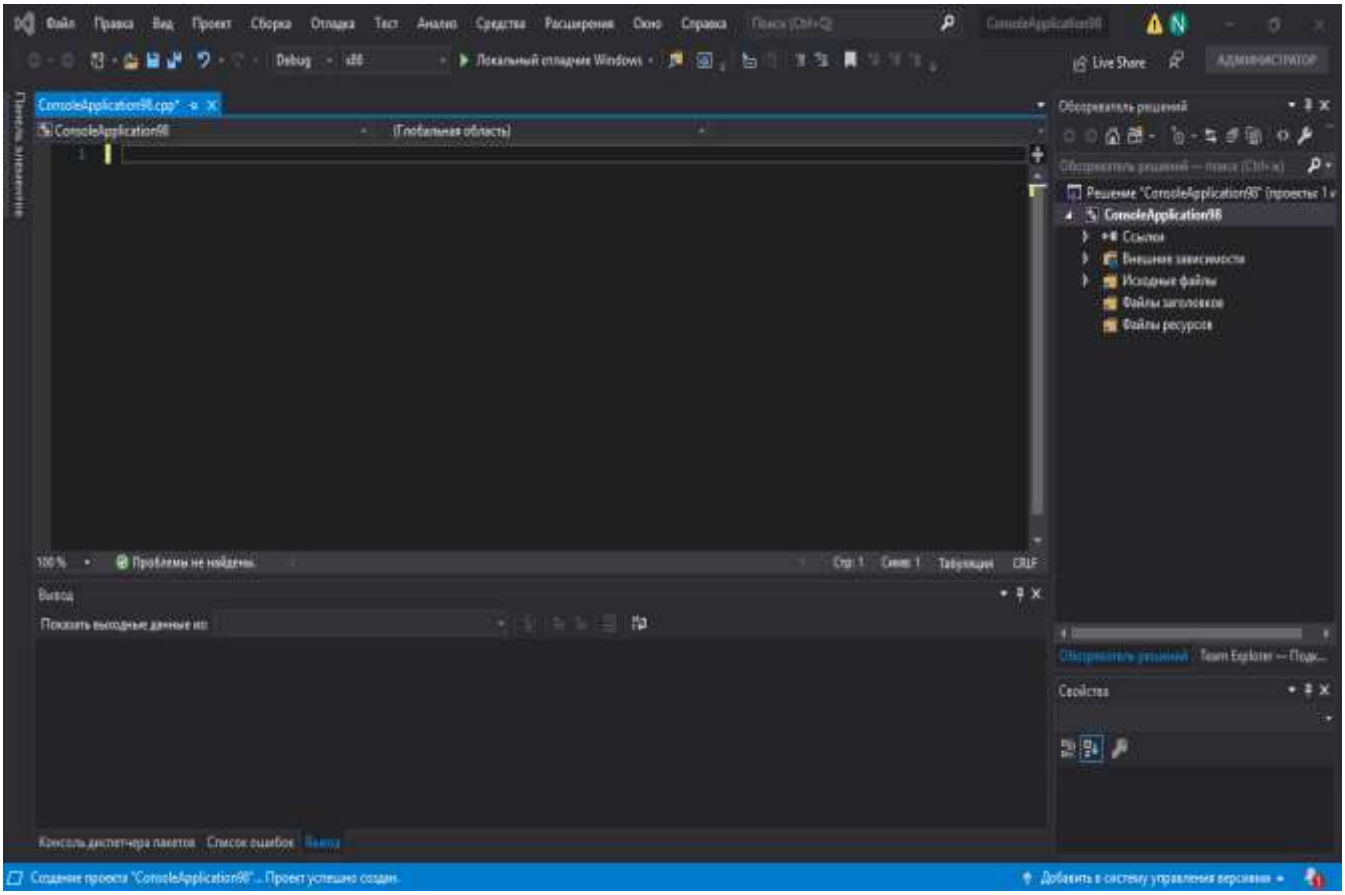

Рисунок 3.2 – Microsoft Visual Studio

Visual Studio містить редактор вихідного коду, який підтримує технологію IntelliSense і дозволяє легко реорганізувати код. Вбудований налагоджувач може запускати як налагоджувачі вихідного коду, так і налагоджувачі машинного рівня. Інші вбудовані інструменти включають редактор форм, веб-редактор, конструктор класів і конструктор схем бази даних для спрощення графічного інтерфейсу користувача програми. Visual Studio дозволяє створювати та підключати сторонні програми (плагіни), щоб розширити функціональність практично на будь-якому рівні.

Visual Studio також дозволяє ефективно використовувати API бота Telegram.

3.3 Розробка програми підбору і візуалізація особистого графіку

Згідно від завдання розробляється інтерфейс користувача з додатком, реалізований за допомогою команд робота (рисунок 3.3). Кількість команд

мінімальна, щоб не перевантажувати користувача, а їх назви відповідають функціям.

Усі команди доступні відразу після запуску програми.

Доступ до усіх команд є одразу після старту роботи з додатком.

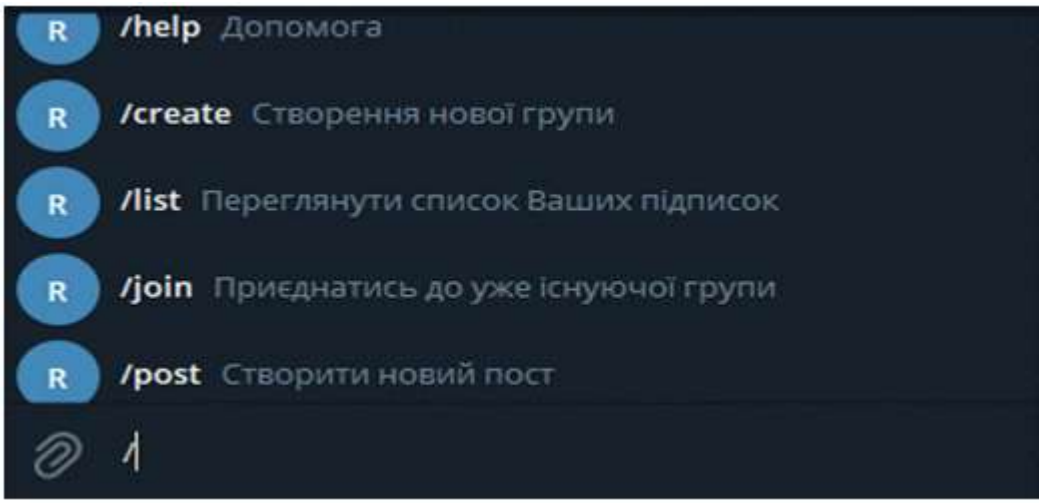

Рисунок 3.3 - Інтерфейс взаємодії користувача

Щоб програма працювала, було створено ряд класів, включаючи User, Message, Group і Subscribe, як показано на малюнку 3.4.

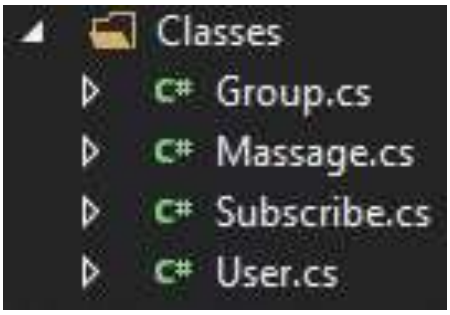

Рисунок 3.4 – Розроблені класи

Розглянемо клас User. У нього 1 поле та один метод, що зображено на рисунку 3.5.

```
public class User
{
    public int Id;

    public Massage ShowAllSubscribe()...
```

Рисунок 3.5 – Клас User

Поле ID — це унікальний номер користувача, який присвоюється при реєстрації в Telegram. Додаток використовується для ідентифікації користувачів, створення підписів, керування групами тощо.

Метод ShowAllSubscribes() викликається користувачем за допомогою спеціальної команди і дозволяє переглянути список усіх підписок.

Єдиним вхідним є ідентифікатор користувача, який здійснює пошук у базі даних. Вихідними даними є об’єкт класу Massage, який передається в модуль обміну повідомленнями.

Клас масажу містить 1 поле і 1 метод, як показано на рисунку 3.6.

```
public class Massage
{
    int telegram_ID;

    public Massage Send(int User_Id)...
```

Рисунок 3.6 – Клас Massage

Це поле використовується для ідентифікації повідомлення та відправлення його користувачеві. Значення цієї змінної генерується Telegram, коли користувач надсилає повідомлення, і є унікальним для кожного повідомлення.

Цей метод відповідає за відправку повідомлення одному користувачеві, а не всій групі. Потрібне спілкування з конкретними користувачами та правильне виконання певних команд.

Вхідне — це ідентифікатор користувача, якому ви хочете надіслати повідомлення, і ваше власне повідомлення, яке буде надіслано у форматі стрічки.

у класі Field Class 2 (рисунок 3.7)

```
public class Group
{
    int ID;
    int ID_Admin;


    Massage SendAll(Massage m)
```

Рисунок 3.7 – Клас Group

GroupID використовується для ідентифікації групи і є унікальним для кожної групи. Він створюється, коли конкретна команда створює групу. для будь-якої взаємодії з певною групою.

Інше поле, ID_Admin, потрібне для ідентифікації адміністратора групи, який надсилатиме інформацію. Отримано з Telegram під час створення групи.

Також використовується в деяких інших командах.

Клас Massage SendAll також містить метод.

Основне завдання цього методу - розсилати інформацію всім абонентам групи.

Щоб отримати всі підписки, які містять ідентифікатор групи, ми отримуємо ідентифікатори всіх передплатників і використовуємо метод Massage.SendMassage, щоб поступово надсилати повідомлення кожному користувачеві.

Клас Subscribe містить 3 поля, як показано на рисунку 3.8.

```
public class Subscribe
{
    int UserId;
    int GroupId;
    int ID;
}
```

Рисунок 3.8 – Клас Subscribe

Поле «ID» — це унікальний номер підписки, згенерований під час створення. Використовується для визначення підписок.

Друге поле містить інформацію про те, на яку групу підпишеться. Отримано під час підписки.

Третє поле показує, на якого користувача ви підписані. Одержується при створенні підписки від Telegram і є унікальним ідентифікатором користувача.

Також реалізовано деякі команди для взаємодії користувача з програмою.

Коли ви вводите команду створення нової групи для інформації, і після того, як користувач введе необхідні дані, вона буде відправлена на сторону сервера, де обробляється і зберігається у форматі JSON.

Алгоритм команди / create показано на рисунку 3.9.

Рисунок 3.9 – Алгоритм роботи команди /create

Після введення команди /join введені дані перевіряються. Якщо вони правильні, створюється запис підписки, інакше - користувач видає повідомлення про помилку при введенні даних.

Алгоритм команди /join показаний на рисунку 3.10

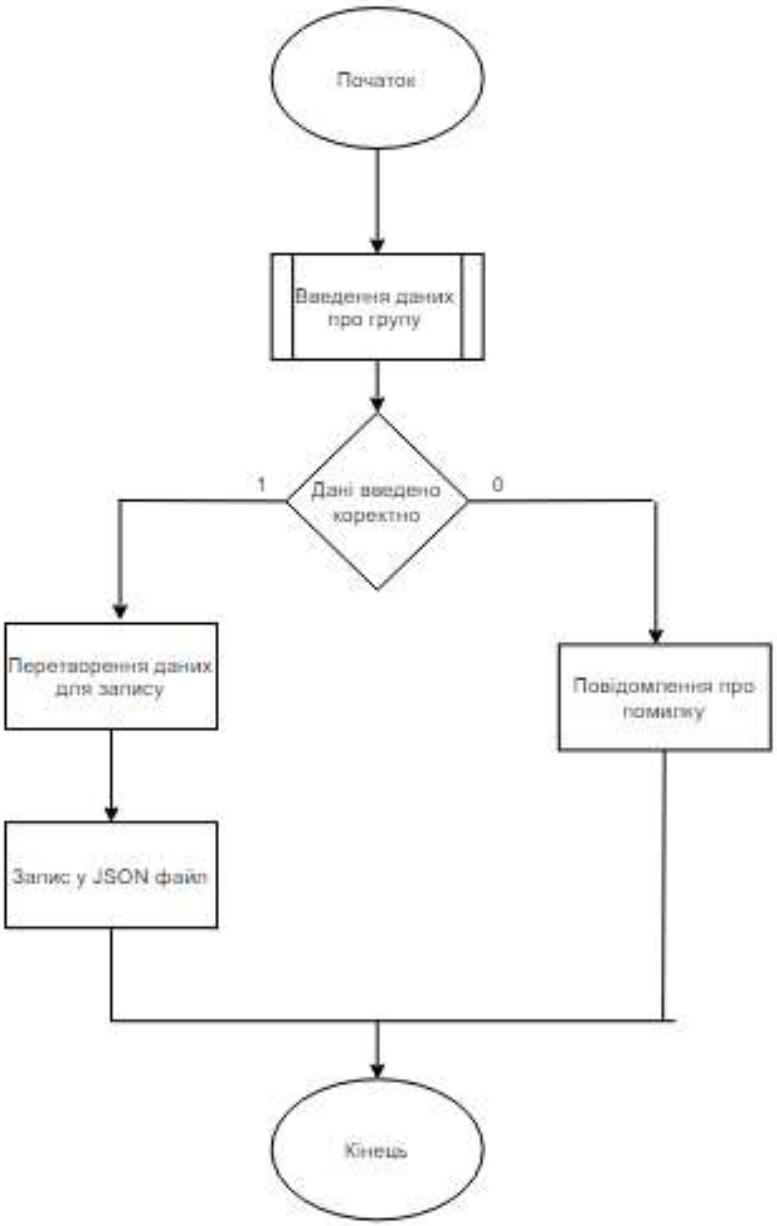

Рисунок 3.10 - Алгоритм команди /join

Коли команда / list вводиться в розділ сервера, підписки конкретного користувача шукаються та надсилаються користувачеві у вигляді тексту.

Алгоритм команди / list показано на рисунку 3.11.

Рисунок 3.11 – Алгоритм команди /list

Після того як команда /post отримує дані користувача, вона надсилає запит на сервер. Надішліть запит на підписку, і повідомлення, створене адміністратором, буде надіслано всім користувачам, підписаним на групу.

Алгоритм команди /post показано на рисунку 3.12.

Рисунок 3.12 – Алгоритм команди /post

/help - надає користувачам коротку інформацію про бота та відповіді на поширені запитання.

3.4 Висновки

В процесі впровадження третьої частини аналізуються основні інструменти, реалізовані Telegram-ботом, такі як ручна розробка та використання API Telegram Bot. Вибрано середовище розробки Visual Studio. Тестовий аналіз показує, що найбільш ефективною технологією для написання серверної частини буде .Net Core.

Розроблено програмний модуль, який містить логіку серверної частини програми..

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Огляд рішень для тестування додатків

У процесі створення програми важливо оцінити функціональність на етапі розробки. Перевірка роботи програми - це спеціальний тест.

У сучасних умовах займатися тестуванням програмістам можна самостійно за допомогою автоматизованих тестів, завдяки плагінам, які дозволяють перевіряти точність функцій, модулів та окремих процесів. На початку надається багато даних, а також результати, написані програмістом. Перевірка коду вихідної функції дає результат, якщо він відповідає специфікації програміста. Однак цей метод тестування не був обраний, оскільки модулі не дуже великі, а розробка повинна бути швидкою.

Тестування програмного забезпечення є важливою частиною процесу розробки програмного забезпечення для перевірки того, що якість або функціональність програми відповідає певним вимогам розробки.

Тестування програмного забезпечення — це остаточний огляд аналізу вимог до програмного забезпечення, специфікацій дизайну та попередньо налаштованого кодування, і є ключовим кроком у забезпеченні якості програмного забезпечення.

Тестування програмного забезпечення — це процес запуску налагоджувача. Тестування програмного забезпечення охоплює обидві фази життєвого циклу програмного забезпечення, і необхідні тести (так зване модульне тестування) зазвичай виконуються після написання кожного модуля. Кодування та модульне тестування є фазою життєвого циклу програмного забезпечення. Після цього етапу також проводяться різноманітні комплексні тестування програмної системи, що є ще однією незалежною фазою життєвого циклу програмного забезпечення, фазою тестування. Початкова оцінка тестування:

Перший тест – це підтвердження якості програмного забезпечення, з одного боку, щоб підтвердити, що програмне забезпечення робить те, що ви очікуєте

(робить це правильно), а з іншого боку, щоб підтвердити якість програмного забезпечення, і програмне забезпечення виправляє подію. Це правда).

Другий тест, надайте розробникам або керівникам проектів інформацію, наприклад зворотний зв'язок, під час підготовки до оцінки ризику.

Третій тест, програмного забезпечення полягає у перевірці не лише самого програмного забезпечення, а й процесу розробки програмного забезпечення. Якщо після розробки програмного забезпечення виникає багато проблем, швидше за все, процес розробки програмного забезпечення не ідеальний. Тому третя мета тестування програмного забезпечення — забезпечення високої якості всього процесу розробки програмного забезпечення.

Якість програмного забезпечення вимірюється кількома способами:

- Робіть правильну справу в потрібний час.
- Відповідати вимогам деяких стандартів застосування, таким як різні методи роботи та вимоги користувачів у різних країнах, а також вимоги до технічного обслуговування та відповідні засоби контролю в дизайні.
- Якість сама по собі означає, що програмне забезпечення відповідає вимогам, перерахованим на початку, а не красі чи складності коду, що програмне забезпечення є високоякісним (якість визначається як відповідність вимогам, а не «хороша» чи «поганий» програмний продукт).
- Якість також означає, що вона задовольняє потреби клієнта (якість також означає «задовольняє потреби клієнта»). Найважливіше в індустрії тестування програмного забезпечення – це виходити з потреб клієнтів, дивитися на продукт з точки зору клієнта, як клієнти будуть використовувати продукт і з якими проблемами вони зіткнуться під час використання. Тільки вирішуючи ці проблеми, можна говорити про підвищення якості програмних продуктів.

Після проходження комплексного тестування збірка програмного забезпечення завершена, дефекти інтерфейсу усунені, і можна переходити до останнього етапу тестування програмного забезпечення - тестування підтвердження. Перевірочне тестування має підтвердити, що програмне

забезпечення відповідає договірним вимогам, тобто чи відповідає воно критеріям перевірки, зазначеним у специфікація вимог до програмного забезпечення:

1. Підтвердьте стандарт тесту Використовуйте програмне забезпечення, щоб переконатися, що виконується серія тестів картриджів. Підтверджувальне тестування також вимагає розробки плану та процедури тестування.

У плані тестування має бути зазначено тип тестування та графік тестування. Процес тестування визначає деякі спеціальні тестові випадки, щоб показати, чи відповідає програмне забезпечення вимогам. план або процес, чи відповідає програмне забезпечення всім договірним завданням і продуктивності, документація є повною та точною, зосереджена на людино-машинному інтерфейсі (наприклад, переносимість, сумісність, помилки), відмовостійкість і ремонтопридатність) тощо), що задовольняє користувача.

Є дві версії результатів тестування: одна, що функції та продуктивність відповідають специфікації програмного забезпечення та є прийнятними для користувачів, а інша – що програмне забезпечення не відповідає специфікації програмного забезпечення, користувач не може прийняти.

Часто серйозні помилки та відхилення під час запланованого будівництва важко виправити, поки проект не досягне цієї стадії, що вимагає консультації з користувачем для пошуку правильного рішення проблеми.

2. Огляд конфігурації. Іншою важливою частиною валідаційного тестування є перевірка конфігурації. Мета аудиту полягає в тому, щоб переконатися, що програмне забезпечення повністю налаштовано, засекречене та містить інформацію, необхідну для підтримки програмного забезпечення.

Тестування системи може складатися з кількох різних тестів, призначених для повного запуску системи та перевірки того, що різні компоненти системи працюють належним чином і виконують покладені на них завдання.

Ось деякі типи тестування системи:

1. Пошуково-пошукове тестування. Тестування відновлення в першу чергу перевіряє відмовостійкість системи. Коли виникає системна помилка, чи можна виправити помилку та перезапустити систему протягом визначеного

інтервалу. Тестування відновлення спочатку використовує різноманітні методи, щоб спричинити збій системи, а потім перевіряє, що система може відновитися якомога швидше.

Для автоматичного відновлення потрібно перевірити правильність перенавантаження, механізму контрольної точки, відновлення даних і перезапуску, а для відновлення вручну потрібно оцінити середній час відновлення та визначити, чи знаходиться він у прийнятному діапазоні.

2. Перевірка безпеки. Перевірки безпеки перевіряють здатність системи запобігати несанкціонованому доступу. Під час перевірок безпеки тестери вдалися до нелегальних вступників і різними способами намагалися прорвати лінію оборони. Наприклад, спроба заблокувати або розшифрувати паролі. Налаштувати програмне забезпечення для знищення механізмів захисту системи; навмисне вимикання системи, спроба незаконного доступу та використання можливості відновлення.

Спроба отримати необхідну інформацію шляхом перегляду несекретних даних тощо. Система не підходить, якщо теоретично достатньо часу та ресурсів. Тому принцип розробки захищеної системи полягає в тому, що вартість незаконного доступу перевищує вартість захищеної інформації. Наразі зловмисники не вигідні.

3. Силове випробування. Тест потужності перевіряє стійкість програми до ненормальних умов. Стрес-тести завжди змушують систему використовувати незвичайні конфігурації ресурсів. Наприклад: Коли нормальна частота переривань становить одне або два в секунду, запустіть тестовий приклад, який генерує десять переривань за секунду. Кількісно підвищити швидкість введення даних і перевірити швидкість відповіді вхідних під задач. Виконайте тест. максимальний простір для зберігання (або інші ресурси). Виконайте тестові випадки, які можуть призвести до збою операційної системи через сильне тремтіння даних у віртуальній пам’яті чи диску тощо.

4. Тестування продуктивності. Для цих систем реального часу та вбудованих систем, хоча програмне забезпечення відповідає функціональним вимогам, воно не відповідає вимогам виробника.

Для практичного ознайомлення у дипломній роботі надаються такі види тестів [19]. Існує кілька типів системних тестів:

- Тестування чорної скриньки;
- Тестування сірої скриньки;
- Тестування білої скриньки.

Для кращого розуміння був проведений розбір з відмінностями у тестуванні чорної та білої скриньки.

Тестування чорного ящика, також відоме як функціональне тестування, в першу чергу призначене для виявлення помилок у вимогах до проектування програмного забезпечення або специфікаціях програмного забезпечення. Розробка програмного забезпечення є специфічною, часто розробленою для конкретних завдань.

Опис функції програмного забезпечення базується на аналізі специфікації вимог на етапі вимог до програмного забезпечення. У процесі проектування програмне забезпечення поділяється на одну або кілька функцій. Щоб забезпечити належне функціонування цих функцій, програмне забезпечення тестується, щоб краще відповідати потребам і вимогам користувачів.

Тестування чорної скриньки є протилежністю тесту білої скриньки. Програма розробки програмного забезпечення вважається коробкою, яку неможливо відкрити. В основному тестування реалізації програмного забезпечення або функціональності.

Метод випробування

1. Метод поділу класів еквівалентності

Розділення класів еквівалентності є типовим і важливим методом тестування чорного ящика, який ділить всі можливі вхідні дані програми на кілька еквівалентних класів, а потім вибирає репрезентаційні дані з кожного сегмента як тестовий приклад.

Тестові випадки містять репрезентативну статистику дійсних класів еквівалентності та недійсних класів еквівалентності для забезпечення цілісності та репрезентативності тестових випадків. Існує два основних етапи розробки тестових випадків з використанням цього підходу: визначення класів еквівалентності, та побудова тестового випаду.

2. Метод граничного аналізу

Аналіз граничних значень – це метод тестування чорного ящика, який досліджує вхідні або вихідні обмеження програми. Фактичне завдання тестування доводить, що тестовий набір з граничними умовами має вищу швидкість повернення тесту, ніж тестовий набір без граничних умов. Граничні умови, згадані тут, відносяться до тих станів класів еквівалентності вхід-вхід, які знаходяться безпосередньо на, за межами або нижче кордону

3. Метод діаграми причин

Метод причинно-наслідкових діаграм також є широко використовуваним методом перевірки чорного ящика, який є простою логічною схемою. Причинно-наслідкові діаграми можуть візуалізувати причинно-наслідкові зв’язки між умовами входу та вихідними діями, що може допомогти тестувальникам зосередитися на комбінаціях вхідних даних, які мають відношення до функціональності програми.

Метод причинно-наслідкової діаграми — це метод тестування, який варіюється. Він підходить для опису комбінації вхідних умов. Відповідно до комбінації вхідних умов, зв'язку обмежень і причинно-наслідкового зв'язку вихідних умов, він аналізує різні комбінації вхідних умов і розробляє тестові випадки. Підходить для вивчення умов входу в програму та різних комбінацій.

4. Помилкові припущення

Метод оцінки дефектів базується на минулому досвіді та інтуїції, посилаючись на дефекти попередніх програмних систем, роблячи висновки про потенційні дефекти та помилки в поточній програмі тестування та цілеспрямовано розробляючи тестові випадки.

Основна ідея розробки тестових випадків з неправильними припущеннями полягає в тому, щоб перерахувати потенційні помилки або особливі випадки в програмі, а потім написати конкретні тестові випадки та розробити тестові випадки відповідно до списку, наприклад, вхідних даних, які з'являються у програма Значенням є "0" або порожній символ - ця умова схильна до помилок; Коли кількість вхідних або вихідних даних невизначена, числа «ні» або «один» також неправильні.

Важливо відзначити, що, читаючи специфікацію припущень, які програміст може зробити для визначення тестових випадків, тестувальники повинні враховувати вхідні дані з точки зору користувача, відповідний внесок у процес тестування, незалежно від того, чи є ця інформація правдоподібною для тестування.

Тестування білого ящика також називають структурним тестуванням, і його основна мета — знайти помилки в процесі кодування програми. Існує багато причин помилок у написанні коду.

Через досвід програмування, знайомство з інструментами розробки та психічний стан процесу кодування, у процесі кодування виникли помилки. Виникла помилка. Ключові синтаксичні помилки програми можна знайти та своєчасно виправити під час налагодження програми. Але важко знайти помилки в програмі, в логічних рішеннях і в тому, як програма виконується

При написанні коду для реальної програми жоден програміст не може гарантувати, що структура написаного коду є безпомилковою, навіть старший програміст. Без гарантії. Під час тестування білого ящика програма вважається відкритою коробкою з оригінальною тестовою програмою, і внутрішня структура коробки також може бути проаналізована, тому цей метод тестування може ретельно вивчити структуру програмного коду.

Існує три типи методів тестування білого ящика, одним з яких є аналіз структури програми. Ви можете спочатку намалювати блок-схему програми відповідно до вихідного коду, а потім проаналізувати структурну схему програми відповідно до блоку. Другий — тестування логічної області, тестування всіх

шляхів відповідно до внутрішньої структури програми, що є методом комплексного тестування шляху.

Його можна реалізувати в чотири етапи:

1. Перший крок — написати план тестування. Розробити курс тестування програмного забезпечення відповідно до технічного завдання, визначити персонал, обсяги, технології, ризики тощо, а також розробити план тестування або план тестування.

2. Другим кроком є написання тестових випадків. Розбив структуру програмного забезпечення відповідно до вихідного коду та його аналіз будь-яким стандартизованим методом, спроектуйте тестові випадки та створіть таблицю тестових наборів.

3. Третій крок - запустити тестовий приклад. Записуйте тести системи та результати випробувань для створення списків несправностей і звітів про несправності на основі раніше написаних прикладів тестів.

4. Четвертий крок – написання випускного іспиту. Підсумуйте минулі випробування, використовуйте час, проаналізуйте кількість виявлених несправностей високого, середнього та низького рівня, оцініть систему та створіть повний остаточний звіт.

Тестування програмного забезпечення «Чорний ящик» відноситься до тестування програмного інтерфейсу. Цей метод розглядає об’єкт, що тестується, як чорний ящик, тестер не розглядає внутрішню логічну структуру та внутрішні характеристики програми, а лише перевіряє, чи відповідає функція програми її функціональному опису відповідно до специфікації. розклад . Тому тестування чорного ящика також відоме як функціональне тестування.

Тестування програмного забезпечення методом білого ящика — це ретельне вивчення деталей програмної програми. Цей підхід розглядає тестовий об’єкт як відкрите поле, яке дозволяє тестувальникам розробляти або вибирати тестові приклади, використовуючи внутрішню логічну структуру програми та пов’язану інформацію, а також точкове тестування всіх логічних шляхів програми шляхом вивчення стану програми в різний час.

Щоб визначити, чи реальна ситуація відповідає очікуванням. Ось чому тестування білої коробки також називають структурним тестуванням. Метою тестування білого ящика є насамперед тестування програмних модулів.

Між тим потрібно в цілому зрозуміти специфіку тестування Usability воно в основному використовується для оцінки дизайну веб-сайту або мобільного додатка, але його також можна використовувати під час спілкування з розумним обладнанням, часто запрошуючи реальних користувачів цільової аудиторії виконувати певні функції продукту в певних сценаріях.

Спостерігайте за фактичною роботою користувачів у процесі фактичного використання, записуйте та детально аналізуйте проблеми, з якими стикаються користувачі під час використання продукту, виявляйте проблеми під час використання продукту, збирайте якісні та кількісні дані, допомагайте покращувати продукт та виявляти цільових користувачів. Задоволеність продуктом.

Простіше кажучи, тестування юзабіліті – це метод виявлення проблем із продуктивністю продукту та його задоволеністю шляхом моніторингу користувачів, які використовують продукт для виконання конкретних завдань.

Також для кращого розуміння розглянуто метод сірого ящика, який є комбінацією обох.

Сірий ящик — це пристрій, який локально ідентифікує програму або системний робочий процес. Тестування сірого ящика, також відоме як аналіз сірого ящика, — це метод налагодження програмного забезпечення, заснований на обмежених знаннях внутрішніх деталей програми. Тестер може знати, як взаємодіють компоненти системи, але не має детального розуміння внутрішніх функцій. які необхідно проаналізувати ззовні.

Тестування сірого ящика часто використовується для додатків веб-сервісів, оскільки Інтернет може забезпечити відносно стабільний інтерфейс, незважаючи на складність і розвиток програми. Сіра зона для тестування не є інвазивною та упередженою, оскільки тестувальникам не потрібно торкатися вихідного коду.

Існує чітка різниця між розробниками та тестувальниками, знижується ризик кадрових конфліктів.

Проте тестування сірих коробок важче знайти та виправити потенційні проблеми, ніж тестування білих ящиків, особливо в програмі, де ви можете повністю зрозуміти внутрішні деталі білого ящика. Тестування сірого ящика. Він враховує знання користувача про конкретну систему та операційне середовище.

Він оцінює дизайн прикладного програмного забезпечення в середовищі компонентів системи. Тестування сірого ящика включає методи та інструменти, засновані на внутрішніх знаннях програми та її інтерактивного середовища, і може використовуватися в тестуванні чорної скриньки для підвищення ефективності тестування, виявлення помилок та аналізу.

Тестування сірого ящика має вхідні та вихідні дані, але тестування розроблено з використанням інформації про код і операції програмного забезпечення, які часто знаходяться за межами поля зору тестувальника

Після розгляду і аналізу усіх нюансів пов’язаних з тестуванням для тестування безпосередньо додатку було обрано метод чорного ящика [20].

Залежно від початкової специфікації, під час тестування можуть бути виконані наступні випробування:

- Перегляд команд доступу користувача. Усі команди мають бути доступні для всіх користувачів.
- Перевірте правильність учасників групи.
- Перевірте правильність засвоєних текстових повідомлень.
- Перевірте точність інформації, витягнутої з медіафайлів.

4.2 Тестування роботи програмного продукту

Було протестовано розроблений додаток, а саме користувацьку частину та серверну частину.

Результати тестування:

1. До боту доєдналось декілька користувачів. Для кожного з них був доступний повний перелік команд (рис. 4.1).

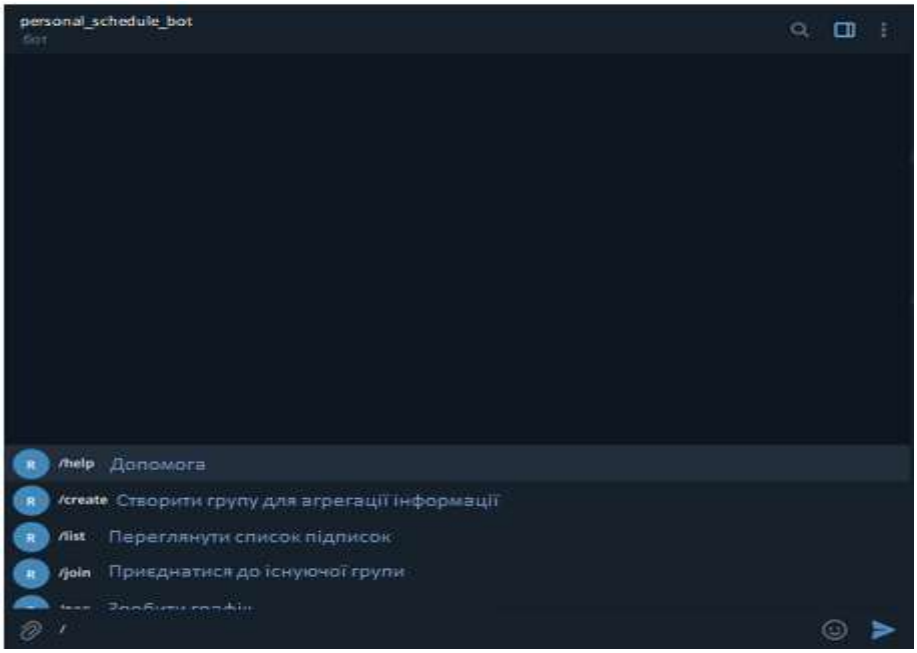

Рисунок 4.1 - Перелік команд

2. Долучення до групи/агрегатора за правильним ID проходить без нюансів(рис.4.2).

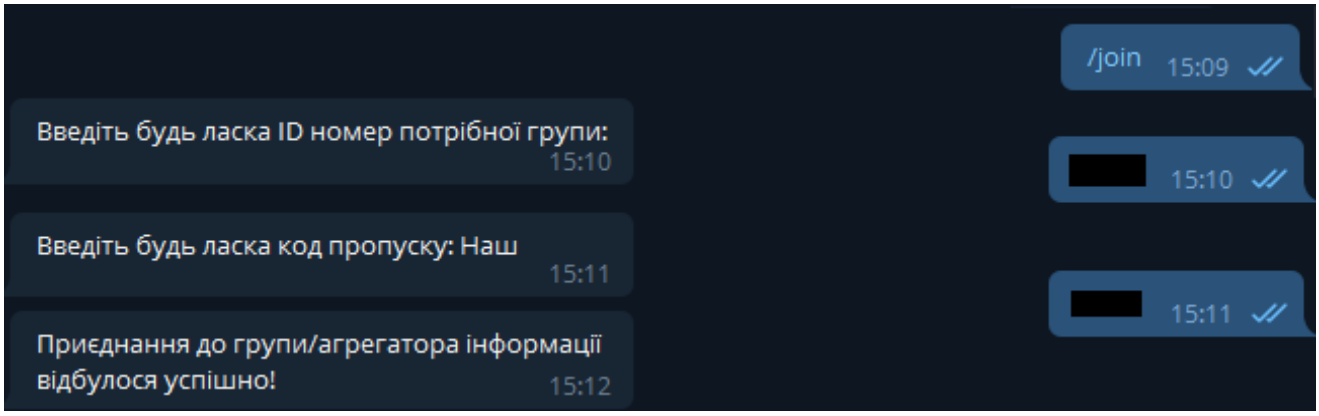

Рисунок 4.2 - Долучення до групи/агрегатора

3. Особистий графік отримую адекватно (рис. 4.3).

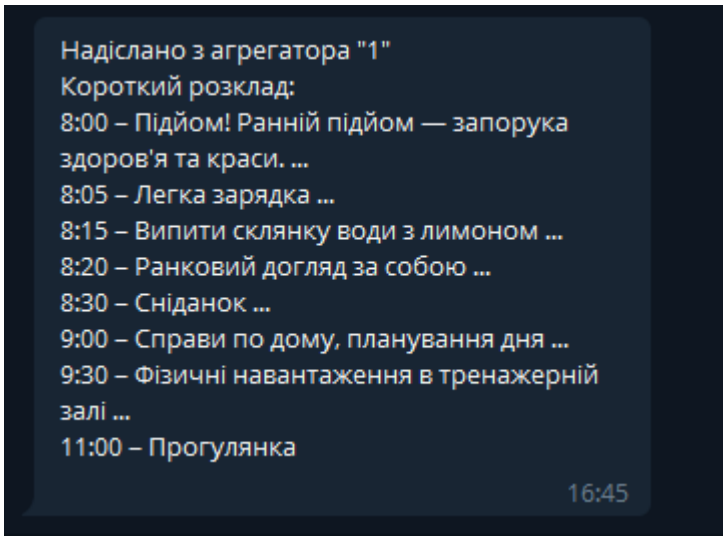

Рисунок 4.3 – Особистий графік

4. Особистий графік з вкладенням надіслано адекватно (рис. 4.4).

Рисунок 4.4 – Особистий графік з вкладенням

5. Долучення до групи/агрегатора за неправильним ID проходить з помилками, користувача не пропускає далі(рис. 4.5).

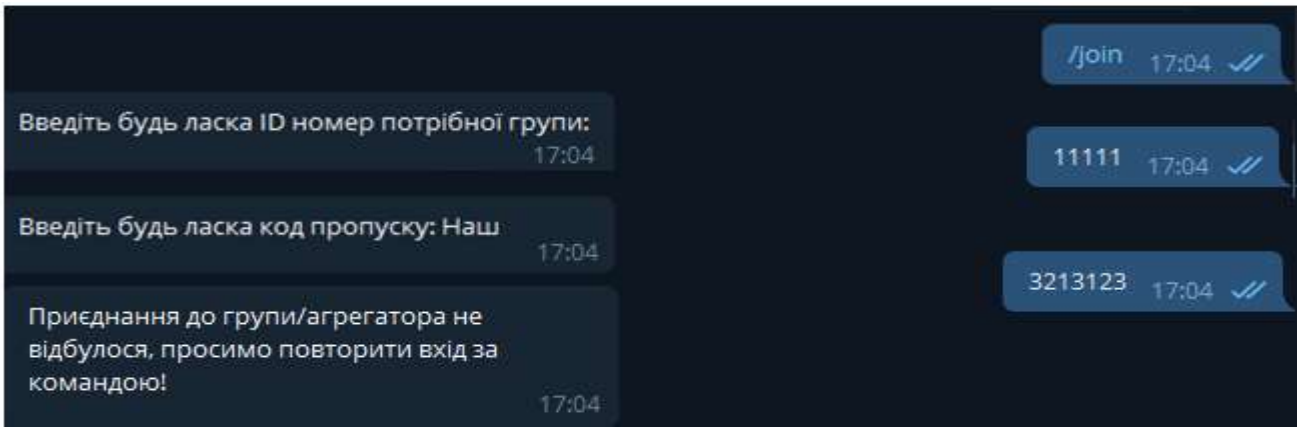

Рисунок 4.5 – Намагання долучитися до групи/агрегатора з некоректним ідентифікаційним кодом

6. Приєднання до групи/агрегатора з невірним кодом не відбувається. Користувачу надається інформація про помилку (рис. 4.6).

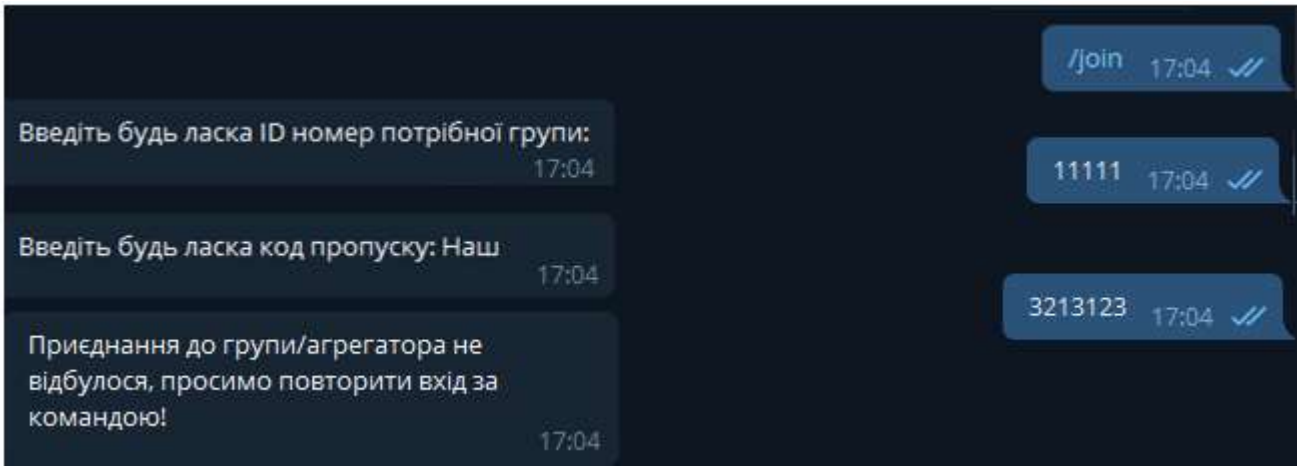

Рисунок 4.6 - Намагання долучитися до групи/агрегатора з некоректним кодом пропуску

7. Перевірка адекватності функціонування команди /list. Усі групи/агрегатори, до яких користувач долучений користувач показуються (рис. 4.7).

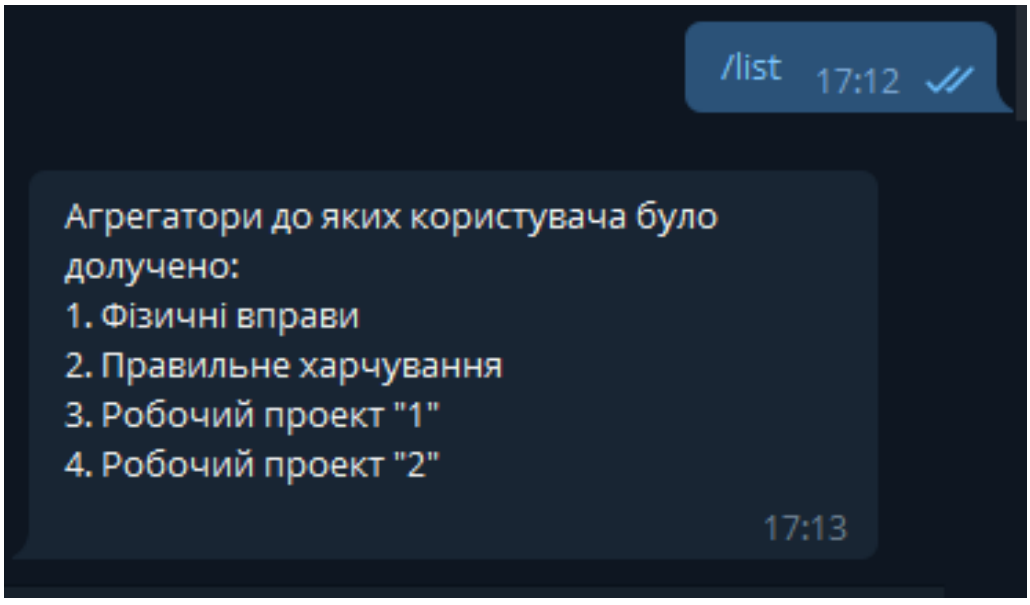

Рисунок 4.7 – Приклад функціонування команди /list

4.3 Розробка інструкції користувача

Щоб працювати з програмою, користувачі повинні мати на своєму комп’ютері встановлену програму Telegram і під’єднатися до Інтернету. Персональний комп'ютер користувача повинен відповідати мінімальній конфігурації, що наведена у таблиці 4.1.

Таблиця 4.1 – Мінімальна конфігурація:

Процесор	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2,4 ГГц
Об’єм оперативної пам’яті	512 Мб
Відеопам’ять	256 Мб
Вінчестер	200 Гб
Операційна система	Windows 7, 8, 8.10,12, Linux,Android

Для початку роботи з ботом потрібно під`єднатись до нього, знайшовши його за тегом у пошуку. Скористайтесь стрічкою пошуку, що зображена на рисунку 4.8 та розташована у верхньому лівому куту програми.

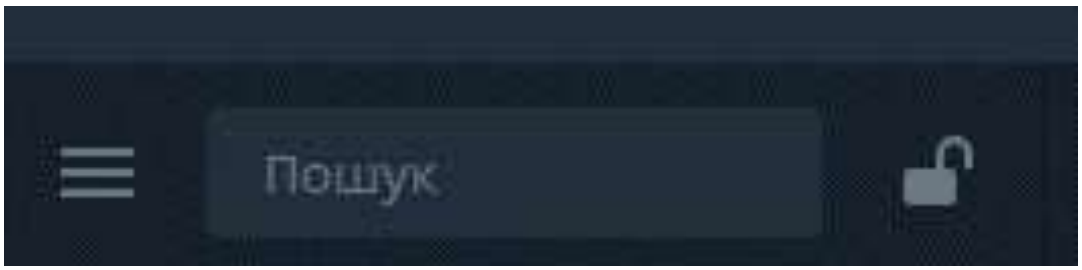

Рисунок 4.8 – Стрічка пошуку

Після підключення необхідно натиснути кнопку «Розпочати», як показано на рисунку 4.9. це знаходиться внизу блоку повідомлень.

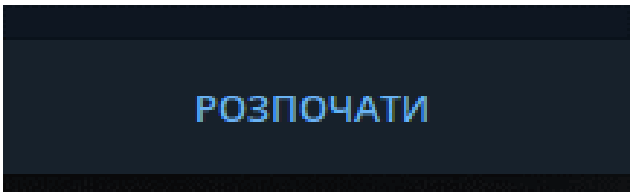

Рисунок 4.9 – Кнопка «Розпочати»

Натискання кнопки / start негайно реєструє команду для автоматичного запуску бота. Ви отримаєте вітання, яке допоможе вам орієнтуватися в інтерфейсі програми (рисунок 4.10).

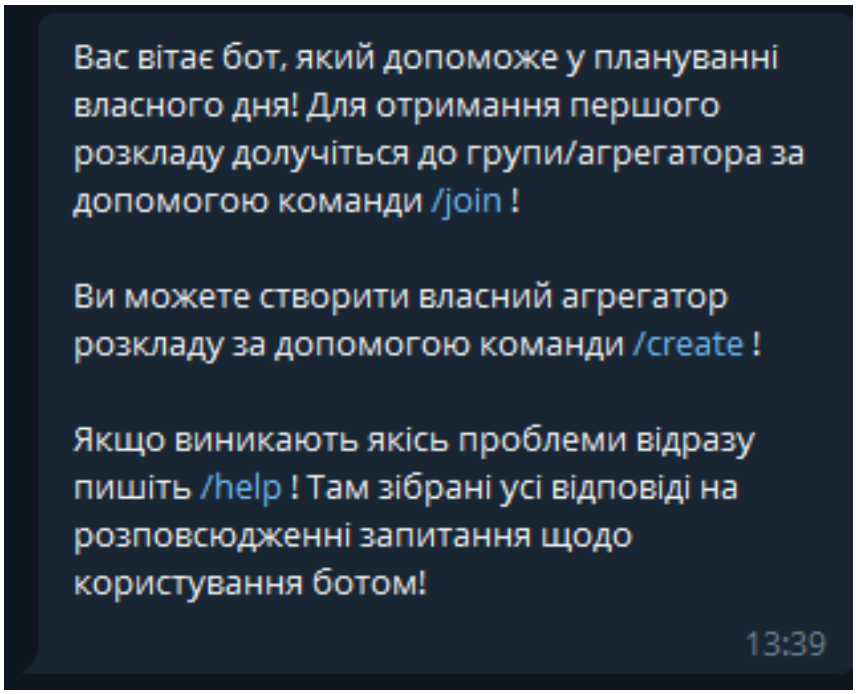

Рисунок 4.10 – Стартове вітання бота

Тепер можна починати користуватися роботом. Усі команди робота доступні відразу. Розгляньте їх.

По-перше, вам потрібно скористатися командою /help. Вона відображає всю необхідну інформацію для взаємодії з програмою: список усіх команд, короткі описи, відповіді на поширені запитання тощо.

Використовуйте команду /join, щоб приєднатися до групи, з якої ви хочете

отримати інформацію/графік. Бот запитує адресу групи, яку можна отримати у адміністратора групи. Далі вам потрібно ввести пароль для успішного членства.

У разі успіху бот повідомить про це. Якщо підключення не вдається, ви отримаєте повідомлення з поясненням причини збою.

Використовуйте команду /list, щоб переглянути список груп передплати. Бот шукатиме назви всіх груп, на які підписаний користувач, згенерує список і поверне його.

Насправді, для спілкування з роботом буде канал інформації користувача. Він надсилає всі повідомлення групі, яка підписується, з підписом групи відправника.

Ви повинні використовувати пошук, щоб знайти повідомлення з певної групи/агрегатора. Для цього введіть назву групи, яку потрібно шукати, у рядку, показаному на рисунку 4.8.

Для швидкого перегляду файлів, зображень, відео та інших повідомлень, які містять більше, ніж текст, можна використовувати панель праворуч, яку показано на рисунку 4.11.

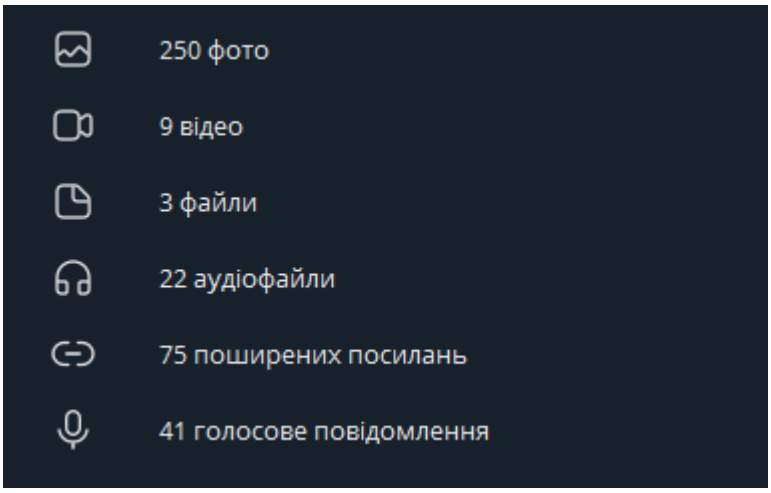

Рисунок 4.11 – Блок з вкладеними файлами

Створіть власну групу та скористайтеся командою /create, щоб помістити туди інформацію. Вам потрібно ввести нову назву групи/агрегатора та пароль.

Після успішного створення групи користувач отримує повідомлення з автоматично створеним кодом, іменем та паролем, введеними раніше. Цю інформацію необхідно надати потенційним клієнтам, щоб вони могли успішно приєднатися до групи/агрегатора.

Використовуйте команду /post, щоб створити власний розклад. Якщо користувач є адміністратором кількох груп, необхідно вибрати один зі списків, до яких він хоче надсилати частини розкладу, та корегуючи інформацію.

Потім вам потрібно надіслати повідомлення бота, яке буде надіслано кожному клієнту в групі/агрегаторі. Оскільки керівники груп також є клієнтами, вони також отримують сповіщення.

4.4 Висновки

Розглянуто основні методи тестування додатків. План був повністю перевірений. Оскільки тестових помилок не виявлено, інтерфейс користувача в програмі відображається коректно. На стороні сервера також немає нічого поганого.

Розроблено посібник користувача.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи було розроблено програмні засоби для реалізації стрічки інформації у месенджері Telegram. Для розробки було використано середовище програмування Visual Studio 2019. Робота розроблена згідно методичних вказівок.

Було проаналізовано стан даного питання на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом. У результаті аналізу обрано мову програмування C# та бібліотеку Telegram Bot Api.

У процесі виконання роботи було зроблено: розроблено алгоритм для підбору особистого графіку, розроблено графічний інтерфейс користувача для взаємодії із серверною частиною, розроблено програмні засоби для реалізації стрічки інформації особистого графіку користувача в меседжері Telegram, проведено тестування програмного продукту.

Перед створенням програмного продукту було розроблено метод пошуку та вибору інформації, для створення персонального розкладу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Статистика Телеграм в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://tlgrm.in.ua/vsjo-o-telegram/stati/statistika-telegram-v-ukraine/.

2. За рік карантину кількість українців в соцмережах зросла на сім мільйонів [Електронний ресурс] – Режим доступу до ресурсу:https://www.dw.com/uk/za-rik-karantynu-kilkist-ukraintsiv-u-sotsmerezhakh-zrosla-na-sim-milioniv/.

3. Статистика Телеграм в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://tlgrm.in.ua/vsjo-o-telegram/stati/statistika-telegram-v-ukraine/.

4. Telegram бот для отправки сообщений вашим клиентам [Електронний ресурс] – Режим доступу до ресурсу: https://www.smsfeedback.ru/telegram/.

5. SendPulse Украина [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/ru/features/chatbot/telegram.

6. 3seller [Електронний ресурс] – Режим доступу до ресурсу: https://3seller.com

7. Застосунок [Електронний ресурс] - https://uk.wikipedia.org/wiki/Застосунок

8. Формат JSON, метод toJSON [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/json.

9. XML — Обзор [Електронний ресурс] – Режим доступу до ресурсу: https://coderlessons.com/tutorials/xml-tekhnologii/vyuchit-xml/xml-obzor.

10. Telegram User ID [Електронний ресурс] – Режим доступу до ресурсу: https://tgrm.su/blog/faq/telegram-user-id/

11. Додаток(модуль) [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Додаток_(програма)

12. Формати даних [Електронний ресурс] – Режим доступу до ресурсу: https://socialdata.org.ua/manual/manual2/

13. Подробная инструкция по Telegram для компьютера [Електронний ресурс] – Режим доступу до ресурсу: https://popravilam.com/blog/067-instrukciya-telegram-dlya-kompyutera.html

14. Діаграми «сутність-зв’язок» [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.org/8-191266.html.

15. Модель «сутність-зв’язок» [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв%27язок»

16. Всё о чём должен знать разработчик телеграм ботов [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/543676/

17. .Net Core [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/ASP.NET_Core.

18. Створення інструкції [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/sintructpzw44ikinml

19. Статичне та динамічне тестування Обзор [Електронний ресурс] – Режим доступу до ресурсу: https://coderlessons.com/tutorials/test324dyna

20. Spring Framework — Обзор [Електронний ресурс] – Режим доступу до ресурсу: https://coderlessons.com/tutorials/java-tekhnologii/uchis-vesne/spring-framework-obzor.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"____ "_____2022 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного забезпечення інтелектуального боту для ведення особистого графіку»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. Г. О. Черноволик
"____"_____2022 р.
Виконав:
студент гр. 2ПІ-18б П. І. Довбиш
"____"_____2022 р.

Вінниця 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного забезпечення інтелектуального боту для ведення особистого графіку».

Галузь застосування – автоматизовані системи обробки даних.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 66 від « 24 » березня 2022 р.

3. Мета та призначення розробки.

Метою є підвищення ефективності та швидкодії процесу пошуку інформації шляхом використання спеціалізованих алгоритмів підбору потрібного контенту для обраного користувача в зручний для нього час, що дозволяє створювати індивідуальний графік.

Призначення роботи – підвищення ефективності використання часу користувачем, шляхом створення індивідуального графіку.

4. Вихідні дані для проведення НДР

1. Zhiheng Huang, Wei Xu, Kai Yu Bidirectional LSTM-CRF Models for Sequence Tagging 2015. URL: https://arxiv.org/pdf/1508.01991.pdf (дата звернення 22.05.2022).

2. Ghosh S., Gunning D. Natural Language Processing Fundamentals. – Birmingham: Packt Publishing, 2018. 374 с.

3. Goldberg Y. Neural Network Methods in Natural Language Processing. – San Rafael: Morgan & Claypool Publishers, 2017. – 309 с.

4. Lane H., Hapke H., Howard C. Natural Language Processing in Action. – New York: Manning Publications, 2019. – 544 с.

5. Технічні вимоги

Вхідні дані – текст, медіа файли; вихідні дані – результат у вигляді індивідуального графіку.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинг програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного забезпечення інтелектуального боту для ведення особистого графіку

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Черноволик Г.О.

Оригінальність	91,7%
Схожість	8,3%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Довбиш П.І.

Керівник роботи _____ Черноволик Г.О.

Додаток В – Лістинг програми

HomeController.cs

```
Using
System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Web;
        using System.Web.Mvc;


        namespace TelegramBotApp.Controllers
        {
           public class HomeController : Controller
           {
              public string Index()
              {
                 return "Test massage";
              }

           }
        }
```

MessageController.cs

```
using
System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Net;
        using System.Net.Http;
        using System.Threading.Tasks;
        using System.Web.Http;
        using System.Web.Http.Results;
        using Telegram.Bot.Types;
        using TelegramBotApp.Models;


        namespace TelegramBotApp.Controllers
        {
           public class MessageController : ApiController
           {
              [Route(@"api/message/update")] //webhook uri part
```

```
        public async Task<OkResult> Update([FromBody]Update update)
        {
            var commands = Bot.Commands;
            var message  = update.Message;
            var client   = await Bot.Get();


            foreach(var command in commands)
            {
                if (command.Contains(message.Text))
                {
                    command.Execute(message, client);
                    break;
                }
            }

            return Ok();
        }


    }
}
```

AppSettings.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TelegramBotApp.Models
{
    public static class AppSettings
    {
        public static string Url { get; set; }  =
"https://telegrambotapp.azurewebsites.net:443/{0}";
```

```
        public static string Name { get; set; } = "r_news_bot";

        public static string Key { get; set; }  = "";

    }
}
```

Bot.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;

using Telegram.Bot;
using TelegramBotApp.Models.Commands;

namespace TelegramBotApp.Models
{
    public static class Bot
    {
        private static TelegramBotClient client;
        private static List<Command> commandsList;

        public static IReadOnlyList<Command> Commands =>
commandsList.AsReadOnly();

        public static async Task<TelegramBotClient> Get()
        {
            if(client != null)
            {
                return client;
            }
```

```
            commandsList = new List<Command>();
            commandsList.Add(new HelloCommand());



            client = new TelegramBotClient(AppSettings.Key);
            var hook = string.Format(AppSettings.Url, "api/message/update");
            await client.SetWebhookAsync(hook);

            return client;
        }
    }
}
```

Comand.cs

```
using Telegram.Bot.Types;
using Telegram.Bot;

namespace TelegramBotApp.Models.Commands
{
    public abstract class Command
    {
        public abstract string Name { get; }

        public abstract void Execute(Message message, TelegramBotClient client);

        public bool Contains(string command)
        {
```

```
            return command.Contains(this.Name) &&
command.Contains(AppSettings.Name);
        }

    }
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace TelegramBotApp
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
```

```
            defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
        );
    }
  }
}
```

WebApiConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TelegramBotApp
{
  public static class WebApiConfig
  {
    public static void Register(HttpConfiguration config)
    {
      config.MapHttpAttributeRoutes();

      config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
      );
    }
  }
}
```

Global.asax.cs

```
using
System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Web;
        using System.Web.Http;
        using System.Web.Mvc;
        using System.Web.Optimization;
        using System.Web.Routing;
        using TelegramBotApp.Models;


        namespace TelegramBotApp
        {
           public class MvcApplication : System.Web.HttpApplication
           {
              protected void Application_Start()
              {
                 AreaRegistration.RegisterAllAreas();
                 GlobalConfiguration.Configure(WebApiConfig.Register);
                 RouteConfig.RegisterRoutes(RouteTable.Routes);
                 Bot.Get();
              }
           }
        }
```

Join.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TelegramBotApp.Models.Commands
{
```

```
public class JoinCommand : Command
{
    public override string Name => "join";

    public override void Execute(Message message, TelegramBotClient client)
    {
        var chatId = message.Chat.Id;
        var messageId = message.MessageId

        client.SendTextMessageAsync(chatId, "Введіть код групи до якої Ви бажаєте приєнатись:", replyToMessageId: messageId);

         var me = await Bot.GetMeAsync();



        Bot.OnMessage += BotOnMessageReceived;
        Bot.OnMessageEdited += BotOnMessageReceived;
        Bot.OnCallbackQuery += BotOnCallbackQueryReceived;
        Bot.OnInlineQuery += BotOnInlineQueryReceived;
        Bot.OnInlineResultChosen += BotOnChosenInlineResultReceived;
        Bot.OnReceiveError += BotOnReceiveError;


         Bot.StartReceiving(Array.Empty<UpdateType>());


         Bot.StopReceiving();


         var code = massage.text;

        client.SendTextMessageAsync(chatId, "Введіть пароль від групи:", replyToMessageId: messageId);

         var password = massage.text;

         Bot.StartReceiving(Array.Empty<UpdateType>());


         Bot.StopReceiving();
              if(conrtolClass.FindGroup(code, password)
              {
                    client.SendTextMessageAsync(chatId, "Ви успішно приєднались до групи!", replyToMessageId: messageId);
```

```
                }
                else
                {
                client.SendTextMessageAsync(chatId, " Невірний код групи або пароль!
                Спробуйте ще раз за допомогою команди /join", replyToMessageId:
        messageId);
                }
}
   }
}
```

Join.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TelegramBotApp.Models.Commands
{
   public class PostCommand : Command
   {
      public override string Name => "post";

      public override void Execute(Message message, TelegramBotClient client)
      {
         var chatId = message.Chat.Id;
         var messageId = message.MessageId

       var arr = controlClass.findUsers(message);
          for(int i=arr.lenghth(); i>0;i--)
          {
                client.SendTextMessageAsync(chatId, message,);
          }
}
}
}
```

List.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TelegramBotApp.Models.Commands
{
   public class ListCommand : Command
   {
      public override string Name => "list";

      public override void Execute(Message message, TelegramBotClient client)
      {

 var chatId = message.Chat.Id;
          var messageId = message.MessageId

        var arr = controlClass.findSubscribes(message);

        string m = "Ваші підписки";

          for(int i=arr.lenghth(); i>0;i--)
          {
                m+=arr[i];
                m+="\n"

          }
          client.SendTextMessageAsync(chatId, m)
}
}
}
```

Help.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TelegramBotApp.Models.Commands
{
   public class HelpCommand : Command
   {
```

```
        public override string Name => "help";

        public override void Execute(Message message, TelegramBotClient client)
        {
            var chatId = message.Chat.Id;
            var messageId = message.MessageId



             client.SendTextMessageAsync(chatId, m)
}
}
}
```

Create.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Telegram.Bot;
using Telegram.Bot.Types;

namespace TelegramBotApp.Models.Commands
{
    public class CreateCommand : Command
    {
        public override string Name => "create";

        public override void Execute(Message message, TelegramBotClient client)
        {
            var chatId = message.Chat.Id;
            var messageId = message.MessageId

            client.SendTextMessageAsync(chatId, " Для створення групи вкажіть назву:",
replyToMessageId: messageId);

             var me = await Bot.GetMeAsync();



            Bot.OnMessage += BotOnMessageReceived;
            Bot.OnMessageEdited += BotOnMessageReceived;
            Bot.OnCallbackQuery += BotOnCallbackQueryReceived;
            Bot.OnInlineQuery += BotOnInlineQueryReceived;
```

```
            Bot.OnInlineResultChosen += BotOnChosenInlineResultReceived;
            Bot.OnReceiveError += BotOnReceiveError;


             Bot.StartReceiving(Array.Empty<UpdateType>());


             Bot.StopReceiving();


             var name = massage.text;

            client.SendTextMessageAsync(chatId, " Вкажіть пароль для підключення:",
replyToMessageId: messageId);

              var password = massage.text;

             Bot.StartReceiving(Array.Empty<UpdateType>());


             Bot.StopReceiving();


                  Group g = controlClass.createGroup(name,password,chatID);

                  string m = "Група створена успішно!\nnНазва:";
                  m+= g.name;
                  m+= "\nНомер";
                  m+=g.id;
                  m+="\nПароль: ";
                  m+=g.password;

                  client.SendTextMessageAsync(chatId, " m);
}
   }
}
```

Додаток Г – Графічна частина

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНОГО БОТУ ДЛЯ ВЕДЕННЯ ОСОБИСТОГО ГРАФІКУ

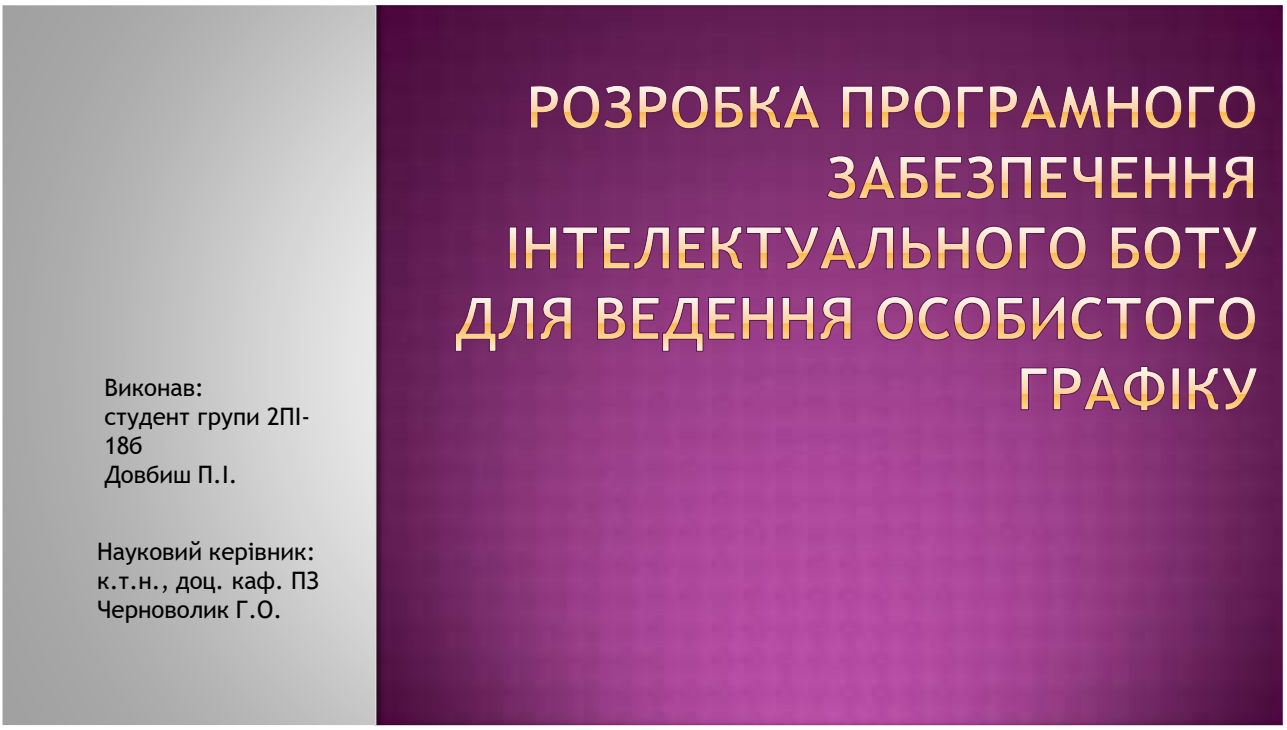

Рисунок Г.1 – Назва роботи

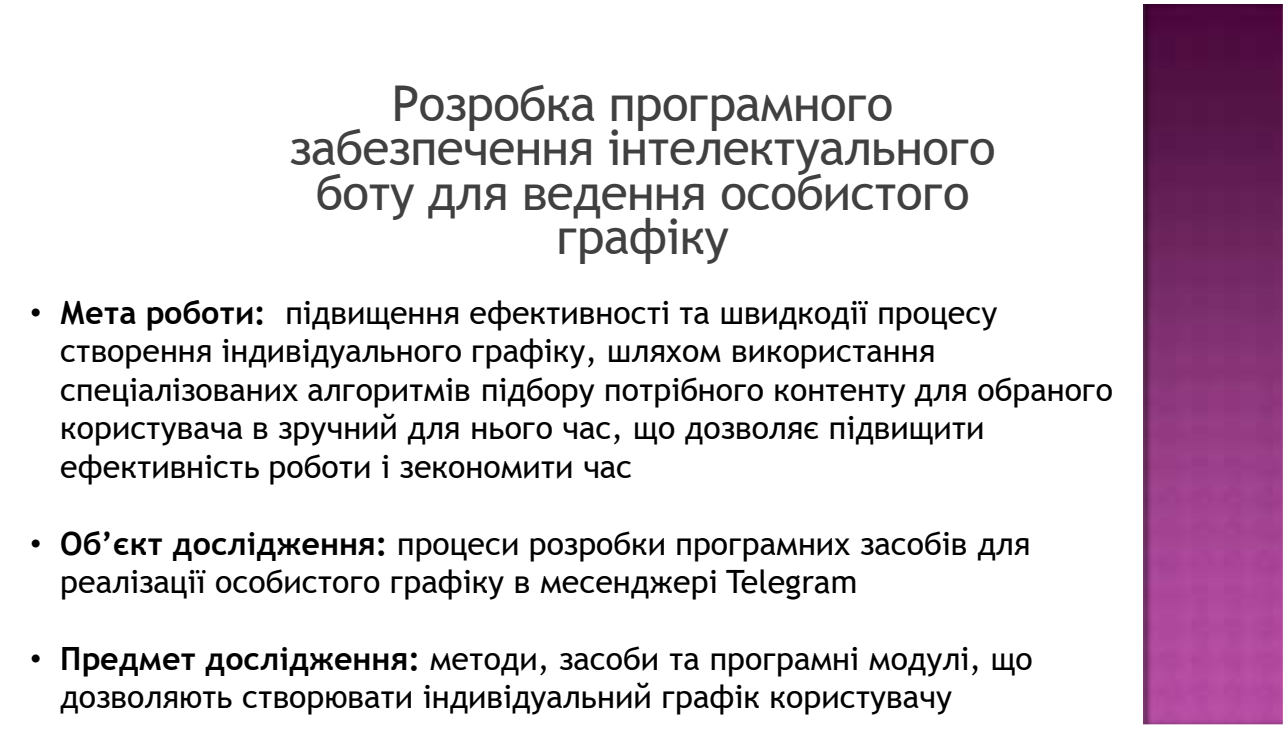

Рисунок Г.2 – Мета, об’єкт і предмет дослідження

Наукова новизна отриманих результатів: Удосконалено метод пошуку та вибору інформації, для створення персонального розкладу з урахуванням потреб конкретного користувача, шляхом визначення його персональної адреси, що підвищує надійність системи.

Практична цінність: Досягнуті реальні результати у використанні алгоритмів і можливості розробки програмного забезпечення, здатного полегшити та прискорити пошук і обробку інформації користувачем для створення персональних графіків.

Рисунок Г.3 – Наукова новизна та практична цінність

Метод пошуку та вибору інформації для створення персоналізованого графіку

Рисунок Г.4 – Метод пошуку та вибору інформації для створення персоналізованого графіку

Аналоги

Критерій	Gmail Bot	Ukrposhta chatbot	Tgrec bot	Russian war tribunal bot	Власний додаток
Зручність інтерфейсу	+	-	+	-	+
Відсутність необхідності реєстрації в інших сервісах	-	-	-	+	+
Безкоштовна розсилка повідомлень	+	+	-	+	+
Збір статистики	+	+	-	-	-
Можливість надсилати посилання, фото, відео	-	-	+	-	+
Сума	**3**	**2**	**3**	**3**	**4**

Рисунок Г.5 – Аналоги

Актуальність розробки

Ефективний розподіл часу відіграє важливу роль у житті людини. З давніх-давен люди планували своє майбутнє, щоб покращити своє життя. Особливо це актуально сьогодні в наш час. Оскільки темп життя різко прискорився, це вимагає відповідної реакції. Ключовим тут є ефективне та швидке використання інформації та її аналіз для конструктивного планування.
Занадто багато вузькоспеціалізованих ресурсів лише погіршує ситуацію, оскільки користувачам доводиться використовувати сторонні ресурси на додаток до своїх улюблених програм або веб-сайтів, що призводить лише до незадоволеності та втрати часу. Це значно уповільнює процес планування власної роботи.
Тому розробка програмного забезпечення інтелектуального боту для ведення особистого графіку є актуальною.

Рисунок Г.6 – Актуальність розробки

Загальна модель додатку

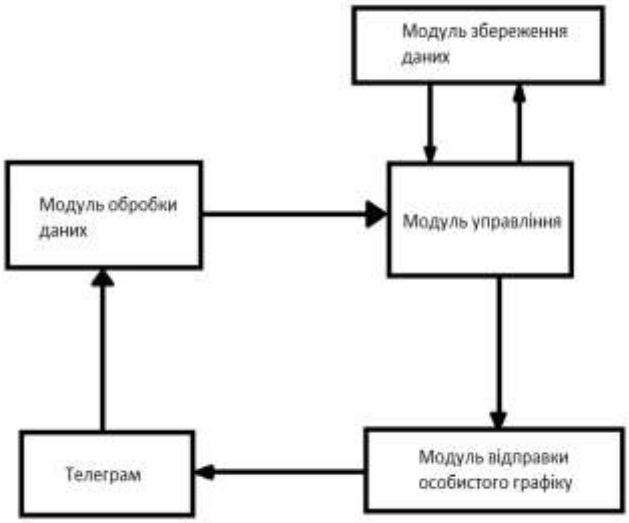

Рисунок Г.7 – Загальна модель додатку

Структура інтерфейсу додатку

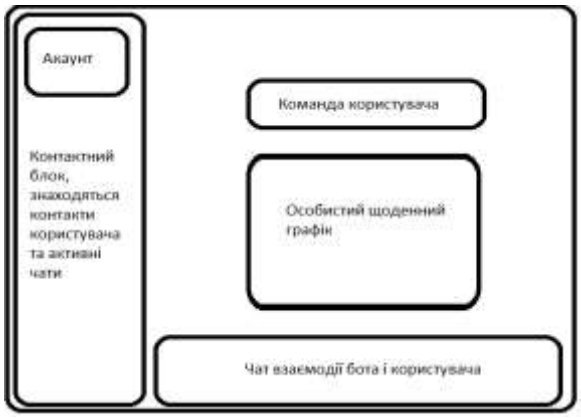

Рисунок Г.8 – Структура інтерфейсу додатку

ER-модель

Рисунок Г.9 – ER-модель

Інтерфейс з командами користувача

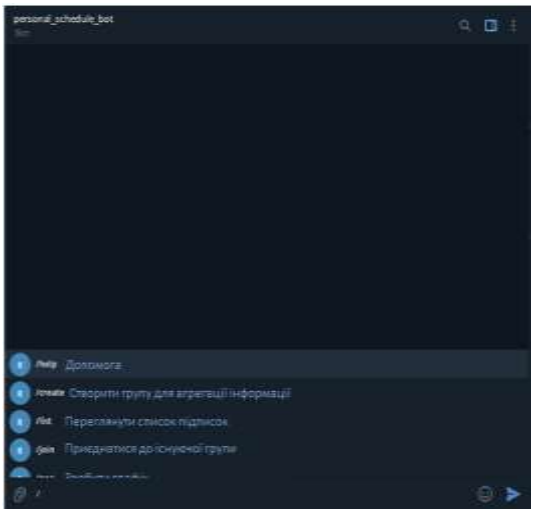

Рисунок Г.10 – Інтерфейс взаємодії користувача з додатком

Блок-схеми алгоритму роботи бота

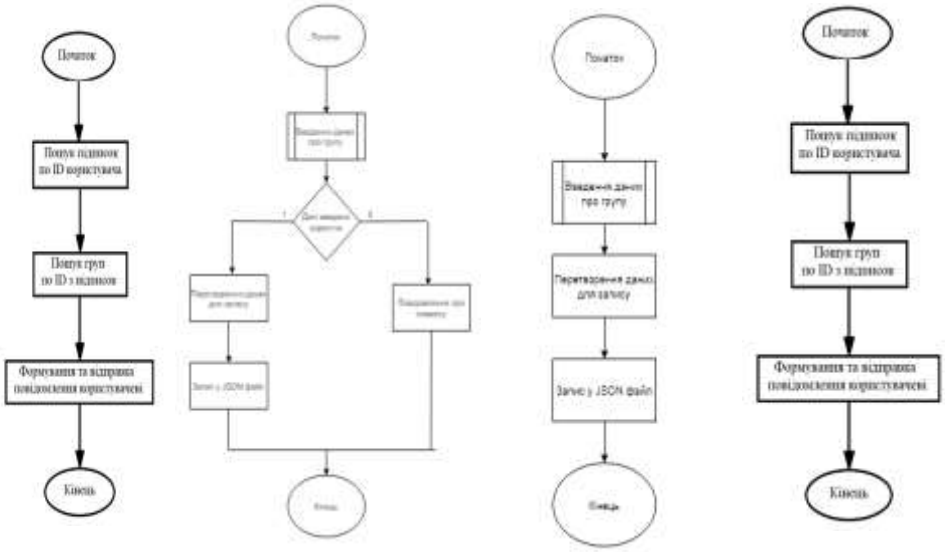

Рисунок Г.11 – Алгоритми команд бота

Тестування приєднання до групи/агрегатора

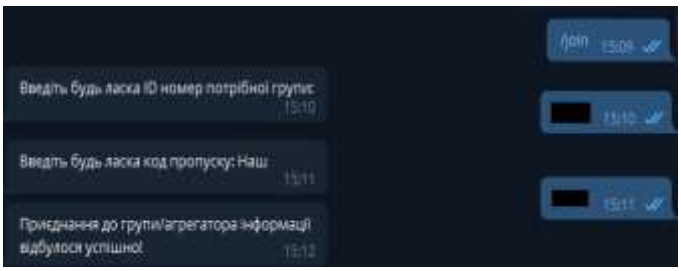

Рисунок Г.12 – Тестування приєднання до групи/агрегатора

Тестування отримання повідомлення з графіком від групи/агрегатора

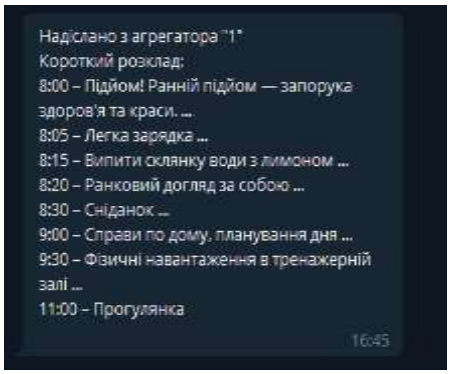

Рисунок Г.13 – Тестування відправлення особистого розкладу ботом

Тестування отримання розкладу в вигляді медіафайлу

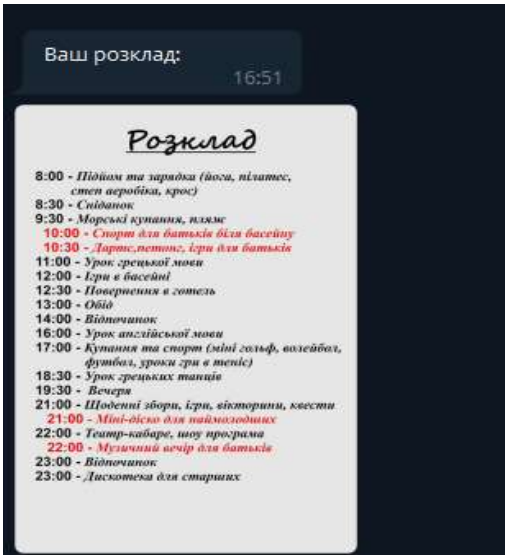

Рисунок Г.14 – Тестування відправлення розкладу в вигляді медіафайлу

Висновки

Під час виконання бакалаврської дипломної роботи було розроблено програмні засоби для реалізації стрічки інформації у месенджері Telegram. Для розробки було використано середовище програмування Visual Studio 2019. Робота розроблена згідно методичних вказівок.
Було проаналізовано стан даного питання на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом. У результаті аналізу обрано мову програмування C# та бібліотеку Telegram Bot Api.
У процесі виконання роботи було зроблено: розроблено алгоритм для підбору особистого графіку, розроблено графічний інтерфейс користувача для взаємодії із серверною частиною, розроблено програмні засоби для реалізації стрічки інформації особистого графіку користувача в меседжері Telegram, проведено тестування програмного продукту.
Перед створенням програмного продукту було розроблено метод пошуку та вибору інформації, для створення персонального розкладу.

Рисунок Г.15 – Висновки

Дякую за увагу!

Рисунок Г.16 – Кінцевий слайд